CAN A DISTRIBUTED ARCHITECTURE BE APPLIED TO PROFILE-BASED E-LEARNING?

Matthew Dyson

Background – This research is investigating the link between profile-based e-learning and distributed systems. Profile-based e-learning is a rapidly emerging field, whereby a user's previous learning history is analyzed and compared to that of other similar users, in order to determine the learning material that will have the greatest effect. The majority of systems favor an individual approach, and currently there is no model for interaction between implementations on different networks

Aims – The aim of this project is to examine the relationship between these two areas in order to determine whether widely implemented e-learning systems based around user profiles can be interlinked through the use of a distributed system in order to improve the quality of learning material suggested to the end user, forming a network whereby systems can co-operate and share learning material.

Method – This document describes the design of a protocol for interoperability of e-Learning nodes on a network, based upon search algorithms that will take a potential users previous history into account, which I will then implement and evaluate.

Proposed Solution - I propose to implement an e-Learning server with underlying interconnectivity protocols, in order that a distributed network (using the public internet as a medium) can share information to gather relevant material for display to a user. This system will then be tested to prove that users can access relevant shared material in order to benefit their learning.

Keywords – Distributed system, e-learning, profile, networking, social learning

I. INTRODUCTION

Current generation e-learning systems focus on providing content to users in a manner that appeals best to the material creator, whereby the easiest approach is taken to display the information the user is required to absorb. This does not take into consideration any particular method that the user may respond well to, and does not offer any flexibility for displaying the same content in different manners to different people.

Profile-based systems try to rectify this issue by adapting the displayed content to match a user's profile, a set of stored information which can be queried algorithmically in order to determine the best material to display based on the user's historical records, such as their learning style (visual/kinesthetic/auditory) or a particular method of teaching.

No consideration is currently given to the wide array of different content servers available, each with vast amounts of teaching material, and potentially user-related data on previous successes and failures to provide user-styled teaching.

In order to attempt proof of the theory that a distributed architecture can be implemented underneath a profile-based e-learning system, I will provide three tiers of deliverables, which are detailed below

Basic Deliverables

To achieve the most basic level of deliverables, I will implement the server-side of the e-learning system, which will go some way to proving the theory that a distributed architecture could be applied. This server will be aimed at organizations that wish to publish their data into a wider e-learning network.

Algorithms to determine the best learning material to display to the user will be developed at this level, although will be limited to data stored locally. This algorithm will take into consideration the users profile when selecting the resources to return, fulfilling the requirements of a profile-based e-learning system. This system will be implemented through a Remote Procedure Call interface, so that a client interface can be built on top. The methods made available through the interface will be documented and their use outlined, in order to create a rough specification.

Intermediate Deliverables

The intermediate level of deliverables will contain the implementation of the protocols necessary for servers to transfer e-learning resources to one another, and the extension of the client/server protocol to make this process transparent to the end user. Again, this will be done through RPC, and will be documented to give a specification of how this theory can be applied to multiple implementations rather than a single one, allowing multiple consumers to create their own bespoke software, which will still interact with others if the protocol is applied correctly.

An algorithm to allow the server to process the pool of incoming and locally stored resources in order to determine the most accurate data to return to the client will also be delivered at this level (an extension on the algorithm forming part of the basic deliverables), essentially creating a distributed search network.

Advanced Deliverables

A client program will be delivered at the advanced stage of development, which will implement all features offered by the e-learning server and wider network. The client software will demonstrate how organizations would implement connectivity to the network to allow users to access e-learning material not only locally, but throughout the distributed system to enhance their learning. This software should successfully detach the end user from the intricacies of the underlying network in order to prove the theory that a distributed architecture is appropriate in an e-learning system.

Although this implementation will demonstrate the functionality provided by the server and distributed network, it will only be a single view on how the protocol laid down by the server specification can be displayed to a user – the purpose of this project is to demonstrate the theory that this architecture can be applied, which will be proved by implementing it. On a wider scale, multiple clients and servers could be developed – it is the underlying protocol and algorithms that will determine whether this architecture is appropriate for use in this field.

Project Direction

As I examine within Related Work (below), the concept I am proposing is not one out of the grasp of current research, however the process of optimizing profile-based material selection algorithms that have been extended with a distributed layer form a challenging hurdle that must be overcome for this proposed next generation of learning systems to become commonplace. My intermediate deliverables should prove that this amalgamation is feasible, and the advanced deliverables will go some way to demonstrating the practicalities of my proposed implementation.

II. Related Work

The concept of e-learning systems has grown in popularity as the adoption of technology into businesses and homes has expanded. For many years it has offered a relatively cost-effective way for users to have an interactive learning experience, although many solutions found within business today are bespoke systems tailored towards the companies' individual needs. Several market leaders in the field have emerged, including some open source alternatives to commercial packages.

One such system is the proprietary VLE (Virtual Learning Environment) and course management system 'Blackboard Learn' (1), which aims to "enrich the education experience" that universities and other educational establishments offer to their students. Being aimed at the education market, Blackboard tailors itself by offering its content split down into individual courses, which can be assigned to users as necessary, and administrated individually by relevant staff members. One common open-source alternative to Blackboard is Moodle[1], which focuses on providing interactive quizzes to users, as well as a high level of syndication to allow data to freely flow in and out of the software as required by the individual implementation. Moodle has a large community developer following due to its high extensibility, as writing new modules to customize the software is extremely easy.

Some of these systems also introduce another element often discussed within the field of e-Learning - social interaction. Qing Li et al discussed this issue in (2), based upon the framework suggested by Qun Jin (3), both seeing the need for users to be able to collaborate and share learning progress with others. Many of the key issues surrounding the prospect of integrating current generation e-Learning into social media is discussed by Chatti et al., who suggest that "we need federated, intelligent and social search engines that build on user recommendations" (4), summarizing the case for building next generation e-Learning software upon a social backend in order to improve upon current systems. They also suggest that "knowledge pull" (the process of automatically gathering content from sources available on the internet to widen the pool of available learning data) is the optimum solution to get the best results from a social search algorithm.

The concept of creating a distributed network for learning material is not entirely new, however there is currently very little investigation into the field. Sampson et al. (5) suggest a similar approach as part of their "Knowledge on Demand" project[2], whereby the roles of content authors, brokers and providers are separated into individual implementations; however they do not go as far as suggesting a truly distributed nature. They do, however, outline the need for a singular standard in order to ensure the interoperability of different e-Learning systems, and hold this as key to the wide adoption of any future system. Similar themes are also picked up by Westerkamp in (6), as he suggests a web-based e-learning network based around a centralized learning server with many clients. He highlights in particular the advantages of a system being available on the internet, including interactivity with other content sources and user data, giving LearnServe[3] as an example of how this can be implemented.

The ideas of social interaction and distributed networking in an e-Learning system are not new concepts, and have been proved individually on a variety of both small- and large-scale projects, however the combination of these two in order to create a profile-based distributed learning network is a relatively new concept, hence this will form the base of my project.

[1] Modular Object-Oriented Dynamic Learning Environment
[2] http://kod.iti.gr/
[3] http://dbis-group.uni-muenster.de/projects/LearnServe/

III. Design

Requirements

There are two main stakeholders in a multi-site profile based e-learning system – the recipient of the learning material, and the content providers (examples including educational establishments and companies using Computer Based Training). People primarily use e-learning software to achieve knowledge of a certain topic in the most time-efficient manner, with the addition of wanting individually tailored results in the case of profile-based systems. The establishments providing the training information often are working towards the same goals – eliminating the overhead costs associated with providing teaching facilities to their staff, however with the system of networking that I am proposing; the element of information sharing is brought forwards. Establishments focusing on a single area of knowledge (a university concentrating research, for example) may not have the best teaching available for other areas, however with my proposed interchanging of information, the server (establishment) with the best information on a topic will be used more frequently, ensuring that the learner is presented with material that is widely accepted as high-quality, therefore achieving the primary goal of both stakeholders.

Taking a broader view of non-functional requirements, there are a large number of areas upon which potential users will evaluate the proposed system. One major factor is the usability of the system – if the introduction of the network layer increases the complexity of the interface too greatly, then the system will have failed, as the learning curve to get to the desired material will make the whole process too complex - users should be able to access and navigate material with ease. The actual design of client and server interfaces is out of the scope of this project (although interfaces will be developed as demonstration models only); however the integration of the wider distributed network should be transparent to the user and administrator. There is also the reliability and performance of the system to consider – any software using a networking layer has a greatly increased risk of under-performance and potential networking issues; however within the scope of this project I hope to eliminate these risks by providing the implementation with adequate functionality to overstep any issues in this area.

Use Cases

At the most simple level, the e-learning system I propose to implement forms a client/server architecture whereby a client program queries the server, and is returned material based upon the decision algorithms I will define later. The server software will require a basic graphical user interface in order that administrators can configure the server, and add learning material to the database. The basic use case for the client and server software is displayed in the sequence diagram Figure 1, which shows the user requesting material through the client interface, which then communicates with the server to decide upon the most suitable material to return (see Decision Algorithms) based upon the user information and material type requested. This material will then be parsed by the client program and displayed to the user in a readable format. Once a user has finished viewing a piece of learning material, they will be prompted to provide a rating, which will then be passed back to the server via the client. This rating system will form the basis of the adaptive profile-based decision algorithm on the server.

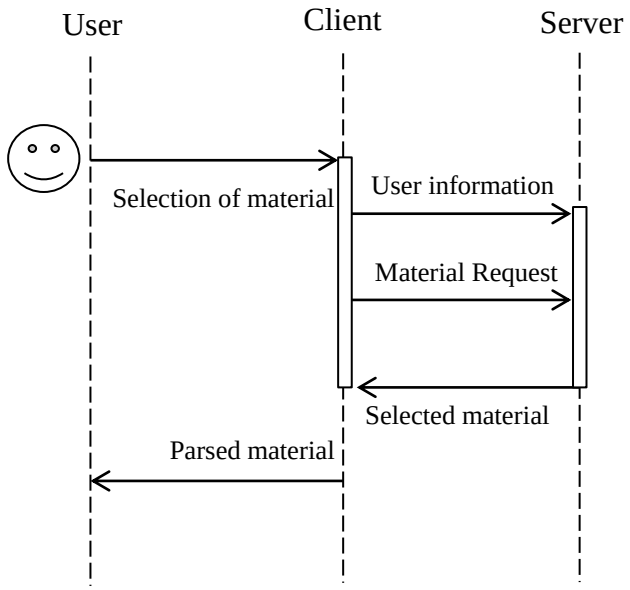

FIGURE 1: SEQUENCE DIAGRAM FOR BASIC DELIVERABLES

Low-level design

In order to make the system as widely accessible as possible, I intend to implement all of my deliverables through a web-based interface, with a Remote Procedure Call layer (see Networking Layer) to allow the different web services to interact, and to demonstrate how the decision algorithms can be tailored to make use of a distributed architecture. Theoretically, this system could be implemented in any language on any system that is connected to a public network (internet-facing), so long as the protocol is adhered to.

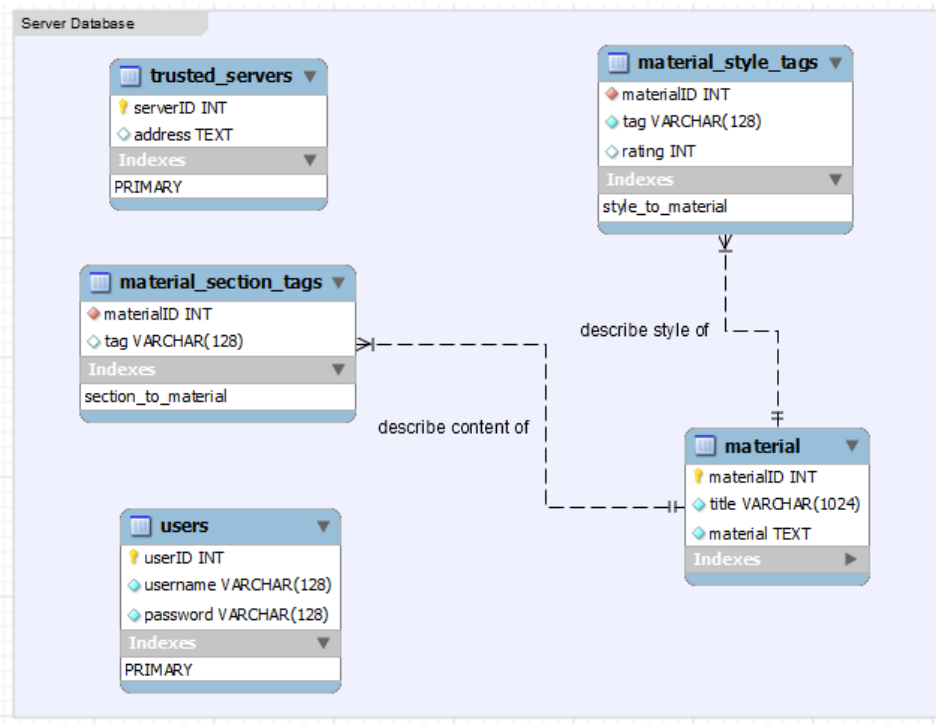

FIGURE 2: SERVER DATABASE ENTITY RELATIONSHIP DIAGRAM

The first deliverable to be developed will be the server. The server is responsible for holding all learning material, and for serving it as necessary to clients and other nodes on the distributed system. It is therefore necessary to implement an interface through which

authorized users can access this functionality and to add material, which ideally should remove the technical intricacies from the user. This server software will be duplicated over multiple different machines for evaluation, in order to create the distributed network

In order to form the base level information storage of the server, I will implement a MySQL database (simplified version shown in Figure 2), which will contain all the teaching

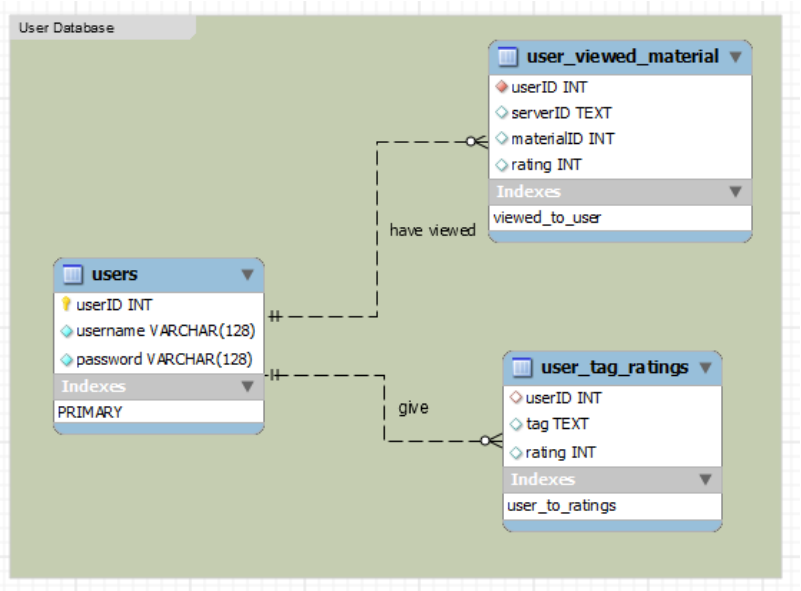

FIGURE 3: CLIENT DATABASE ENTITY RELATIONSHIP DIAGRAM

information necessary to create an e-learning system, along with the associated style and tag information (discussed later), information on other trusted servers (to form the distributed network) and user login information for housekeeping purposes.

The client database is also simplistic, containing only three tables (see Figure 3) which store the user's personal information (can be expanded from what is shown in Figure 3, simplified in this example), their relationship to material 'tags' and the material that they have previously viewed.

The tagging system I am proposing in this design is similar to that you find on many media types on the internet, such as blogs, download websites, and is present within the HTML specification itself. Within the definition of the HTML META tag, a programmer can define 'keywords' (7) in order to describe the content of a page in a manner that can be interpreted by search engine crawlers. Similarly, I intend to allow creators of material on the server to allocate two different kinds of tags to each article – one containing information on the area of expertise that the material is relevant to, and another containing tags relevant to the style in which the learning content is provided. Figure 4 provides a simple example of how tags would be applied to an article. In order for these tags to be relevant throughout the entirety of the network, it will be necessary for each server to offer a list of its tags to other servers

FIGURE 4: EXAMPLE MATERIAL TAG FORMAT

Content Title	TCP & UDP Packets
Content Material	"In this article, we will examine two common types of packets commonly used on the wider internet, with a video demonstration of their differences…"
Material Area Tags	computers; networking; internet; packets
Material Style Tags	visual; video

Decision Algorithms

For the decision algorithms implemented on the server to be able to make accurate suggestions on learning material based on user profiles, data must be stored that can be extrapolated to determine how a user reacts to certain styles of material. As mentioned above, tags will be stored for every piece of material kept on a server, and also for each user on a client implementation. In order for the system to learn how well users react to certain tags, the user will be asked to rate their interaction with the learning material after each session, through some kind of numerical representation. The client will then pass the users feedback to the server, where the appropriate database tables will be updated, as demonstrated in Figure 5.

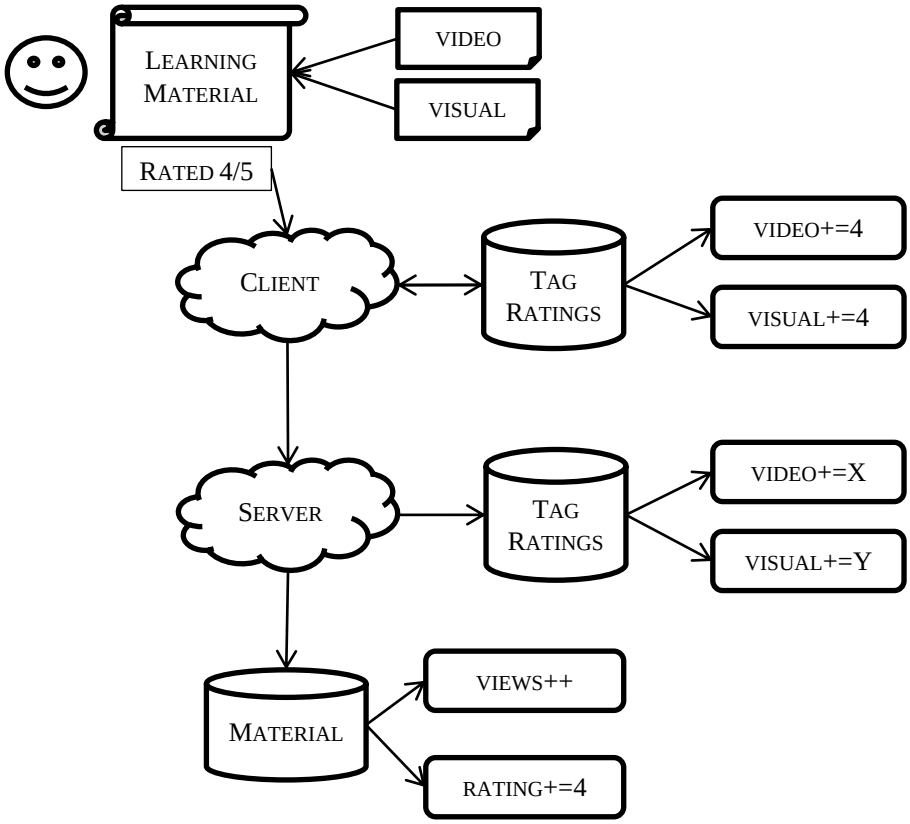

FIGURE 5: FLOW DIAGRAM FOR FEEDBACK PROPAGATION

In this example, the user has given a rating of 4 out of 5 to a piece of learning material that is tagged with 'video' and 'visual' style tags. The client software then immediately updates its internal database for the user in question, adding 4 to their score of the tags, and storing information relevant to the learning material to ensure it isn't shown to the user again (omitted on diagram for simplicity). The client software then passes this information on to the server, which will do one of two things. If the material it has been passed is not stored locally, it will use the networking layer to pass the rating forwards to the relevant server. If the material is stored locally (whether the information has come from a client or another server does not matter at this stage), then the internal tag ratings table will be updated with the relevant score (see below), and the material table will have the number of views incremented, and the overall rating increased by 4, so the average rating can be computed.

The two variables (X and Y) shown on the diagram require slightly more complex calculation. For instance, if a user rates a single piece of material highly, they do not necessarily rate all the tagged aspects highly. For this reason, passing the direct feedback score to the server would be counter-intuitive to the rating process, and a different approach

is necessary. For each user style tag, the number of times the tag has been rated will also be stored, and the mean score for that tag will be passed instead. This can be represented by Figure 6, where x is the style tag rating passed to the server, t is the tag, $r(t)$ is the rating(s) given to the tag by the user, and $f(t)$ is the number of times a user has rated the tag

$$x = \frac{\sum r(t)}{f(t)}$$

FIGURE 6: FORMULA FOR STYLE TAG RATING CALCULATION

Using the data that is provided by this accumulation of user feedback, it can be decided how effective a piece of material may be to a user, given their own tag history. For instance, a user that responds well to learning content with videos (a visual learner, perhaps) would have a high numeric corresponding with their individual 'video' tag. When this information is passed to a server, the server can examine its database of content relating to the desired topic (sent from the client) to ascertain which articles contain a high 'video' tag rating, which can then be returned. Iterating this method over multiple tags will further refine the process, and ensure that the optimal material is returned to the user for the given topic. A pseudocode implementation of this algorithm is given in Figure 7.

```
findContent(contentTags[tag], userStyleTags[tag->rating], quantity)
    fetch all matches from database having tags within contentTags[]
    for each matching content tag
        get article details

        matches[thisArticle][details]=article details
        matches[thisArticle][sectionTagsFound]++

        for each style tag of article
            if in userStyleTags
                suitabilityScore += user tag rating
                matchingTags++

        if matchingTags>0
            matches[thisArticle][suitability]=suitabilityScore
            matches[thisArticle][matchedTags]=matchingTags

    get content from other servers if wanted/necessary

    order matches array by (suitabilityScore/matchedTags) and sectionTagsFound
    return top matches from array (number of returns = quantity)
```

FIGURE 7: PSEUDOCODE IMPLEMENTATION OF SEARCH ALGORITHM

When a request for content is received by a server, three sets of information will be included: the content tag(s) that the user is looking for, the set of style tags to ratings that the user has in their history, and the quantity of pieces of material required. The server will then iterate through each requested content tag, pulling articles from the database that have that tag. For every found article, the style tags will be iterated through, and if the user has a given rating for this tag, their score will be appended to an overall 'suitability score' for the article. A sorting routine will then be applied based on the variables collected through this iteration process (also being applied to material gathered through the network) in order to determine the most suitable material for the user. The process of making this networking layer fit seamlessly into the search and sorting algorithm forms the crux of this project, and must be

achieved without the user being aware of the contents original location and within an acceptable timeframe for the implementation to be deemed successful.

This algorithm will obviously need some modification once implemented to give the best results, for instance by placing a higher weighting on the number of section tags found within the sorting procedure, and taking into account the quantity of matched style tags as well as the average score.

Using this process of iterating over multiple style tags within each article, the system can build up a clearer picture of how the user may respond, and hence can return the most suitable material. The feedback propagation process discussed above will ensure that each server holds the full set of ratings for its material, and hence the above algorithm will become more effective over time, as the server can build a clearer picture of the relationship between articles and style tags. By implementing this algorithm, and achieving its optimization over a distributed network, the goals of this project will have been realized.

Networking Layer

As discussed, the need for establishments to share information now exists; therefore a networking protocol for multiple e-learning servers to communicate is the key goal of this

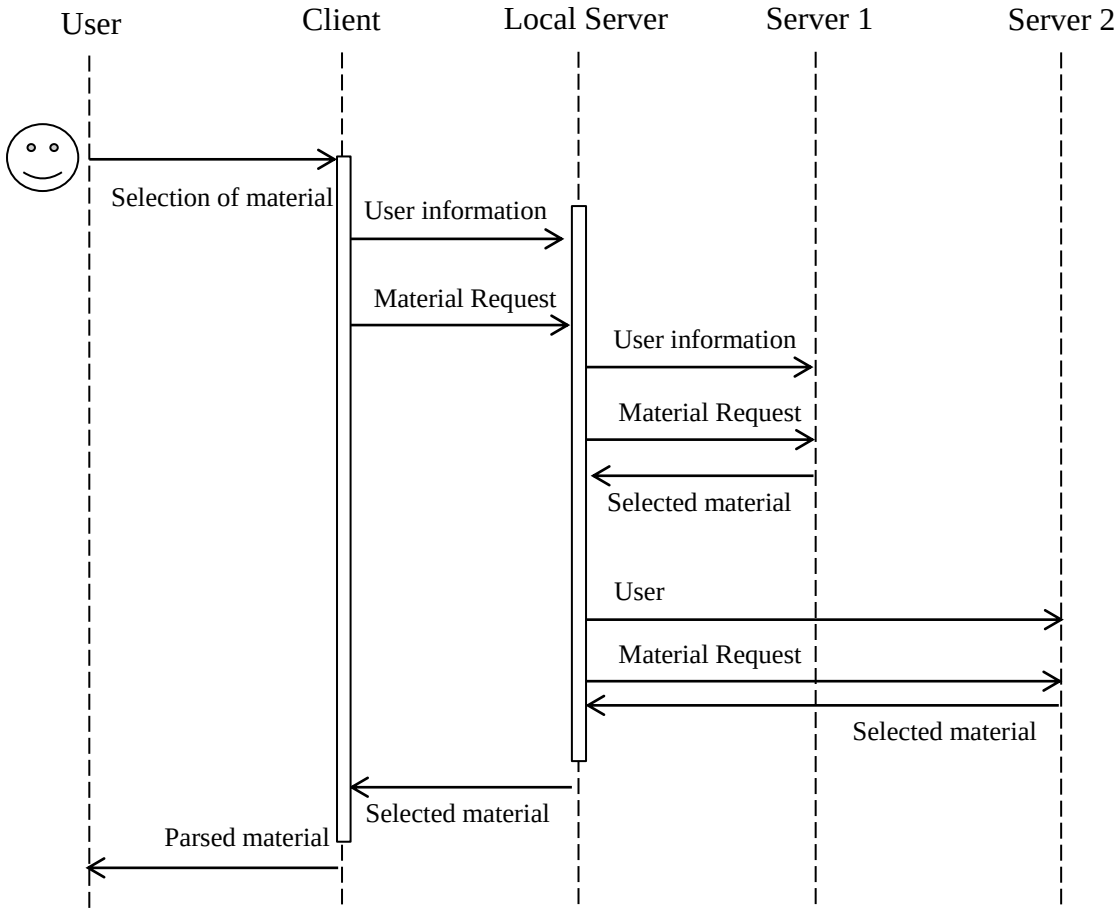

FIGURE 8: SEQUENCE DIAGRAM FOR MULTI-SERVER NETWORK

research. One issue that could potentially halt adoption of this e-learning network is the element of trust, therefore within the scope of this project it is assumed that a 'handshake' agreement is built into the protocol to ensure that both connected servers are permitted to share each other's data (as would be necessary in the real world), and that this is the basis for server acquisition, rather than any form of peer-to-peer networking.

The process for a user selecting a piece of material is similar to that shown in Figure 1: Sequence Diagram for basic deliverables; and to the client the change in process is completely transparent – demonstrated in Figure 8. Once a request is received from a client, the local server (the server that a client is connected directly to) will forward the request on to every server in its memory in turn, waiting a specified amount of time for a return before moving on to the next one. At the end of this process, the local server will assess each given piece of material for its suitability, and will then return the material it selects to be most suitable (see Decision Algorithms).

Given suitable allowances for network performance and processing power, and given that the network search algorithm is not recursive (in this content $(A \leftrightarrow B) \wedge (B \leftrightarrow C) \neq (A \leftrightarrow C)$ where $(A \leftrightarrow B)$ represents A trusting B), the response time of the network-enhanced search algorithm should be directly proportional to the number of servers the local server has in its database, which is an acceptable timeframe.

The implementation of this data transfer will be built upon the Remote Procedure Call architecture, to remove the need for any kind of persistent server software to be held in memory on a serving machine. The functionality required by servers and client machines will be available through a published interface that will respond to incoming requests by way of a JSON or similar return, data being passed into the server where necessary by HTTP POST, removing the need for any bespoke implementation of communication protocols, which would over-complicate potential adoption of the system.

Figure 9 shows a list of the proposed functions to be implemented by the RPC server, which covers the functionality needed by both clients and other servers to carry out all the tasks outlined above.

FIGURE 9: TABLE OF PROPOSED RPC FUNCTIONS

Method	**Response**
checkAuth	Server checks authentication from requester (client or server) and responds accordingly
getContentTags	A list of content tags currently held in the local database
getStyleTags	A list of style tags currently held in the local database
getMaterial	Requires the users profile information (style tags and associated weight, previously viewed material) and the content requested (content tags), returning the material decided upon by the content decision algorithm, including other server inputs if appropriate
materialRating	Updates the rating stored locally (or passes on to appropriate server) according to the passed information

IV. REFERENCES

1. **Blackboard Inc.** Blackboard Learn Platform. [Online] [Cited: December 29, 2010.] http://www.blackboard.com/Teaching-Learning/Learn-Platform.aspx.
2. *Technology Supports for Distributed and Collaborative Learning over the Internet.* **Qing Li, Rynson Lau, Timothy Shih, Frederick Li.** 2, s.l. : ACM Transactions on Internet Technology, 2008, Vol. 8.
3. *Design of a virtual community based interactive learning environment.* **Jin, Qun.** 1-2, s.l. : Information Sciences, 2002, Vol. 140.
4. *The future of e-learning: a shift to knowledge networking and social software.* **Chatti, M.A., Jarke, M. and Frosch-Wilke, D.** 4-5, s.l. : International Journal of Knowledge and Learning, 2007, Vol. 3.
5. *An Architecture for Web-based e-Learning Promoting Re-usable Adaptive Educational e-Content.* **Demetrios Sampson, Charalampos Karagiannidis & Fabrizio Cardinali.** 4, s.l. : Educational Technology & Society, 2002, Vol. 5.
6. *E-Learning as a Web Service.* **Westerkamp, Peter.** s.l. : Seventh International Database Engineering and Applications Symposium, 2003.
7. **Word Wide Web Consortium.** The global structure of an HTML document. [Online] [Cited: December 22, 2010.] http://www.w3.org/TR/html4/struct/global.html#h-7.4.4.

V. TABLE OF FIGURES