Final Major Project - Internet Engineering (H622)

Improve and supplement existing teaching methods with a system for broadcasting interactive content, in real-time, to students/pupils in a lecture/classroom

Outline Project Specification

Author: Thomas Usher *Supervisor:* Dave Price
Date: October 3, 2010 *Version:* 0.2 (Draft)

1 Project Description

With the increasing popularity of tablet computers such as Apple's iPad to enhance textbooks in education, using them as a resource for enhancing interaction between students and lecturers is the next logical step. The aim of this project is to build a system where lecturers can broadcast content (web sites, images, trivia, interactive exercises, etc.) to all students during a lecturer in real-time, increasing interactivity and enabling lecturers to receive immediate feedback from their students.

An example of how this system would work:

1. A lecturer would use the tools provided to construct a set of interactive slides before a lecture.
2. During the lecture, all students would access a certain URL in their tablet's browser.
3. The lecturer would choose to broadcast a certain interactive slide to all students when necessary.
4. All students' tablets would instantly update with the content.
5. If the slide is interactive, the lecturer would immediately receive notifications and statistics when a student responds.

The system will consist of three primary parts:

- A web-based management interface for lecturers; this will facilitate the creation of 'slides' and lesson plans. It will also serve as an area for lecturers to review responses and statistics from previous lectures.
- A distribution server to allow for real-time polling from clients.
- A web-based client to be accessed by students which receives and renders content.

The primary goal of this project is to improve interactivity during lectures, helping to engage students in the subject, improve attention-span and providing additional ways for students to feel personally involved in the subject. Secondarily, facilitating real-time feedback will allow lecturers to tailor their approach based on the class response - for example, the lecturer might broadcast a set of questions about the current topic, if a large percentage of students answer incorrectly, the lecturer might wish to attempt another explanation.

The system should be built around an extensible framework; allowing additional types of content to be broadcast, and to allow the system to be used in other use cases, such as school classrooms and business meetings. It should also be usable on any system with a relatively modern web-browser; while this project will target tablet computers, it should not use any tablet or device specific implementations.

2 Work to be Tackled

The administration system
To be built in a suitable server-side web programming language (such as Python[1], Ruby[2] or PHP[3]) - this will need to be built to allow lecturers to easily create different types of content for broadcast - likely making use of HTML5[4] and JavaScript[5]. It should also provide an interface for data interchange between the event server and the client, by outputting a suitable data interchange format (XML[6] or JSON[7]).

A distribution server
As a typical server is likely to struggle with the number of simultaneous requests generated by broadcasting content, this will most likely be built on top of an existing evented I/O framework (such as Node.js[8], Twisted[9] or EventMachine[10]. It will be necessary to learn callback-based programming techniques to ensure the server is non-blocking and can handle numerous simultaneous requests. This will need to implement some form of caching mechanism to reduce requests made to the admin system, possibly through the use of a persistent key-value store (MemcacheDB[11] or Redis[12]), It should be able to fetch and broadcast content from the administration system.

A web-based client
Will need to be able to receive content in real-time without additional HTTP requests, this will require the implementation of some form of real-time communication protocol/technique (such as WebSockets[13], Flash Sockets, or long-polling). On receiving content, the client should be able to parse and render it.

User authentication
Implementing a way to authenticate students and staff to ensure only enrolled students can receive interactive content.

3 Project Deliverables

- Report on languages, libraries and software to be used, including:
  - Server-side web programming languages and associated frameworks.
  - Real-time communication protocols.
  - Evented I/O systems.
  - Caching systems.
- Administration system, deployable on any Unix-based system.
- Distribution server, deployable on any Unix-based system.
- Web-based client, accessible through the distribution server.
- Progress Report
- Final Report

Initial Bibliography

[1] Python documentation. `http://docs.python.org/`.

[2] Ruby documentation. `http://www.ruby-lang.org/en/documentation/`.

[3] PHP documentation. http://php.net/docs.php.

[4] Ian Hickson. A vocabulary and associated APIs for HTML and XHTML, October 2010. http://dev.w3.org/html5/spec/.

[5] Javascript - mozilla developer center, September 2010. https://developer.mozilla.org/en/JavaScript.

[6] Tim Bray, Jean Paoli, C.M. Sperberg-McQueen, Eve Maler, and Franois Yergeau. Extensible markup language (XML) 1.0 (fifth edition), November 2008. http://www.w3.org/TR/REC-xml/.

[7] JSON. http://www.json.org/.

[8] Node API. http://nodejs.org/api.html.

[9] Twisted documentation. http://twistedmatrix.com/trac/wiki/Documentation.

[10] Eventmachine documentation. http://eventmachine.rubyforge.org/.

[11] MemcacheDB. http://memcachedb.org/.

[12] Redis. http://code.google.com/p/redis/.

[13] Ian Hickson. The websockets API, September 2010. http://dev.w3.org/html5/websockets/.