Computer Science 75
Fall 2009
Scribe Notes

Lecture 0: August 31, 2009
Andrew Sellergren

Contents

Computer Science 75
Fall 2009
Scribe Notes

Lecture 0: August 31, 2009
Andrew Sellergren

1 Welcome (0:00–5:00)

- O hai! Welcome to Computer Science E-75: Building Dynamic Websites!
- Since this course is all about dynamic websites, it seems appropriate to begin with static websites.
- We don't expect works of art in terms of design, but rather in terms of underlying functionality. We want you to be programmers, not just designers.
- This is a course in which students with *some* programming background will flourish and students with *no* programming background will also thrive if they are willing to work hard and bootstrap themselves.
- The course is structured around 5 projects, 4 of which we specify and 1 of which you come up with on your own and present at the end-of-the-semester Computer Science Fair. You will have three weeks to complete these projects and on average you will require about 10 hours over the course of those three weeks to accomplish the task at hand.
- What do we mean by dynamic? Well, for starters, the websites we design will serve up more than just static content using HTML and CSS. The content of our websites will actually change based on the user who's viewing it. We'll accomplish this by focusing on a LAMP (Linux Apache MySQL PHP) framework as well as employing Ajax. Ajax is the fantastic new technology that allows for sleeker user interfaces that don't require page refreshes nearly as often. It's responsible, for example, for the drag feature of Google Maps which seemed like such a giant leap forward from MapQuest. These days, the mentality is to shunt as much work to the client as you can in order to reduce load on the server.
- We're taking a holistic approach in this course in the sense that we'll be investigating why certain things work and others don't—things that aren't necessarily familiar to typical web developers. We'll be using tools to examine HTTP header traffic, for example, one of them being Firebug, a plugin for Firefox.
- Because of the diverse backgrounds of students in this course, no doubt some of the material will occasionally be review, but it will be important for us to cover all our bases so that no one is left in the dark.
-

2 DNS (5:00–55:00)

- What actually happens when you pull up a web browser, type cnn.com into the address bar, and hit Enter? The host name, in this case cnn.com,

is translated or *resolved* into a numeric IP address which is useful to computers.

- Who did we query to get this IP address? First, the request will go to the nearest DNS server, which hopefully has cached the answer. If it has the IP address in its database, it returns that to our browser in a packet.
- Now that we have the IP address, what happens next? Think of a virtual envelope. In the upper left corner, we'll have the return address, which is our computer's IP address. We'll address the envelope to the IP address that we just looked up.
- How does this virtual envelope get to its destination? It actually travels through a number of routers or gateways. If you've ever run the `ipconfig` command on Windows or `ifconfig` on a Mac, you know that your computer is assigned both an IP address and a default gateway. This is the "closest" gateway through which all of its traffic will be routed. This default gateway then knows where to pass the packet next so that it can get one step closer to its destination. Generally, there are fewer than 30 gateways between the origin of a request and its destination.
- If we want to take a peek at the gateways along this route, we can use a program called `traceroute`. David will SSH into his own personal domain (since Harvard blocks this kind of route tracing) and then execute the command `traceroute cnn.com`. Now he can see the IP addresses of all the gateways which passed his information along.
- If we examine this program's output, we see that our information gets routed through a Qwest ISP and also a server that's presumably in New Jersey (since its IP begins with EWR, the airport code for Newark). The `Level3.net` suffix corresponds to one of the large Internet backbones. Ultimately, we hit a dead end (designated by `* * *`) because someone decided they wouldn't divulge where they were sending the information.
- Let's do the same for stanford.edu. We see that it goes through Harvard's servers again, then through `nox.org`, the Northern crossroads—a hub in the Northeast where multiple ISPs interconnect. Then it looks like we're going through Washington, D.C., Atlanta, Houston, and Los Angeles. Eventually we reach stanford.edu. What's different about this time compared to last? Well, the request times are noticeably longer. This is because the data actually had to travel longer distances which aren't insignificant, it turns out! Even so, it only took a total of 90 milliseconds for our data to travel 3000 miles. Pretty good!
- Would web traffic experience the same bottlenecks we're seeing here? Not necessarily. The protocol that `ping` and `traceroute` use is called ICMP. A lot of system administrators disable that specific protocol (which is why we see asterisks for some of the steps).

- If we access cnn.co.jp, the Japanese version of CNN, we see that we get relatively fast results considering that between two of the steps, we must hop over an entire ocean.
- Note that differences between two runs of `traceroute` could be attributable to different networks or just random chance.
- If you'd like to run these commands at home, run `traceroute` from Terminal on a Mac or the command `tracert` on a Windows machine.
- IPv4 is the current protocol for assigning IP addresses, which specifies that each much be comprised of 4 8-bit numbers for a total of 32 bits. This means there are $2^32 \approx 4$ billion IP addresses available in the world. This may sound like a lot, but it's actually not enough. That's why IPv6, which specifies 128-bit addresses, is being rolled out.
- Harvard has IP addresses in the range of 140.247.x.y, which means that we have a total of $2^16 = 65,536$ IP addresses to assign. In contrast, MIT has IP addresses in the range of 18.x.y.z, which means they have many more at their disposal. Still, it's worth noting that there's not a one-to-one mapping between URLs and IP addresses because more than one website can be hosted on the same webserver, which has its own IP address. Our own webserver, cs75.net, in fact, will host all of your websites (100+) this semester. One of the consequences of this is that none of you will be able to use SSL because it requires a unique IP address.
- So we've had one request go out to look up the IP address of the URL and then another request to the IP address itself. What happens once the request gets to the webserver? The webserver will retrieve the file—in the case of cnn.com, it will be the default `index.html` file—and send it back to your browser. Your browser will then parse the file and display it according to the standards of HTML (e.g. bold for a `<b>` tag).
- If there are pictures or other files linked from the file, then another request must go out for each additional file. This has consequences, of course, because each request has a certain amount of overhead associated with it. You can see how many of these requests are made by enabling the Net panel of Firebug (more details on that later). For cs75.net, there are 16 requests that go across the wire. If you think that's a lot, try cnn.com: there are over 100 requests made.
- One of the concepts we'll talk about this semester is scalability. When you have multiple users accessing the same resource or many thousands of users accessing the webserver period, user experience tends to degrade as request time increases.
- Question: is it better to have 1 request for 100 KB or 100 requests for 1 KB each? It depends. In the case of CNN, for example, it would certainly

be better to load all the text content for the site before trying to load any videos so that the user might have something to read while he's waiting.

- Google is certainly a good example of an uncluttered site, but even it has added some extra links in recent years. Interestingly, the "I'm Feeling Lucky" button costs them a considerable amount of money because it whisks the user away to the first search result, bypassing all the advertisements.
- Think now about a large-scale operation like Google and the consequences of adding an extra space at the end of a long file of code. If that file is requested a billion times over the course of a day, that's a billion extra bytes that were needlessly transferred across the wire!
- As part of Project 0, you will purchase your own domain name so that your websites for this course will have a much cooler address than they otherwise would. And don't worry, we'll take care of hosting your websites on the course's server!
- GoDaddy.com is the registrar we're recommending, but you're welcome to use whoever you want. Know that GoDaddy will try to upsell you at every turn, so you'll get a lot of good practice at clicking the "No Thanks" link.
- Note that you can choose from a number of Top-Level Domains (TLD) when picking your hostname. This is the familiar last part of the web address (e.g. .com, .net, .org, .info). Some TLDs are restricted (e.g. .edu, .gov), but for the most part you have free choice if the domain name isn't already taken.
- Why don't most American websites have .us as their TLD? Well, simply because we took charge of .com![1] Websites around the world tend to have country-code top-level domains (ccTLD) like .aq for Antarctica or .va for Vatican City. Interestingly, .tv is actually a ccTLD for Tuvalu but is often used for vanity's sake to stand for television. (Check out cs75.tv!)
- Once the course is over, you're welcome to use your domain name for whatever purposes you wish (hopefully benevolent, appropriate ones) although you'll eventually have to find someone else to host your files.
- There are smaller registrars out there now that might over you a domain name for less, but be careful to choose one that's at least slightly reputable so that there's a good chance it'll still be around in a year and beyond.
- There are a few central bodies who handle domain name registration, namely ICANN and IANA. Most likely, they charge middlemen like GoDaddy a certain fee for accessing the database of domain names and inputting new entries.

[1] America is the best. Just ask this guy.

- Once you've bought a domain name, you're going to tell your registrar who your nameservers are. Ultimately, it will be the course's nameservers, i.e. ns1.cs75.net and ns2.cs75.net, that know all about your domain name and what IP address it maps to—they are the so-called *authoritative nameservers* for your domain. We have a total of four IP addresses—one of these is used for all of your websites.
- What if you didn't have a course affiliation? You can tell GoDaddy not what the nameservers are, but actually what the IP address is of your website.

3 More About the Course (55:00–75:00)

- We have three teaching staff, in addition to David. Dan Armendariz will act as head teaching fellow and Siddarth Chandrasekaran will be an on-campus teaching fellow. Andrew Sellergren (me!) will be an online teaching fellow.
- Sid will lead an on-campus section once a week and Andrew will lead an online section once a week. Each is meant to dive into details that we don't have time to thoroughly discuss in lecture. Online section will take place using conferencing software called Elluminate which allows for audio, video, and screen sharing.
- In terms of prerequisites for the course, we assume only that you have *some* prior programming experience in a high-level language such as Java, JavaScript, PHP, Perl, C, C++, C#. If you only have experience making websites using HTML and CSS or programs like Dreamweaver or you've only taken one semester of computer science (e.g. E-50a), then you'll have to work a little harder, but you can still ultimately succeed in this course.
- All lectures will be videotaped and available online, usually within 72 hours of being so eloquently delivered by David in person.[2]
- The expectations of the course are to watch all lectures and to complete 5 projects, 4 of which we will assign and 1 of which you will design yourself. Project 0 is very simple and only requires you to setup your domain name and make a very basic website. Projects 1, 2, 3, and the Final Project will be the more time-consuming ones.
- What will you take away from this course? Today we'll cover the basics of DNS and HTTP and next time we'll introduce you to PHP, a well-documented, function-rich (albeit occasionally ugly) server-side scripting language which will empower you to create dynamic websites. Soon after, we'll hit upon XML, a meta-language that will allow you to implement

[2] (scoff).

configuration and data files. For Project 1, this will enable you to implement a flat-text file that can then be parsed by your website in order to display the menu of Three Aces Pizza. After XML (and a brief discussion of XPath and the DOM), we'll get into the more sophisticated database language called SQL (specifically MySQL). We'll follow that up with in-depth discussion of JavaScript and Ajax and how to use them to create effective user interfaces. Finally, we'll get into security issues.

- Projects! Project 0, as mentioned, will simply get you set up on the course's VPS and teh interwebs. Project 1 will be your implementation of the Three Aces menu (or at least part of it) as well as an online ordering system. The trick will be making it easy to manage so that your not-so-technically-savvy friend, the owner, will be able to maintain it after you finish—so we'll use XML, which Project 2 will be your E*Trade-like website that allows users to register and buy/sell stocks using live CSV (Comma-separated Values) data from Yahoo Finance. It will be punctuated by a competition in which each of you will have your own stock portfolio to manage. Project 3 will be a mashup whereby you will combine Google Maps' API (Application Programming Interface) and Google News' RSS feed to display markers with news headlines centered around a location provided by the user. Lastly, the Final Project will be of your own design (after staff approval). In December, you will have the opportunity to display your Final Project at the Computer Science Fair.

- Books! Not required! Check out the Resources section of the course website for free supplemental materials. If you *do* believe that you would benefit from additional reading material, check out the Books section of the syllabus.

- The course website! Let it be your lodestar this semester! Once you're done with Project 0, you'll be equipped with a user account for the course website which will allow you to access the bulletin board and check grades online.

- If you have questions which don't involve your own work specifically (i.e. What's wrong with my code?), you should post them to the course's bulletin board. If you have a more private question, feel free to e-mail us at help@cs75.net.

- Office Hours! Mostly by appointment, but even if you are remote, you'll be able to interact with us via the Virtual Terminal Room. This uses Elluminate's software, which will allow us to chat with you but also to screen share with you and even take control of your terminal, if necessary, to help with debugging and the like.

- After the course is over, if you want your website to live on, you can pay for web hosting yourself. One popular web hosting service is Dreamhost (although we've had a bad experience with them and wouldn't give them

our recommendation anymore). Still, the feature set they offer is representative of what is available these days. With most web hosting companies, you can get unlimited e-mail addresses, unlimited databases, and several GBs of storage space all for a very low monthly fee. You should probably look to spend between $10 and $50 per month.

- One of the advantages of doing our own web hosting is that we have root access over the server. We've set up the server as a Virtual Private Server (VPS) through a company called ServInt which uses special software to partition a single machine so it will function as multiple servers. The resources of the machine, i.e. hard disk space and CPU cores and RAM, are likewise divided among multiple installations of the operating system, in our case CentOS 5. One neat trick which this enables is the ability to transfer an entire server from one machine to another without any downtime whatsoever.

- On top of the VPS, we run what's called a panel. cPanel is perhaps the most popular (despite being somewhat poorly designed) of these web-hosting control panels, but we use DirectAdmin, provided free for the university, which is just as reliable and much easier to use. DirectAdmin provides a convenient GUI wrapper so that we can abstract away a lot of the messy details of administering a Linux server via SSH and the command line.

- Of course, you can also access your account by SSHing to cs75.net and executing Linux commands. For example, you can write your code directly from the command line if you choose to. If you choose to develop locally, you can upload files to your website via SFTP. More details on that in section.

- Ultimately, everything you need to develop websites for this course is freely available online. So you can write all your code on your home computer if you so choose and only upload the files in order to submit your projects. In fact, we encourage you to develop on your home machine using XAMPP, a software bundle that includes Apache, MySQL, PHP and Perl. With this bundle, you can use your own computer as a web server and test out your web pages before you make them live and public by putting them on the course's web server.

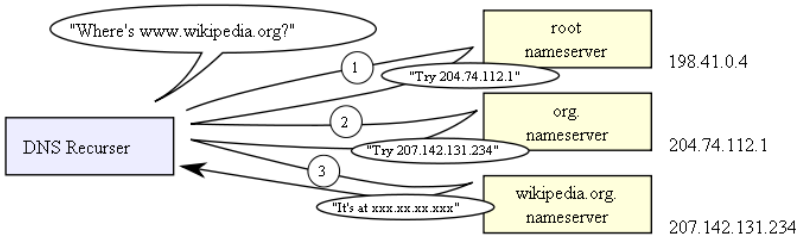
- Another technology we'll make available to you is that of virtual machines. Using free software called VirtualBox, you'll be able to set up an environment on your home computer which is nearly identical to cs75.net. In this way will you be able to develop offline and locally without running into too many configuration problems when you go to transfer your site to the live server.[3] Generally, VirtualBox will work fairly well for software

[3]David has vetted this software in his extremely nerdy endeavor to play the 8-bit game King's Quest, which requires DOS, on his MacBook Air. I'm not sure whether to applaud or weep.

applications that don't require a lot of graphics processing.

4 More on DNS and HTTP (90:00–109:00)

- Let's take a look at a diagram representing a DNS lookup so that we can delve into one detail that David overlooked in our earlier discussion:



- As the above diagrams conveys, DNS lookups are recursive and the entire system of DNS servers is hierarchical. When you type in wikipedia.org into your address bar, your browser will begin by querying one of the 13 root nameservers[4] That root nameserver will spit out the IP address of the .org nameserver, which will spit out the IP address of the wikipedia.org nameserver, and so on until your browser finds the correct IP address.

- This diagram would suggest that the internet doesn't scale very well if there are only 13 root nameservers in the entire world. However, thanks to caching, these nameservers don't get queried all that often. Once wikipedia.org's IP address has been looked up once, that information will be stored by users or DNS servers or both so that in the future, time won't have to be wasted looking it up again. Luckily, this information doesn't change all that often. However, one downside is that if you as a sysadmin make a mistake and hardcode the wrong IP address into your DNS records or a webserver whose IP address has been cached actually goes down then users will often hit a deadend.

- What's the deal with dynamic DNS? You can sign up for free with a company like DynDNS which will provide you with a DNS name to map to an IP address of interest. For example, if you want to SSH or RDP to your home computer, you might sign up for this service because your IP address is liable to change every once in a while. You could, actually, run a whole server this way, but it's not recommended. There are certainly more robust solutions available.

- HTTP stands for Hypertext Transfer Protocol. But what does it really mean? Recall our discussion of the virtual envelope that we sent to a

[4]That is, if this is the first time you've ever visited wikipedia.org. Come on, be honest, I know it isn't. It's probably not even the first time you've visited during this *class*.

webserver. What's inside the envelope? Basically, it's just a request for content. More technically, it might be a message like "GET /index.html HTTP/1.0."

- We don't really care what browser you use in this course, but we will be using Firefox for the majority of our demonstrations because it offers a plethora of web development tools, including Firebug and Live HTTP Headers. We *will* require that your website render identically on two major browsers. This is not meant to drive you completely insane, but just as a real-world experience in cross-browser compatibility.

- If we open up Live HTTP Headers, we can actually sniff out the virtual envelope that we're sending to the webserver. If we make a request to cnn.com, this is the first two lines of the first header which is sent:

  ```
  GET / HTTP/1.1
  Host: www.cnn.com
  ```

  If all of your websites are being hosted on our course server, then all requests will contain the same IP address. How will it know which website to spit out? Thanks to virtual hosting, the request will contain not only an IP address, but also a single line of text in the HTTP header that represents what the user actually typed into the address bar. Here, for example, it says Host: www.cnn.com. This will be interpreted by the webserver itself so that it knows which home directory to access and which files to return.

- A quick note: we'll require your websites to have well-formed XHTML. On a basic level, this means that every time you open a tag, you also close it. And with respect to quotation marks, you close with what you opened with—single quotation marks with single quotation marks or double with double.

- Moreover, your XHTML must be *valid*. This means that it conforms to the W3C standards. Practically, this means it passes the validation test at validator.w3.org. You can't, for example, make up your own tag names.

- We'll encourage you to use CSS to improve the style of your websites, but it won't be our primary focus in the course. You can embed CSS directly in the XHTML or otherwise link to it via stylesheets. We don't care whether your CSS is valid or not.

- We touched upon cross-browser issues earlier. This is one of the biggest pains in web development today due to slight aesthetic and functional differences between IE, Firefox, Safari, and Chrome. We'll make good use of YUI's libraries to help standardize styles across browsers.

- Next week, we'll get into some actual programming!