Computer Science 010
Introduction to Computer Applications
Siena College
Fall 2010

Topic Notes: Networks

Our next major topic is networking – in particular **the** network – the *Internet*.

But first, some basics. What is a *network*? Simply put, it is a connection among computers that allow those computers to share information. The computers and other devices connected to a network are often called *nodes*. Nodes are connected within a network via a *router* or similar device.

We will think of the information we wish to send on a network as a *bitstream*, just a collection of 0's and 1's. Usually, these bitstreams are similar to the file formats we just discussed. The network is just a way to transmit use of binary data. Just as the contents of a file are meaningless unless we have some rules to lay out the bits of the file in an organized way and to interpret them as the information they are supposed to represent, the information transmitted on a network is a meaningless stream of 0's and 1's unless we have rules for how to organize it and interpret it.

Siena has a network on campus. I have a network at home.

There is no direct connection between my home network and Siena's network – that connection comes via the Internet (an "inter network" or "network of networks").

Network Basics

We will consider networks in three layers:

- Low-level infrastructure
  - Ethernet
  - TCP/IP
  - packets
- Higher-level architecture (topology)
  - nodes, routers, wired/wireless links
- Services delivered across the network
  - file downloads (ftp)
  - world wide web (http)
  - email (smtp)

  – ...

Low-Level Infrastructure

There are three low-level technologies required for a network to function:

1. Rules for "talking" on the communication medium (who can be sending a signal on the wire). *Ethernet* is a very common example.
2. An *addressing* scheme to allow information sent from one computer to another. *IP* is the Internet addressing scheme.
3. Some structure in this communication (a *protocol*) to manage a "conversation" between two computers. *TCP* is a very common protocol for reliable communication.

Ethernet is built into most modern computers and other networked devices.

- An Ethernet port looks kind of like a phone jack, and an RJ45 connector clicks into the port.
- The Ethernet protocol manages collisions on the communication medium (the wire). If more than one device attempts to "talk" at the same time, the "back off" protocol allows the devices to try again at different times. Ethernet will allow each device to have a turn to talk, since each device is required to let other devices have a chance to talk.

IP (the *Internet Protocol*) network addressing scheme gives each computer on the network a unique address.

This address is used to *route* the data from a sender to a receiver.

These addresses take the form of a 32-bit number, written as a sequence of 4 8-bit numbers, separated by .'s. The names we know (like `siena.edu`, `google.com`, `courses.teresco.org`) are translated to these numbers. We can look these up using the `nslookup` tool.

Every computer on the Internet has an *IP address*. When your computer sends something onto the network, it must provide the appropriate IP address of the destination.

TCP (the *Transmission Control Protocol*) manages the IP "conversations":

- making the initial connection (get the other computer's attention)
- agreeing on how to connect
- how to deal with errors, if part of the "conversation" is lost, how does a retransmission get requested and sent?

Packet Switching

Suppose we would like to have a “conversation” involving 8 MB of data (*e.g.*, a file download).

We could send the data as one large “chunk” of 8 MB of data (think: a big freight train that can carry all of our data, or sending a single package containing a 1000 page document through the mail).

Or, we could break down the data into smaller, independent pieces (called *packets*), and send them out (think: a fleet of smaller data trucks, or sending that 1000 page document in 1000 separate envelopes, with pages numbered so we can put them in order when they arrive).

Which of these approaches will be more reliable and robust when there are traffic disruptions (as there will be on a network)?

Well, if the big chunk of data does not arrive successfully (which would be the case if any part of it was lost along the way), the entire chunk would need to be retransmitted. And since it’s so large and will take a long time to transmit, there is a very good chance that a problem will arise.

On the other hand, if we sent out a large collection of smaller packets, there is a very good chance that some of those packets will not arrive successfully. But..there is also a very good chance that the majority of them will arrive successfully. In this case, we need only retransmit the ones that failed.

All bitstreams on the Internet are broken down into packets.

Each packet has the sender’s IP address and the destination IP address.

One of the reasons that the Internet is so reliable is that each packet can take a different route to its destination. If a link becomes unavailable, unreliable, or simply too busy, traffic can be routed along other paths.

Internet Services

The world wide web is just one example of a *service* that can be transmitted via the Internet.

There are thousands of services, many of which we can see at `http://www.iana.org/assignments/port-numbers`, the list of *ports* assigned to services by the *Internet Assigned Numbers Authority (IANA)*.

If we looked through this list, we would see only two deal with the web: `http` on port 80, and `https` on port 443.

Some others that are important and perhaps familiar:

- 21: file transfer protocol (ftp)
- 22: secure shell (ssh)
- 23: telnet

- 25: simple mail transfer protocol (smtp; email transmission)
- 143: internet message access protocol (imap; email fetch from server)

These port numbers corresponding to services allow incoming network data that arrives at a computer to be directed to the appropriate program running on that computer. Port 80 traffic goes to the web server, port 22 traffic goes to the interactive remote login server, port 25 goes to the email server, *etc.*.

Low-Level Internet Summary

Key ideas about the low level network:

- Rules for traffic control (Ethernet)
  - avoid and resolve collisions
- Common addressing scheme (IP)
  - host name (`siena.edu`) translated to IP address
- A conversation protocol (TCP)
  - conversation sent in packets
- Services offered over the connection
  - http, ftp, smtp, IM, etc,

Higher-Level Network Architecture

How does your network device get information to "the Internet"?

- The network device needs a connection to an *Internet Service Provider (ISP)*
- This connection is often called "The Last Mile"
- Typical connection types:
  - "dialup" over a traditional telephone line (24-52 Kbps)
  - DSL - a digital subscriber line, provided by phone companies (400-800 Kbps)
  - Cable (1+ Mbps)
  - Wireless/cell phone network
- Often "asymmetric" – faster down (from net) than up (to net)

- DSL is likely dedicated, cable is shared with neighbors

This gets you to the ISP. As an individual, you are making a connection to the ISP's *Local Area Network (LAN)*.

The ISP then needs to connect to a *Network Access Point (NAP)*.

- Typically done with cable connections such as T1 (1.5 Mbps) or T3 (42 Mbps) lines.
- NAPs are owned by large telecommunication companies.
- NAPs are interconnected into a *Wide Area Network (WAN)* using cable connections such as OC3 (155 Mbps), OC48 (2.5 Gbps), OC192 (9.6 Gbps). These make up the Internet *Backbone*.

See a visualization of what the Internet "looks like" at `http://en.wikipedia.org/wiki/File:Internet_map_1024.jpg`

A Network of Routers

Recall that all Internet traffic is broken down into a set of packets. It is these packets that need to be routed from one computer to another on the Internet.

This is achieved by sending the packets through a network of routers.

Each connection point on the Internet has a *router computer*.

A router keeps track of:

- which other routers it is connected to
- how much traffic is going to the other routers

Each packet (of a bitstream) is examined by each router it goes through. The router decides the best direction (*i.e.,* route) to send the packet to next.

At the destination, the original bitstream is sssembled from its many packets, many of which may have been lost and retransmitted, and which may have arrived in any order.

Each router needs only understand where to send a packet next to get it closer to its destination. No one router has knowledge of the entire path the packet has taken and/or will be taking.

Domain Name Service (DNS)

How do all those Internet names (*hostnames* or *domains*) that we know so well get translated into IP addresses, and who decides which names go to which addresses?

The answer to both is that there is a central authority in charge of names and numbers:

The Internet Corporation for Assigned Names and Numbers (ICANN), `http://www.icann.org/`

Any name to be recognized on the Internet must be registered with ICANN. Any IP addresses associated with those names must be assigned by ICANN.

Individuals or organizations do not contact ICANN directly to register names and obtain numbers. Other companies/organizations called *domain registrars* do these registrations.

You've almost certainly seen ads for some of these like GoDaddy or Network Solutions. I use one in France called Gandi.

The individual or organization that desires a name finds one that's not already in use and pays one of these domain registrars to add that name to the appropriate *top-level domain* (`.com`, `.org`, `.net`, *etc.*) and optionally allocates one or more IP addresses for that domain's use.

With the domain registered (*e.g.*, `siena.edu`, `teresco.org`), a domain name server can then be set up to map names within the domain (*e.g.*, `www.siena.edu`, `blackboard.siena.edu`, `courses.teresco.org`) to IP addresses.

Each domain has an *authoritative domain name server* that is the one place that manages name to number mapping for that domain.

But how does every computer in the world know how to find the authoritative domain name server for a particular domain like `siena.edu` or `teresco.org`?

ICANN maintains a set of name servers called *root servers* that know how to look up the authoritative domain server for each *top-level domain* (*e.g.*, `.com`, `.org`, `.net`). So the `.org` server knows how to find my name server for `teresco.org`.

But does that mean every time anyone on the planet needs anything in my domain, my server needs to do that translation? No, because the name servers throughout the Internet will maintain a *cache* of the names they have looked up in the recent past. If the same name is requested again, the number in the cache is used and the authoritative server is not queried again.

This system is hierarchical in the sense that a local name server looks up names for one local network (like Siena's). All computers at Siena will first query that server. If the server already knows the number for a given name (and it will for the most common and popular lookups), it returns it and the work is done. If not, it knows who to ask: the name server at Siena's network provider. If the network provider's servers know the number for the given name, it returns it to Siena's server (which remembers it for a while, just in case it's asked again) and provides the number to the computer that issued the request. Only when the request gets all the way up the chain to the root servers without being answered does the authoritative domain name server need to answer the query.

Some tools to try out:

- nslookup: look up IP addresses for host names
- whois: look up information about domain registrations

Client-Server Model for Network Services

We've talked about services on the Internet, but how are these services provided?

The basic method here is the *client-server model*. A central *service provider* or *server* on the network awaits requests from other machines, the *clients*, for whatever information that server can provide.

The server has a known address (its IP address, which we obtain from a name using DNS), so clients know "who" to ask.

When a request arrives at the server, it sets up communication with the client to find out what information is being requested. If the information is available, it is sent to the client.

This model is used by a wide variety of services on the Internet:

- DNS
- Web
- time servers
- some file sharing (like our SOSAD files in lab)
- email

Let's think about what happens when you wish to view a web page.

- You enter a URI (aside: this is a *uniform resource identifier*, also often referred to as a URL, or uniform resource locator), such as `http://xkcd.com` into your web browser (the client).
- The browser looks up the name by contacting a DNS server (and possibly having that request forwarded to other servers as we discussed earlier).
- The name server responds with the IP address 72.26.203.99.
- The browser then sends a request for `http` service from the web server at IP address 72.26.203.99.
- The server receives the request and sends back a response, in this case the contents of a web page.
- The browser receives this information and displays the page.

Peer to Peer Model

While the client-server model is very popular, and works well in many circumstances, it is not always the most appropriate.

Perhaps the most glaring problem is that all clients need to rely on the same server.

- The server may be limited in what services it can provide.
- The server may be too slow to handle all of the requests coming from the clients.
- The server may fail or be unreachable on the network.

Consider a famous example from Internet history: Napster – shut down by a court ruling.

In a pure *Peer to Peer* model, there is no central server to provide services, but many *peers*. Each peer knows the IP addresses of many of its peers.

Each peer runs software that listens for requests and has some resources to share (the ability to service some requests).

A request for a resource by a peer in the network is sent to all known peers. If one of those peers can satisfy the request, it does so. If not, it forwards the request to other peers.

If a resource is available somewhere in the peer to peer network, it eventually is located and provided to the original requestor.

Examples: BitTorrent, Limewire

World Wide Web

We next will look in some more detail at the one nternet service many of us use more than any other: the *World Wide Web (WWW)*.

Tim Berners-Lee's big idea: use a client-server model to implement *hypertext* for sharing of documents at CERN, the European Particle Physics Laboratory, in 1989.

Hypertext is what allows *links* within a document to lead to another.

The "language" of the web is *Hyper Text Markup Language (HTML)*. This is the description of the page that tells the *browser* (the web client) how to display the content.

- The HTML page is *rendered* by the browser.
- The exact standards for how the rendering is to be done are established by the *World Wide Web Consortium (W3C)*, `http://www.w3.org`.

- See the HTML that generated the rendered page in your browser with "Page Source" or "View Source" options.

Recall that the web is a client-server system. The browser (client) sends a request to a web server, receives back the HTML for the page, and the browser renders it to the screen.

The "hypertext" idea comes in when the HTML page includes a *link*:

```
<a href="http://courses.teresco.org/cs010_f10/">CSIS 010</a>
```

When the page is rendered, the text "CSIS 010" is displayed as a link, and when the user clicks on that link in the browser, it generates a request for that page and displays it when the server has provided it.

A Web Server

A *web server* is a computer connected to the Internet that has a program listening for http requests and responds to those requests.

The most popular web server program is called *Apache*. Apache is just a program like any other, except that its job is to respond to web server requests.

The server also needs to have the pages and images that the web server program will send out. These are in files on the web server just like other files.

The web server program interprets the request and provides the appropriate file.

The URI includes information used by the server to find the appropriate file. For example, the URI

```
http://courses.teresco.org/cs010_f10/lectures/lect10/index.html
```

consists of three main parts:

1. `http` indicates Hyper Text Transfer Protocol (the standard web service) is to be used, which uses port 80.
2. `courses.teresco.org` is the host name of the web server. We have already seen how DNS converts this to the appropriate IP address.
3. `cs010_f10/lectures/lect10/index.html` provides the name of the file on the web server that we wish to display.

In this case, the web server running at `courses.teresco.org` will look on its hard disk in the location that contains web server files for the site `courses.teresco.org`. The server configuration defines this as a *directory* (folder) named `/home/www/courses`.

Within that directory, the server looks for the file specified in the rest of the URI: `cs010_f10/lectures/lect`

This means it looks in `/home/www/courses` for a directory named `cs010_f10`, that in turn contains a directory `lectures`, which in turn contains a directory `lect10`, which in turn contains the file `index.html`. And that is the file the web server sends back across the network to the client (browser).

Web Clients (Browsers)

A brief history:

- Berners-Lee's first web client was called "WorldWideWeb" first made available in late 1990 and 1991.
- The "Mosaic" browser from NCSA (National Center for Supercomputing Applications at the University of Illinois) was the first widely popular browser, starting in 1993.
- Mosaic went on to become "Netscape Navigator" and later "Mozilla" and eventually today's popular "Firefox" browser.
- Microsoft's entry came in 1995 with "Internet Explorer", which became and remains the most frequently used browser.
- "Opera" is a commercial browser that has a small but loyal user base.
- Apple's "Safari" browser arrived in 2003, and is the most popular Mac-based browser today.
- Google entered the browser market in 2008, with "Chrome".

Any browser can render any web page correctly, as long as the web page follows W3C standards and the browser follows those standards.

The browser formats text and images according to the HTML it receives from a web server.

Browsers have evolved to allow *plugins* for things like animations and sounds, and to have programs involved in the display of pages using technologies like Javascript.

Building a Web Page on the School of Science Server

Next, we will build a very simple web page on Siena's School of Science web server, `www.cs.siena.edu`, also known as `ares.cs.siena.edu`.

Regular users like you and I are not allowed to log into `ares`. We can only transfer files to it using a *secure copy* or *secure file transfer*.

Our first step is to create a file of HTML text that can be rendered by a browser into the web page we wish to see displayed. There are countless *web authoring tools* available to aid in this process, but we will keep it very simple. We will use a plain text editor and insert the markup (the HTML *tags*) that will tell the browser how to display.

The file in this case is on the SOSAD file server (the "Z Drive") that you know and love from labs. I have named it `cs010test.html`. The `.html` extension indicates that it will contain HTML text.

A nice simple way to edit the file is with Windows Notepad.

Once we have the file ready to test out in a browser, we can load it into our browser with a `file:` URL. The easiest way to do this is with the browser's "Open File" function.

We can then go back and make changes to the HTML file and reload them in the browser to see what happens.

Once we're ready to publish, we need to transfer the file to the web server. We can use "WinSCP" for this. This program lets us transfer files. We will connect WinSCP to `ares.cs.siena.edu`. When prompted, log in with your regular SoS username and password. When you are connected to the server, you will need to navigate to the `public_html` folder, as that is where your web pages need to be placed to be visible on the server. Select the files to transfer and execute the transfer.

We can now view the pages at a specific URL in our browser (or any browser on the planet, for that matter). My file is found at:

```
http://www.cs.siena.edu/~jteresco/cs010test.html
```

Any files I place in my `public_html` folder on the server will be seen at a similar URL. Your URLs will use your SoS username in place of mine.