LBSC 690: Information Technology Lecture 02 Networking and the Internet

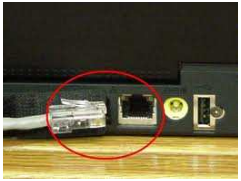
William Webber
CIS, University of Maryland

Spring semester, 2012

The Network

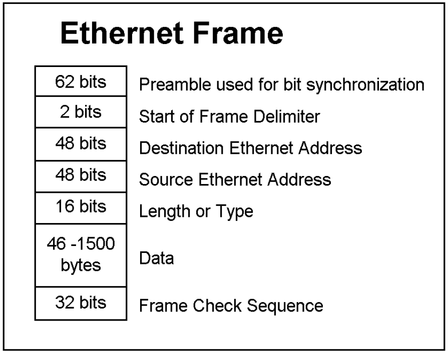
- Networking is pushing bits down wires (and over wavelengths) between one computer and another.
- Essentially, we need:
  - A transport medium (copper wire, fiber, spectrum)
  - A way of transmitting (modulating, digital $\rightarrow$ analogue), transporting, and receiving (demodulating, analogue $\rightarrow$ digital) bits
  - A way of addressing the bits from source to destination

Ethernet: the medium



- Transport medium is cable (coaxial, twisted pair, or fiber-optic)
- Connected with other computers in a local area network by hubs
- Bits are:
  - Coaxial, twisted pair: positive, negative voltage for 1, 0
  - Fiber-optic: pulses of light

Ethernet: transportation



Ethernet Frame

62 bits	Preamble used for bit synchronization
2 bits	Start of Frame Delimiter
48 bits	Destination Ethernet Address
48 bits	Source Ethernet Address
16 bits	Length or Type
46 -1500 bytes	Data
32 bits	Frame Check Sequence

- Data sent out over network in *frame*
- Frame is a sequence of bit segments, defined by protocol
- Frame wraps data in envelope with from, to address

Ethernet: addresses

```
f0:de:f1:0e:de:2f
```

- Each Ethernet device has a 48-bit identifier, called a “MAC address”; generally printed using 12 hexadecimal values
- Each Ethernet frame contains destination and source MAC address
- All messages broadcast to every machine on network
- Device knows that frame is “for it” if it contains its MAC address

Ethernet: messaging protocol

Basic problem on shared medium: who gets to transmit?

- Ethernet solution: anyone can transmit whenever
- Having transmitted, they check to see if someone else transmitted at the same time
- If so, they wait for random period of time, transmit again

Diversion: hexadecimal numbers

Decimal	181
Binary	1011 0101
Hex	B5

- Decimal is based ten (0-9); binary base 2 (0-1); hexadecimal base 16 (0-E).
- One hexadecimal digit represents four binary digits (bits).
- Hexadecimal frequently used to represent “inherently” binary values

Excursus

- What happens if two devices on the same LAN have the same MAC address?
- If MAC addresses are assigned purely at random, how likely is it that any two given addresses will be the same?
- If Ethernet frames are broadcast to every computer on the LAN, what stops one computer snooping on another's communications?
- What about wireless?

Ethernet: error detection and correction

Ethernet frame contains two segments for error detection, correction:

1. Preamble: alternating pattern of 101010 . . . bits, to detect the frame, followed by 11 to start body of frame
2. Frame check: Pattern of bits, calculated depending on content of rest of frame. Automatically detects error bursts up to 32 bits long.

Simplest check code is a single (“parity”) bit, set so that the number of 1 bits in a message is even. What is the maximum number of errors that this code can detect?

Beyond the LAN

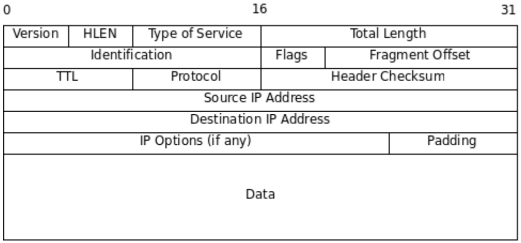
- Ethernet only takes us as far as the door of the lab (or the router-modem on our desk)
- We need a way of talking beyond our own local area network
- . . . and of communicating with computers on local area networks using a different technology (e.g. wireless)

IP: the Internet Protocol

Internet protocol: allows communication between any two machines worldwide (and off planet!). Requirements:

1. How to package up data
2. How to address machines worldwide
3. How to find the way from one machine to another
4. How to transit over the various network technologies

IP: the datagram packet



Protocol Field: 0x01 = ICMP, 0x06 = TCP, 0x11 = UDP

- IP packet, like Ethernet frame, combines header with data
- Header contains to, from address, and other management fields
- Data can be up to 65k in size

IP: addresses

Internet addresses, under IP version 4 (IPv4), are made up of 4 numbers, from 0 to 255, separated by ".". E.g.:

192.168.1.125	the IP address of my laptop as I write this (at home)
128.8.237.26	the IP address of the server hosting the University of Maryland
184.26.100.110	the IP address of the server hosting the Whitehouse home page

Excursus: the IP address space

IP addresses: four values from 0 to 255

- How many bits in the binary representation? How many bytes?
- How many distinct addresses does this support? Does that seem enough?
- The internet is (very slowly) migrating to the new IPv6 protocol, which has 128 bits. How many addresses does this support?

IP addresses: allocation

- IP addresses are allocated by central organization to registrars, and further subdivided to ISPs, companies, etc., to maintain uniqueness.
- There are also certain “private” IP addresses (e.g. 192.168.1.XXX) which are only on the local network; bridge to internet must translate
- And 127.0.0.1 always means “the current computer”

IP routing: finding a way from one machine to another

- Beyond local network, packet must be sent through a router.
- LAN typically has a single router (“gateway”) to the internet
- Internet backbone uses complicated route discovery algorithms that weigh:
  - Availability
  - Distance
  - Load

  to dynamically find a route for packets

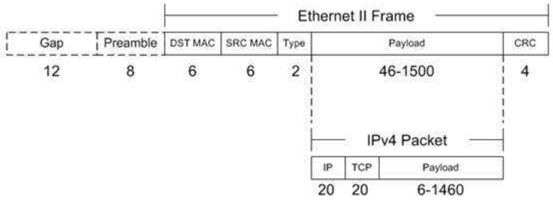
IP routing: from UMD to UniMelb

Source IP: 128.8.107.183   Destination IP: 128.250.148.40

Command: `traceroute www.unimelb.edu.au`

```
1 Vlan224.css-dual-r1.net.umd.edu (128.8.225.252) 0.368 ms
2 Gi3-10.css-core-r1.net.umd.edu (128.8.0.29) 0.381 ms
3 Gi3-2.ptx-fw-r1.net.umd.edu (128.8.0.86) 0.548 ms
4 129.2.0.246 (129.2.0.246) 0.446 ms
5 clpk-umd-i2.maxgigapop.net (206.196.177.152) 0.708 ms
6 xe-7-2-0-0.lvl3-t640.maxgigapop.net (206.196.178.90) 2.235 ms
7 i2-lvl3.maxgigapop.net (206.196.178.46) 27.103 ms
8 ae-8.10.rtr.atla.net.internet2.edu (64.57.28.6) 15.438 ms
9 xe-1-0-0.0.rtr.hous.net.internet2.edu (64.57.28.112) 38.818 ms
10 * ge-6-1-0.0.rtr.losa.net.internet2.edu (64.57.28.96) 567.616 ms
11 * * *
12 so-1-0-0.bb1.a.hnl.aarnet.net.au (202.158.194.109) 149.043 ms
13 so-2-1-0.bb1.a.syd.aarnet.net.au (202.158.194.105) 228.471 ms
14 so-2-0-0.bb1.a.mel.aarnet.net.au (202.158.194.33) 240.650 ms
15 ge-0-0-0.bb1.b.mel.aarnet.net.au (202.158.194.182) 240.428 ms
16 tengigabitethernet2-1.er2.unimelb.cpe.aarnet.net.au (202.158.200.99) 255.421 ms
17 gw1.er2.unimelb.cpe.aarnet.net.au (202.158.206.162) 255.276 ms
...
```

IP transport: routing over diverse networks



- Each time packet goes through gateway, router, it potentially transits from one network protocol to another:
  - 802.11g from laptop to wireless router
  - Ethernet from wireless router to DSL modem
  - ATM from DSL modem to telephone exchange
  - SONET for internet backbone
- IP packet is wrapped in the (lower-level) packet/frame format of transport layer
- This can mean splitting packet into multiple frames (e.g. up to 65k IP into 1.5k Ethernet)

The protocol stack

Protocol	Layer
HTTP	Application
TCP	Transport
IP	Internet
Ethernet	Data link
IEEE 802.3u	Physical

- There are layers above IP, and layers below Ethernet
- Each layer wraps data from the layer above
- Each layer relies upon the layer below to carry out lower-level operations
- This defines a protocol stack

TCP: packets to connections

Protocol	Layer
HTTP	Application
TCP	Transport
IP	Internet
Ethernet	Data link
IEEE 802.3u	Physical

- Ethernet, IP packet-based systems:
  - No guarantee packets will arrive in sequence or at all.
  - No standard reply mechanism.
- TCP (transmission control protocol) provides reliable, persistent, two-way connections
- Implemented as another packet format, wrapped in IP.

TCP as transport layer and IP as inter-network layer are so ubiquitous that the two are almost always combined: TCP-IP.

TCP ports: linking programs, not just computers

- IP links computers, not programs
- TCP implements program addressing through concept of *ports*; 16-bit numbers (like phone extensions)
- Servers listen on well-known, low-number ports:
  - Unencrypted web servers listen by default on port 80.
  - Encrypted web servers listen on port 443.
  - Mail servers listen on port 25.
- Clients (e.g. web browser) open arbitrary high-numbered ports to contact server, tell server their port number

The application layer

Protocol	Layer
HTTP	Application
TCP	Transport
IP	Internet
Ethernet	Data link
IEEE 802.3u	Physical

- TCP provides connection, but programs still need to agree on how to talk to each other
- This need filled by application-specific protocols
- Different programs have different protocols for communication. We'll look just at just one: HTTP.

HTTP: requesting and sending web pages

- HTTP (Hyper-Text Transport Protocol) provides protocol for web server, browser to communicate.
- A URL typed into a browser location box, such as:

  `http://www.umd.edu/directories/colleges.cfm`

  has a few main components

http:	the protocol (others include https:, ftp:).
www.umd.edu	the server to send the request to (port defaults here to port 80, but can be specified by ":<PORT>" at end).
directories/colleges.cfm	the "path" to request from the server. Often refers to a file to retrieve, but can have arbitrary interpretation to server.

HTTP: request header

```
GET /directories/colleges.cfm HTTP/1.1
Host: www.umd.edu
User-Agent: Mozilla/5.0 (Ubuntu; X11; Linux x86_64; rv:9.0.1)
    Gecko/20100101 Firefox/9.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection: keep-alive
Cookie: __switchTo5x=XXX; __unam=XXX; __utma=XXX; __utmz=XXX;
    CFID=XXX; CFTOKEN=XXX; __utmb=XXX; __utmc=XXX
Cache-Control: max-age=0
```

Figure: HTTP request

What my browser sends when requesting the UMD directory of colleges and schools (cookie contents blacked out)

HTTP: response header

```
HTTP/1.1 200 OK
Date: Mon, 06 Feb 2012 16:09:23 GMT
Server: Apache
Transfer-Encoding: chunked
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

Figure: HTTP response header

The start of what the UMD web server sends my browser; the actual web page itself follows after this.

Calling servers by names, not numbers

- IP addresses are 32-bit numbers (e.g. 128.8.237.26, for the UMD web server).
- But numbers are hard to remember.
- Also, numbers encode specifics of network topology that might change (if a web site moves to a different hosting provider, for instance).
- It is easier, and superior, to use names to refer to servers.

DNS: a directory for the internet

- DNS (the Domain Name System) provides a distributed database of host name to IP address (and vice versa) mappings:
  - User provides host name (e.g. www.umd.edu)
  - Program requests IP address for name from nearest domain name server (e.g. 128.8.237.26)
  - DNS request has its own application-layer protocol, on top of TCP or alternative transport layer called UDP, in turn on top of IP.
  - Program uses returned IP address to address IP packet to host.
- Your computer must know the IP address (not host name) of DNS server! (but this often provided by dynamic configuration)

DNS: anatomy of hostnames

- Hostnames have hierarchical structure, reading from right to left.
- For instance, “http://www.cis.unimelb.edu.au/”:

.au	This is an Australian domain
.edu	Among au domains, an educational one
.unimelb	Amongst au educational domains, this is called “unimelb” (University of Melbourne)
www.cis	Amongst unimelb domains, this is the “cis” subdomain (Computing and Information systems), and the “www” machine

- Common TLD (top-level domains) are national domains, plus “.com”, “.edu”, “.gov”, “.org” etc.. Latter are ambiguously international or US domains.

DNS: a distributed, hierarchical database

Every DNS server cannot store every domain name!

- Requests forwarded to top-level domain server, redirected to authoritative server
- If changing host name to IP address mapping, only need to change authoritative server for for domain.
- Lookups cached by other servers for speed; cached results expire after a period.

End-to-end: fetching a web page

1. User types URL into browser, or clicks link, e.g. `http://www.umd.edu/directories/colleges.cfm`
2. Browser parses URL into protocol (`http:`), hostname (`www.umd.edu`), and path (`/directories/colleges.cfm`)
3. Browser opens TCP-IP connection to DNS server by IP address (at work, `128.8.76.2`) (details as per web page request below).
4. Browser creates DNS protocol request for IP address for hostname `www.umd.edu`, sends to DNS server.
5. DNS server `128.8.76.2` recursively asks authoritative DNS server for `umd.edu` domain, `128.8.5.2`.
6. DNS server responds to browser with IP address `128.8.237.77`, also using DNS protocol (and caches the answer).
7. Web browser requests operating system to open TCP connection to `128.8.237.77` on port 80.
8. Operating system opens high port, say 12020, on local machine, returns connection to it to web browser.
9. Web browser constructs HTTP request, starting:

   GET /directories/colleges.cfm HTTP/1.1
   Host: www.umd.edu

   writes it to connection.
10. Operating system wraps HTTP request in TCP packet giving to port of 12020, from port of 80, sequence number of 1, other details.
11. Wraps TCP packet in IP packet giving to IP address of `128.8.237.77`, from address of `128.8.107.183` (my laptop address).
12. TCP packet split up into 1500 byte chunks (if necessary), wrapped in Ethernet frames, sent to LAN gateway.
13. LAN gateway joins up Ethernet frames, strips header and footer, reconstitutes IP packet, looks at address, figures out next step on route.
14. Gateway wraps IP packet in networking protocol for step to next router.
15. Subsequent routers repeat steps 13 and 14, until IP packet arrives at server.
16. Operating system of server receives IP packet, peeks at TCP header to find port 80.
17. Operating system strips (but remembers) IP and TCP wrappings, extracts contents (which are an HTTP request, but OS doesn't care), writes contents to web server's end of connection.
18. Server splits off new, high port to manage this connection (say, 25121); will inform browser of new port number with return TCP packet.
19. Web server reads HTTP request, sees that it is for path `/directories/colleges.cfm`, takes whatever steps necessary to construct content for this path (in this case, runs a script in a web development language called Cold Fusion).
20. Web server construct HTTP response, with header starting

    HTTP/1.1 200 OK
    Date: Mon, 06 Feb 2012 16:09:23 GMT

    and after header, the content of the web page.
21. Operating system of server wraps response in TCP packet, giving local high port of 25121, remote high port of 12020.
22. Wraps TCP packet in IP package, with from address of `128.8.237.77`, and to address of `128.8.76.2`.
23. IP packet routed back to machine running browser, using steps 13 and 14.
24. Operating system at browser end receives Ethernet frames containing response IP packet; reconstitutes packet.
25. Operating system peeks in IP packet, sees port of `12020`.
26. Operating system extracts content of IP packet (which happens to be an HTTP response), and writes it to connection at port 12020.
27. Browser reads HTTP response, displays web page.

And this takes perhaps a hundredth of a second for a local page, half a second to go across the globe.

Summary: networking on the internet

Protocol	Layer
HTTP	Application
TCP	Transport
IP	Internet
Ethernet	Data link
IEEE 802.3u	Physical

The network stack:

- Upper layers call lower layers for transport
- Applications, services only see relevant layer

Host	IP
www.umd.edu	128.8.237.77
apple.com	17.149.160.49
whitehouse.gov	23.1.44.110
unimelb.edu.au	128.250.148.40
bbc.co.uk	212.58.224.21

The domain-name system:

- Maps names to IP addresses
- Logical name independent of cyber-physical (IP address) location

Philosophy: the importance of open standards

- The internet relies crucially on common, and therefore open, standards
- Developed originally as a joint academic/defense/government project.
  - Very hard to believe it would have emerged from private industry (at least from large IT firms).
  - At time of public emergence, Microsoft thought future instead was in multimedia CD-ROMs.
- Standards-making process still surprisingly (to a cynic like me) open and informal.
  - Many standards developed through RFC process.
  - Emphasis on working systems rather than perfect idealizations.

Pragmatics: web services

- Web has accelerated move to client-server architecture:
  - Client machines runs interface software (e.g. web browser), on your desktop, laptop, or mobile.
  - Server runs backend service (e.g. web server) on centralized machine, typically in dedicated data center.
- Physical location of server often of only incidental practical interest.
- Recent interest in cloud computing:
  - Client data held on servers, not local machine
  - Web service providers do not maintain individual servers
  - Instead, buy resources (compute time, memory, disk space), dynamically and virtually provided by distributed server farms.

Pragmatics: setting up a web site

1. Go to domain registry, e.g. `http://gkg.net`, and buy domain name of your choice (if available!) for around \$10 per year.
2. Rent shared hosting space, e.g. from `http://www.siteground.com`, with database and scripting language, for around \$50 to \$100 per year.
   - You will get remote access to a shared server sitting in a data center in (say) Dallas.
3. Develop and upload your site – the expensive part.

Implications: Where is a web site?

A web site can be:

- Run by a company in Germany
- Hosted on a machine in London
- . . . or server farms across Europe
- Registered with a Canadian domain name register
- Using a US (`.com`) address

Where is the website? This matters particularly in the law: under whose jurisdiction is it?

Is the website where the domain is?

Emerging practice (at least for US govt) is to treat domain name as defining website location:

- PIPA, SOPA legislation defined `.com`, `.edu`, `.us` etc. sites as “in the US”, and sites with non-US country-specific TLD (`.uk`, `.se`) as “outside the US”.
- `megaupload.com` executives criminally prosecuted in US, though company registered in HK, executives (mostly) German, and living in NZ.
- The Pirate Bay changes its URL from `thepiratebay.org` to `thepiratebay.se` to avoid seizure by US government

But what about `bit.ly`?

Is the website where the data is?

- High-volume services (e.g. Google) replicate themselves around the world, shift data to where it is closest to current users.
- With Cloud Computing, even smaller sites can be automatically shunted between data centers internationally.
- Which country's (for instance) privacy laws apply to data that is distributed internationally?

Who is responsible for a web page?

There are various strategies that users can use to determine who is responsible for a web page:

- Look for explicit attributions of responsibility (e.g. “about us links”)
- Look up the directory path hierarchy (e.g. for `http://www.umiacs.umd.edu/~wew/teaching/690/spring12/`, look at `http://www.umiacs.umd.edu/~wew/teaching/690/`, `http://www.umiacs.umd.edu/~wew/teaching/`, `http://www.umiacs.umd.edu/~wew/`, etc.)

Who is responsible for a web page? (continued)

```
$ whois umd.edu
Domain Name: UMD.EDU

Registrant:
   University of Maryland
   Office of Information Technology
   College Park, MD 20742
   UNITED STATES

Administrative Contact:
   University of Maryland
   Office of Information Technology
   Bldg 224, Room 3309-C
   College Park, MD 20742-2411
   UNITED STATES
   (301) 405-3003
   dnsadmin@noc.umd.edu

Technical Contact:
   University of Maryland
   Office of Information Technology
   Network Operations Center
   College Park, MD 20742
   UNITED STATES
   (301) 405-3003
   dnsadmin@noc.umd.edu

Domain record activated:    31-Jul-1985
Domain record last updated: 28-Jun-2011
Domain expires:             31-Jul-2012
```

- See who the domain is named for, using `whois` or similar tool. You may need to prune the domain (for instance, `www.umiacs.umd.edu` to `umd.edu`)

Who is responsible for a web page? (continued)

- The top-level domain (TLD) may give you clues. For instance,
  - A `.edu` domain is only given to accredited tertiary US institutions.
  - A `.gov.uk` domain is only given to UK government entities

  But most extensions have few or no registration requirements:
  - At one stage, you had to have a registered Australian business name to get a `.com.au` domain name.
  - Now, though, anyone in this room (with a credit card) could buy a `.com.au` domain name in five minutes for $30.
- A `traceroute` request or IP geolocation service may tell you where the server is hosted.
  - But this generally means little about the organization behind it

APPENDIX

The following appendix summarizes the process of fetching a web page from a web server.

End-to-end: fetching a web page I

1. User types URL into browser, or clicks link, e.g. `http://www.umd.edu/directories/colleges.cfm`
2. Browser parses URL into protocol (`http:`), hostname (`www.umd.edu`), and path (`/directories/colleges.cfm`)
3. Browser opens TCP-IP connection to DNS server by IP address (at work, `128.8.76.2`) (details as per web page request below).
4. Browser creates DNS protocol request for IP address for hostname `www.umd.edu`, sends to DNS server.
5. DNS server `128.8.76.2` recursively asks authoritative DNS server for `umd.edu` domain, `128.8.5.2`.
6. DNS server responds to browser with IP address `128.8.237.77`, also using DNS protocol (and caches the answer).

End-to-end: fetching a web page II

7. Web browser requests operating system to open TCP connection to `128.8.237.77` on port 80.
8. Operating system opens high port, say 12020, on local machine, returns connection to it to web browser.
9. Web browser constructs HTTP request, starting:

   GET /directories/colleges.cfm HTTP/1.1
   Host: www.umd.edu

   writes it to connection.
10. Operating system wraps HTTP request in TCP packet giving to port of 12020, from port of 80, sequence number of 1, other details.
11. Wraps TCP packet in IP packet giving to IP address of `128.8.237.77`, from address of `128.8.107.183` (my laptop address).
12. TCP packet split up into 1500 byte chunks (if necessary), wrapped in Ethernet frames, sent to LAN gateway.

End-to-end: fetching a web page III

13. LAN gateway joins up Ethernet frames, strips header and footer, reconstitutes IP packet, looks at address, figures out next step on route.
14. Gateway wraps IP packet in networking protocol for step to next router.
15. Subsequent routers repeat steps 13 and 14, until IP packet arrives at server.
16. Operating system of server receives IP packet, peeks at TCP header to find port 80.
17. Operating system strips (but remembers) IP and TCP wrappings, extracts contents (which are an HTTP request, but OS doesn’t care), writes contents to web server’s end of connection.
18. Server splits off new, high port to manage this connection (say, 25121); will inform browser of new port number with return TCP packet.

End-to-end: fetching a web page IV

19. Web server reads HTTP request, sees that it is for path `/directories/colleges.cfm`, takes whatever steps necessary to construct content for this path (in this case, runs a script in a web development language called Cold Fusion).
20. Web server construct HTTP response, with header starting

    HTTP/1.1 200 OK
    Date: Mon, 06 Feb 2012 16:09:23 GMT

    and after header, the content of the web page.
21. Operating system of server wraps response in TCP packet, giving local high port of 25121, remote high port of 12020.
22. Wraps TCP packet in IP package, with from address of `128.8.237.77`, and to address of `128.8.76.2`.
23. IP packet routed back to machine running browser, using steps 13 and 14.

End-to-end: fetching a web page V

24. Operating system at browser end receives Ethernet frames containing response IP packet; reconstitutes packet.
25. Operating system peeks in IP packet, sees port of `12020`.
26. Operating system extracts content of IP packet (which happens to be an HTTP response), and writes it to connection at port 12020.
27. Browser reads HTTP response, displays web page.

And this takes perhaps a hundredth of a second for a local page, half a second to go across the globe.