A “Simple” Task

Send information from one computer to another

- hosts: endpoints of a network
- The plumbing is called a link.
Not Really So Simple…
Not Really So Simple…

AT&T

Quest

Sprint

Cogent

Swat

Google
Not Really So Simple…

AT&T

Quest

Cogent

Print

Swat

Google
Not Really So Simple…
We only need... ?
We only need... naming and addressing

Agreeing on how to describe/express a host, application, network, etc.
We only need... moving data to the destination

Routing: deciding how to get it there
We only need... moving data to the destination

Forwarding: copying data across devices/links
We only need... reliability and fault tolerance

how can we ...guarantee that the data arrives?
...handle link or device failures?
We only need... security and privacy
We only need... to manage complexity and scale up

Layering abstraction: divide responsibility
Protocols: standardize behavior for interoperability
We only need…

• Manage complexity and scale up
• Naming and addressing
• Moving data to the destination
• Reliability and fault tolerance
• Resource allocation, Security, Privacy..
We only need...

• Manage complexity and scale up
• Naming and addressing
• Moving data to the destination
• Reliability and fault tolerance
• Resource allocation, Security, Privacy..

(Lots of others too.)
Five-Layer Internet Model

Application: the application (e.g., the Web, Email)

Transport: end-to-end connections, reliability

Network: routing

Link (data-link): framing, error detection

Physical: 1’s and 0’s/bits across a medium (copper, the air, fiber)
Application Layer
(HTTP, FTP, SMTP, Skype)

- Does whatever an application does!
Transport Layer (TCP, UDP)

- Provides
  - Ordering
  - Error checking
  - Delivery guarantee
  - Congestion control
  - Flow control

- Or doesn’t!
Network Layer (IP)

• **Routers**: choose paths through network

Transport layer data + header becomes payload for the network layer
Link Layer (Ethernet, WiFi, Cable)

- Who’s turn is it to send right now?
- Break message into frames
- Media access: can it send the frame now?

- Send frame, handle “collisions”

Network layer data + header becomes payload for the link layer
Physical layer – move actual bits! (Cat 5, Coax, Air, Fiber Optics)

- 2.4GHz Radio
- DS/FH Radio (1-11Mbps)
- 802.11b Wireless Access Point
- Cat5 Cable (4 wires)
- 100Base TX Ethernet 100Mbps
- Ethernet switch/router
- To campus backbone
- 62.5/125um 850nm MMF 1000BaseSX Ethernet 1000Mbps
Layering and encapsulation

- **Layer**
- **Application**
- **Transport: reliability**
- **Network: routing**
- **Link: framing, error detection**
- **Physical**
Layering: Separation of Functions

• explicit structure allows identification, relationship of complex system’s pieces
  – layered reference model for discussion
  – reusable component design
• modularization eases maintenance
  – change of implementation of layer’s service transparent to rest of system,
  – e.g., change in postal route doesn’t effect delivery of letter
Abstraction!

• Hides the complex details of a process

• Use abstract representation of relevant properties make reasoning simpler

• Ex: Your knowledge of postal system:
  – Letters with addresses go in, come out other side
Five-Layer Internet Model

- Application: the application (e.g., the Web, Email)
- Transport: end-to-end connections, reliability
- Network: routing
- Link (data-link): framing, error detection
- Physical: 1’s and 0’s/bits across a medium (copper, the air, fiber)
OSI Seven-Layer Model

Application: the application (e.g., the Web, Email)

Presentation: formatting, encoding, encryption

Session: sockets, remote procedure call

Transport: end-to-end connections, reliability

Network: routing

Link (data-link): framing, error detection

Physical: 1’s and 0’s/bits across a medium (copper, the air, fiber)
OSI Seven-Layer Model

Application: the application (e.g., the Web, Email)

Presentation: formatting, encoding, encryption

Session: sockets, remote procedure call

Transport: end-to-end connections, reliability

Network: routing

Link (data-link): framing, error detection

Physical: 1’s and 0’s/bits across a medium (copper, the air, fiber)
Five-Layer Internet Model

Application: the application (e.g., the Web, Email)

Transport: end-to-end connections, reliability

Network: routing

Link (data-link): framing, error detection

Physical: 1’s and 0’s/bits across a medium (copper, the air, fiber)
Internet Protocol Suite

The Hourglass Model

Applications
Transport
Data Link
Physical

“Thin Waist”

NET_1
NET_2
...
NET_n

IP
TCP
UDP
FTP
HTTP
VoIP
SSH
Putting this all together

• **ROUGHLY**, what happens when I click on a Web page from Swarthmore?

My computer  

?  

Internet  

www.google.com
Application Layer: Web request (HTTP)

- Turn click into HTTP request

GET http://www.google.com/ HTTP/1.1
Host: www.google.com
...
Application Layer: Name resolution (DNS)

- Where is www.google.com?

My computer
(132.239.9.64)

What's the address for www.google.com

Local DNS server
(132.239.51.18)

Oh, you can find it at 66.102.7.104
Transport Layer: TCP

- Break message into packets (TCP segments)
- Should be delivered reliably & in-order

GET http://www.google.com HTTP/1.1
Host: www.google.com
...
Network Layer: Global Network Addressing

- Address each packet so it can traverse network and arrive at host

My computer (132.239.9.64)

www.google.com (66.102.7.104)
Network Layer: (IP) At Each Router

• Where do I send this to get it closer to Google?

• Which is the best route to take?
Link & Physical Layers (Ethernet)

- Forward to the next node!
- Share the physical medium.
- Detect errors.
Message Encapsulation

- Higher layer within lower layer
- Each layer has different concerns, provides abstract services to those above
TCP/IP Protocol Stack

- Application Layer
  - HTTP

- Transport Layer
  - TCP

- Network Layer
  - IP
  - Ethernet interface
  - SONET interface

- Link Layer
  - Ethernet interface
  - SONET interface

Slide 38
Five-Layer Internet Model

- **Application**: the application (e.g., the Web, Email)
- **Transport**: end-to-end connections, reliability
- **Network**: routing
- **Link (data-link)**: framing, error detection
- **Physical**: 1’s and 0’s/bits across a medium (copper, the air, fiber)
What is a protocol?

Protocol: message format + transfer procedure

Human Protocol

Network Protocols (defined in RFCs)
What is a protocol?

Goal: get message from sender to receiver

Protocol: message format + transfer procedure

• Expectations of operation
  – first you do x, then I do y, then you do z, ...

• Multiparty! so no central control
  – sender and receiver are separate processes
A “Simple” analogous task: Post-it Note

Write a protocol to write a note / post—it to your housemate

Protocol: message format + transfer procedure

• Message format: (from, to), message contents
• Transfer procedure: post on refrigerator
Message = Header + Data

Header: information to support protocol
- Source and destination addresses
- State of protocol operation
- Error control (to check integrity of received data)

Data (a.k.a Payload or Body) usually very small
A “Simple” analogous task: Postal Mail

Alice

Message

Mila

To: 575 Albatross Street, Seattle, WA
From: Chicago, IL

The post office does NOT care about what’s in here, and shouldn’t be looking at it...
A “Simple” analogous task: Postal Mail

- **Mail Sending Protocol**
  - Message format: (from, to), message contents
  - Transfer procedure: post mail in mailbox (agreed upon convention)

The post office does NOT care about what’s in here, and shouldn’t be looking at it...

**To:** 575 Albatross Street, Seattle, WA  
**From:** Chicago, IL
A “Simple” analogous task: Postal Mail: other protocols in use?

**Mail Protocol**
- Message format: (from, to), message contents
- Transfer procedure: post mail in mailbox (agreed upon convention)

**Card Protocol (within the mail protocol!)**
- Message format: (from, to), message contents
Message Encapsulation

- Higher layer within lower layer
- Each layer has different concerns, provides abstract services to those above
Layering and encapsulation

Layer
Application
Transport: reliability
Network: routing
Link: framing, error detection
Physical
Layering: Separation of Functions

• explicit structure allows identification, relationship of complex system’s pieces
  – layered reference model for discussion
  – reusable component design
• modularization eases maintenance
  – change of implementation of layer’s service transparent to rest of system,
  – e.g., change in postal route doesn’t effect delivery of letter
Abstraction!

• Hides the complex details of a process

• Use abstract representation of relevant properties make reasoning simpler

• Ex: Alice and Mila’s knowledge of postal system:
  – Letters with addresses go in, come out other side
TCP/IP Protocol Stack

Application Layer

Transport Layer

Network Layer

Link Layer
Creating a network app

write programs that:
• run on (different) end systems
• communicate over network
• e.g., web server software communicates with browser software

no need to write software for network-core devices
• network-core devices do not run user applications
• applications on end systems allows for rapid app development, propagation

Adapted from Kurose and Ross
What IS A Web Browser?
HTTP and the Web

First, a review...

• web page consists of objects
• object can be HTML file, JPEG image, Java applet, audio file,...
• web page consists of base HTML-file which includes several referenced objects
• each object is addressable by a URL, e.g.,

  www.someschool.edu/someDept/pic.gif

  ^ host name                      ^ path name

Adapted from: Kurose & Ross
HTTP: Hypertext transfer protocol

- client/server model
  - **client**: browser that requests, receives, (using HTTP protocol) and “displays” Web objects
  - **server**: Web server sends (using HTTP protocol) objects in response to requests

Adapted from: Kurose and Ross
HTTP Overview

1. User types in a URL.
   http://some.host.name.tld/directory/name/file.ext

   host name

   path name
HTTP Overview

2. Browser establishes connection with server.
   Looks up “some.host.name.tld”
   Calls connect()
3. Browser requests the corresponding data.
   GET /directory/name/file.ext HTTP/1.0
   Host: some.host.name.tld
   [other optional fields, for example:]
   User-agent: Mozilla/5.0 (Windows NT 6.1; WOW64)
   Accept-language: en
HTTP Overview

4. Server responds with the requested data.
   HTTP/1.0 200 OK
   Content-Type: text/html
   Content-Length: 1299
   Date: Sun, 01 Sep 2013 21:26:38 GMT
   [Blank line]
   (Data data data data data...)
HTTP Overview

5. Browser renders the response, fetches any additional objects, and closes the connection.
HTTP Overview

1. User types in a URL.
2. Browser establishes connection with server.
3. Browser requests the corresponding data.
4. Server responds with the requested data.
5. Browser renders the response, fetches other objects, and closes the connection.

It’s a document retrieval system, where documents point to (link to) each other, forming a “web”.
HTTP Overview (Lab 1)

1. User types in a URL.
2. Browser establishes connection with server.
3. Browser requests the corresponding data.
4. Server responds with the requested data.
5. Browser renders the response, fetches other objects, and closes the connection.

It’s a document retrieval system, where documents point to (link to) each other, forming a “web”.
Trying out HTTP (client side) for yourself

1. Telnet to your favorite Web server:

```
telnet demo.cs.swarthmore.edu 80
```
Opens TCP connection to port 80 (default HTTP server port) at example server. Anything typed is sent to server on port 80 at demo.cs.swarthmore.edu

2. Type in a GET HTTP request:

```
GET / HTTP/1.1
Host: demo.cs.swarthmore.edu
```
(Hit carriage return twice) This is a minimal, but complete, GET request to the HTTP server.

3. Look at response message sent by HTTP server!
Example

$ telnet demo.cs.swarthmore.edu 80
Trying 130.58.68.26...
Connected to demo.cs.swarthmore.edu.
Escape character is '^]'.
GET / HTTP/1.1
Host: demo.cs.swarthmore.edu

HTTP/1.1 200 OK
Vary: Accept-Encoding
Content-Type: text/html
Accept-Ranges: bytes
ETag: "316912886"
Last-Modified: Wed, 04 Jan 2017 17:47:31 GMT
Content-Length: 1062
Date: Wed, 05 Sep 2018 17:27:34 GMT
Server: lighttpd/1.4.35
Example

$ telnet demo.cs.swarthmore.edu 80
Trying 130.58.68.26...
Connected to demo.cs.swarthmore.edu.
Escape character is '^[].'
GET / HTTP/1.1
Host: demo.cs.swarthmore.edu

Response
headers

<html><head><title>Demo Server</title></head>
<body>
.....
</body>
</html>

Response
body
(This is what
you should be
saving in lab 1.)
HTTP request message

- two types of HTTP messages: request, response
- **HTTP request message**: ASCII (human-readable format)

```
GET /index.html HTTP/1.1
Host: web.cs.swarthmore.edu
User-Agent: Firefox/3.6.10
Accept: text/html,application/xhtml+xml
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7
Keep-Alive: 115
Connection: keep-alive
```