SIMPLE WEB REQUEST
Simple Web Request

Thanks to Karen Reid for material in these slides
The request

- How do we tell the server what we want?
- How do we even find the server?
- How do the server and browser talk to each other?
HTTP Request

request

GET / HTTP/1.1
Connection: Keep-Alive
Host: www.tkf.toronto.on.ca

reply

HTTP/1.1 301 Moved Permanently
Date: Tues, 23 July 2002
Server: Apache/1.3.22(unix)
Content-Type: text/html
How do we find the server?

- Every computer on the Internet has an Internet address.
- Called an IP address (Internet Protocol)
- An IP address is 4 numbers separated by dots.

www.tkf.toronto.on.ca = 207.245.2.3
Domain Name Servers

- **browser**
  - www.tkf.toronto.on.ca?
  - www.tkf.toronto.on.ca?

- **local name server**
  - 207.245.2.3
  - www.tkf.toronto.on.ca?
  - ca name server
  - on name server

- **root name server**

- **“ca” name server**
  - “ca” name server
  - toronto name server

- **“on” name server**
  - www.tkf.toronto.on.ca?
  - on server

- **“toronto” name server**
  - www.tkf.toronto.on.ca?
  - toronto server

- **207.245.2.3**
This is getting complicated!

Number of messages?
10-12
Actually, it’s worse than that

- The web page for www.tkf.toronto.on.ca doesn’t really live at 207.245.2.3

  request
  
  HTTP/1.1 301 Moved Permanently
  Date: Tues, 23 July 2002
  Server: Apache/1.3.22(unix)
  Location: http://www.kites.org
  Content-Type: text/html

  reply

- So we need to go look up www.kites.org
Now what?

- Okay, we have the address.
- What do we do with it?
- Let’s look at how two computers communicate.
- HTTP is a high-level protocol
- HTTP is specific to the web.
- Computers communicate for other reasons.
Computers use several layers of general protocols to communicate.

To understand why these layers are important, think about how a company sends you an invoice for a purchase.
## Protocols

### Invoice:

**Customer:** Karen Reid  
**Order No:** 5379

<table>
<thead>
<tr>
<th>Qty</th>
<th>Item</th>
<th>Unit Price</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Athalon</td>
<td>219.00</td>
<td>219.00</td>
</tr>
<tr>
<td>2</td>
<td>128 MB</td>
<td>149.95</td>
<td>299.90</td>
</tr>
</tbody>
</table>

**Subtotal:** 518.90  
**Tax:** 77.84  
**TOTAL:** 596.74

**Payable to:** CPUS are us  
**Amount:** $596.74  
**Address:**

Karen Reid  
CPUS are us  
0 College Street  
Toronto Ontario M5S 3G4

---

*We deliver!*

---

*Courier*
### TCP/IP: Protocol Framework

<table>
<thead>
<tr>
<th>TCP/IP Layers</th>
<th>TCP/IP Protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Layer</td>
<td>HTTP</td>
</tr>
<tr>
<td>Transport Layer</td>
<td>TCP</td>
</tr>
<tr>
<td>Network Layer</td>
<td>IP</td>
</tr>
<tr>
<td>Network Interface Layer</td>
<td>Ethernet</td>
</tr>
</tbody>
</table>
TCP/IP: Sending/Receiving Data

Application Layer
Telnet, HTTP, FTP, SMTP

Transport Layer
TCP, UDP

Network Layer
IP

Physical Layer
Ethernet, X.25, Token Ring

Sending Data
Receiving Data
TCP/IP

- Transmission Control Protocol.
- Tells us how to package up the data.

<table>
<thead>
<tr>
<th>source address</th>
<th>dest. address</th>
</tr>
</thead>
<tbody>
<tr>
<td>bytes</td>
<td>ack</td>
</tr>
<tr>
<td></td>
<td>port</td>
</tr>
<tr>
<td>data</td>
<td></td>
</tr>
</tbody>
</table>

Example analogy:
You can think that HTTP is what goes in the envelop, TCP is the envelop, and IP is the truck.
TCP Connection

3-way handshake
SYN

Hi 24.197.0.67
Connection port 80?

Hi 128.100.27.9
Let’s talk

okay

Send me a file

Here’s some data

Got it

Here’s some more

Got it

Got it

I’m done

I’m done too

ack

ack

ack

ack

fin

fin
TCP: Three-way handshake

sequence number = $J$

socket
connect (blocks)
connect returns

client

server sequence number = $K$

socket, bind, listen
accept (blocks)

SYN $J$

SYN $K$, ack $J+1$

ack $K+1$

accept returns
Each TCP packet is given a header
- sequence number
- checksum

make packets

put in an IP envelope with another header

To 24.197.0.67
When something goes wrong

- A packet might not arrive
  - traffic overload
  - bit corruption
- Receiver asks for missing packets to be resent. Want to send data as fast as possible.
- Strategies:
  - Send packets as fast as possible (too many lost)
  - Send packets at a certain rate (can go faster)
  - Wait for the ack (too slow)
  - Window-based (adaptive)
TCP Congestion Control

- **Window-based:**
  - some number of packets allowed to be sent and not ack’d
  - as successful ack’s arrive, grow window
  - if packet loss is detected, cut window size
TCP Congestion Control

The diagram illustrates the relationship between time and window size, with packet losses indicated by the red circles. The graph shows how the window size increases and decreases in response to packet losses over time.
- **Client-Server model**: a client process wants to talk to a server process
- Client must find server - **DNS lookup**
- Client must find service on server - **ports**
- Finally **establish a connection** so they can talk
Routing

mcl-gpb.gw.utoronto.ca

sf-cs27.gw.utoronto.ca

browser

gate-gateway.gw.utoronto.ca

fe6-3.gwy2-tor.bb.attcanada.ca

srp2-0.core2-tor.bb.attcanada.ca

pos8-0-0.hcap2-tor.bb.attcanada.ca

atm1-0-101.adsl1-tor.bb.attcanada.ca

206.222.73.174

border.onet.on.ca

pos10-1.peer-toroon.ip.grouptelecom.net

www.tkf.toronto.on.ca
Plus the redirect (20 hops)
At least 5 different cities

- my office
- CSLab
- U of T
- Scarborough, ON
- Toronto, ON
- Chicago
- Washington, DC
- St. Louis, Missouri
- Atlanta, Georgia
How many messages?

- It depends on the size of the web page we retrieve.
- If the web page is 100 Kbytes (small!) it will be broken up into ~80 IP packets.

\[
10 \text{ (DNS)} + \\
22 \text{ (Connect with toronto.on.ca)} + \\
6 \text{ (DNS)} + \\
+ 80 \times 20 \text{ hops} = 1638 \text{ messages!}
\]
Types of Connection (TCP/UDP)

- **Connection oriented model**
  - Like phone calls
  - Uses Transmission Control Protocol (TCP)
  - Defined ordering of messages and acks

- **Connectionless model**
  - Like sending letters via postal service
  - Uses User Datagram Protocol (UDP)
  - More efficient and good for sending broadcasts to many machines