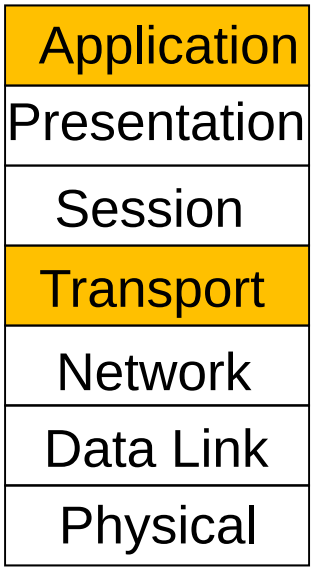
CSE561 – Application Transports

David Wetherall
djw@cs.washington.edu

Application use of the network

- Focus:
  - What transports do real applications need?

- Transports, Applications
- HTTP as example



Clark Discussion

- What are the contributions of the paper?
  - Which is/was the more lasting?
- Why is predicted service useful vs. guaranteed service?
  - What apps want which one?
- What is the overall architecture?
  - What is novel about it?
- How does admission control work?
  - What is the service interface?

Transports we have

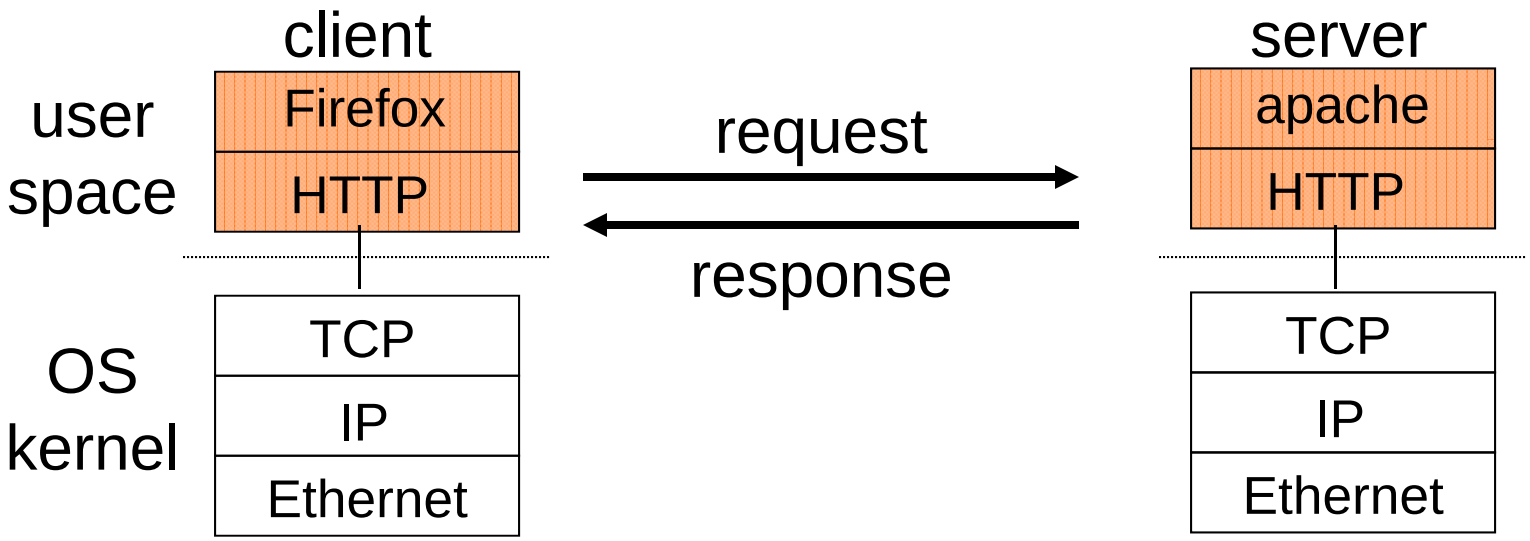
- TCP
  - Reliable, congestion controlled bytestream
- UDP
  - Unreliable individual short messages
  - Error detection if you are nice
  - (Packets!)

Example applications and their needs

- Video conferencing
  - Unreliable video stream (congestion friendly)
- Video-on-demand (streaming media)
  - Reliable bytestream with buffered playback (congestion control)
- DNS
  - Request / reply
  - Reliable, short messages
- Web
  - Series of related request / replies
  - Reliable, variable length messages (congestion control)

- Not exactly a great match to what we have …

Web Protocol Stacks



- To view the URL http://server/page.html the client makes a TCP connection to port 80 of the server, by it's IP address, sends the HTTP request, receives the HTML for page.html as the response, repeats the process for inline images, and displays it.

HTTP Request/Response

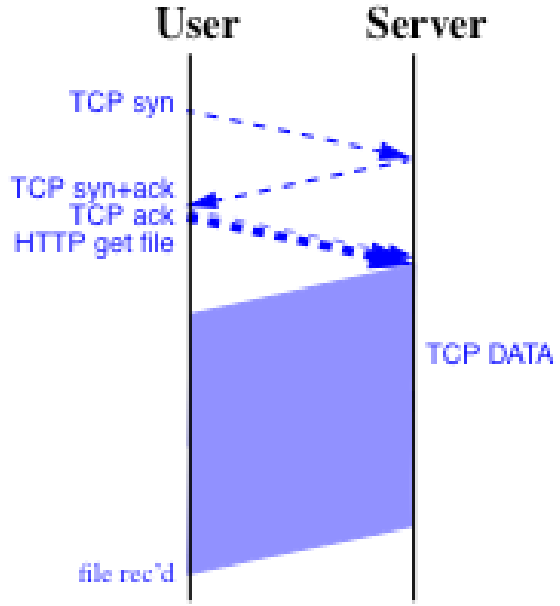
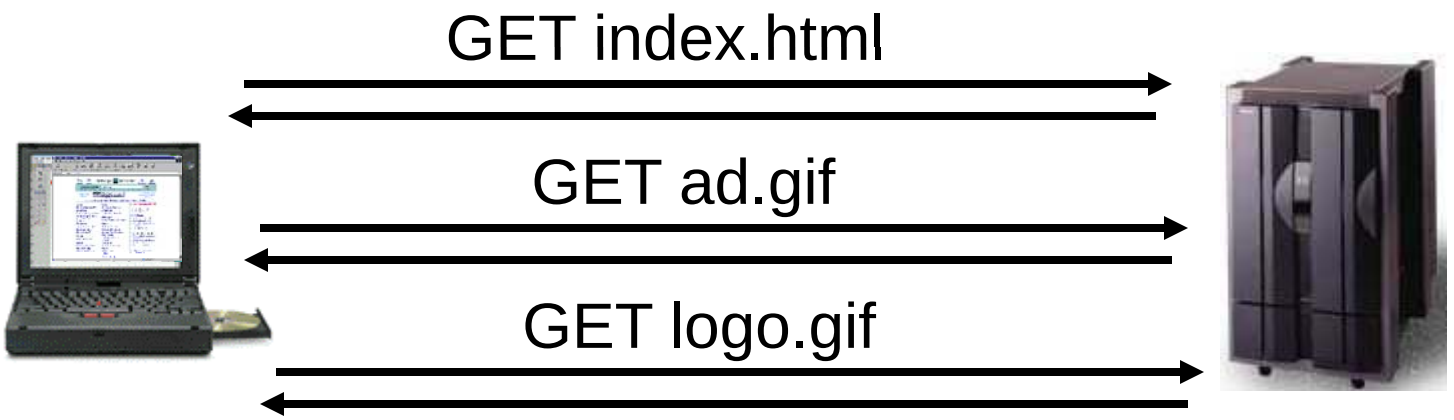


FIGURE 3. HTTP File Transfer

1 RTT	channel OPEN
0.5 RTT	send request
0.5 RTT	file starts to arrive
Ftrans	time to transmit the file

2 RTT + Ftrans
= time to get a file in HTTP

Simple HTTP 1.0



- HTTP is a tiny, text-based language
- The GET method requests an object
- There are HTTP headers, like “Content-Length:”, etc.
- Try “telnet server 80” then “GET index.html HTTP/1.0”
  - Other methods: POST, HEAD,… google for details

HTTP Request/Response in Action

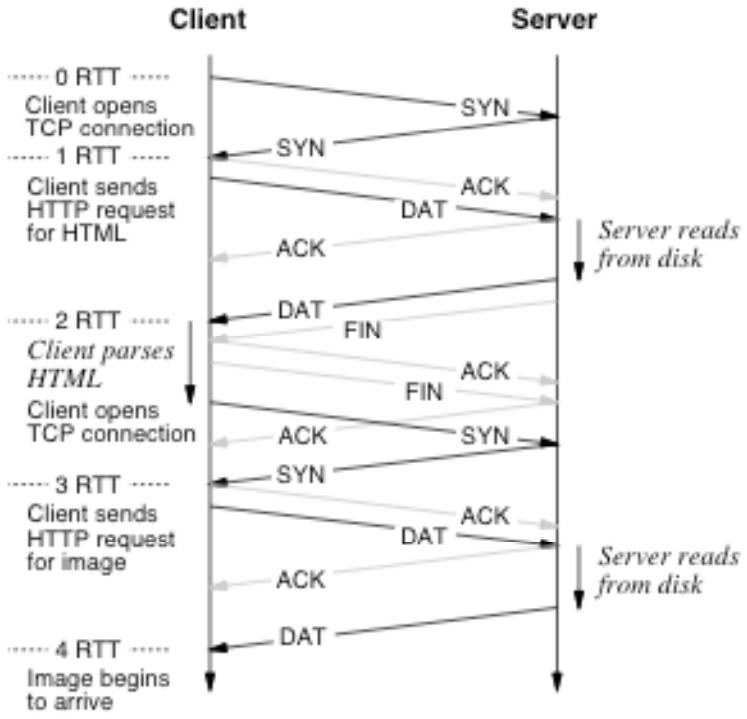


Figure 3-1: Packet exchanges and round-trip times for HTTP

- Problem is that:
  - Web pages are made up of many files. Most are very small (< 10k)
  - files are mapped to connections

  For each file
  - Setup/Teardown
    - Time-Wait table bloat
  - 2RTT "first byte" latency
  - Slow Start+ AIMD Congestion Avoidance

The goals of HTTP and TCP protocols are not aligned!

TCP Behavior for Short Connections

RTT=70ms

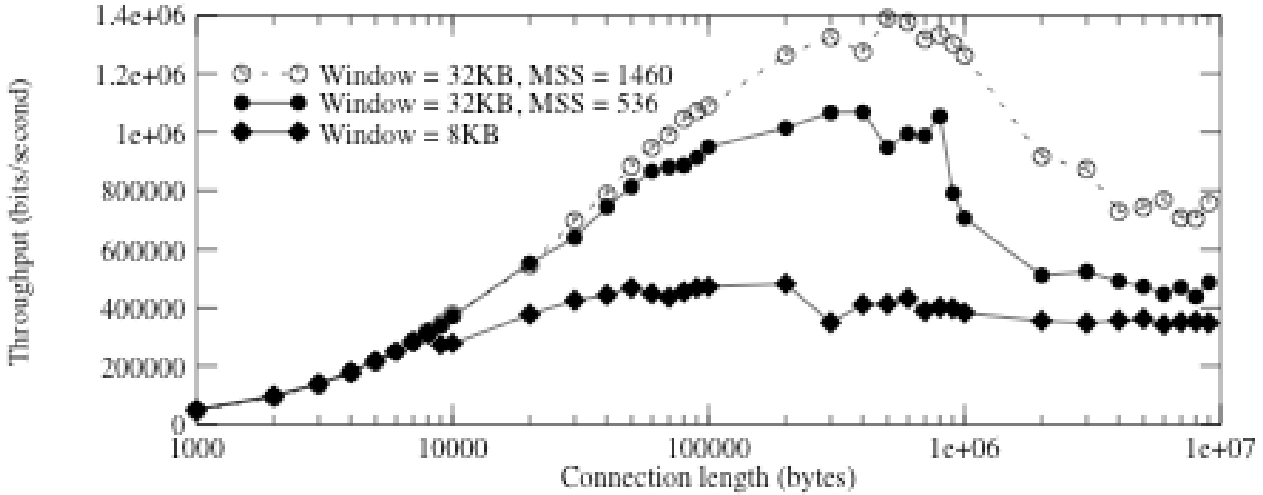


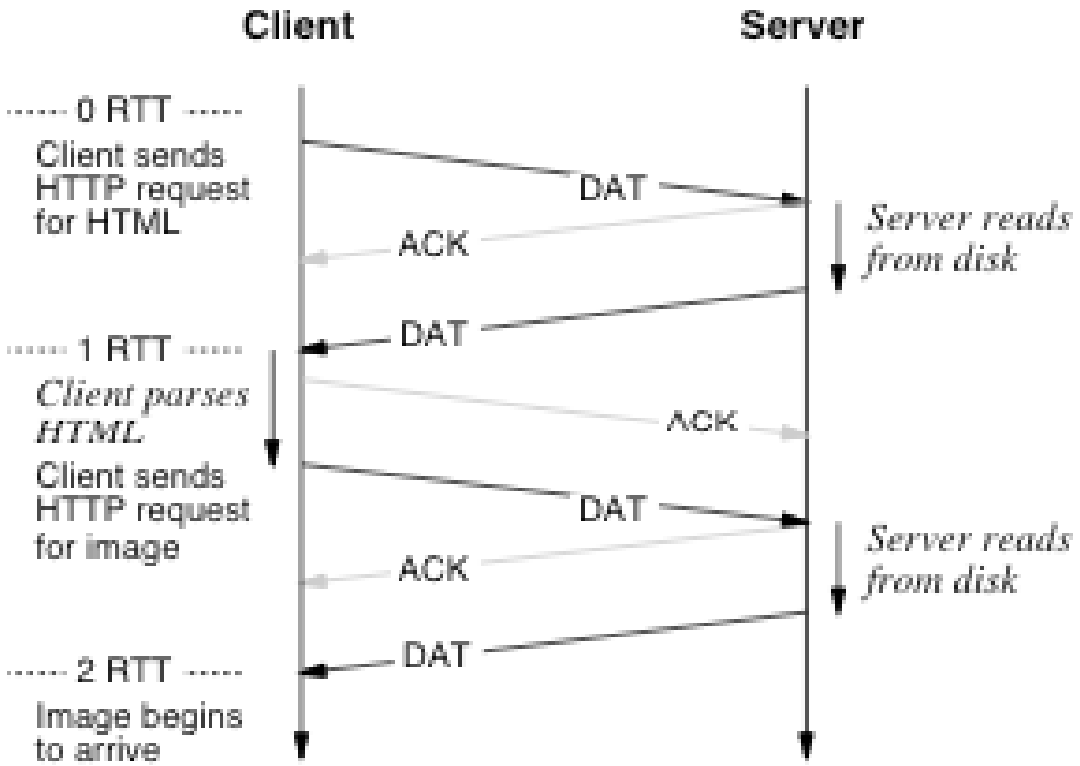
Figure 3-2: Throughput vs. connection length, RTT = 70 msec

Figure 3-2 shows that, in the remote case, using a TCP connection to transfer only 2 Kbytes results in a throughput less than 10% of best-case value. Even a 20 Kbyte transfer achieves only about 50% of the throughput available with a reasonable window size. This reduced throughput translates into increased latency for document retrieval. The figure also shows that, for this 70 msec RTT, use of too small a window size limits the throughput no matter how many bytes are transferred.

HTTP1.1: Persistent Connections

- Idea: Use one TCP connection for multiple page downloads (or just HTTP methods)
- Q: What are the advantages?
- Q: What are the disadvantages?
  - *Application layer multiplexing*

HTTP/1.1



- Also pipelining: send multiple request before responses done

Effect of Persistent HTTP

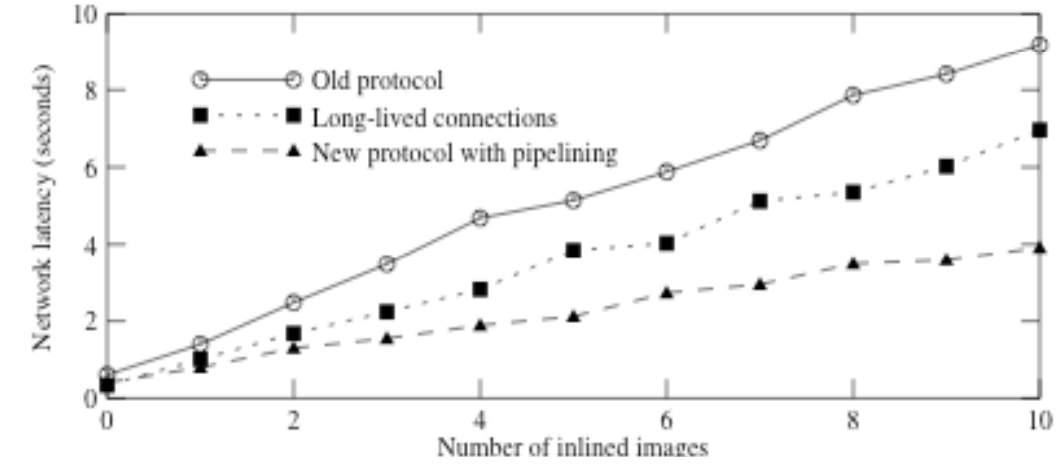


Image size=2544

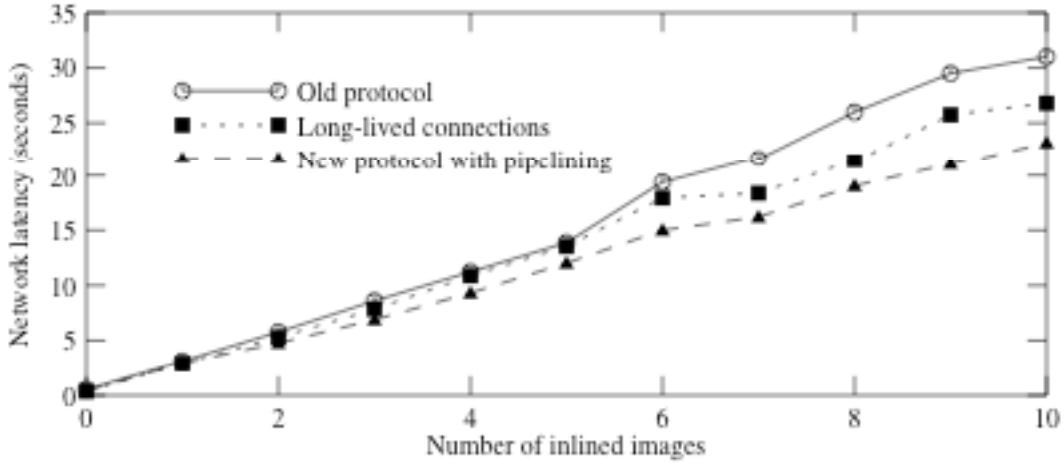


Image size=45566

Figure 6-2: Latencies for a remote server, image size = 45566 bytes