What now? What about performance? Goals depend on who you're talking about

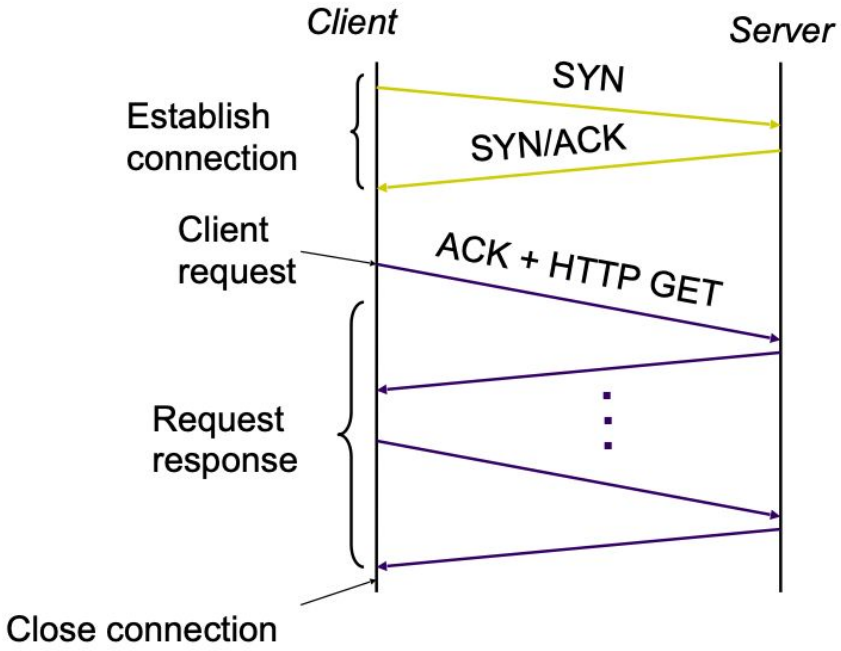
	User	Network operators	Content provider
wish	fast downloads high availability	no overload	happy users cost-effective infrastructure
solution	Improve HTTP to compensate for TCP weakspots	Caching and Replication	

What now? What about performance? Goals depend on who you're talking about

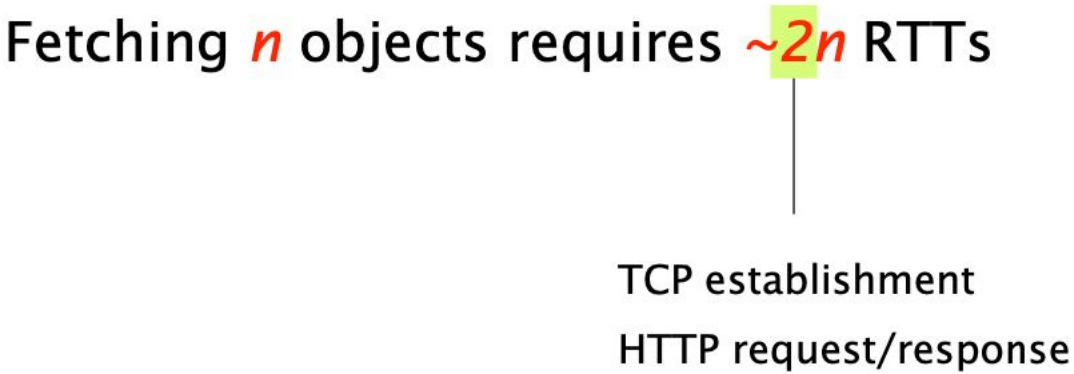
User

wish	fast downloads
	high availability
solution	Improve HTTP to compensate for TCP weakspots

Recall TCP forces a client to open a connection before exchanging any data

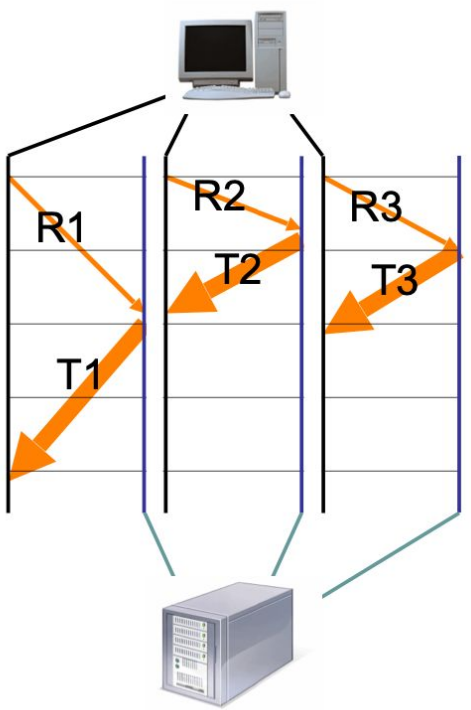


Nearly all websites have multiple objects, naive HTTP opens one TCP connection for each...



One solution to that problem is to use multiple TCP connections in parallel

User	Happy!
Content provider	Happy!
Network operator	Not Happy!
	Why?



Another solution is to use persistent connections across multiple requests (the default in HTTP/1.1)

Avoid overhead of connection set-up and teardown

clients or servers can tear down the connection

Allow TCP to learn more accurate RTT estimate

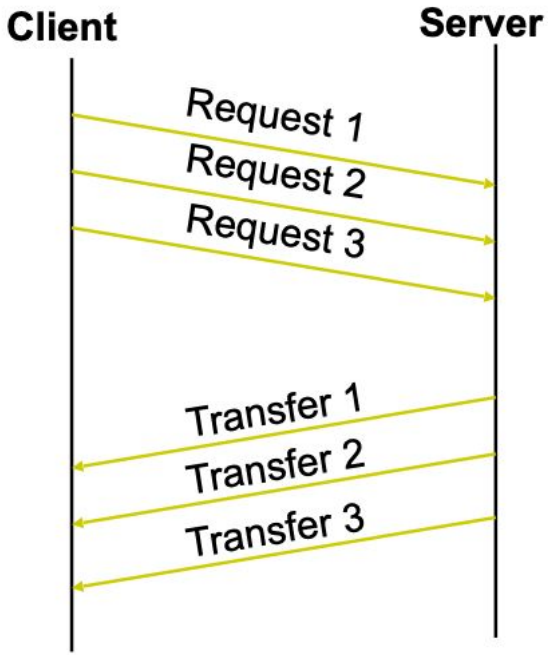
and with it, more precise timeout value

Allow TCP congestion window to increase

and therefore to leverage higher bandwidth

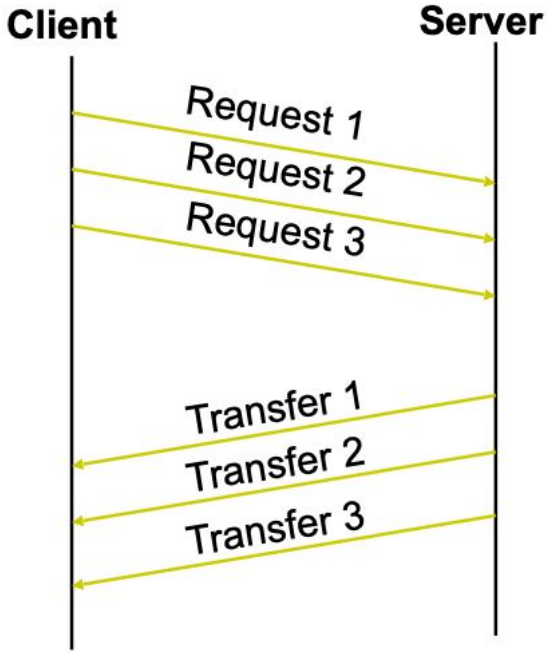
Yet another solution is to pipeline requests & replies asynchronously, on one connection

- batch requests and responses to reduce the number of packets
- multiple requests can be packed into one TCP segment

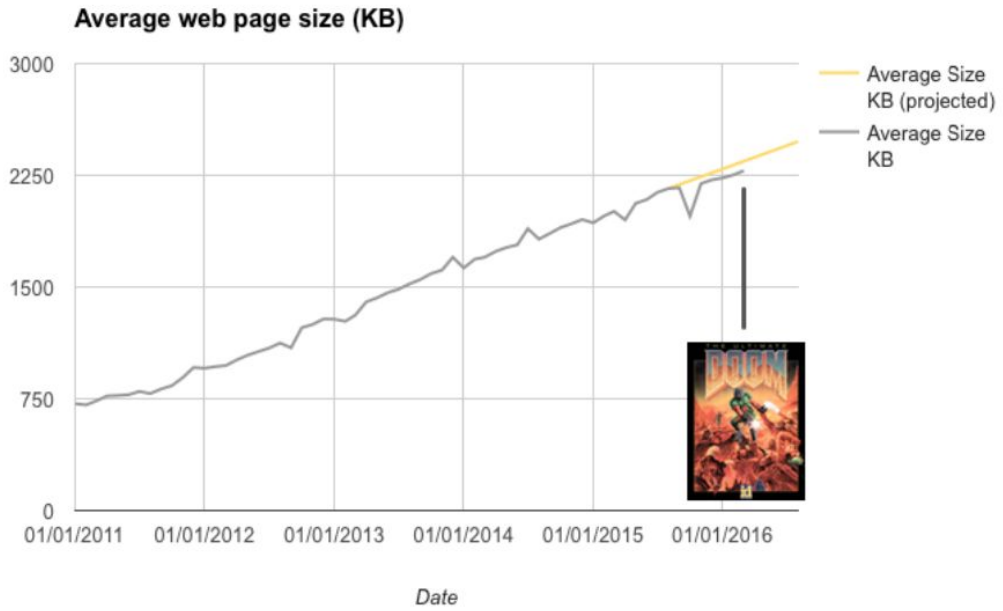


Yet another solution is to pipeline requests & replies asynchronously, on one connection

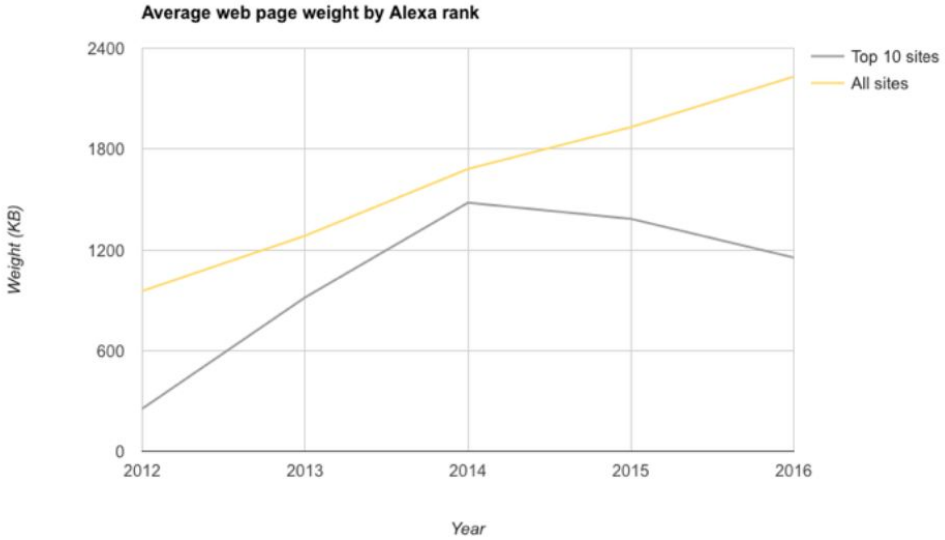
- Pipelined connections aren't actually used
- But they seemed like a huge win
- What happened?!
  - .. primarily two reasons
- Reason 1: Bugs
  - One manifestation: images on page are swapped!
  - Often blamed on proxy servers
  - My guess: bad adaptation of multithreaded non-pipelined version
- Reason 2: Head-of-line blocking



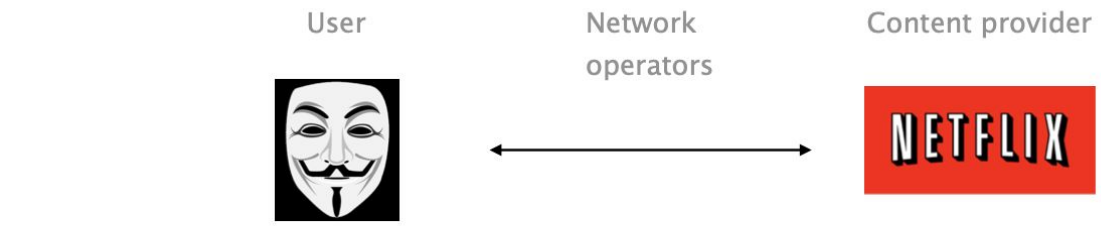
The average webpage size nowadays is as large as the original DOOM...



Top web sites have decreased in size though because they care about TCP performance



What now? What about performance? Goals depend on who you're talking about



	User	Network operators	Content provider
wish	fast downloads high availability	no overload	happy users cost-effective infrastructure

solution

Caching and Replication

Caching leverages the fact that highly popular content largely overlaps

Just think of how many times
you request the  logo
per day

vs

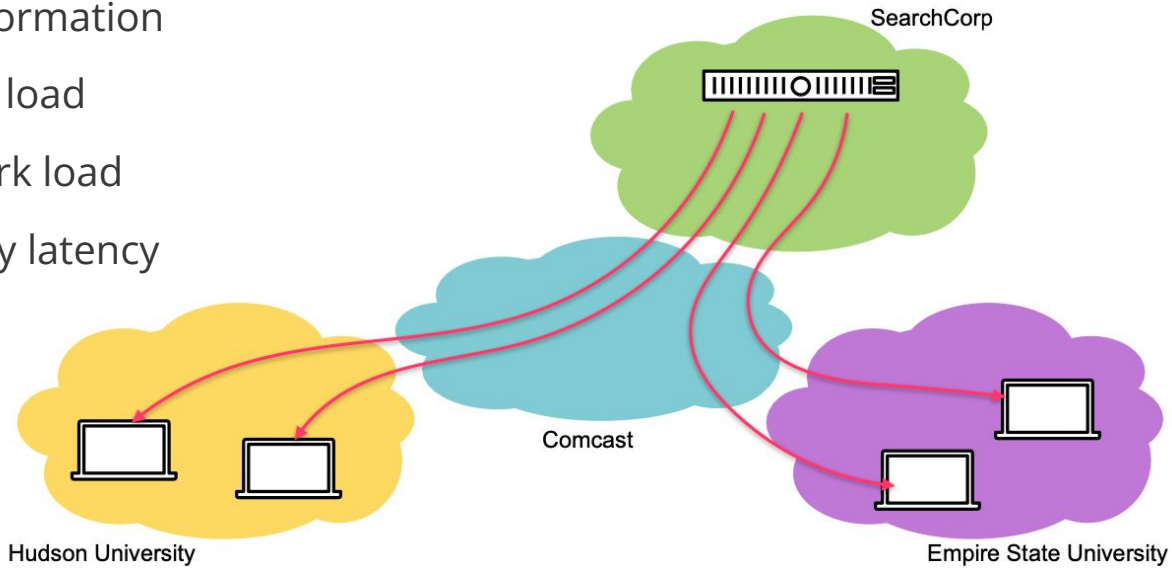
how often it *actually* changes

Caching it saves time for your browser
and decrease network and server load

HTTP Caching

No caching

- Many clients transfer same information
- Generates unnecessary server load
- Generates unnecessary network load
- Clients experience unnecessary latency



Yet, a significant portion of the HTTP objects are “uncachable"

Examples	dynamic data	stock prices, scores, ...
	scripts	results based on parameters
	cookies	results may be based on passed data
	SSL	cannot cache encrypted data
	advertising	wants to measure # of hits ($$$)

To limit staleness of cached objects, HTTP enables a client to validate cached objects

Server hints when an object expires (kind of TTL) as well as the last modified date of an object

Client conditionally requests a resource using the “if-modified-since” header in the HTTP request

Server compares this against “last modified” time of the resource and returns:

- **Not Modified** if the resource has not changed
- **OK** with the latest version

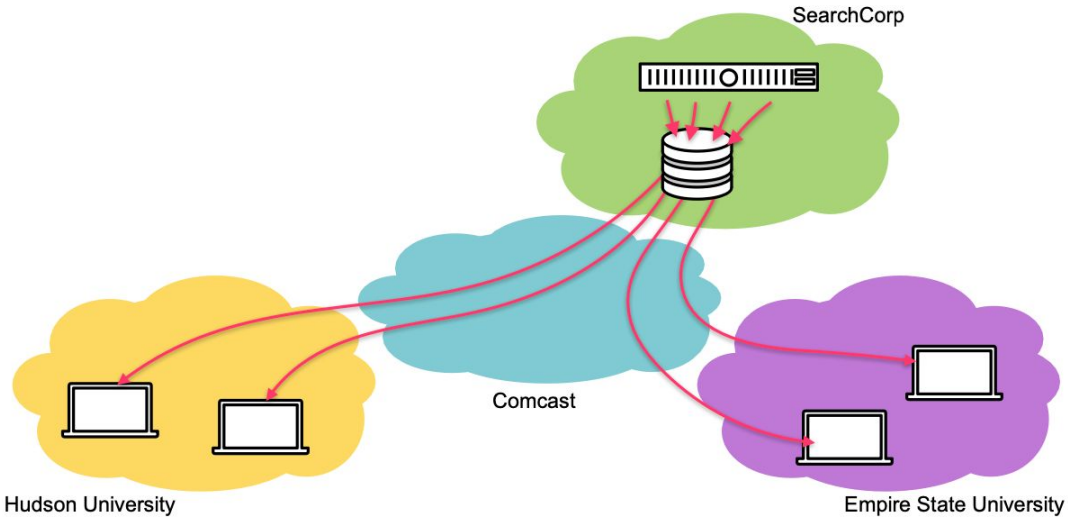
Caching can be (and is) performed at different locations

client	browser cache
close to the client	forward proxy Content Distribution Network (CDN)
close to the destination	reverse proxy

HTTP Caching

Reverse proxies

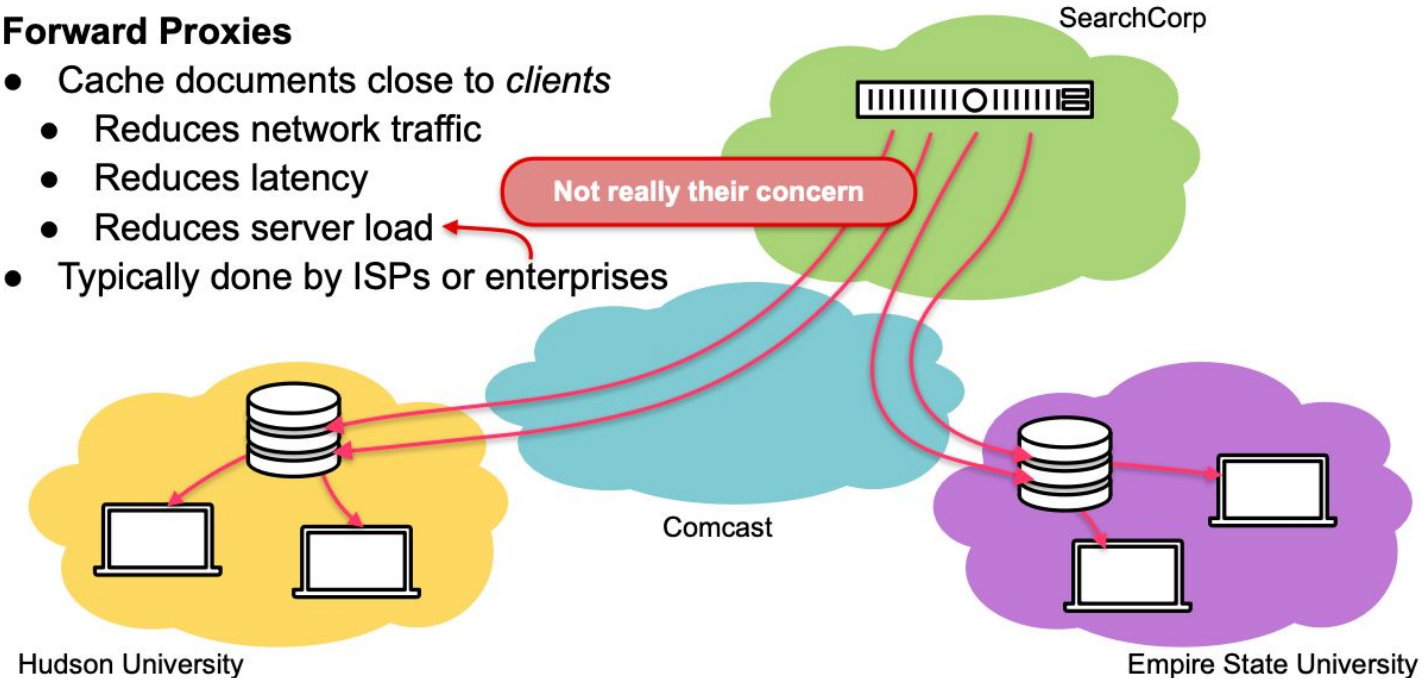
- Cache documents close to servers
    - Reduces server load
- Typically done by content provider



HTTP Caching

Forward Proxies

- Cache documents close to *clients*
  - Reduces network traffic
  - Reduces latency
  - Reduces server load
- Typically done by ISPs or enterprises



Content Delivery Networks

- Replication is a huge benefit to availability, scalability, and performance
  - We saw this with DNS
  - Can spread the load
  - Places content closer to clients (less latency)

  - Caching is a form of opportunistic replication
    - .. but what if a given organization doesn't have a forward proxy?
    - .. what if content provider and wants its content always replicated?

    - Idea: Caching and replication as a service — "CDNs 1.0"

CDNs “1.0”

- Large-scale distributed storage infrastructure
  - (Usually) administered by one entity
  - e.g., Akamai has 275,000+ servers in 136 countries
- Any server can host content for the many clients of the CDN (virtual hosting)
- How does content provider get its data onto Akamai’s servers?
- Two major ways
  - Pull
  - Push
  - .. we’ll come back to these in a moment