CS 4/55231
Internet Engineering

Kent State University
Dept. of Computer Science

LECT-10

Today's Topic

Advanced Topics in HTTP

More Definitions

- Connection
  - A transport layer virtual circuit established between two programs for the purpose of communication.

More Definitions

- Client
  - A program that establishes connections for the purpose of sending requests.
- User agent
  - The client which initiates a request. These are often browsers, editors, spiders (web-traversing robots), or other end user tools.

More Definitions

- Server
  - An application program that accepts connections in order to service requests by sending back responses. Any server may act as an origin server, proxy, gateway, or tunnel, switching behavior based on the nature of each request.
- Origin server
  - The server on which a given resource resides or is to be created.

More Definitions

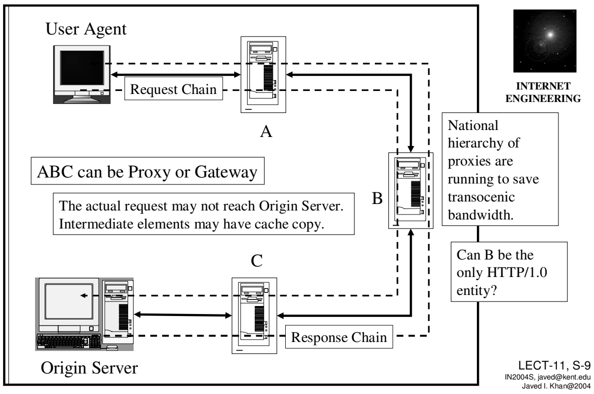
- Proxy
  - An intermediary program which acts as both a server and a client for the purpose of making requests on behalf of other clients. Requests are serviced internally or by passing them on, with possible translation, to other servers. A proxy must implement both the client and server requirements of this specification.

More Definitions

- <u>Gateway</u>
  - A server which acts as an intermediary for some other server. Unlike a proxy, a gateway receives requests as if it were the origin server for the requested resource; the requesting client may not be aware that it is communicating with a gateway.

More Definitions

- <u>Tunnel</u>
  - An intermediary program which is acting as a blind relay between two connections.
  - Once active, a tunnel is not considered a party to the HTTP communication, though the tunnel may have been initiated by an HTTP request.
  - The tunnel ceases to exist when both ends of the relayed connections are closed.



ABC can be Proxy or Gateway

The actual request may not reach Origin Server. Intermediate elements may have cache copy.

National hierarchy of proxies are running to save transocenic bandwidth.

Can B be the only HTTP/1.0 entity?

Persistent Connection

Motivation

- Because of inline objects generally a client sends multiple HTTP requests in close time interval.
- There is a relative large overhead with the opening and closing of TCP connections.
- The ability to send multiple HTTP requests over one transport connection can improve the performance.

Advantages for Every One!

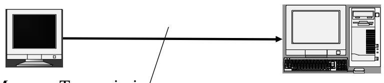
- Network: Lesser TCP open and close.
  - CPU time, memory overhead saved.
  - Reduced network congestion
- User: Pipe lined HTTP requests
  - One request does not have to wait for other.
- HTTP: Graceful operations
  - Error can be reported even if a response fails without closing down the TCP.

HTTP 1.1 Persistent Connection

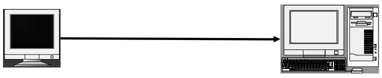
- Default is persistence
  - All HTTP 1.1. connections are now by default persistent!
  - Client may assume that the server will maintain a persistent connection.
- New connection header:
  - The connection close signaling takes place using the Connection header field “close”.
  - Once a close has been signaled, that requests becomes the last one for the connection.
- Must define message length:
  - In order to remain persistent, all messages on the connection must have a self-defined message length (i.e., one not defined by closure of the connection).

Flash back: HTTP General Headers

Header	*1.0*	*1.1*	*Explanation*
Cache Control	N	Y	Cacheing Directions.
Connection	N	Y	Connection Maintenance routine
Date	Y	Y	Date of time of message origination.
Forwarded	N	Y	Used by gateways to trace intermediate steps and avoid loops
Keep-Alive	N	Y	Diagnostic Information.
MIME-version	Y	Y	Contains the mime version used to encode the message.
Pragma	Y	Y	Contains implementation directives (such as no cacheing)
Upgrade	N	Y	Lists additional protocols a client supports and would like to use if server agrees.

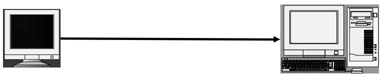


- Message Transmission
  - Default HTTP/1.1: “keep connection open”.
  - Header “connection: close” in a request means close the connection after servicing the request.
  - A set of requests can be pipelined. Client does not have to wait for previous responses.
  - But don’t forget
    - for HTTP/1.0 default is not persistent.
    - A proxy server MUST signal persistent connections separately with its clients and the origin servers (or other proxy servers) that it connects to.

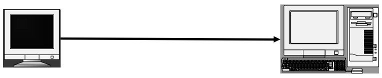


- Practical Considerations
  - Servers usually have some time-out value beyond which they will no longer maintain an inactive connection.
  - When a client or server wishes to time-out it SHOULD issue a graceful close on the transport connection.
  - Clients and servers SHOULD both constantly watch for the other side of the transport close, and respond to it as appropriate.
    - If a client or server does not detect the other side's close promptly it could cause unnecessary resource drain on the network.

Proxy may have higher time-out value. Why?



- Practical Considerations (continued..)
  - A client, server, or proxy MAY close the transport connection at any time. For example, a client MAY have started to send a new request at the same time that the server has decided to close the "idle” connection.
  - This means that clients, servers, and proxies MUST be able to recover from asynchronous close events.
  - Client software SHOULD reopen the transport connection and retransmit the aborted request without user interaction so long as the request method is idempotent.
  - Non idempotent methods MUST NOT be automatically retried, although user agents MAY offer a human operator the choice of retrying the request.



- Practical Considerations (continued..)
  - However, this automatic retry SHOULD NOT be repeated if the second request fails.
  - Servers SHOULD always respond to at least one request per connection, if at all possible. Servers SHOULD NOT close a connection in the middle of transmitting a response, unless a network or client failure is suspected.
  - Clients that use persistent connections SHOULD limit the number of simultaneous connections that they maintain to a given server. A single-user client SHOULD maintain at most 2 connections with any server or proxy.

What will be the maximum recommended connections per proxy server where N is the number of simultaneously active users?

- Conversation between HTTP 1.1 Client to HTTP 1.1 Server:
  - Client may sends part of a request
  - Server may in between send status 100 (continue).
  - An HTTP/1.1 client must be ready to receive status 100 after sending any regular expression.
    - It means the server understands the first part of the request, and is still willing to receive rest of the request.
  - Client sends rest of the request, or if rest has been transmitted, do nothing.

- Connection Establishment HTTP/1.1 Client
  - An HTTP/1.1 server MUST either respond with 100 (Continue) status and continue to read from the input stream, or respond with an error status. If it responds with an error status, it MAY close the transport (TCP) connection.
  - If an HTTP/1.1 client has not seen an HTTP/1.1 or later response from the server, it should assume that the server implements HTTP/1.0 or older and will not use the 100 (Continue) response.
  - If in this case the client sees the connection close before receiving any status from the server, the client SHOULD retry the request using the "binary exponential back-off" algorithm to be assured of obtaining a reliable response.

Binary Exponential Backoff of HTTP/1.1 Client

1. Initiate a new connection to the server
2. Transmit the request-headers
3. Initialize R (based on the round-trip time it took to establish the connection, or to a constant value of 5 seconds).
4. Compute T = R * (2**N), where N is the number of previous retries.
5. Wait either for an response from the server, or for T seconds (whichever comes first) →
6. If no error response is received, after T seconds transmit the body of the request.
7. If the connection is closed prematurely, repeat from step 1 until the request is accepted, an error response is received, or the user becomes impatient and terminates the retry process.

If status 100 comes back it is an HTTP/1.1. If then connection closes down after that then resend without waiting for status code.

Idempotent Methods

Methods may also have the property of "idempotence" in that (aside from error or expiration issues) the side-effects of N > 0 identical requests is the same as for a single request. The methods GET, HEAD, PUT and DELETE share this property.

Advantages of Persistent Connection

- By opening and closing fewer TCP connections, CPU time and memory is saved.
- Pipelining allows a client to make multiple requests without waiting for each response, allowing a single TCP connection to be used much more efficiently, with much lower elapsed time.
- Network congestion is reduced by reducing the number of packets caused by TCP opens, and by allowing TCP sufficient time to determine the congestion state of the network.
- HTTP can evolve more gracefully; since errors can be reported without the penalty of closing the TCP connection.