Web overview

Sergiy Kukharyev

telegram: 095-397-398-1
http://github.com/epam-sergej/FE

Client/Server architecture

Server side:

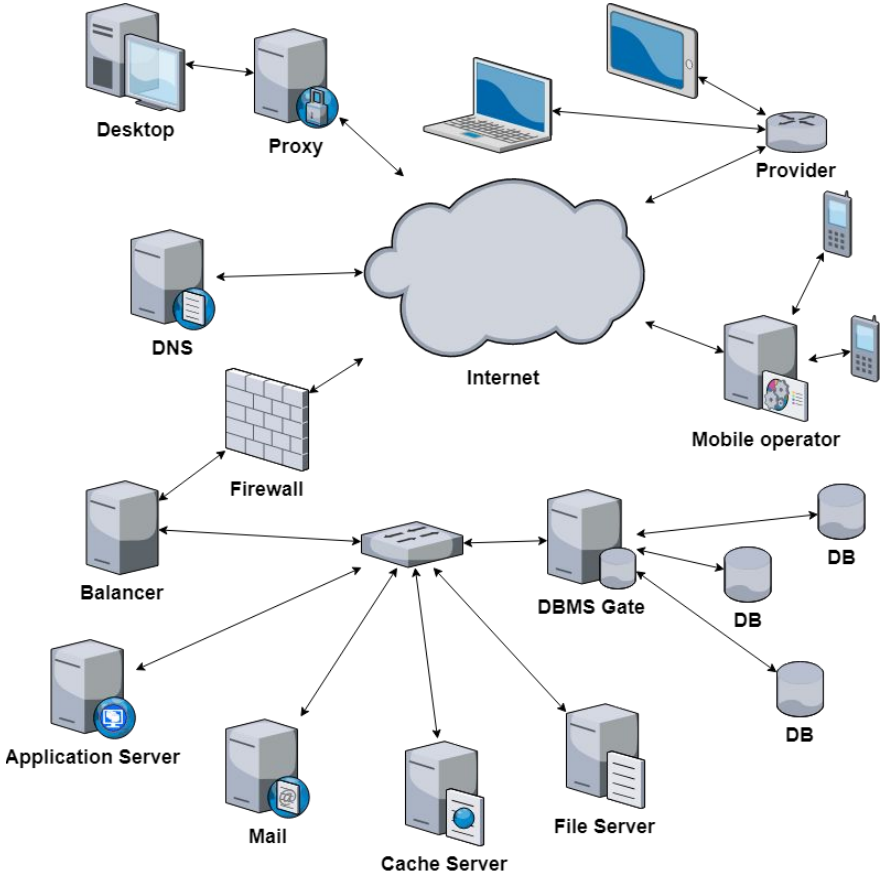
- Firewall
- Balancer Server's
- Application Server's
- Cache Server's
- File Server's
- DBMS Server's
- Mail Server's
- etc

Client side:

- Proxy
- Desktop
- etc

Others:

- DNS
- Provider
- etc



HTTP Protocol Training Objectives

1. Create low level HTTP protocol understanding
2. Provide knowledge for HTTP debugging tools and practices
3. Provide HTTP performance improvement tips
4. Provide Internet Browser tips
5. Create HTTP troubleshooting skill
6. Understanding, not just knowledge

Why is this training is required?

- Issue is not always on application level – low level troubleshooting skill is required
- Performance improvement is performed on low level
- Web Service integration troubleshooting skill
- Web-Caching usage require specific knowledge

HTTP overview plan

1. HTTP History & Specification
2. Methods, Response Statuses, Headers
3. URL Format
4. Cache Control
5. Sessions & Cookies
6. Performance Improvements and Load Balancing
7. Browser Features, Debugging and Tracing

1. HTTP History & Specification

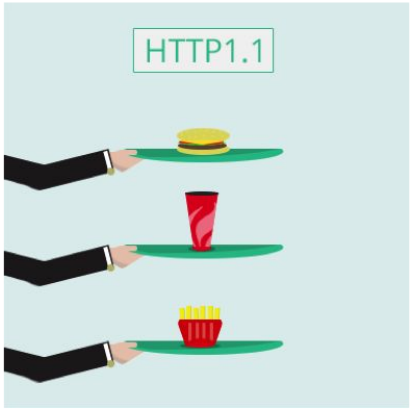
HTTP History

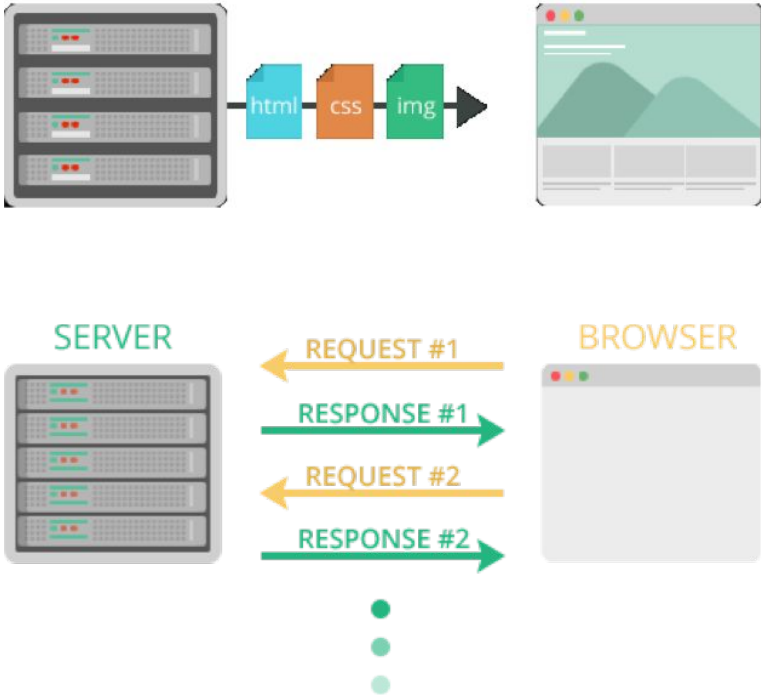
HTTP Versions

- 0.9 (1991) – to aggregate used technologies to specification, support only one method GET and no headers. Currently fully deprecated.
- 1.0 (1996) – RFC 1945, define multiple methods and headers, still widely used and supported.
- 1.1 (1999) – RFC 2616, persistent connections, content negotiation, content compression, extended proxy support, etc. Currently active version.

HTTP 1.x Interaction

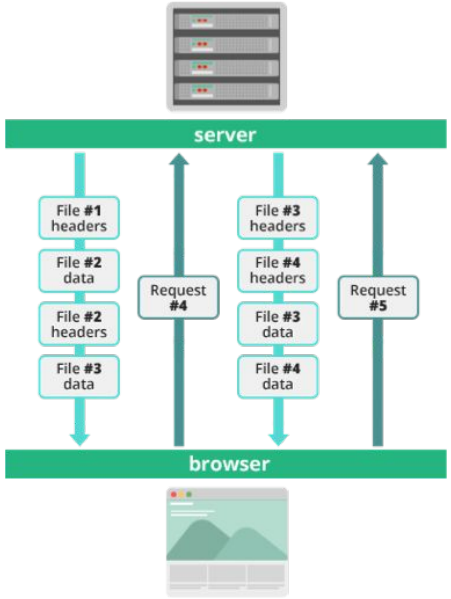
- Client locates the server
- Client opens connection to server
- Client sends request to server
- Server returns response to client
- Server closes connection

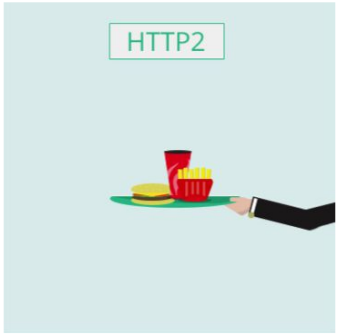


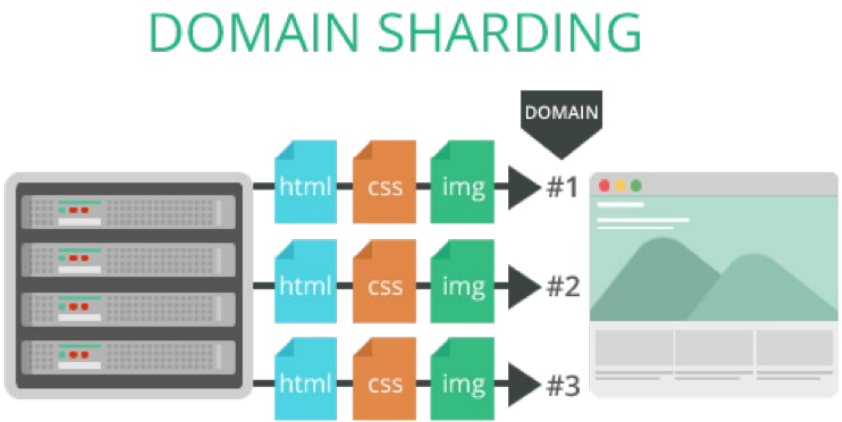


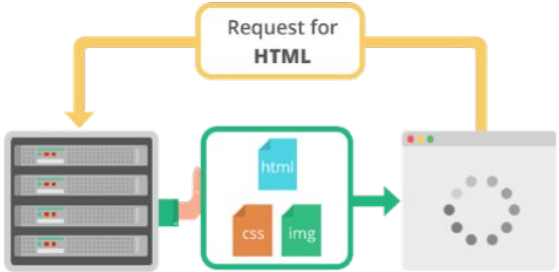
HTTP 2.0 Introduction

- Multiplexed streams
- Server push
- Binary protocol
- Stream prioritization
- Statefull headers compression









2. Methods, Response Statuses, Headers

HTTP Request Format

<Header line>
Method*<space>*URI*<space>*Version

<Header key1>:<Header value1>
<Header key2>:<Header value2>
<Header key3>:<Header value3>
<Header key4>:<Header value4>
...

<Empty line>

<HTTP BODY>

GET / HTTP/1.1

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, ..., */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727; .NET CLR 3.0.04506.30; InfoPath.2; .NET CLR 3.0.04506.648)
Host: www.epam.com:80
Connection: Keep-Alive
…

<Empty line>

<HTTP BODY>

HTTP Response Format

<Header line>
Protocol<*space*>Code<*space*>Message

<Header key1>:<Header value1>
<Header key2>:<Header value2>
<Header key3>:<Header value3>
<Header key4>:<Header value4>
…

<Empty line>

<HTTP BODY>

HTTP/1.1 200 OK
Date: Tue, 19 Aug 2008 09:39:48 GMT
Server: Microsoft-IIS/6.0
Cache-Control: max-age=2592000;post-check=31449600,pre-check=31449600
X-Powered-By: ASP.NET
X-AspNet-Version: 2.0.50727
Cache-Control: private
Content-Type: text/html; charset=utf-8
Content-Length: 57915

<html>
<head>
<title>EPAM Systems - #1 offshore software development outsourcing company in Russia & Eastern Europe</title>...

HTTP Methods

GET
The most common method used on the Web. Just requests a representation of the specified resource. No request body is expected, request parameters are passed in URI in the URL encoding fashion. Amount of parameters are limited by specification (URI length), but mostly browsers/servers support less length. URL is enough to reproduce the request. It can be easily forwarded or bookmarked.

HEAD
Asks for the response identical to the one that would correspond to a GET request, but without the response body. This is useful for retrieving response headers, without having to transport the entire content. Used by load-balancers to check nodes alive. HTTP servers are supposed to implement at least the GET and HEAD methods

POST
Submits data to be processed (e.g. from an HTML form) to the specified resource. Parameters and other data is included in the body of the request (parameters also can be included into URI). URL is not enough to reproduce request, can't be bookmarked. Response can't be cached.

PUT
Uploads a representation of the specified resource. Mostly used together with WebDAV (Web-based Distributed Authoring and Versioning)

DELETE
Deletes the specified resource

TRACE
Echoes back the received request, client can see what intermediate servers are adding or changing in the request

OPTIONS
Returns the HTTP methods that the server supports for specified URI

CONNECT
Converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS) through an unencrypted HTTP proxy

HTTP Response Status Groups

1xx Informational

Request received, continuing process. This class of status code indicates a provisional response, consisting only of the Status-Line and optional headers, and is terminated by an empty line. Since HTTP/1.0 did not define any 1xx status codes, servers must not send a 1xx response to an HTTP/1.0 client

2xx Success

The action was successfully received, understood, and accepted. This class of status code indicates that the client's request was successfully received, understood, and accepted.

3xx Redirection

The client must take additional action to complete the request. This class of status code indicates that further action needs to be taken by the user agent in order to fulfill the request. The action required may be carried out by the user agent without interaction with the user if and only if the method used in the second request is GET or HEAD.

4xx Client Error

The request contains bad syntax or cannot be fulfilled. The 4xx class of status code is intended for cases in which the client seems to have erred. Except when responding to a HEAD request, the server should include an entity containing an explanation of the error situation, and whether it is a temporary or permanent condition.

5xx Server Error

The server failed to fulfill an apparently valid request. Indicate cases in which the server is aware that it has erred or is incapable of performing the request. Except when responding to a HEAD request, the server should include an entity containing an explanation of the error situation, and whether it is a temporary or permanent condition.

HTTP Response Status Groups Samples

200 OK
Standard response for successful HTTP requests

302 Found
This is the most popular redirect code, but also an example of industrial practice contradicting the standard – per HTTP 1.1 303 must be used for this purpose, but it's still used from HTTP 1.0 times. Response Location header is used to point to new resource location

304 Not Modified
Indicates the request URL has not been modified since last requested. Typically, the HTTP client provides a header If-Modified-Since or If-None-Match header to specify a time against or hash which to compare. Utilizing this saves bandwidth and reprocessing on both the server and client. Browser utilize this when resource is available in the browser cache, but only for the first time be browser session

401 Unauthorized
Authentication is possible but has failed or not yet been provided

403 Forbidden
The request was a legal request, but the server is refusing to respond to it. Unlike a 401 Unauthorized response, authenticating will make no difference

404 Not Found
HTTP standard response code indicating that the client was able to communicate with the server but either the server could not find what was requested, or it was configured not to fulfill the request

500 Internal Server Error
HTTP standard response code indicating that the server failed to perform response because of error

502 Bad Gateway
Server is configured to delegate request fulfillment and was unable to connect underneath server

HTTP Most Common Request Headers (part 1)

Header	Description
Accept	This field contains a semicolon-separated list of Content-Type values which are accepted in the response to this request. E.g.: Accept: text/plain, text/html
Accept-Encoding	Similar to Accept, but lists the Content-Encoding types which are acceptable in the response. E.g.: Accept-Encoding: x-compress; x-zip
Accept-Language	Similar to Accept, but lists the Language values which are preferable in the response. Language is specified per ISO 639. E.g.: Accept-Language: en-us
User-Agent	Gives the software program used by the original client. This is for statistical purposes and the tracing of protocol violations
Referrer	This **optional** header allows the client to specify, for the server's benefit, the address of the document from which the request's URI was obtained. (Correct spelling is Referrer)
Authorization	used with GET to make it conditional: if the requested document has not changed since the time specified the document will not be sent with Not Modified 304 reply

HTTP Most Common Request Headers (part 2)

Header	Description
If-Modified-Since	used with GET to make it conditional: if the requested document has not changed since the time specified the document will not be sent with Not Modified 304 reply
Pragma	Pragma directives should be understood by servers to which they are relevant, e.g. a proxy server; currently only one pragma is defined. E.g.: “no-cache”
Cache-Control	used to specify directives that MUST be obeyed by all caching mechanisms along the request/response chain
Connection	specify options that are desired for that particular connection
Content-Length	indicates the size of the entity-body, in decimal number of OCTETs, sent to the recipient
Host	Specifies the Internet host and port number of the resource being requested. A client **MUST include** a Host header field in **all HTTP/1.1 request messages**

HTTP Most Common Response Headers (part 1)

Header	Description
Content-MD5	**MD5 digest of the body for the purpose of end-to-end message integrity check**
Location	Mostly used to redirect the recipient to a location other than the Request-URI for completion of the request or identification of a new resource
Server	Contains information about the software used by the origin server to handle the request
Via	MUST be used by gateways and proxies to indicate the intermediate protocols and recipients between the user agent and the server on requests, and between the origin server and the client on responses. E.g.: Via: 1.0 fred, 1.1 nowhere.com (Apache/1.1)
WWW-Authenticate	MUST be included in 401 (Unauthorized) response messages. Value consists of at least one challenge that indicates the authentication scheme(s) and parameters applicable

HTTP Most Common Response Headers (part 2)

Header	Description
Cache-Control	used to specify directives that MUST be obeyed by all caching mechanisms along the request/response chain
Content-Encoding	indicates what additional content coding have been applied to the entity-body, and thus what decoding mechanisms must be applied in order to obtain the media (can be also used in requests)
Content-Length	indicates the size of the body, in decimal number of OCTETs, sent to the recipient or, in the case of the HEAD method, the size of the entity-body that would have been sent had the request been a GET
Accept-Ranges	allows the server to indicate its acceptance of range requests for a resource
Cache-Control	used to specify directives that MUST be obeyed by all caching mechanisms along the request/response chain

HTTP Authentication: Basic and Digest

```
GET / HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible;
MSIE 6.0; Windows NT 5.1; SV1)
Host: hrm.epam.com
Proxy-Connection: Keep-Alive

HTTP/1.1 401 Unauthorized
WWW-Authenticate: Basic realm="HTTP
Authentication"
Connection: close
```

```
GET / HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible;
MSIE 6.0; Windows NT 5.1; SV1)
Host: hrm.epam.com
Proxy-Connection: Keep-Alive
Authorization: Basic
bWlrYWxhaV96ZW5rZXZpY2g6TmF0LXNo
YTY5

HTTP/1.1 200 OK
Date: Mon, 08 Sep 2008 15:19:56 GMT
Server: Apache/2.0.55 (Win32)
mod_ssl/2.0.55 OpenSSL/0.9.8a
mod_jk2/2.0.4
```

HTTP File Upload Form

```
<form name="DocumentForm" method="POST"
action="/pmc/document/detail.do"
enctype="multipart/form-data">

...

<input type="hidden" name="webBase" value="">
<input type="file" name="documentBody" value="">

…

</form>
```

HTTP File Upload Request

```
POST /pmc/document/detail.do?id=-1221462&subpage=subdetail&typeId=64 HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Referer: http://pmcmsq.epam.com/pmc/document/detail.do?dispatch=add&appId=40022&projectId=400062
Accept-Language: en-us
Content-Type: multipart/form-data; boundary=---------------------------7d821d2132070e
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727)
Host: pmcmsq.epam.com
Content-Length: 11352
Proxy-Connection: Keep-Alive
Pragma: no-cache
Cookie: pmcId=fHwJ37oeerhtFx8XbiAcWYk

-----------------------------7d821d2132070e
Content-Disposition: form-data; name="creatorEmail"

Mikalai_Zenkevich@epam.com
-----------------------------7d821d2132070e
Content-Disposition: form-data; name="documentBody"; filename="d:\test.txt"
Content-Type: text/plain

It a test file to sniff file upload session for PMC.
-----------------------------7d821d2132070e
Content-Disposition: form-data; name="versionComment"


-----------------------------7d821d2132070e--
```

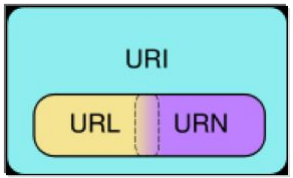
3. URL Format

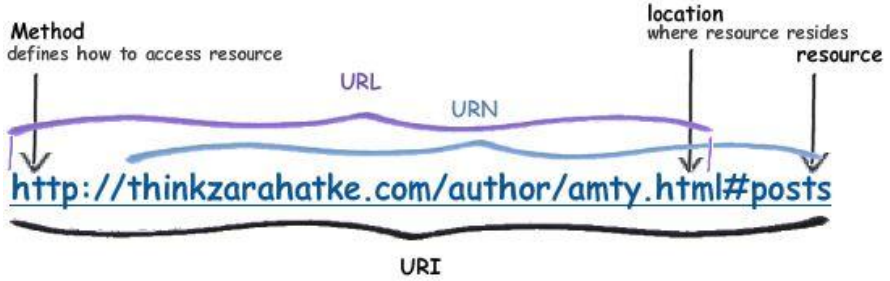
URL / URI / URN

URL – **Uniform Resource Locator** is a compact string of characters used to represent a resource available on the Internet.
Samples: http://example.org/, mailto:bob@example.com.

URN – **Uniform Resource Names** are intended to serve as persistent, location-independent resource identifiers and are designed to make it easy to map other namespaces (that share the properties of URNs) into URN-space.
Format: "urn:" + <NID> ":" <NSS>.
Samples: urn:isbn:0451450523, urn:www.agxml.org:schemas:all:2:0.

URI – **Uniform Resource Identifier** is a compact string of characters used to identify or name a resource on the Internet. The main purpose of this identification is to enable interaction with representations of the resource over a network. A URI may be classified as a locator (URL) or a name (URN), or both.





URL / URI Format

foo://user:pass@example.com:842/there/index.htm;type=animal?name=ferret#nose

Part	Name	Description
foo	Scheme	Protocol or schema name
user	Username	Realm authentication user name. Optional
pass	Password	Realm authentication password. Optional
example.com	Hostname	DNS hostname or IP address of the server. Used to setup TCP socket connection
842	Port	TCP socket port listened by server. Optional – if not specified then used default for protocol. E.g. 80 for http, 443 for https (see below)
there	Path	Path to document. Optional
index.html	Filename and extension	Requested document file name and extension. Optional, welcome or index file used in this case
type=animal	Extension Parameters	Additional parameter(s) for the document access. Optional
name=ferret	Query	Request parameter(s). Optional
nose	Fragment	Navigation to the document fragment. Optional. In HTML called anchor tag

URL Encoding

URL encoding, is a mechanism for encoding information in a Uniform Resource Identifier (URI) under certain circumstances.
When a character from the reserved set has special meaning, then the character must be percent-encoded. Percent-encoding a reserved character involves converting the character to its corresponding value in ASCII and then representing that value as a pair of hexadecimal digits. The digits, preceded by a percent sign ("%").
When data that has been entered into HTML forms is submitted, the form field names and values are encoded. The encoding used by default is based on a very early version of the general URI percent-encoding rules, with a number of modifications such as newline normalization and replacing spaces with "+" instead of "%20". The MIME type of data encoded this way is application/x-www-form-urlencoded.
Binary data in a URI must divide the data into 8-bit bytes and percent-encode each byte in the same manner.

+	$	,	/	?	%	#	[	]
%3D	%2B	%24	%2C	%3F	%25	%23	%5B	%5D
!	*	(	)	;	:	@	&	=
%21	%2A	%27	%28	%29	%3B	%3A	%40	%36

4. Cache Control

What is it?

Cache is a collection of data duplicating original values stored elsewhere or computed earlier, where the original data is expensive to fetch or to compute, compared to the cost of reading the cache. In other words, a cache is a temporary storage area where frequently accessed data can be stored for rapid access. Web cache sits between one or more Web servers and a client or many clients, and watches requests come by, saving copies of the responses. Then, if there is another request for the same URL, it can use the response that it has, instead of asking the origin server for it again.

There are three main reasons for using web cache:
To reduce latency — request satisfaction from the cache (which is closer to the client) instead of the origin server takes less time.
To reduce network traffic — reduces the amount of bandwidth used by clients and keeps bandwidth requirements lower and more manageable.
To reduce server load – reduce the number of requests need to be served.

Main issues with using web cache:
Possibility for client to get obsolete or inconsistent response.
Sometimes using transforms to fighting with web cache – make browser to obtain latest (updated) version of the resource or perform call each time.

Where to perform caching?

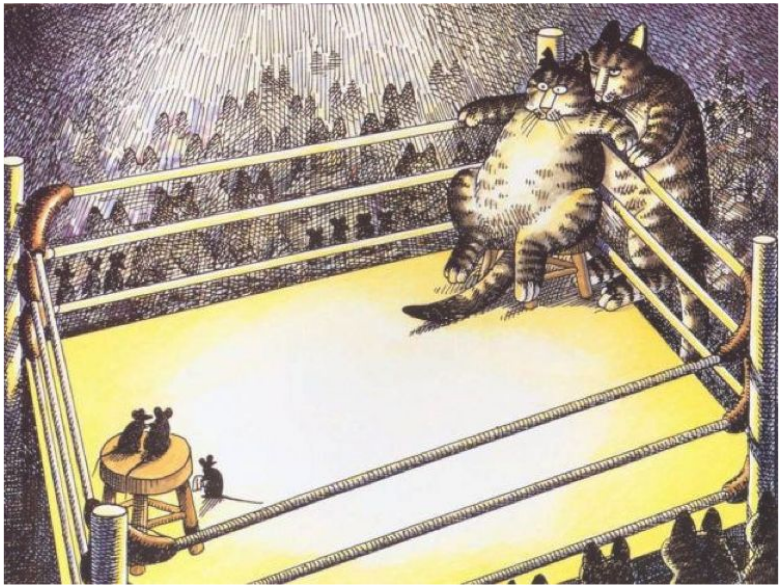
- **User agent cache** – caching performed by client.
- **Proxy cache** (forward proxy) are usually deployed by internet service providers, schools and corporations to save bandwidth. Interception proxy caches (transparent) are a variant that doesn't require clients to be explicitly configured to use them.
- **Gateway cache** (reverse proxy or web accelerators) – operate on behalf of the origin server, and to clients are indistinguishable from it.

Ways to control cache

- On protocol level (headers level) – provide cache related HTTP headers in accordance with specification to control required cache.

  Caching depends not only on protocol and server settings, but also on proxy or client settings. Problem: different clients have different user settings related to cache.

- On application level – fully enable caching on protocol level and control it by application with special tricks



HTTP basic mechanisms

HTTP defines three basic mechanisms for controlling caches:

- Freshness allows a response to be used without re-checking it on the origin server, and can be controlled by both the server and the client. For example, the Expires response header gives a date when the document becomes stale, and the Cache-Control: max-age directive tells the cache how many seconds the response is fresh for.

- Validation can be used to check whether a cached response is still good after it becomes stale. For example, if the response has a Last-Modified header, a cache can make a conditional request using the If-Modified-Since header to see if it has changed.

- Invalidation is usually a side effect of another request that passes through the cache. For example, if URL associated with a cached response subsequently gets a POST, PUT or DELETE request, the cached response will be invalidated.

Protocol Cache Control: Request Headers

Header	Description
Cache-Control	Used to specify directives that MUST be obeyed by all caching mechanisms along the request/response chain. See next slides for details.
If-Match	The If-Match request-header field is used by a client (having one or more entities previously obtained from the resource) to verify one of those entities by including a list of their associated entity tags in the If-Match header field.
If-Modified-Since	The If-Modified-Since header is used and if the requested variant has not been modified since the time specified in this field, an entity will not be returned from the server; instead, a 304 (not modified) response will be returned without any message-body.
If-None-Match	The If-None-Match header is used by a client (having one or more entities previously obtained from the resource) to verify that none of those entities is current by including a list of their associated entity tags in the If-None-Match header field.
If-Range	If a client has a partial copy of an entity in its cache, and wishes to have an up-to-date copy of the entire entity, it could use the Range request-header with a conditional GET (using either or both of If-Unmodified-Since and If-Match.) However, if the condition fails, the client would then have to make a second request to obtain the entire current entity-body.
If-Unmodified-Since	The If-Unmodified-Since header is used and if the requested resource has not been modified since the time specified in this field, the server should perform the requested operation as if the If-Unmodified-Since header were not present. Not used with GET and POST.
Pragma	When the no-cache directive is present in a request message, an application should forward the request toward the origin server even if it has a cached copy of what is being requested.

Protocol Cache Control: Response Headers

Header	Description
Age	The age the object has been in a proxy cache in seconds
Cache-Control	Used to specify directives that MUST be obeyed by all caching mechanisms along the request/response chain. See next slides for details.
ETag	Used to determine change in content at a given URL. When a new HTTP response contains the same ETag as an older HTTP response, the contents are determined to be the same without further downloading. One method of generating the ETag is based on the last modified time of the file and the size of the file. Thus, generating a HEAD request and checking the ETag header of the response is an effective way for a browser to determine whether a previously cached response needs to be fetched again. **Note:** HTTP 1.1 does not specify any particular way for a server to generate an entity tag, they are analogous to a message digest or checksum for a file.
Expires	Date/time after which the response is considered stale. The presence of an Expires field does not imply that the original resource will change or cease to exist at, before, or after that time.
Last-Modified	The Last-Modified header returns the date and time that the requested resource was last updated. It did not always work as well as hoped in HTTP 1.0, so the ETag header now supplements it. The Last-Modified header implies that HTTP resources are static files. Like the ETag header, the web server does not typically generate the Last-Modified header for your scripts, although you can output it yourself if you desire.
Pragma	When the no-cache directive is present in a request message, an application should forward the request toward the origin server even if it has a cached copy of what is being requested
Vary	Use of server-driven content negotiation as indicated by the presence of a Vary header field in a response, alters the conditions and procedure by which a cache can use the response for subsequent requests.

HTTP Session Samples with Cache enabled

```
GET /ru/flight_info/index.html HTTP/1.1
Accept: image/gif, ..., */*
Referer: http://www.s7.ru/
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Host: www.s7.ru
Proxy-Connection: Keep-Alive
Cookie: JSESSIONID=2EDA67671743140456938C046CD64FDD;
test_cookieless=1; bannerIndex=1;
botMemorySession=activeFormName%3Dform_buy%3B

HTTP/1.1 200 OK
Date: Mon, 08 Sep 2008 12:44:50 GMT
Server: Apache
Last-Modified: Fri, 05 Sep 2008 07:17:54 GMT
ETag: "2000000001ba1-3ff0-45620de3ec337"-gzip
Accept-Ranges: bytes
Cache-Control: max-age=900
Expires: Mon, 08 Sep 2008 12:59:50 GMT
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 4163
Content-Type: text/html
Proxy-Connection: Keep-Alive

df35454…
```

```
GET /ru/flight_info/index.html HTTP/1.1
Accept: image/gif, ..., */*
Referer: http://www.s7.ru/ru/special_programs/index.html
Accept-Language: en-us
Accept-Encoding: gzip, deflate
If-Modified-Since: Fri, 05 Sep 2008 07:17:54 GMT;
length=16368
User-Agent: Mozilla/4.0 ...; InfoPath.2; .NET CLR
3.0.04506.648)
Host: www.s7.ru
Proxy-Connection: Keep-Alive
Cookie: JSESSIONID=2EDA67671743140456938C046CD64FDD;
test_cookieless=1; bannerIndex=3;


HTTP/1.1 304 Not Modified
Date: Mon, 08 Sep 2008 12:50:23 GMT
Server: Apache
ETag: "2000000001ba1-3ff0-45620de3ec337"
Expires: Mon, 08 Sep 2008 13:05:23 GMT
Cache-Control: max-age=900
Vary: Accept-Encoding
Proxy-Connection: Close
```

HTTP Session Samples with Cache disabled

```
GET /servlet/getServerDate HTTP/1.1
Accept: */*
Referer: http://www.s7.ru/
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727)
Host: www.s7.ru
Proxy-Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Mon, 08 Sep 2008 12:43:05 GMT
Server: Apache
Pragma: no-cache
Cache-Control: no-cache
Expires: -1
Content-Length: 99
Content-Type: text/javascript;charset=utf-8
Set-Cookie: JSESSIONID=2EDA67671743140456938C046CD64FDD; Path=/
Proxy-Connection: Keep-Alive

timeZone=180;serverMinutes=43;serverHours=16;initialBOTDay=8;initialBOTMonth=9;initialBOTYear=2008;
```

Some Cache Related Browser Tips

- If browser have resource in cache - call If-Modified-Since only first time. All further calls are not validated – cached version is used.
- Refresh (F5) – cause document re-load with If-Modified-Since. Ctrl-F5 – without any caching possibilities.
- F5 do not resets the HTML form. Ctrl-F5 does.
- HTTPS doesn’t allow to use If-Modified-Since. For the session all resources loaded at first consume.

5. Sessions & Cookies

Cookies

HTTP cookies - parcels of text sent by a server to a web client and then sent back unchanged by the client each time it accesses that server. Cookies are used for authenticating, session tracking, and maintaining specific information about users, such as site preferences or the contents of their electronic shopping carts.
Web page may contain images or other components retrieval of these components are stored on servers in other domains. Cookies that are set during called third-party cookies. Cookies that have an expiration date are called persistent cookies. The expiration date tells the browser when to delete the cookie.

If no expiration date is provided, the cookie is deleted at the end of the user session, that is, when the user quits the browser. This cookies are called session cookies.

As a result, specifying an expiration date is a means for making cookies to survive across browser sessions.

Cookies

	Details
Name	Name of the cookie
Value	Value of the cookie
Expire Date	Defines the expiry date and time of a cookie. If the expiry date is in the future when the browser exits, they will be remembered in a persistent cache inside the browser. If not, the cookie is discarded and unavailable next time you run the browser.
Domain	Defines the domain scope of a cookie. For security reasons, cookies can only be sent back to the original web-server. However, in large web-server farms, the web pages may be distributed for load balancing reasons. The domain attribute provides a way to widen the scope to any machine within a domain.
Path	Defines the path scope of a cookie. The scope of a cookie can be limited to a certain part of the document tree within the web server. By defining a node within the document hierarchy, the cookie will only be sent to the web server when requesting a page that exists at that path or lower down
Is Secure	Boolean value that defines whether the secure protocol is required. If it is activated, then the cookie is only sent to a server when the secure protocol is used
Max. Length	The specification says that cookies data stored is not expected to exceed 4 kilobytes (including all attributes)
Max. Amount	The specification says that browsers need support no more than 300 different cookies. Of those, no more than 20 should be associated with any particular server. **But:** all browsers are different

Cookies Session Sample

```
GET /index.jsp HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, */*
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)

HTTP/1.1 200 OK
Date: Mon, 08 Sep 2008 12:44:50 GMT
Server: Apache
Last-Modified: Fri, 05 Sep 2008 07:17:54 GMT
Expires: Mon, 08 Sep 2008 12:59:50 GMT
Content-Length: 4163
Content-Type: text/html
Set-Cookie: JSESSIONID=2EDA67671743140456938C046CD64FDD; Path=/

GET /page.jsp HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, */*
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Cookie: JSESSIONID=2EDA67671743140456938C046CD64FDD
```

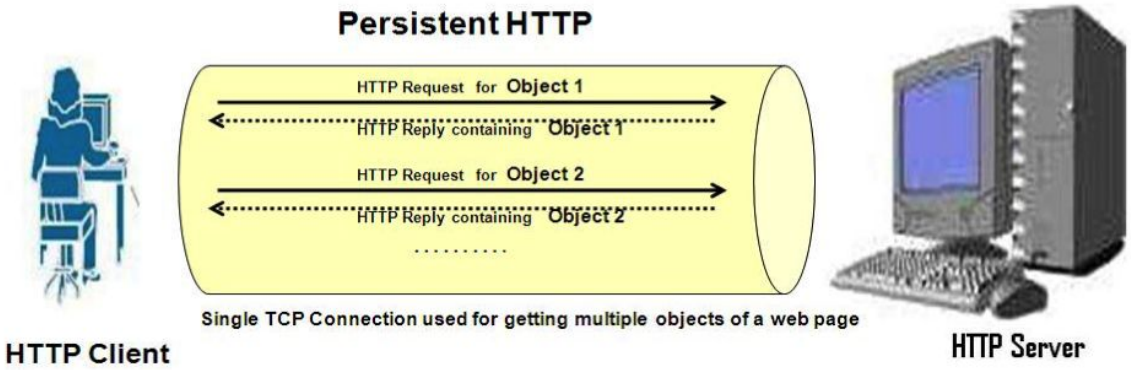
6. Performance Improvements and Load Balancing

Persistent Connections

- HTTP persistent connections, also called HTTP keep-alive, or HTTP connection reuse, is the idea of using the same TCP connection to send and receive multiple HTTP requests/responses, as opposed to opening a new connection for every single request/response pair.

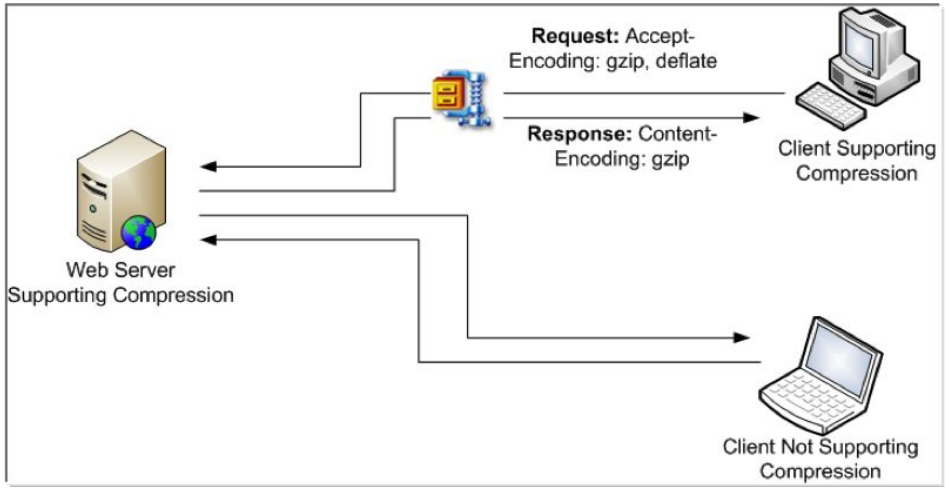
- In HTTP/0.9 and 1.0, the connection is closed after a single request/response pair. In HTTP/1.1 a keep-alive-mechanism was introduced, where a connection could be reused for more than one request.

- Such persistent connections reduce lag perceptibly, because the client does not need to re-negotiate the TCP connection after the first request has been sent.



Content Compression

HTTP compression is a capability of HTTP /1.1 protocol version to make better use of available bandwidth. Data is compressed before it is sent from the server: compliant browsers will announce what methods are supported to the server before downloading the correct format; unsupported browsers will download uncompressed data. Data is usually compressed with either deflate or gzip modules specific to the server software.



Compression Session Sample

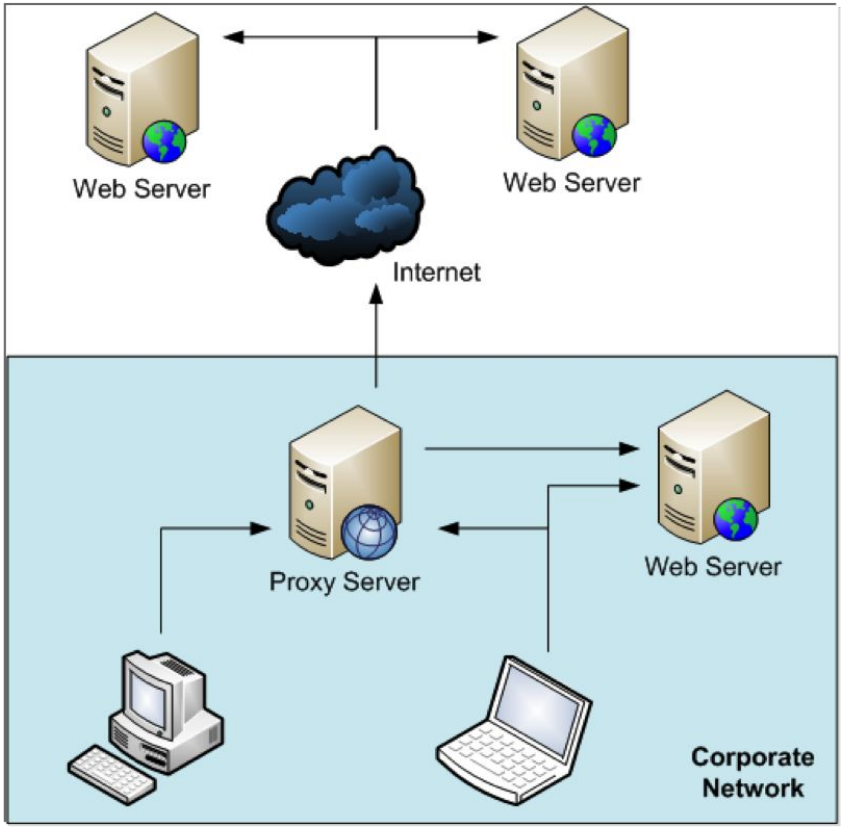
```
GET / HTTP/1.1
Accept: image/gif, image/jpeg, application/x-shockwave-flash, application/msword, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727)
Host: www.s7.ru
Proxy-Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Mon, 08 Sep 2008 12:03:00 GMT
Server: Apache
Last-Modified: Mon, 08 Sep 2008 09:13:30 GMT
ETag: "300000000028e-fb24-4565ed5316de1"-gzip
Accept-Ranges: bytes
Cache-Control: max-age=900
Expires: Mon, 08 Sep 2008 12:18:00 GMT
Content-Encoding: gzip
Content-Length: 10739
Content-Type: text/html
Proxy-Connection: Keep-Alive

‹        н}ksG’азU„яС/ґИсды:Н‘IhKў†¤д™ќ›@4Ъ&ЩР
```

HTTP Proxy Server

Proxy Servers

- Private – inside corporate network
- Public – available for everybody

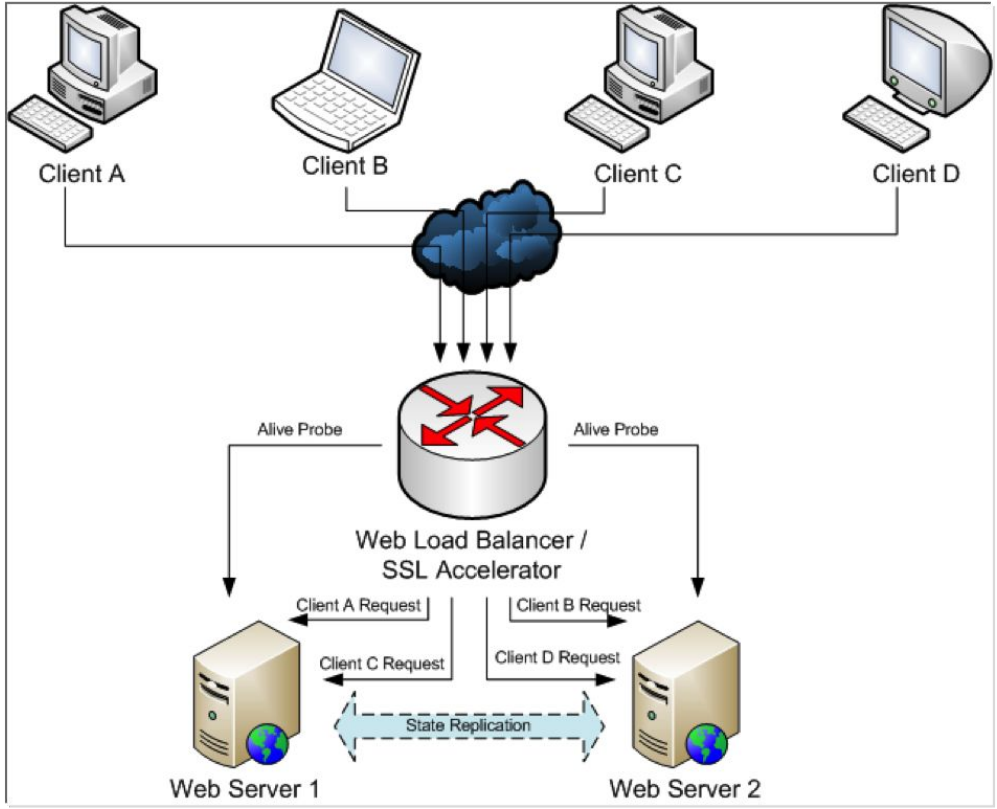
- Impersonate users (NAT is other option)
- Allows content filtering and measuring (do not mix up with traffic blocking)
- Additional cache level for shared resources
- Additional Internet access protection



Load Balancing

- Load Balancing – spread work or requests to logical services between multiple equals physical deployments (nodes, instances) with purpose to increase performance, number of concurrent requests serving and availability
- Cluster – physical deployments (nodes, computers, appl. servers) working together closely so that in many respects they can be considered as a single service
- Load balancing may be performed with multiple approaches starting from resource addressing (e.g. DNS balancing), by special network equipment and configuration, by using front proxy server
- Failover LB – if any failure of cluster node determined by balancer and excluded from further request serving. All requests intended for this deployment are routed to others.
- Session replication for stateful services balancing - service state is not lost after cluster node failure and available for client on other nodes

HTTP Load Balancing



Stateful services and Load Balancing

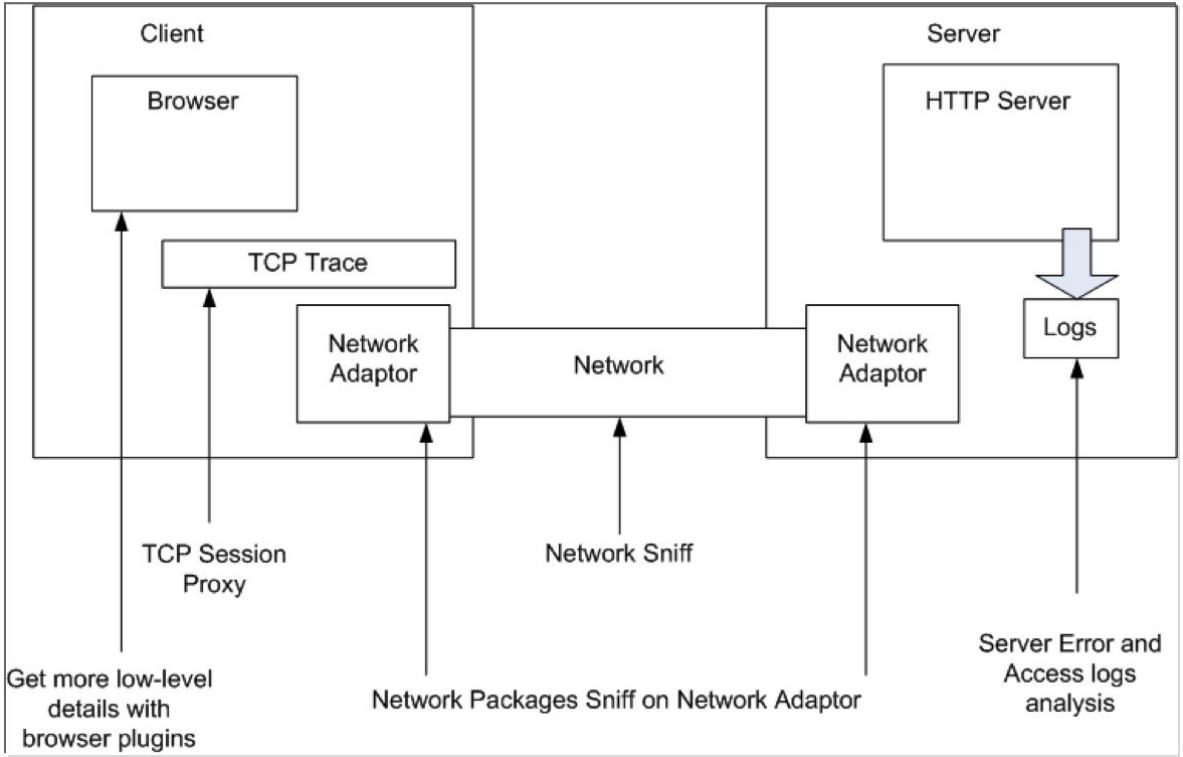
For balancing stateful services 2 main approaches:

- Replicate sessions including storing session at DB (also allows soft session shift after failure)
- Route all requests from the same client to the same node

    - Sticky IP – limitation for some network configurations
    - Sticky User Name - only for authorized users
    - Sticky Cookies – requires session cookies to be enabled

7. Browser Features, Debugging and Tracing

Tracing Options



Command Line Utilities Features

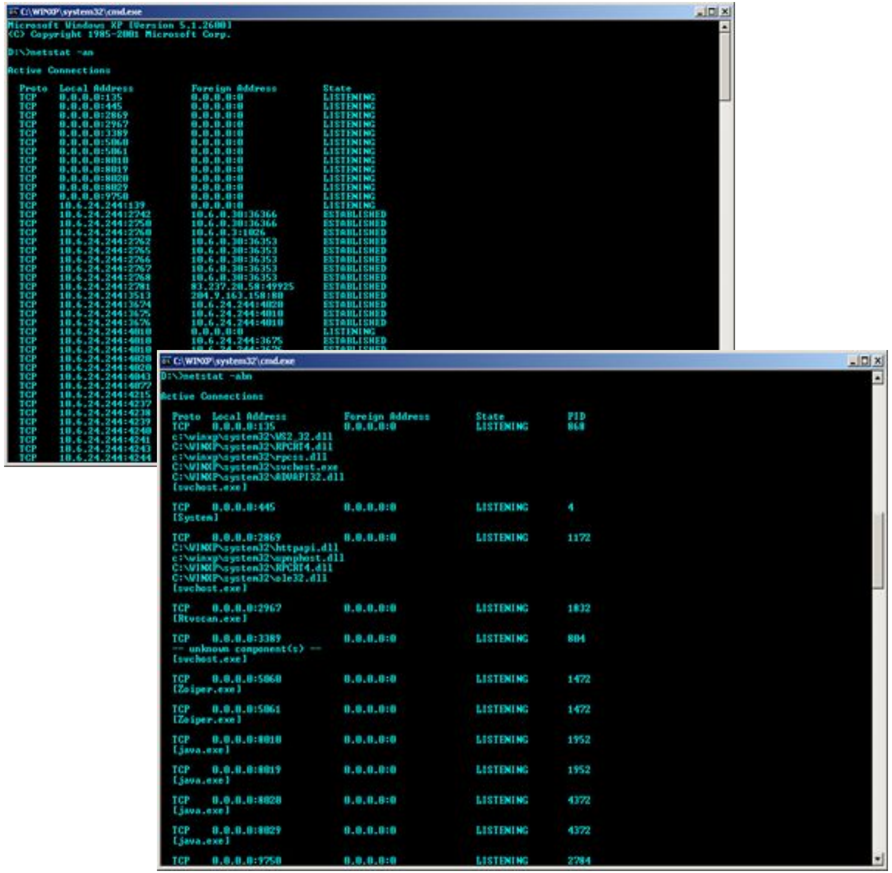
- Always available (not platform dependent)
- Easy emulating of HTTP client
- Allows quick check port is listened
- Allows quick check that HTTP server is listen given port
- Allows verify what process is currently occupied required port
- Not easy to issue complicated HTTP request by typing

Command Line Utilities

netstat –an
netstat -abn

telnet

- Issue telnet connection command
  C:\>telnet www.epam.com 80
- Specify your request (sometimes input echo is disabled):
  GET / HTTP/1.0 ↙
  ↙
  Note: for HTTP/1.1 you MUST provide Host header, otherwise you will get 400 error
- View server response
  HTTP/1.1 200 OK

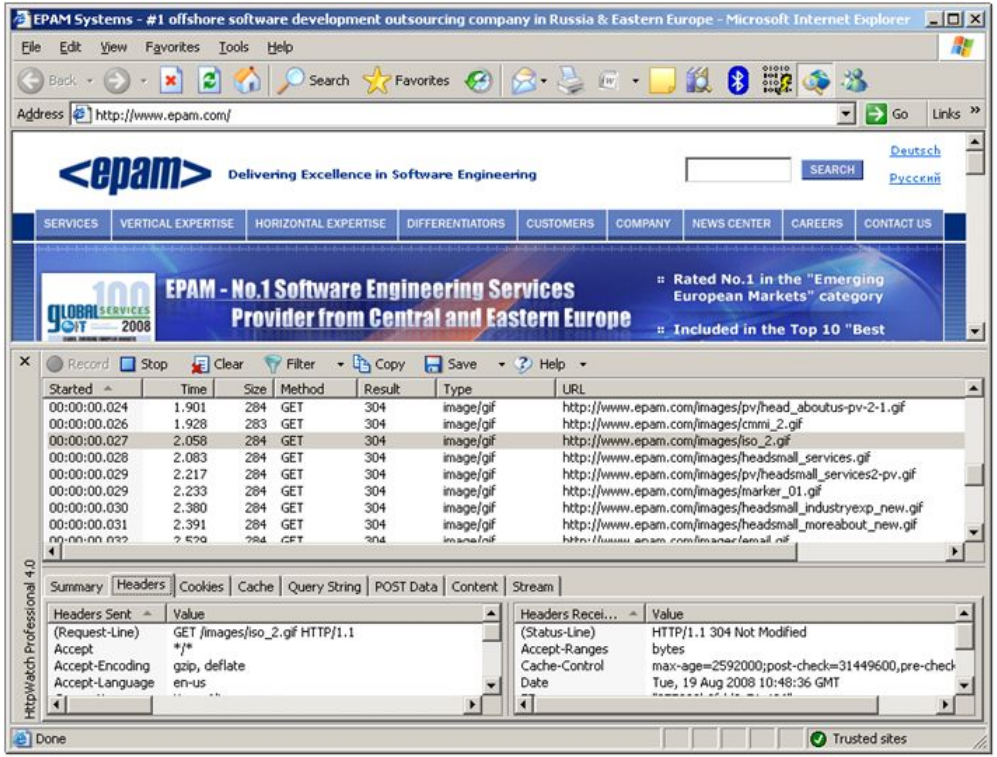


Browser Plugins Features

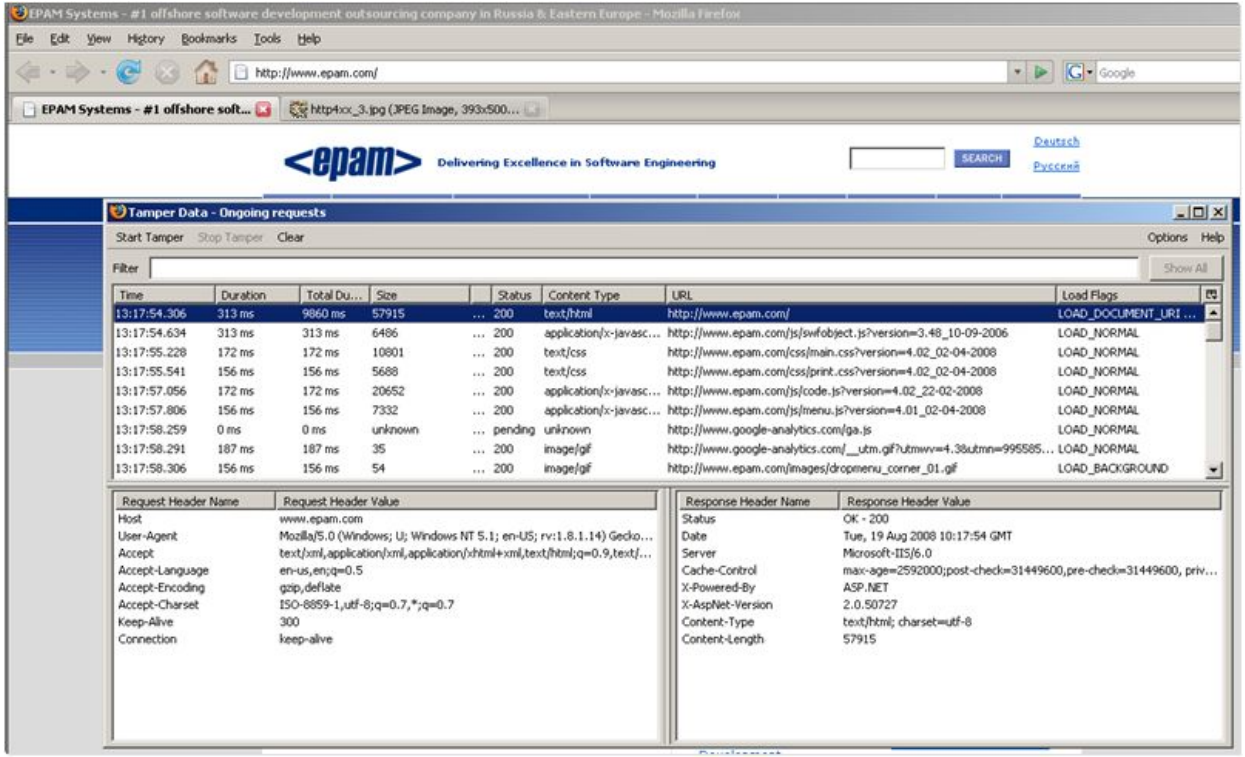
- HTTP Watch for IE – not free, not purchased by EPAM
- Tamper Data for FireFox – free, allowed by EPAM

- Check all request/response details issued by browser
- Depends on browser, no universal tool of such kind
- Not suitable for other clients debugging
- Allows troubleshooting of all page resources load
- Filter and highlight required requests/response to check hidden fileds
- Measure page load velocity
- Check caching productivity
- Copy and save requests/response
- Alter requests on fly (Tamper): bypass client side validation and check server side validation

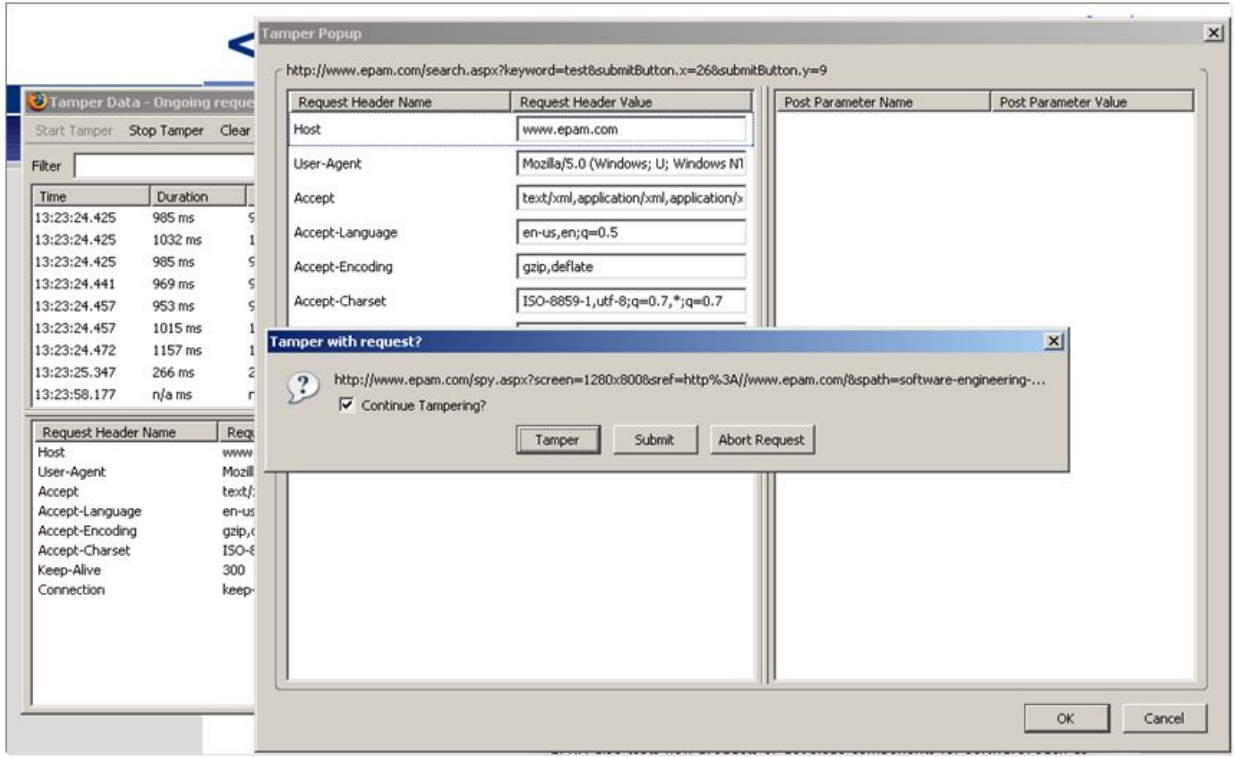
HTTP Watch (IE)



Tamper (Firefox Plugin). Slide #1



Tamper (Firefox Plugin). Slide #2

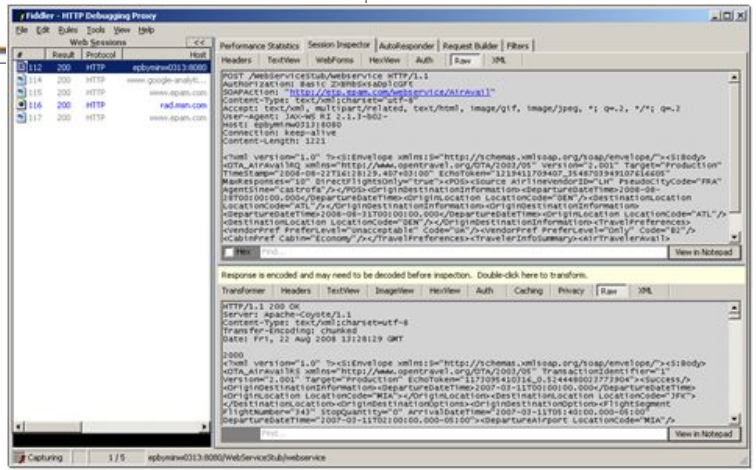


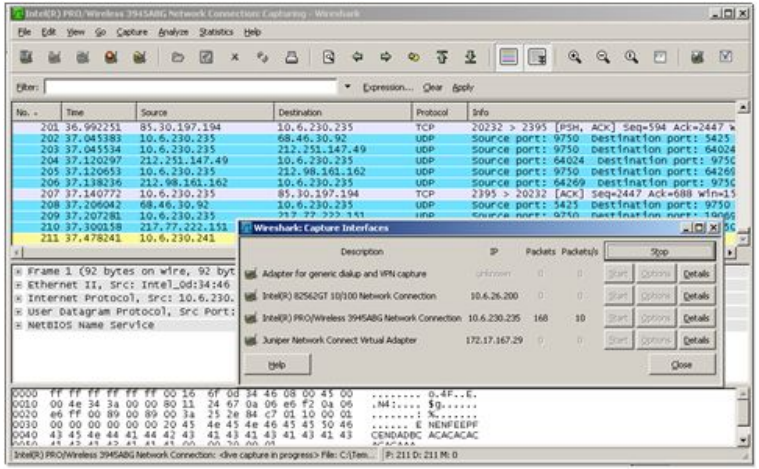
Traffic Sniffers Features

- Comm View – not free, not purchased by EPAM, but evaluation version is availble
- WireShark – free, considered to be allowed by EPAM

- Suitable not only for HTTP problems troubleshooting
- Debugging on very low level, allows investigation of very deep problems on network level
- Not easy to configure correct filtering and restore HTTP session
- You must remember about what network adapter is currently in use
- Doesn’t allow easy cache troubleshooting

Traffic Analysis Demo





Thank you!