●Bad requests handled by the Proxy:
(the following requests will cause error from client, return with "HTTP/1.1 400 Bad Request")

1.Methods can only be GET, POST and CONNECT, other http method would not be implemented and would be reported as bad request directly to the client. This request would not be sent to server.

DELETE http://www.baidu.com/ HTTP/1.1
Host: www.baidu.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0)

2.Protocol can only be 1.1/1.0. Other protocol could not be not issued.

GET http://www.baidu.com/ HTTP/1.2
Host: www.baidu.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0)

3.The path cannot be empty. After parse the path form the header, if the path is empty, the proxy would handle this request as bad request directly and send 400 to client, and server would not receive any request.

GET HTTP/1.1
Host: www.baidu.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0)

4.The path should be right: Even if with the right host information and not empty path, the proxy could not determine whether the request is valid, it would connected with the server, and let server handle this path, if server could not find the right response, it would send back a suitable response and the proxy would forward this response to the client.

GET http://ww HTTP/1.1
Host: www.baidu.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0)

5.The host cannot be empty: the proxy find that the host it needs to connect is empty, it would directly send response to the client and not forward to connect the remote

server.
GET http://www.baidu.com/ HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0)

6.The host cannot be wrong: the proxy could not find whether it is a valid host, thus it would connect with the whatever provided by the request header and try to make the connection, if it can not make the connection, it would find the host is not valid. Then the proxy would send 400 to client to inform the error.
GET http://www.baidu.com/ HTTP/1.1
Host: rthytjytityhrt
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0)

7.Port number should be 0 – 65535, the port number must in this range. Since the proxy need to connect with the server with both valid hostname and port number, when it could not make the connection, it would send 400 response to the client.
GET http://www.baidu.com/ HTTP/1.1
Host: www.baidu.com: 77777777
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0)

8.Port number should be 0 – 65535 and can not also be negative.
GET http://www.baidu.com/ HTTP/1.1
Host: www.baidu.com: -888
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0)

9.Port number should be a number and cannot be any other invalid message.
GET http://www.baidu.com/ HTTP/1.1
Host: www.baidu.com: hhiisia
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0)

10. Header cannot be empty, which means either the client does not send any information to the proxy, or the proxy could not receive the request header, it would then send a error message back to inform the client, unless the client close the connection.

11.The port number with connect should be right: cannot connect to the remote server. Since https default port number is 443, it is possible that other host with https would have other port number which would be provided in the header, and the port number must by right, other wise the proxy could not make the connection and send a 400 response to the client.
CONNECT apis.google.com: 7658768 HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Proxy-Connection: keep-alive
Connection: keep-alive
Host: apis.google.com:443

12.The Content-Length with POST should not be empty. The Post header must be provided content length thus the proxy would know how many bytes it need to receive from the client and send back to the server, if content-length is not in the header, the proxy would directly send 400 response to the client.
POST http://ocsp.pki.goog/GTSGIAG3 HTTP/1.1
Host: ocsp.pki.goog
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Content-Type: application/ocsp-request
Connection: keep-alive

13.The format should be right (should end with "\r\n\r\n"). Since the header is ended with "\r\n\r\n", the proxy receives the header according to the end character, otherwise it would not stop receive the message until the client close the connection, and if the proxy received incomplete message, it would report the error.
What our proxy can not handle properly it that it would continually receive request if it not find the "\r\n\r\n", when the bad request happen, it hard to detect and block on the receiving.

- Good cases handled by the proxy:

1.Implement GET request

GET http://www.baidu.com/img/bg-1.0.0.gif HTTP/1.1
Host: www.baidu.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.baidu.com/
Cookie: BAIDUID=0D2B2CA2BCCCE52554C02AB2F9A5BA3E:FG=1
DNT: 1
Connection: keep-alive

2.Implement CONNECT

CONNECT video-api.yql.yahoo.com:443 HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Proxy-Connection: keep-alive
Connection: keep-alive
Host: video-api.yql.yahoo.com:443

3.Impletment of POST

POST http://ocsp.pki.goog/GTSGIAG3 HTTP/1.1
Host: ocsp.pki.goog
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Length: 75
Content-Type: application/ocsp-request
Connection: keep-alive

4. Handle response with content-length. The proxy would parse the header and find the content length of the response, and it would keep receiving the bytes until it get the length number, then it would send what it gets to the client.

5. Can handle chunked response, it would parse the transfer-coding and if it is non-identity, it would regard the all the transfer-encoding as chuncked. It has no idea how many bytes to receive until it got one chunk which is "0\r\n\r\n" , and that means the end of the response.

HTTP/1.1 200 OK
Cache-Control: no-cache, no-store
Pragma: no-cache
Transfer-Encoding: chunked
Content-Type: image/jpeg; charset=utf-8
Expires: -1
Server: Microsoft-IIS/10.0
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Arr-Disable-Session-Affinity: True
Date: Fri, 02 Mar 2018 19:56:42 GMT

6. Can handle responses with "no–store": not store into cache. The proxy would parse this cache-control field and to determine whether to cache the response, and if It is no-store, it would not cache this response.

HTTP/1.1 200 OK
Cache-Control: no-cache, no-store
Pragma: no-cache
Transfer-Encoding: chunked
Content-Type: image/jpeg; charset=utf-8
Expires: -1

7. Can handle responses with "no–cache": need revalidation. The proxy would mark this response as need revalidation, thus the next time it receives the same request, it would get the Etag from the cached response and append it to the new request, then sends this request to the server to revalidate the response which has been cached.

HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Transfer-Encoding: chunked
Content-Type: image/jpeg; charset=utf-8
Expires: -1

8.Can handle response with "must-revalidation": need revalidation. The proxy would also append the Etag to the request to do the revalidation for the request.

9. Can handle normal response
HTTP/1.1 200 OK
Date: Fri, 02 Mar 2018 20:10:15 GMT
Server: Apache
Last-Modified: Sun, 24 Aug 2014 21:32:59 GMT
ETag: "121d834d5-1d6-50166d07258c0"
Accept-Ranges: bytes
Content-Length: 470
Connection: close
Content-Type: text/css

10. Can handle responses with ETag: send request with ETag if need validation
HTTP/1.1 200 OK
Date: Fri, 02 Mar 2018 20:10:15 GMT
Server: Apache
Last-Modified: Sun, 24 Aug 2014 21:32:59 GMT
ETag: "121d834d5-1d6-50166d07258c0"
Accept-Ranges: bytes
Content-Length: 470
Connection: close
Content-Type: text/css

11. can handle responses with "s-maxage": calculate expire time
HTTP/1.1 200 OK

Date: Fri, 02 Mar 2018 20:10:15 GMT
Server: Apache
s-maxage = 1777
Last-Modified: Sun, 24 Aug 2014 21:32:59 GMT

12. Can handle responses with "max-age": calculate expire time. The proxy would plus the current time with the max-age and set this to the response's expire time. When it receive the same request, it would determine whether this response is expired.
HTTP/1.1 200 OK
Date: Fri, 02 Mar 2018 20:10:15 GMT
Server: Apache
max-age = 1
Last-Modified: Sun, 24 Aug 2014 21:32:59 GMT

13. Can handle responses with "Last-Modified": calculate expire time together with current time (expire time = current time + (current time – last modified time) * 0.1)
HTTP/1.1 200 OK
Date: Fri, 02 Mar 2018 20:10:15 GMT
Server: Apache
Last-Modified: Sun, 24 Aug 2014 21:32:59 GMT

14. Can handle request with a different port number
GET http://152.3.64.14:8080/library/skin/default/neo-default.css HTTP/1.1
Host: 152.3.64.14:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/css,*/*;q=0.1
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://152.3.64.14:8080/library/content/gateway/about.html
Cookie: JSESSIONID=168983c4-e8f4-4299-94a8-6c90900212f9.vcm-3385.vm.duke.edu
Connection: keep-alive

•Case that can not handle

1. Since the proxy would not change the header and send what it get to the server, some remote server would not parse the header thus the server would get a wrong ulr and could not find the page.

2. If the response header not include transfer-encoding, and content length, the proxy would keep receiving the message until the server close the connection, the proxy sometime would be blocked.

3.Since the cache policy we implement, we only delete the response from the cache when cache is full, and we would delete the first one on the cache list, which would not ensure the one we delete is expired and may keep other expired response in the cache. But one benefit is that, due to the concurrency, when each thread need to change the cache, it would require the lock, and delete the first one is quick, rather than pass through cache and find the expired one and delete it.