HTTP Redirection to IP Addresses

Bin Ni (nibin@quantil.com) 7/22/2019

Background

Many website owners and CDN providers want the web servers or proxy servers to participate in load balancing. For example, when a server is overloaded, redirect the requests to some other servers; or based on the geo information of the client IP address, send the client to a server that can provide better performance. Compared to the DNS-based load balancing, this method is more real time and more accurate. It can also take the URL, header fields or other components in the HTTP request into consideration.

What they can use today are the 30X redirections or the “Alt-Svc” header, which have at least the following limitations:
1. 30X redirections break the cookie, if you redirect to another hostname or IP address.
2. 30X redirections break HTTPS if you redirect to another IP address.
3. The “Alt-Svc” header is only advisory for future requests. It can’t be used to force the client to retry the current request with the alternate server (like the 30X responses do).

We need another way to tell the client/browser to retry the current request with some other IP address and keep everything else the same. This kind of redirection will mostly benefit large object downloads that could take minutes and the overhead of a few more RTT is negligible.

Proposal

A simple addition to the HTTP protocol is proposed here:
A new status code: 312 with 3 new response header fields: "x-redirect-ipv4", "x-redirect-ipv6" and “x-redirect-context”. The first two fields are for passing the redirected IP(s) to the client. The 3rd one is to ask the client to forward some information to the redirected server in the next request.

When a server wants to redirect the client to a different IP for whatever reason, it should respond with the proposed status code. The response header **must** contain at least one of "x-redirect-ipv4" and "x-redirect-ipv6" with the redirected IP address(es) specified. When the client receives this response, it should pick one of the returned IPs and make exactly the same HTTP(S) request again to that IP. The server **may** also return an “x-redirect-context” header field in the 312 response. Its value should be a name:value pair the client **may** add to the request header to the redirected server. This can serve as a hint to the redirected server to help it decide how to process the request, for example, to avoid too many levels of redirection. The servers can also use this context to pass an unique ID to each redirected server to help future troubleshooting.

Examples

This proposal should work with both HTTP/1.1 and HTTP/2.

```
Request #1. IP address obtained through regular DNS.
```

```
GET /dir/file.txt HTTP/2
host: www.mydomain.com
...
```

```
Response to request #1:
```

```
HTTP/2 312
x-redirect-ipv4: 12.34.56.78
x-redirect-ipv6: 2001:0db8:85a3:0000:0000:8a2e:0370:7334
x-redirect-context: req-unique-id:234756ad0ef
...
```

```
Request #2, made to 12.34.56.78
```

```
GET /dir/file.txt HTTP/2
host: www.mydomain.com
req-unique-id: 234756ad0ef
...
```

```
Response to request #2:
```

```
HTTP/2 200
...
```

The “x-redirect-ipv4” and “x-redirect-ipv6” headers can return a comma separated list of IPs:

```
Request #1. IP address obtained through regular DNS.
```

```
GET /dir/file.txt HTTP/1.1
host: www.mydomain.com
...
```

```
Response to request #1:
```

```
HTTP/1.1 312
x-redirect-ipv4: 12.34.56.78, 11.22.33.44
x-redirect-ipv6: 2001:0db8:85a3:0000:0000:8a2e:0370:7334
x-redirect-context: req-redirect-cnt:1
...
```

```
Request #2, made to 11.22.33.44
```

```
GET /dir/file.txt HTTP/1.1
host: www.mydomain.com
req-redirect-cnt: 1
...
```

```
Response to request #2:
```

```
HTTP/1.1 200
...
```

Client Considerations

Here are a few recommendations for clients to robustly handle this new reponse.

1. If a status code 312 is received without the “x-redirect-ipv4” or “x-redirect-ipv6” header fields, the client can retry the existing IP a few times before displaying errors to the user.
2. Similar to the handling of 30X redirections, the client can maintain a counter for the number of times a request has been redirected and abort the request when it reaches some upper limit.

Or, Extending RFC 7838

Another way to do this is to extend RFC 7838 to introduce this status code 312 to force the client to retry the current request with the alternate server. This will of course make the “alt-svc” header no longer “advisory”, but only when with this special status code.