
Munich Personal RePEc Archive

Opportunities for Using the Protocol HTTP/2 to Reduce Lag When Loading Web Applications

Petrov, Pavel and Petrova, Stefka

-

October 2016

Online at https://mpra.ub.uni-muenchen.de/75284/
MPRA Paper No. 75284, posted 27 Nov 2016 09:14 UTC

Възможности за използване на протокол HTTP/2 за намаляване на лага при зареждане на уеб приложения

Павел Петров
Стефка Петрова

Opportunities for Using the Protocol HTTP/2 to Reduce Lag When Loading Web Applications

Pavel Petrov
Stefka Petrova

Abstract

This article examines the evolution of the protocol HTTP - from version 0.9 through 1.0, and 1.1 to version 2. The factors affecting latency and lag when loading web pages are discussed in details. The research concludes that nowadays network bandwidth plays an increasingly small role about the lag. The results of empirical research how much the lag is reduced when using HTTP/2 while loading an average by volume and by content web page are presented. In the tests are used different versions of the protocol HTTP - HTTP/1.0, HTTP/1.1, HTTPS/1.1 and HTTPS/2. An experimental software is created and an external program to simulate network latency is used.

Keywords: HTTP, HTTP/1.0, HTTP/1.1, HTTP/2, Hypertext Transfer Protocol, web applications, latency, lag, web server, web client, Apache, Google Chrome, Mozilla Firefox

Въведение

Както е добре известно, Световната мрежа, предоставяща достъп до свързани помежду си документи разположени на различни сървъри свързани в Интернет, използва като основен комуникационен протокол HTTP (HyperText Transfer Protocol). Този протокол осигурява връзката между информационните ресурси посредством хипервръзки. Световната мрежа (World Wide Web) предизвика истинска революция в информационните технологии и бум в развитието и използването на Интернет. Много често днес когато се говори за Интернет се има в предвид именно Световната мрежа, тъй като по-голямата част от ресурсите се използват именно чрез нея.

Протоколът HTTP работи на приложно ниво по модела Open Systems Interconnection (OSI) на ISO[1] и обменът на данни между клиента и сървъра се извършва по класическата схема заявка-отговор. Оригиналният протокол HTTP, познат като **HTTP/0.9**[2], е максимално опростен и е бил подходящ за технологичното ниво през 90-те години на миналия век[3]. При него заявката, която клиентът изпраща към сървъра, се е само от един ред, състоящ от думата GET, един интервал и адреса на документа. Отговорът, който връща сървърът на клиента е самото съдържание на искания документ. Обменяните съобщения между двете страни изглеждат по следния начин:

Заявка:
GET / [\n]
Отговор:
<html>
<head><title>Sample Title</title></head>
<body>Sample Web Page Content</body>

[1] ISO/IEC 7498-1:1994, Information technology -- Open Systems Interconnection -- Basic Reference Model: The Basic Model, <http://www.ecma-international.org/activities/Communications/TG11/s020269e.pdf>
[2] Tim BL, The Original HTTP as defined in 1991, <https://www.w3.org/Protocols/HTTP/AsImplemented.html>
[3] Към момента сървъри, които поддържат виртуални хостове базирани на имена, не трябва да поддържат HTTP/0.9, съгласно последния стандарт за HTTP/1.1: R. Fielding, J. Reschke, RFC 7230, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", 2014, <https://tools.ietf.org/rfc/rfc7230.txt>. На практика повечето уеб сървъри в днешно време не го поддържат.

</html>

Мрежовата връзка между клиента и сървъра се прекъсва след всеки отговор, което налага при изпращане на следващи заявки да се отваря нова мрежова връзка.

В следващата версия на протокола - **HTTP/1.0**[4], създадена през 1993 г. и стандартизирана от IETF през 1996 г., заявката се усложнява, като към нея се добавят нови части - първият ред се запазва, той става така наречения начален ред, и се добавят заглавна част и тяло. В началния ред, освен ключовата дума GET, вече може да се използват и други думи, указващи начина, по който сървъра да обработи заявката. Тези ключови думи са известни още като методи и в HTTP 1.0 са GET, POST и HEAD. Допълнително, след адреса на документа се указва протоколът и версията му.

Методът GET си остава основното средство за изискване на документи, но не съдържа тяло. За разлика от него методът POST има тяло и в него могат да се предадат големи по обем данни към сървъра. Методът HEAD е сходен на GET, с изключение на това, че сървърът не трябва да връща тяло на отговора, а само заглавната част, т.е. получава се служебна информация за документа, но без всъщност той да се изпраща, което най-често се използва при проверка за валидност на хипервръзки, достъпност на ресурса или кога той последно е бил променян.

От страна на сървъра отговорът също се усложнява и се състои вече не само от искания документ, а преди това се прибавя начален ред (status line), заглавна част (header) и накрая следва тяло (body), в което всъщност е съдържанието на документа. Началният ред започва с данни за протокола и версията му, следвано от код за статуса - едно трицифрено число, и евентуално съобщение за статуса. Заглавната част съдържа служебни данни по отношение на информацията на данните, съдържащи се в тялото на съобщението. Мрежовата връзка между клиента и сървъра отново се прекъсва след всеки отговор.

За разлика от HTTP/0.9, в който липсва заглавна част, то в HTTP/1.0 тя играе важна роля, тъй като в отговора там се съдържа много полезна информация: посочва се какъв е типа на данните, изпращани в тялото (поле Content-Type), какъв е техния размер (поле Content-Length), как са кодирани (поле Content-Encoding), кога последно документът е бил променян (поле Last-Modified) и др. Обменяните съобщения между двете страни изглеждат по следния начин:

Заявка:
GET /img.jpg HTTP/1.0
Referer: /index.html [\n\n]
Отговор:
HTTP/1.0 404 Not Found
Content-Type: text/html
Content-Length: 17 [\n\n]
Object not found.

Следващата версия на протокола HTTP - версия **HTTP/1.1** се създава през 1997 г. (RFC 2068), допълва се през 1999 г. (RFC 2616)[5] и претърпява последни промени през 2014 г. (RFC 7230 - RFC 7240), като документацията се разделя в множество отделни описания за това какво представлява HTTP/1.1 и как клиентът и сървърът трябва да работят по него. Най-характерните моменти са: заявката и отговорът запазват основната си структура от 3 части; в заглавната част на заявката се добавя задължително поле "Host:" с цел да се реализират заявки към виртуални хостове; осигурява се възможност за поддържане отворена мрежовата връзка (поле Connection: Keep-Alive) за продължителни периоди от време и използването й

[4] T. Berners-Lee, R. Fielding, H. Frystyk, RFC 1945, "Hypertext Transfer Protocol - HTTP/1.0", 1996, <http://www.rfc-editor.org/rfc/rfc1945.txt>

[5] R. Fielding, J. Gettys, J. Mogul et al., RFC 2616, Hypertext Transfer Protocol - HTTP/1.1, 1999, <https://tools.ietf.org/rfc/rfc2616.txt>

за други заявки-отговори; има възможност за изпращане на част от файл; броят на кодовете за статуса се увеличава значително; възможните полета в заглавните части и на заявката и на отговора също се увеличават значително и др. Обменяните съобщения между двете страни изглеждат по следния начин:

Заявка:

GET /img.jpg HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:47.0) Gecko/20100101 Firefox/47.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: bg,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive [\n\n]

Отговор:

HTTP/1.1 404 Not Found
Accept-Ranges: bytes
Cache-Control: max-age=604800
Content-Encoding: gzip
Content-Length: 606
Content-Type: text/html
Date: Sun, 23 Oct 2016 08:51:37 GMT
Etag: "359670651+gzip"
Expires: Sun, 30 Oct 2016 08:51:37 GMT
Last-Modified: Fri, 09 Aug 2013 23:54:35 GMT
Server: ECS (ewr/1445)
Vary: Accept-Encoding
X-Cache: HIT
x-ec-custom-error: 1 [\n\n]
[следват двоични данни в тялото на отговора, които тук не са представени]

Както се вижда, протоколът HTTP до версия 1.1 преминава през еволюционно развитие, имащо за цел да се обогатят неговите възможности като се запази до известна степен съвместимост с предходните версии, доколкото това е възможно. Тъй като протокола е текстов, лесно може да се проследява съдържанието на служебните части - заглавните части на заявката и отговора. Тялото, което в повечето случаи представлява същността и съдържанието на обменените данни, е възможно да бъде компресирано. За съжаление това най-голямо предимство - текстовия характер на протокола, се оказва и неговия най-голям недостатък от гледна точка намаляване на обема на предаваните данни.

Фактори, влияещи върху латенцията и лага при зареждане на уеб страници

Латенцията е един общ термин с широко значение в информатиката, означаващо времезакъснението при изпълнение на дадена операция. По отношение на компютърните мрежи латенцията включва в себе си времезабавянето, което възниква по време на предаване на данните, тяхното обработване и представяне за крайния потребител. Често латенцията още се нарича и лаг, когато се отнася до цялостното субективно усещане за времезабавяне при интерактивна работа със система, работеща в режим онлайн.

В зависимост от същността на уеб приложението, под латенция може да се разбира времезабавянето само в едната посока - от момента на генериране на съобщението при източника до момента на неговото получаване, но също така под латенция може да се разбира времезабавянето и в двете посоки на комуникационната връзка - в този случай включва времето от изпращане на първоначалната заявка, нейната обработка от отсрещната страна, генериране на отговор и последващото му получаване от първата страна. При втория

вариант латенцията е около два пъти по-голяма спрямо първия вариант.

За латенцията допринасят множество независещи един от друг фактори и тя може да се представи като сума от времето за подготовка на съобщението от страна на клиента плюс времето необходимо на сигнала да пропътува през физическата медия (оптични кабели, радиовълни, електрически сигнали и прочие) плюс времето за обработка на сигнала в междинните устройства плюс времето необходимо за обработка на съобщението от страна на сървъра, генериране на отговор, изпращането му обратно на клиента и отново плюс времето за пропътуване на сигнала във физическата медия плюс времето за обработка в междинните устройства плюс времето необходимо на клиента да обработи и евентуално да визуализира отговора.

В компютърна мрежа, в която има относително малък трафик, който се явява част от максималната пропускателна способност на средата и ако приемем, че настройките на междинните устройства и настройките от страна на клиента и сървъра са извършени правилно (оптимално), то в този случай определящо за големината на латенцията има скоростта на разпространение на сигнала, която е ограничена от физичните закони - скоростта на разпространение на светлината във вакуум (около 300000 km/s). В среда на локална мрежа, където клиентът, сървърът и междинните устройства са сравнително близко разположени едни до други при сравнително прости заявки обикновено времезабавянето може да спадне до 1-2 ms. В среда на глобална мрежа, където клиентът и сървърът са сравнително на големи разстояния един от друг, при същия тип прости заявки времезабавенето може да надхвърли 500 ms, ако например комуникационната връзка се извършва с помощта на спътник на геостационарна орбита. В този случай сигналът трябва да измине 4 пъти разстоянието от Земята до спътника на геостационарна орбита (около 36 хил. км. над Екватора), при което времезабавянето от междинните устройства става пренебрежимо по-малко.

Не по този начин стои въпросът, когато в компютърната мрежа има относително голям трафик, който се доближава до максималната пропускателна способност на средата. В подобни случаи междинните устройства започват да буферират пакетите или да търсят други маршрути, при което част от данните може да се загубят, което да наложи тяхното повторно изпращане. Тогава основен дял за латенцията започва да играе друг фактор - времето за обработка в междинните устройства.

Освен разстоянието и натоварването на междинните мрежови устройства, голямо влияние за латенцията оказват също така клиентът или сървърът. Ако към сървъра се отправят заявки, които изискват един по-продължителен период от време за тяхната обработка или ако от страна на сървъра се изисква повече време за интерпретиране на отговора, то това също може да доведе до увеличаване на латенцията, а от там и на лага.

Особено неблагоприятно влияние за лага, като едно субективно усещане за времезабавянето, се оказва загубата на пакети. Тази загуба може да се получи поради най-различни причини и може да няма пряко отношение към латенцията, но независимо поради какви причини един пакет с данни е загубен, то неговото повторно изискване, изпращане и получаване в крайна сметка оказва влияние върху завършеността на конкретната операция и субективното усещане на потребителя за работа в режим онлайн. В подобни случаи, когато се елиминира причината за загуба на пакети, лагът се намалява.

Можем да обобщим, че определящи фактори за лага са: пропускателната способност на компютърната мрежа, нейното моментно натоварване, разстоянието между двете страни на комуникационната връзка, наличието на загуби на пакети, натоварването и възможността за обработка на данни от страна на сървъра и от страна на клиента.

През последните години пропускателната способност играе все по-малка роля за намаляване на лага. Данните от изследвания в световен мащаб показват, че средната скорост на една връзка е от порядъка на 6,3 Mbps (виж табл.1), което означава, че много потребители

имат достъп до високоскоростна Интернет връзка и все повече пропускателната способност на каналите за връзка вече няма да се явява ограничаващ фактор при зареждането на уеб страници. Съответно други фактори ще имат по-голямо влияние върху намаляването на латенцията и лага.

На практика, след като един потребител е осигурен с повече от 4 Mbps скорост на връзка по-нататъчно увеличаване на скоростта на достъп ще доведе до минимални изменения по отношение на латенцията и лага. Интересен факт е, че България е сред водещите страни по отношение на дела на Интернет връзките със скорост над 4 Mbps и над 10 Mbps (виж табл.2), но за съжаление ръстът на годишна база е отрицателен.

Таблица 1. Средна скорост на Интернет връзката по държави към първото тримесечие на 2016 г.[6]

	Country/Region	Q1 2016 Avg. Mbps	QoQ Change	YoY Change
–	Global	6.3	12%	23%
1	South Korea	29.0	8.6%	24%
2	Norway	21.3	14%	68%
3	Sweden	20.6	8.3%	32%
4	Hong Kong	19.9	19%	19%
5	Switzerland	18.7	12%	25%
6	Latvia	18.3	9.8%	33%
7	Japan	18.2	4.6%	20%
8	Netherlands	17.9	5.5%	20%
9	Czech Republic	17.8	12%	31%
10	Finland	17.7	6.9%	30%

Таблица 2. Дял на Интернет връзките със скорост над 4 Mbps и над 10 Mbps по държави към първото тримесечие на 2016 г.[7]

	Country/Region	% Above 4 Mbps	QoQ Change	YoY Change
–	Global	73%	5.4%	16%
1	South Korea	97%	0.4%	1.4%
2	Malta	97%	1.4%	4.5%
3	Isle Of Man	97%	1.8%	4.6%
4	Bulgaria	97%	2.4%	-0.1%
5	Thailand	96%	0.8%	15%
6	Switzerland	96%	2.3%	2.8%
7	Denmark	96%	1.6%	2.3%
8	Israel	96%	1.9%	2.9%
9	Netherlands	95%	-0.2%	3.0%
10	Romania	95%	4.2%	5.7%

	Country/Region	% Above 10 Mbps	QoQ Change	YoY Change
–	Global	35%	10%	34%
1	South Korea	84%	4.0%	9.5%
2	Switzerland	68%	9.8%	16%
3	Netherlands	67%	4.2%	14%
4	Belgium	66%	8.9%	40%
5	Bulgaria	66%	22%	20%
6	Hong Kong	66%	7.8%	10%
7	Norway	65%	5.8%	69%
8	Iceland	65%	14%	103%
9	Japan	65%	3.9%	17%
10	Singapore	65%	9.9%	40%

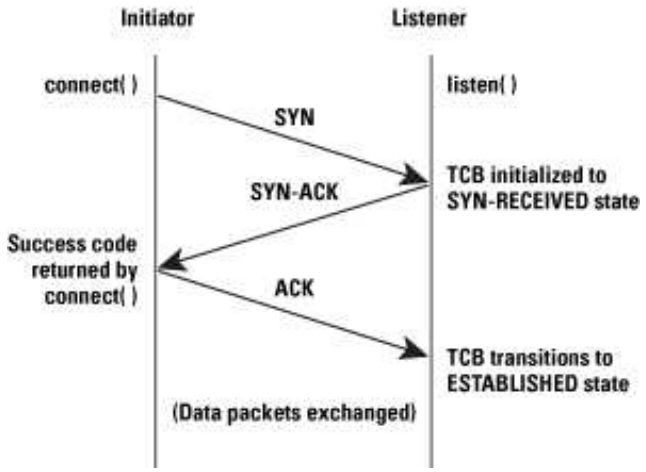
[6] QoQ - Quarter-over-Quarter, YoY - Year-over-year - изменение на показателя в проценти на тримесечна и годишна база. Източник: Akamai, The State of the Internet / Q1 2016 Report, V.9, N.1, p.12. <https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/akamai-state-of-the-internet-report-q1-2016.pdf>

[7] Пак там, pp.13-14.

За да се намали влиянието на фактора "разстояние между двете страни на комуникационната връзка" върху латенцията, през последните години широко разпространение получиха мрежите за дистрибуция на съдържание (Content Delivery Network, CDN), чиято цел е посредством географско разпределение на мрежовата инфраструктура да пренесат ресурсите по-близо до крайния потребител и така да се намали латенцията при зареждане на отделните ресурси.

Съвременните уеб страници се различават съществено от тези отпреди 20 г., когато се създаваше протоколът HTTP/1.1. В онзи период пропускателната способност на компютърните мрежи и най-вече на така наречения фактор "последен километър" или "последна миля" е най-важният лимитиращ фактор за бързото зареждане на една уеб страница. В днешно време това не е така. Днес за пълното визуализиране на една уеб страница в повечето случаи се налага да се свалят множество допълнителни файлове, при това много често от различни хостове. Този подход на работа не може да оползотвори пълния капацитет на TCP/IP връзките, които са оптимизирани за поддържането на мрежовата връзка за по-дълъг период от време. Основен лимитиращ фактор се оказва самата същност на HTTP, който не използва ефективно ресурсите на TCP/IP.

Започването на множество и кратки по времетраене мрежови връзки увеличава значително латенцията, тъй като т.нар. "три пасово ръкостискане" преди отваряне на една TCP/IP връзка, изисква предварително двете страни да си разменят съобщения (фиг. 1), чрез които двете страни да договорят параметрите на мрежовата връзка.



Фигура 1. Нормално "три пасово ръкостискане" при осъществяване на TCP/IP връзка.[8]

Основният проблем на протоколите HTTP отпреди вер. 2 е, че те са синхронни. При тях се изисква клиентът да изчаква сървърът да върне целия отговор, след което да изпрати нова заявка. При този начин на работа, ако сървърът по-бавно генерира някакъв ресурс, то това на практика блокира цялата последваща комуникация. За преодоляване на този проблем повечето браузъри отварят едновременно няколко мрежови връзки към сървъра, с помощта на които могат да получават няколко ресурса едновременно (например Google Chrome отваря едновременно 6 мрежови връзки). Това ограничение до няколко връзки е на базата на един домейн и поради тази причина се прилага тактиката част от ресурсите да се разполагат на други домейни с цел да се позволи да се използват повече мрежови връзки (това е т.нар.

[8] SYN - SYNchronize; SYN-ACK - SYNchronize-ACKnowledgement; ACK - ACKnowledge.. Източник: Wesley M. Eddy, Defenses Against TCP SYN Flooding Attacks, The Internet Protocol Journal - v.9 n.4/2006, <http://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-34/syn-flooding-attacks.html>

подход за "domain sharding"). Друга тактика е малките файлове от един и същи тип да се обединят в един голям файл, така че да се минимизира изчакването, получаващо се в резултат на множество последователни двойки заявка-отговори. Например малките изображения се обединяват в така наречените спрайтове ("CSS image sprites").

Протоколът HTTP версия 2 се опитва да реши проблемите, свързани с латенцията, чрез оптимизация на начина по който се заявяват и изпращат ресурсите. Той се различава съществено от протоколите версии 0.9/1.0/1.1 - той не е синхронен, а асинхронен; не е текстов, а двоичен; използва само една TCP/IP връзка за домейн, чрез която се извършва мултиплексиране, т.е. предаване на множество потоци данни едновременно. Последното се реализира като всяка двойка заявка-отговор се асоциира със свой собствен поток и съответно данните, които трябва да се изпратят, се разделят в отделни фреймове[9]. Фреймовете се асоциират с определени потоци и по този начин през една мрежова връзка може да се предават множество потоци от данни. Потоците са относително независими едни от други, така че блокирането на някой отговор или заявка не пречи на работата на останалите потоци. По този начин множество заявки-отговори могат да бъдат изпълнявани едновременно и потоците могат да бъдат приоритизирани. Мултиплексирането намалява необходимостта да се разполагат ресурсите на различни домейни (domain sharding) или да се използват спрайтове, но не премахва изцяло тази необходимост.

Протоколът HTTP/2 поддържа възможност за компресиране на заглавните части. Това не се извършва чрез класическите алгоритми за компресиране на данни, а чрез организационна техника, използването на която осигурява да няма повторно изпращане на полета от заглавните части, които вече са били изпратени. За целта и клиентът, и сървърът поддържат таблици с изпратените и съответно получените полета в заглавните части, както и техните стойности. При първата двойка заявка-отговор се изпраща пълната заглавна част, а при последващи заявки-отговори дублиращите се полета се пропускат. Икономията достига средно до 0,5KB при някой заявки-отговори и това може да окаже много ефективен механизъм за намаляване на размера на заглавната част.

Друга важна особеност на HTTP/2 е възможността сървърът да изпраща ресурси, които не са били изрично заявени от клиента (server push). Тези ресурси се кешират от страна на клиента за бъдеща употреба. Сървърът може да започне да изпраща тези ресурси непосредствено след като връзката е била установена, без дори да чака клиентът да изпрати заявка. При този начин на работа е възможно да се увеличи ненужно мрежовия трафик, но като цяло уеб страниците ще се зареждат по-бързо.

Изследване на намаляването на лага при използване на HTTP/2

Съгласно данни от httparchive.org за първите 500 хил. най-посещавани сайта за период от 1 година към 15.10.2016 (съгласно Alexa rank), за пълното зареждане и визуализиране на уеб страница средно са необходими около 102 заявки за ресурси, а средния обем на напълно заредената уеб страница е около 2,3MB[10]. От тях за HTML файлове средно се падат 10,1 заявки и 52KB, за JavaScript файлове - 23 заявки и 408KB, за CSS файлове - 7,1 заявки и 76KB, за изображения - 54 заявки и 1,6MB, за флаш - 0,4 заявки и 25KB, за шрифтове - 2,6 заявки и 84KB, за други ресурси - 4,1 заявки и 14KB.

С цел емпирично изследване на намаляването на лага при зареждане на уеб страници чрез използване на различни протоколи HTTP - версии 1.0, 1.1 и 2, създадохме опитна установка, чрез която да установим лага при различни стойности на мрежова латенция. Важна особеност при използване на HTTP/2 е, че съвременните браузъри поддържат версия 2

[9] HTTP/2 предвижда различни видове фреймове за различни цели например HEADERS, DATA, GOAWAY, PRIORITY, SETTINGS, PUSH_PROMISE и др. M. Belshe, R. Peon, M. Thomson, RFC 7540, Hypertext Transfer Protocol Version 2 (HTTP/2), <https://tools.ietf.org/rfc/rfc7540.txt>

[10] Trends, <http://httparchive.org/trends.php?s=All&minlabel=Oct+15+2015&maxlabel=Oct+15+2016>

само, ако връзката е криптирана, въпреки че това не се изисква изрично по стандарта. Затова реално използвахме HTTPS/2 и допълнително добавихме изследване за HTTPS/1.1.

При проведените тестове са използвани компоненти със следните характеристики: операционна система - Windows 7, 32 bits; процесор - i5-2430M; оперативна памет - 4GB; уеб сървър - Apache/2.4.18; уеб браузъри - Google Chrome 54.0.2840 и Mozilla Firefox 47.0.1 (най-актуални версии към м.октомври 2016).

Тъй като клиентът и сървърът физически са разположени на една машина, за симулиране на мрежова латенция използвахме допълнителен софтуер - clumsy 0.2[11]. С негова помощ се задържат за определен период от време изпращаните и пристигащи TCP/IP пакети и по този начин може да се симулира териториална отдалеченост на двете страни на мрежовата връзка. Разбира се, при използване на реална глобална мрежа, латенцията би била друга и би варирала постоянно, тъй като маршрутът по който се движат пакетите може динамично да се променя. Променящият се маршрут оказва влияние както на броя на устройствата, през които преминават пакетите в двете посоки, така и на разстоянието, което изминава сигнала, движейки се със скоростта на светлината. Дори и маршрута, по който се движат пакетите, да не се променя, е възможно отделни мрежови устройства да са подложени на пиково натоварване, което да забавя тяхната работа. Очевидно е, че при симулацията на латенция тези фактори са премахнати, т.е. приемаме, че между клиента и сървъра няма други устройства и съответно латенцията се дължи единствено на отдалечеността им.

При настройване на уеб сървъра за работа с HTTP/2 са следвани указанията от документацията на Apache[12], а именно - в конфигурациония файл httpd.conf е активирана директивата за зареждане на mod_http2, добавена е директива за превключване от вер. 1.1 към вер. 2:

LoadModule http2_module modules/mod_http2.so

Protocols h2 h2c http/1.1

Тъй като при работа се установи, че браузърът Mozilla Firefox не превключва автоматично към използване от по-ниска на по-висока версия, се наложи допълнително да се променят две директиви за работа с SSL. Директивите SSLCipherSuite и SSLProtocol бяха променени по следния начин:

SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-DSS-AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-SHA:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!3DES:!MD5:!PSK

SSLProtocol All -SSLv2 -SSLv3

При настройване на уеб сървъра за работа с HTTP/1.0, отново бяха следвани указанията от документацията на Apache[13] и бяха добавени директивите:

SetEnv downgrade-1.0

SetEnv force-response-1.0

Тези настройки бяха включвани и изключвани при необходимост, тъй като по подразбиране уеб сървърът използва вер. 1.1 и при HTTP и при HTTPS. При провеждане на

[11] clumsy 0.2, <http://jagt.github.io/clumsy/>

[12] Apache HTTP Server Version 2.4, HTTP/2 guide, <https://httpd.apache.org/docs/2.4/howto/http2.html>

[13] Apache HTTP Server Version 2.4, Environment Variables in Apache, Special Purpose Environment Variables, <http://httpd.apache.org/docs/2.4/env.html#special>

тестовете се спазваше следния подход: уеб сървърът се спираше; извършваха се съответните настройки и пак се стартираше; на програмата за симулиране на латенцията за задаваха съответни параметри; три пъти се зареждаше ресурса с цел "подгряване" на системата и след това в нови "инкогнито" прозорци се извършваше теста последователно в двата браузъра по три пъти, след което времето се осредняваше за всеки един от тях.

Програмният модул, който генерира уеб страница, която от своя страна изисква зареждането на допълнителни ресурси, се изпълнява от интерпретатор на PHP 7 и има следното съдържание:

```
<?php
$files = 102;
$size = 23; //KB
$latency = 0; // milliseconds
if(!isset($_GET['t'])) {
        print<<<EOT
<html>
<head>
<script>
var begin=0, end=0, delta = 0;
function print(s) {
        document.getElementById("txt").innerHTML += (Date.now()) + ":" + s + "<br />";
}
document.addEventListener("DOMContentLoaded", function(event) {
        begin = Date.now();
        print("DOMContentLoaded");
});
window.onload = function () {
                end = Date.now();
                print("load");
                print("delta="+(end-begin));
}
</script>
</head>
<body>
<div id="txt"></div>
EOT;
        usleep($latency*1000);
        echo "<script>print('START: $files files, $size KB');</script>\n";
        for($i=0; $i<$files; $i++) echo "<script src='?t=$i' async></script>\n";
        print<<<EOT
</body>
</html>
EOT;
} else {
        usleep($latency*1000);
        echo "print('$_GET[t]'); tmp = '" . str_repeat("1234567890", $size*100) . "';";
}
?>
```

В променливите $files и $size се задава съответно колко допълнителни ресурса трябва да се заредят за цялостното визуализиране на уеб страницата и какъв да бъде техния размер. Допълнително в променливата $latency може да се зададе времезабавяне, което да симулира

извършването на повече обработки от страна на сървъра при връщането на ресурса (например връзка с бази от данни) или неговото по-голямо натоварване от други клиенти.

При провеждането на тестовете сме задали стойности на тези променливи, които максимално близко да възпроизведат средната уеб страница към днешния момент, съгласно гореспоменатите статистически данни от изследванията на httparchive.org, а именно - уеб страница, която зарежда допълнително 102 ресурса с общ обем 2,3MB.

В таблици 3 и 4 е показан интервала от време в секунди от момента на зареждане на една средно статистическа уеб страницата до момента на зареждане на всички ресурси, необходими за визуализирането ѝ (лаг) при използване на различни протоколи и симулирано мрежово времезабавяне (латенция) при получаване и изпращане на TCP/IP пакетите.

Таблица 3. Лаг в зависимост от използвания протокол и симулирано мрежово времезабавяне при използване на браузър Google Chrome 54.0.2840

протокол	**симулирано мрежово времезабавяне**			
	0 ms	**10 ms**	**100 ms**	**250 ms**
HTTP/1.0	0,5 s	6,7 s	21,3 s	47,6 s
HTTP/1.1	0,4 s	1,9 s	6,3 s	13,1 s
HTTPS/1.1	0,5 s	3,3 s	9,9 s	23,1 s
HTTPS/2	0,3 s	7,6 s	24,6 s	57,5 s

Таблица 4. Лаг в зависимост от използвания протокол и симулирано мрежово времезабавяне при използване на браузър Mozilla Firefox 47.0.1

протокол	**симулирано мрежово времезабавяне**			
	0 ms	**10 ms**	**100 ms**	**250 ms**
HTTP/1.0	0,5 s	5,7 s	18,2 s	27,6 s
HTTP/1.1	0,6 s	1,1 s	4,9 s	9,8 s
HTTPS/1.1	0,8 s	2,3 s	10,7 s	23,2 s
HTTPS/2	0,2 s	8,4 s	25,2 s	58,7 s

Според нас, при представените данни от съществено значение не са конкретните абсолютни стойности, а по-скоро съотношенията между тях. В някои от тестовете по-добре се представяше единият браузър, а при други тестове по-добре се представяше втория и това също не е от съществено значение. По-интересните моменти в резултатите от проведените тестове са представени на черен фон. Учудващо впечатление прави браузърът Firefox когато работи с HTTP/1.0. По-бързото зареждане при симулирано мрежово времезабавяне от 250 ms се дължи на факта, че той отваря едновременно повече на брой мрежови връзки спрямо браузърът Chrome. Резултатите от изследванията показват подобряване, в смисъл намаляване на лага, при използване на HTTP/2 само при изключено симулиране на мрежово времезабавяне. В останалите случаи лагът се оказва дори по-голям, отколкото ако се използваше протокол HTTP/1.0. Забелязва се, че като цяло използването на HTTPS/1.1 увеличава лага и това е напълно естествено, тъй като за криптиране и декриптиране се използват допълнителни изчислителни ресурси. Това се явява едно допълнително междинно звено при комуникацията, което внася допълнително времезабавяне.

Възможно е програмата, която сме използвали за симулиране на мрежово времезабавяне да изкривява по някакъв неизвестен, все още за нас начин, резултатите от проведените тестове с HTTP/2. Необходими са допълнителни проучвания с други средства, за да се установи дали наистина съществува някакъв специфичен проблем между програмата clumsy 0.2 и използването на HTTP/2. При използването ѝ при тестване с другите протоколи не се забелязват съществени отклонения спрямо очакваните резултати. Единствено при

резултатите при работа с HTTP/2 получените данни рязко контрастират с нашите очаквания за ползите от използването му по отношение на намаляване на лага.

Заключение

Протоколът HTTP изминава дълъг път по време на своето развитие. Преди последната версия е правен опит да се обогатят неговите възможности, като се запази в максимална степен съвместимост с предходните версии, доколкото това е възможно. В новата версия е подходено радикално и тепърва предстои той да се доказва в практиката. Проведените от нас експерименти не можаха категорично да установят намаляване на лага при цялостно зареждане на средностатистическа уеб страница. Намаляване беше установено само при експерименти с изключена симулация за времезабавяне. При включване на симулацията резултатите рязко се отклоняват от очакванията ни и на този етап нямаме отговор на какво може да се дължи това.

Използвана литература

1. ISO/IEC 7498-1:1994, Information technology -- Open Systems Interconnection -- Basic Reference Model: The Basic Model, <http://www.ecma-international.org/activities/Communications/TG11/s020269e.pdf>
2. Tim BL, The Original HTTP as defined in 1991, <https://www.w3.org/Protocols/HTTP/AsImplemented.html>
3. HTTP/1.1: R. Fielding, J. Reschke, RFC 7230, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", 2014, <https://tools.ietf.org/rfc/rfc7230.txt>
4. T. Berners-Lee, R. Fielding, H. Frystyk, RFC 1945, "Hypertext Transfer Protocol - HTTP/1.0", 1996, <http://www.rfc-editor.org/rfc/rfc1945.txt>
5. R. Fielding, J. Gettys, J. Mogul et al., RFC 2616, Hypertext Transfer Protocol - HTTP/1.1, 1999, <https://tools.ietf.org/rfc/rfc2616.txt>
6. Akamai, The State of the Internet / Q1 2016 Report, V.9, N.1, p.12. <https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/akamai-state-of-the-internet-report-q1-2016.pdf>
7. Wesley M. Eddy, Defenses Against TCP SYN Flooding Attacks, The Internet Protocol Journal - v.9 n.4/2006, <http://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-34/syn-flooding-attacks.html>
8. M. Belshe, R. Peon, M. Thomson, RFC 7540, Hypertext Transfer Protocol Version 2 (HTTP/2), <https://tools.ietf.org/rfc/rfc7540.txt>
9. Trends, <http://httparchive.org/trends.php?s=All&minlabel=Oct+15+2015&maxlabel=Oct+15+2016>
10. clumsy 0.2, <http://jagt.github.io/clumsy/>
11. Apache HTTP Server Version 2.4, HTTP/2 guide, <https://httpd.apache.org/docs/2.4/howto/http2.html>
12. Apache HTTP Server Version 2.4, Environment Variables in Apache, Special Purpose Environment Variables, <http://httpd.apache.org/docs/2.4/env.html#special>

За контакти

доц. д-р Павел Петров
Икономически университет - Варна
petrov@ue-varna.bg

д-р Стефка Петрова
Икономически университет - Варна
s.petrova@ue-varna.bg