http2 explained

Daniel Stenberg

İçindekiler

http2'nin açıklaması

Bu doküman HTTP/2(RFC 7540) protokolünü, arkaplanını, konseptini ve mevcut uygulamalarının gelecekte neler getireceğini açıklayan ayrıntılı bir belgedir.

Bu proje https://daniel.haxx.se/http2/ adresinde yer almaktadır.

https://github.com/bagder/http2-explained dokümanın içeriğinin tamamı için kaynak kodunun adresidir.

KATKIDA BULUNMAK İÇİN

Katkıda bulunmak ve iyileştirmeler sunmak isteyen geliştiricilerden memnuniyet duyarım ve katkılarını desteklerim. Çekme isteklerini (pull requests) https://github.com/bagder/http2-explained/pulls adresinden iletiniz ayrıca https://github.com/bagder/http2-explained/issues adresinden sorun bildirebilir (issue) ya da daniel-http2@haxx.se adresine önerilerinizi elektronik posta gönderebilirsiniz!

/ Daniel Stenberg

1. Arkaplan

Bu belge http2'yi teknik açıdan ve protokol düzeyinde açıklamaktadır. Daniel'in Nisan 2014'de Stokholm'de yaptığı bir sunum ile başladı ve tüm detayları, isabetli açıklamaları ile birlikte tam bir dokümana dönüştü.

RFC 7540 son http2 şartnamesinin resmi adıdır ve 15 Mayıs 2015'de yayınlanmıştır(https://www.rfc-editor.org/rfc/rfc7540.txt).

Bu dokümanda bulunan tüm hatalar bana aittir ve kendi ihmalimin sonucudur. Lütfen güncellenen sürümlerde düzeltilmesi için hataları belirtin.

Protokolü açıklamak için geçerli bir teknik terim olan "HTTP/2" yerine okunabilirlik ve daha iyi bir akış yakalamak adına "http2" kelimesini tercih ettim.

1.1 Yazar

Benim adım Daniel Stenberg ve Mozilla'da çalışıyorum. Açık kaynak ve ağ ile 20 yıldan fazla bir süre sayısız projede çalıştım. Muhtemelen beni öncü curl ve libcurl geliştiricisi olarak biliyorsunuz. Birkaç yıl IETF HTTPbis çalışma grubunda yer aldım ve orada http 1.1 yeniliklerini takip ettim, aynı zamanda http2 standartlaştırma çalışmalarına dahil oldum.

Elektronik posta: daniel@haxx.se

Twitter: @bagder

Web: daniel.haxx.se

Blog: daniel.haxx.se/blog

1.2 Yardım!

Eğer bu dokümanda hatalar, eksiklikler ve bariz yalanlar bulursanız lütfen bu bölümlerin yenilenen sürümünü bana gönderin ve ben de bu sürümlerdeki hataları düzelteceğim. Yardımcı olan herkese tesekkür ederim. Bu belgeyi zamanla daha iyi hale getirmeyi umuyorum.

Bu doküman https://daniel.haxx.se/http2 adresinde mevcuttur.

1.3 Lisans

1.4 Doküman tarihçesi

Bu dokümanın ilk sürümü, 25 Nisan 2014 tarihinde yayınlandı. En son doküman sürümlerindeki büyük değişiklikler aşağıdadır.

Sürüm 1.13

- Ana sürüm Markdown sözdizimine dönüştürüldü
- 13: Daha fazla kaynak, güncel bağlantılar ve açıklamalar eklendi
- 12: Taslağına referans vererek QUIC açıklaması güncellendi
- 8.5: Yeni rakamlarla güncellendi
- 3.4: Ortalama artık 40 TCP bağlantısıdır
- 6.4: Teknik özelliklerin ne dediği güncellendi

Sürüm 1.12

- 1.1: HTTP/2 artık resmi bir RFC'de yer almaktadır.
- 6.5.1: HPACK RFC'ye bağlantı verildi.
- 9.1: http2 için Firefox 36+ yapılandırma ayarlarından bahsedildi
- 12.1: QUIC hakkında bölüm eklendi.

Sürüm 1.11

- Katkıda bulunan arkadaşlar tarafından birçok dil iyileştirmesi yapıldı
- 8.3.1: Nginx and Apache httpd spesifik aktivitelerinden bahsedildi.

Sürüm 1.10

- 1: Protokol tamam oldu.
- 4.1: 2014 yılından itibaren kullanılan üslup yenilendi
- Ön: Burada resim eklendi ve "http2'nin açıklaması" denildi, bağlantı düzenlendi
- 1.4: Doküman tarihçesi bölümü eklendi
- Birçok yazım ve dil bilgisi hatası düzeltildi
- 14: Hataları iletenler sayesinde teşekkürler bölümü eklendi
- 2.4:HTTP büyüme grafiği için daha iyi etiketler
- 6.3: Çoklama treninde vagon sıralaması düzeltildi
- 6.5.1: HPACK taslak-12

Sürüm 1.9

- HTTP/2 taslak-17 and HPACK taslak-11 güncellendi
- "10. Chromium'da http2" (== şimdi bir sayfa daha uzun) bölümü eklendi
- Birçok bölüm düzenlemesi
- Şimdi 30 uygulamada
- 8.5: Bazı güncel kullanım rakamları
- 8.3: internet explorer'a da atıf
- 8.3.1 "Eksik uygulamalar" bölümü eklendi
- 8.4.3: Ayrıca TLS'in başarı oranı artışından bahseder

2. HTTP'nin Bugünü

HTTP 1.1, İnternet'teki neredeyse her şey için kullanılan bir protokoldür. Bunun avantajlarından yararlanan protokollerde ve altyapıda büyük yatırımlar yapılmıştır, çünkü bugün kendiliğinden yeni şeyler yapmaktan çok, HTTP'nin üstünde çalışmak daha kolay olmaktadır.

2.1 HTTP 1.1 devasadır

HTTP oluşturulduğunda ve dünyaya yayıldığında, muhtemelen basit ve anlaşılır bir protokol olarak algılanıyordu, fakat zaman bunun yanlış olduğunu kanıtladı. RFC 1945'de HTTP 1.0, 1996'da yayınlanan 60 sayfalık bir beyannamedir. HTTP 1.1'i açıklayan RFC 2616, yalnızca 3 sene sonra 1999'da yayınlanmıştır ve önemli ölçüde artış göstererek 176 sayfaya yükselmiştir. Bununla birlikte, IETF(İnternet Mühendisliği Görev Grubu) bu beyannamenin güncellemesi üzerinde çalışırken, bu beyanname bölünmüş ve toplamda daha büyük sayfa sayısı ile altı dokümana dönüştürülmüş(RFC7230 ve ailesi ile sonuçlanır). Herhangi bir sayımla, HTTP 1.1 büyüktür ve sayısız ayrıntı, incelik ve en azından çok sayıda isteğe bağlı parça içermektedir.

2.2 Seçenekler dünyası

HTTP 1.1'in daha sonraki uzantılar için kullanılabilecek çok sayıda minik ayrıntı ve seçeneğe sahip olma özelliği, neredeyse hiçbir uygulamanın hiçbir zaman hiçbir yerde uygulayamayacağı bir yazılım ekosistemi geliştirmiştir ve "hiçbir şey" kavramının tam olarak ne olduğunu söylemek mümkün değildir. Bu başlangıçta az kullanılan özelliklerin, çok az sayıda uygulamanın yapıldığı ve özelliklerini uygulayanlardan çok az yararlanıldığı bir duruma neden oldu.

Daha sonraları, sunucu ve istemciler bu tür özelliklerin kullanımını arttırmaya başladığında, bu "birlikte çalışabilirlik" sorununa neden oldu. HTTP boruhattı, böyle bir özelliğin temel bir örneğidir.

2.3 TCP’nin yetersiz kullanımı

HTTP 1.1, TCP'nin sunduğu tüm gücü ve performanstan tam anlamıyla yararlanan bir zorluk yaşıyor. HTTP istemcileri ve tarayıcılar, sayfa yükleme sürelerini azaltan çözümler bulmak için çok yaratıcı olmalıdır.

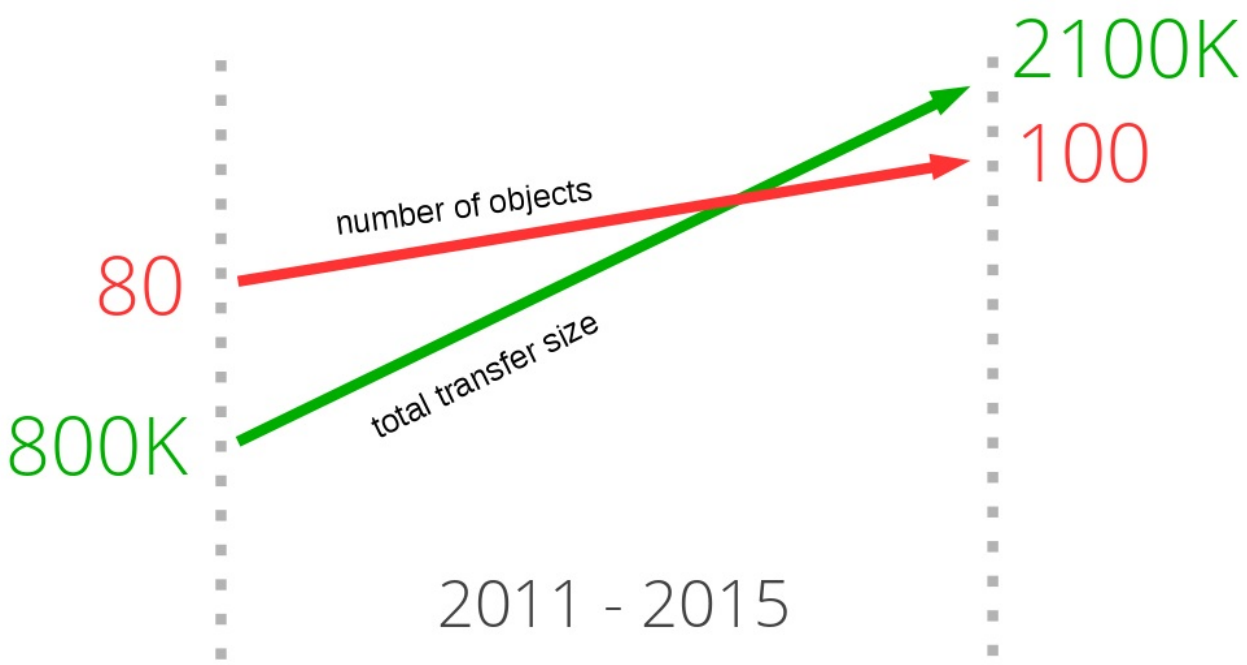
Yıllar boyunca paralel olarak devam eden diğer girişimler TCP’nin bu kadar kolay değiştirilmediğini doğruladı ve bu nedenle hem TCP hem de protokolleri iyileştirmeye çalışıyoruz.

Basitçe söylemek gerekirse, TCP daha fazla veri göndermek veya almak adına oluşabilecek duraklamalar ve boş sürelerden kaçınmak için daha iyi kullanılabilir. Sıradaki bölümlerde bu eksikliklerin bazıları vurgulanacaktır.

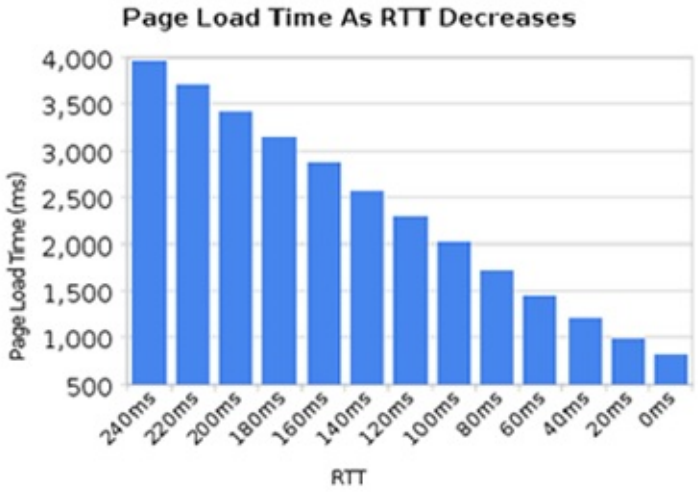
2.4 Aktarım boyutları ve nesne sayısı

Bugün Web'deki en popüler sitelerin bazılarına ve bunların ön sayfalarını indirmek için neye ihtiyaç olduğuna bakıldığında net bir model ortaya çıkıyor. Yıllar geçtikce alınan veri yavaş yavaş 1.9MB'ın üzerine çıkmıştır. Bu bağlamda daha önemli olan şey, her sayfayı görüntülemek için ortalama olarak yüzü aşkın bireysel kaynağa ihtiyaç duyulmasıdır.

Aşağıdaki grafiğin gösterdiği gibi, eğilim bir süredir devam ediyor ve yakın zamanda bunun değişeceğine dair hiç bir gösterge yoktur. Grafik dünyadaki en popüler web sitelerine hizmet vermek için toplam aktarım boyutunun (yeşil renkte) ve ortalama olarak kullanılan toplam talep sayısının (kırmızı renkte) son dört yılda nasıl değiştiğini göstermektedir.



2.5 Gecikme öldürür



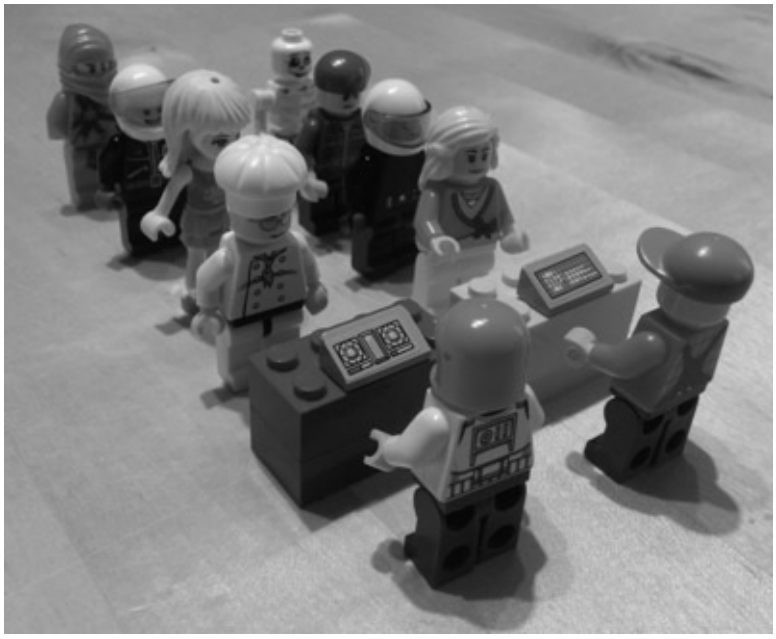
HTTP 1.1'in gecikmeye karşı çok hassas olması, kısmen HTTP boruhattının buyuk bir kullanıcı oranına sahip olması ile ilgilidir.

Geçtiğimiz birkaç yılda kullanıcılara mevcut bant genişliğinde büyük bir artış gösterse de, gecikmeyi azaltma yolunda aynı seviyede bir gelişmeye rastlamadık. Geçmişteki mobil teknolojiler gibi yüksek gecikme süresi olan bağlantılar, yüksek bant genişliği bağlantısı sağladı fakat bu dahi iyi ve hızlı bir web deneyimi elde etmeyi mümkün kılmadı.

Gerçekten düşük gecikmeye ihtiyaç duyan başka bir kullanım örneği, video konferans, oyun ve benzeri türde videolardır, ki bunlarda önceden oluşturulmuş akışlar yoktur.

2.6. Satır başı engelleme

HTTP Boru Hattı, bir önceki isteğe yanıt beklerken başka bir istek göndermenin bir yoludur. Bu durum bankadaki veya süper marketteki bir işlem alanında sıra oluşmasına çok benzer. Önünüzdeki kişi işini hızlıca halledecek dakik bir müşteri midir, yoksa işe başlamadan önce sonsuza dek süreceğine inanan sinir bozucu bir müşteri midir bilmiyorsunuz: satır başı engelleme işte budur.



Satır seçimi konusunda dikkatli olabilirsiniz, bu yüzden doğru olduğunu gerçekten düşündüğünüz bir satırı seçebilirsiniz veya bazen kendinize ait yeni bir satırdan başlayabilirsiniz fakat bunun sonunda seçtiğiniz bu satırı değiştiremezsiniz.

Yeni bir satır oluşturmak da bir performans ve kaynak cezasıyla ilişkilidir, bu yüzden daha küçük satır sayılarının ötesinde ölçeklenebilir değildir. Bunun için mükemmel bir çözüm yoktur.

Bugün bile, 2015, çoğu masaüstü web tarayıcısında varsayılan olarak HTTP boruhattı devre dışı bırakılmıştır.

Bu konuyla ilgili ekler Firefox'un bugzilla entry 264354 adresinde bulunur.

3. Gecikmelerin üstesinden gelmek için yapılanlar

Her zaman olduğu gibi sorunlarla karşı karşıya kaldıklarında, insanlar geçici çözüm bulmak için toplanırlar. Geçici çözümlerin bazıları zekice ve kullanışlı, bazıları ise berbattır.

3.1 Birleştirme

Birleştirme, çok sayıda küçük görüntüyü tek bir büyük görüntü olması için bir araya getirdiğinizde bu durumu tanımlamak için sıklıkla kullanılan bir terimdir. Ardından daha küçük olanlarını göstermek adına büyük resmin parçalarını kesmek için javascript'i veya CSS'yi kullanırsınız.

Bir site bu hileyi hız için kullanır. HTTP 1.1'de tek bir büyük görüntü elde etmek, 100 küçük görüntüyü tek tek bir araya getirmekten daha hızlıdır.

Tabi ki bu, sitenin yalnızca bir veya iki küçük resmi ve benzerlerini göstermek isteyen site sayfaları için dezavantajlara sahiptir. Aynı zamanda, muhtemelen en çok kullanılan resimlerin kalmasına izin vermek yerine, tüm resimleri aynı anda önbellekten çıkarılabilecek hale getirir.

3.2 İçerilme

İçerilme, resimleri tek tek göndermekten kaçınmanın diğer bir hilesidir ve bu verileri kullanarak yapılır. Kaynak Konumlandırıcılar(URLs) css dosyalarında gömülmüştür. Bu birleştirme olayında olduğu gibi benzer faydalara ve zararlara sahiptir.

```
.icon1 {
    background: url(<data>) no-repeat;
}

.icon2 {
    background: url(<data>) no-repeat;
}
```

3.3 Bitiştirme

Büyük bir site, bir sürü farklı javascript dosyası ile sonuçlanabilir. Kullanıcı arayüzünü kontrol eden araçlar, geliştiricilerin hepsini bir araya getirmelerine yardım ederek tarayıcının onlarca küçük dosya yerine tek bir büyük dosyaya ulaşmasını sağlar. Çok az veri gerektiğinde çok fazla veri gönderilir. Bir değişiklik yapılması gerektiğinde çok fazla veri yeniden

yüklenmelidir.

Bu uygulama tabi ki çoğunlukla söz konusu geliştiricilere rahatsızlık veriyor.

3.4 Püskürtme

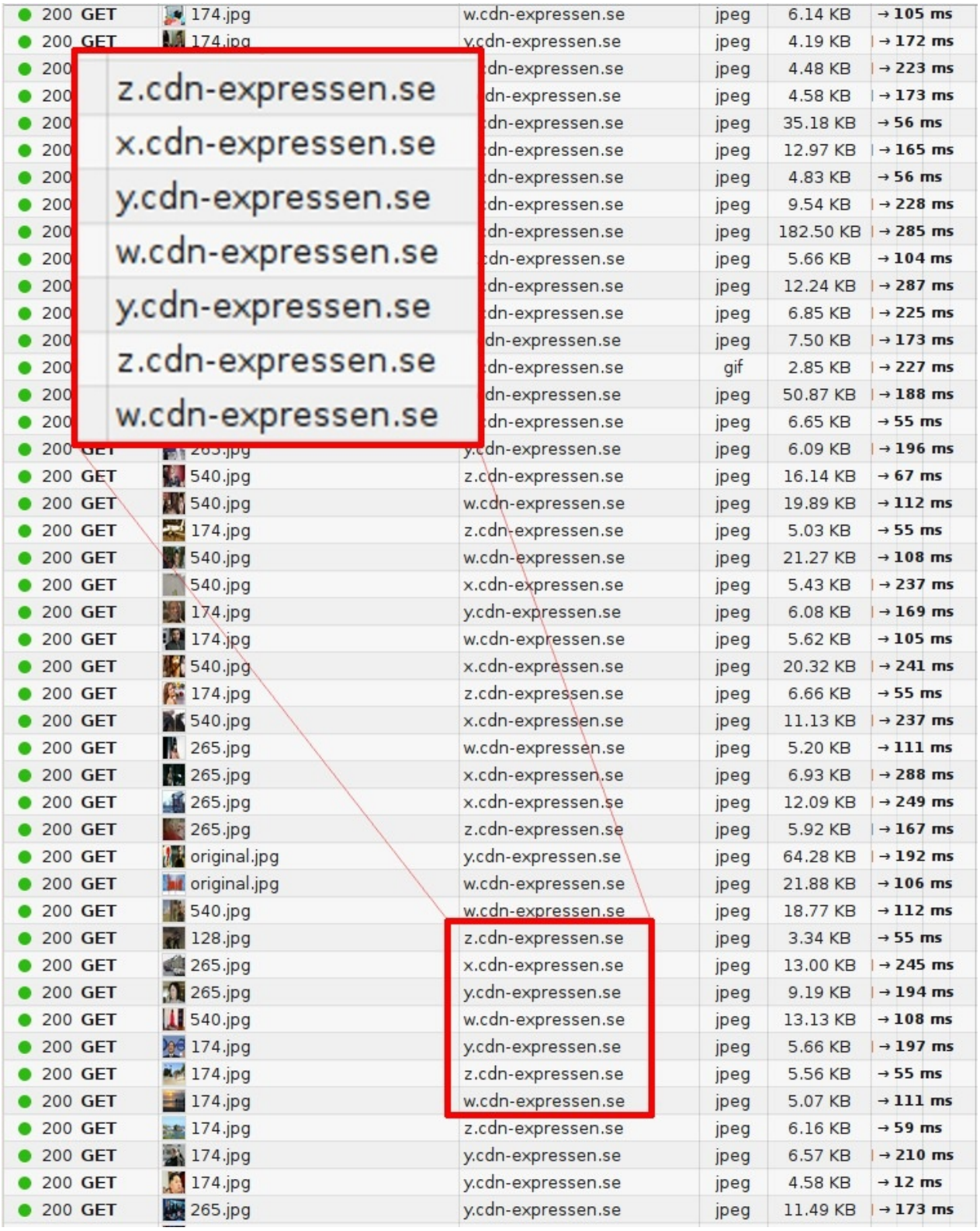
Söz edeceğim nihai performans hilesi sıklıkla "püskürtme" olarak adlandırılır. Temel olarak hizmetinizin mümkün olabildiğince çok sayıda farklı barındırıcıya hizmet etmesi anlamına geliyor. İlk bakışta bu garip gözükse de bunun arkasında bır mantık vardır.

Başlangıçta HTTP 1.1 beyannamesi, bir istemcinin her bir ana bilgisayar için en fazla iki TCP bağlantısı kullanmasına izin verdiğini belirtti. Dolayısıyla, akıllı siteleri ihlal etmemek için, yeni barındırıcı adları keşfedildi ve sitenize daha fazla bağlantı kurabilir – voilà - ve sayfa yükleme sürelerini azaltabilirsiniz.

Zamanla bu sınırlama kaldırıldı ve bugün müşteriler barındırıcı başına 6-8 bağlantıyı kolayca kullanıyor ancak sitelerin bağlantı sayısını artırmak için bu tekniği kullanmaya devam etmesi için hala bir sınırı var. Nesnelerin sayısı arttıkça -daha önce de gösterildiği gibi- çok sayıda bağlantı daha sonra HTTP'nin iyi performans gösterdiğinden ve sitenizi daha hızlı hale getirdiğinden emin olmak için kullanılır. Sitelerin bu tekniği kullanarak tek bir site için 50'den fazla hatta 100'e kadar veya daha fazla bağlantıyı kullanması olağandışı değildir. Httparchive.org tarafından yayınlanan son istatistikler, siteyi görüntülemek için dünyanın en büyük 300K URL'lerinin ortalama 40(!) TCP bağlantısı gerektirdiğini ve eğilim bunun zaman içinde yavaş ilerlediğini gösteriyor.

Başka bir sebep de, bu günlerin oldukça önemli olabileceği gibi, çerezleri kullanmayan ayrı bir ana makine adına resim veya benzeri kaynaklar koymaktır. Çerezsiz resim hostlarını kullanarak bazen çok daha küçük HTTP isteklerine izin vererek performansı arttırabilirsiniz!

Aşağıdaki resim, İsveç'in en iyi web sitelerinden birine göz atarken, taleplerin çeşitli ana bilgisayar adları üzerinden nasıl dağıtıldığını ve bir paket izinin nasıl göründüğünü göstermektedir.


z.cdn-expressen.se
x.cdn-expressen.se
y.cdn-expressen.se
w.cdn-expressen.se
y.cdn-expressen.se
z.cdn-expressen.se
w.cdn-expressen.se

4. HTTP'nin güncellenmesi

Geliştirilmiş bir protokol yapmak hoş olmaz mıydı? Elbette olurdu...

1. Gecikmelere daha az duyarlı olsun
2. Boruhattı ve satır başı engelleme sorununu düzeltsin
3. Her bir barındırıcıya olan bağlantı sayısını artırmaya gerek duymazsın
4. Mevcut tüm arayüzleri, tüm içeriği, URI formatını saklasın
5. IETF'nin HTTPbis çalışma grubu içinde yapılansın

4.1. IETF ve HTTPbis çalışma grubu

İnternet Mühendisliği Görev Gücü(IETF), çoğunlukla protokol seviyesinde, internet standartlarını geliştiren ve tanıtan bir organizasyondur. TCP, DNS, FTP'den en iyi uygulamalara, HTTP'ye ve çok sayıda protokol türevine varıncaya kadar her şeyi belgeleyen RFC dokümanları için dünya çapında yaygınca bilinirler.

IETF içinde, bir hedefe doğru çalışmak için "çalışma grupları" sınırlı bir kapsam ile oluşturulmuştur. Ürettikleri şey için bazı belirlenmiş yönergeler ve sınırlamalar içeren bir "charter" (tanımlama) oluştururlar. Tartışmalara ve gelişmelere herkesin katılmasına izin verilir. Katılan ve bir şeyler söyleyen herkes, sonuca etki eden aynı ağırlığa ve şansa sahiptir ve herkes, kendisi için çalıştığı şirkete pek saygı duymaksızın, bir birey olarak sayılır.

HTTPbis çalışma grubu (adın açıklaması için daha sonra inceleyelim) 2007 yazında kuruldu ve HTTP 1.1 şartnamesinin güncellenmesi görevini üstlendi. Bu grup içinde HTTP'nin bir sonraki sürümü ile ilgili tartışmalar 2012'nin sonlarında gerçekten başladı. HTTP 1.1 güncelleme çalışması 2014 yılının başında tamamlandı ve RFC 7230 serisi ile sonuçlandı.

HTTPbis WG için gerçekleşen son görüşmeler, Haziran 2014'ün başında New York'ta yapıldı. Geri kalan tartışmalar ve resmi RFC'yi almak için gerçekleştirilen IETF prosedürleri ertesi yıl devam etti.

HTTP sahasındaki daha büyük oyuncuların bazıları çalışma grubu tartışmalarında ve toplantılarda eksik kaldı. Burada herhangi bir şirket veya ürün adından bahsetmek istemiyorum, ancak bugün Internet'teki bazı aktörlerin IETF'nin bu şirketlerin karışımı olmadan iyi olacağından emin oldukları açıkça görünüyor...

4.1.1. İsmin "bis" bölümü

Grup, HTTPbis olarak adlandırıldı ve burada "bis" kısmı iki Latin alfabesinden Latin adverb for two geliyor. Bis genellikle bir güncelleme için IETF içindeki adın bir son ek veya bir parçası olarak kullanılır veya ikinci bir beyanname'ye geçmek;bu durumda HTTP 1.1 güncellemesi.

4.2. http2 SPDY'den başladı

SPDY Google tarafından geliştirilen ve öncülük eden bir protokoldür. Kesinlikle bunu açık alanda geliştirdiler ve herkesi katılmaya davet ettiler, ancak hem popüler bir tarayıcı uygulaması hem de iyi kullanılan servislere sahip önemli bir sunucu popülasyonu üzerinde kontrol sahibi olduklarından yararlandıkları açıktır.

HTTPbis grubu http2 üzerinde çalışmaya başlamanın zamanı olduğuna karar verdiklerinde, SPDY zaten bir çalışma konsepti olduğunu kanıtlamıştı. İnternette konuşlandırmanın mümkün olduğunu göstermiş ve ne kadar iyi performansı olduğunu gösteren yayınlanmış rakamlar bulunmaktadır. Http2 çalışması, temelde http2 taslak-00'a küçük bir arama ve değiştirme ile yapılan SPDY/3 taslağı ile başladı.

5. http2 kavramları

Peki http2 ne yapar? HTTPbis gruplarının ne yapmak için ne gibi sınırları vardır?

Sınırlar aslında oldukça katıydı ve ekibin yenilik yapma kabiliyetine çok fazla engel koyuyordu:

- http2, HTTP paradigmalarını korumak zorundadır. Hala, istemcinin TCP üzerinden sunucuya istek gönderdiği bir protokoldür.
- http:// ve https:// URL'leri değiştirilemez. Bunun için yeni bir plan olamaz. Bu tür URL'leri kullanarak içerik miktarı, değişiklik beklemek için çok büyük.
- HTTP1 sunucuları ve istemcileri onlarca yıldır dolaşıyorlar, onları http2 sunucularına proxyleştirebilmeliyiz.
- Daha sonra, proxy'ler http2 özelliklerini HTTP 1.1 istemcilerine birebir eşleyemez olmalıdır.
- Protokoldeki isteğe bağlı parçaları çıkarın veya azaltın. Bu gerçekten bir şart değil, SPDY ve Google ekibinden gelen daha mantra idi. Her şeyin gerekli olduğu düşünüldüğünde, şimdi hiçbir şeyin uygulanamayacağı tuzağına düşürebilir.
- Artık küçük sürüm yok. İstemcilerin ve sunucuların http2 ile uyumlu olduğu ya da olmadığı kararlaştırıldı. Protokolü genişletmek veya değişiklik yapmak için bir ihtiyaç ortaya çıkarsa, http3 doğar. Http2'de artık küçük sürümler yok.

5.1. Varolan URI şemaları için http2

Daha önce de belirtildiği gibi, mevcut URI şemaları değiştirilemez, bu yüzden http2 mevcutları kullanmalıdır. Bugün HTTP 1.x için kullanıldıklarından, protokolü http2'ye yükseltmenin veya başka bir şekilde sunucudan eski protokoller yerine http2'yi kullanmasını isteyebilecek bir yola ihtiyacımız var.

HTTP 1.1, bunu yapmak için tanımlanmış bir yönteme sahiptir; yani Sunucu, eski protokol üzerinden böyle bir istekte bulunurken, yeni bir protokolü kullanarak ek bir gidiş dönüş ücreti ile bir yanıt göndermesini sağlayan Upgrade: başlığını içerir.

Bu gidiş dönüş cezası, SPDY ekibinin kabul etmeyeceği bir şey değildi ve yalnızca TLS üzerinden SPDY'yi uyguladıkları için müzakereyi önemli ölçüde -kısmen- sağlayan yeni bir TLS uzantısı geliştirdiler. Next Protocol Negotiation için NPN olarak adlandırılan bu uzantıyı kullanarak, sunucu istemciye hangi protokolleri sağladığını bildirir ve istemci daha sonra tercih ettiği protokolü kullanabilir.

5.2. https:// için http2

Http2'nin odak noktası, TLS üzerinde düzgün davranmasını sağlamak olmuştur. SPDY TLS gerektirir ve http2'de TLS'i zorunlu hale getirmek için güçlü bir itme olmuştur, ancak http2 isteğe bağlı olarak TLS ile birlikte gönderildiği için fikir birliğine varılamamıştır. Bununla birlikte, iki önde gelen uygulayıcı, http2'yi TLS üzerinden uygulayacaklarını açıkça belirtti: günümüzün önde gelen web tarayıcılarından ikisi olan Mozilla Firefox ve Google Chrome liderleri.

Yalnızca TLS'yi seçme nedenleri, kullanıcıların gizliliği konusundaki çekinceleri ve TLS ile yapılan yeni protokollerin daha yüksek başarı oranına sahip olduğunu gösteren erken ölçümlere saygıyı içerir. Bunun nedeni 80 numaralı bağlantı noktasını aşan HTTP 1.1 olmasıdır ve bu bağlantı noktasında diğer tüm protokoller kullanıldığında bazı -orta kutularla- etkileşime giren veya trafiği yok eden yaygın varsayım olmasıdır.

Zorunlu TLS konusu, e-posta listelerinde ve toplantılarda titrek sesler çıkardı -iyi mi yoksa kötü mü- gibi. Bu sorunu bir HTTPbis katılımcısının karşısına çıkardığınızda bunun farkında olun!

Benzer şekilde, http2'nin TLS kullanırken zorunlu olması gereken şifrelerin bir listesini mi yoksa belki de bir seti kara listeye alması mı gerektiği konusunda veya TLS'den hiçbir şey talep etmemesi gerekip gerekmediğine ilişkin uzun süreli tartışmalar sürüyor, ya da belki de kara listeye alınmalı ya da TLS "katmanı" ndan hiçbir şey talep etmemeli, TLS çalışma

grubuna bırakılmalıdır. Beyanname, TLS'nin en azından sürüm 1.2 olması gerektiğini ve şifre paketi kısıtlamaları olduğunu belirtti.

5.3. TLS üzerinden http2 anlaşması

Next Protocol Negotiation (NPN), SPDY'yi TLS sunucularıyla pazarlık etmesi için kullanılmış protokoldür. Uygun bir standart olmadığı için, IETF aracılığıyla oluşturuldu ve sonuç ALPN oldu: Application Layer Protocol Negotiation. SPDY istemcileri ve sunucuları hala NPN kullanırken, ALPN http2 tarafından kullanılmak üzere yükseltilmektedir.

NPN'nin önce varolduğu ve ALPN'nin standardizasyona geçmesi biraz zaman aldığından, http2 müzakere ederken çok erken http2 istemcileri ve http2 sunucuları uygulamakta ve bu uzantıları kullanmaya başlamıştır. Ayrıca SPDY için NPN kullanılır ve birçok sunucu hem SPDY hem de http2 sunar, bu nedenle bu sunucularda hem NPN hem de ALPN'yi desteklemek mantıklıdır.

5.4. http:// için http2

Daha önce de belirtildiği gibi, düz metin HTTP 1.1 için http2 sunucuya bir yükseltme(Upgrade): başlığı sunar. Sunucu http2 konuşursa, "101 Anahtarlama"(“101 Switching”) durumu ile yanıt verir ve o andan itibaren bu bağlantı http2 olarak devam eder. Tabi ki bu yükseltme prosedürü tam bir ağ gidiş-dönüş süresine yol açar, ancak, http2 bağantısını daha uzun süre canlı tutmak ve onu tipik bir HTTP1 bagantısından daha fazla kullanmak mümkündür.

Bazı tarayıcıların sözcüleri http2'yi uygulamayacaklarını söyleseler de, Internet Explorer ekibi destekleyecelerini söyledi. curl ve diğer tarayıcı dışı birkaç istemci de açık metin(clear-text) http2'yi destekler.

Bugün, TLS olmadan http2'yi destekleyen hiçbir başlıca tarayıcı yok.

6. Http2 protokolü

Arkaplan hakkında, bizi ilgilendiren tarih ve siyaset hakkında yeterince bilgi bulunuyor. Protokolün özelliklerine, http2'yi oluşturan ikili sayılar ve kavramlara bakalım.

6.1. İkili protokol

http2, ikili bir protokoldür.

Bir dakika izin ver yeter. Daha önce internet protokolleri ile ilgilendiyseniz, bu değişime bir şekilde tepki göstereceksiniz, çünkü insanlar telnet ve benzeri yollarla istekte bulunabileceğinden metin / ascii tabanlı protokollerin nasıl daha üstün olduğunu argümanlarınızı sıralayarak ispatlamaya çalışacaktır...

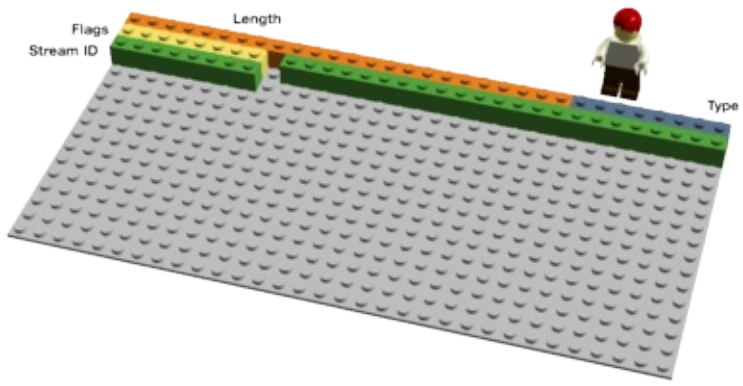
http2, çerçevelemeyi çok daha kolay hale getirmek için ikilidir. Çerçevelerin başlangıcını ve bitişini bulmak HTTP 1.1'de ve aslında genel olarak metin tabanlı diğer protokollerde de karmaşık durumlardan biridir. İsteğe bağlı beyaz boşluğun dışına ve aynı şey için farklı yollarla ilerleyerek uygulama daha basit hale gelir?

Ayrıca, gerçek protokol bölümlerini çerçeveden ayırmak daha kolaydır, ki HTTP1 karmaşıktır.

Protokolün sıkıştırma özelliğine sahip olması ve genellikle TLS ile çalışması da, metnin önemini düşürür; zira yine de hat üzerinde metin görmezsiniz. Http2'deki protokol seviyesinde neler olduğunu tam olarak anlamak için bir Wireshark gibi ağdaki paketleri inceleyebileceğiniz bir uygulama kullanma fikrine alışmamız yeterlidir.

Bu protokolün hata ayıklamasının muhtemelen curl gibi araçlarla veya ağ akışının Wireshark ve benzerleriyle analiz ederek yapılması gerekecektir.

6.2. İkili format



http2, ikili çerçeveler gönderir. Gönderilebilen farklı çerçeve türleri de vardır ve hepsi aynı ayarlara sahiptir: Uzunluk, Tip, Bayraklar, Akış Tanımlayıcı ve çerçeve yükü.

Http2 beyannamesinde tanımlanan on farklı çerçeve türü vardır ve HTTP 1.1 özelliklerine eşleyen iki temel çerçeve VERİ ve BAŞLIK'tır. Çerçevelerin bazılarını daha ayrıntılı olarak açıklayacağım.

6.3. Çoklu akışlar

Önceki bölümde bahsedilen Akım Tanımlayıcı, http2 üzerinden gönderilen her kareyi bir "akış" ile ilişkilendirir. Akış, http2 bağlantısı içinde istemci ve sunucu arasında değiştirilen bağımsız, çift yönlü bir çerçeve dizisidir.

Tek bir http2 bağlantısı, eş zamanlı birden fazla açık akış içerebilir; bu uç noktalarda çoklu akışlardan çerçeveler araya girebilir. Akışlar kurulabilir ve tek taraflı olarak kullanılabilir veya istemci veya sunucu tarafından paylaşılabilir ve iki uç nokta tarafından da kapatılabilir. Çerçevelerin bir akış içinde gönderilme sırası önemlidir. Alıcılar, çerçeveleri aldığı sıraya göre işlerler.

Akışların çoğullaması, birçok akıştan gelen paketlerin aynı bağlantı üzerinden karışabilmesi anlamına gelir. İki bireysel tren tek bir tren haline gelebilir ve daha sonra diğer tarafta tekrar ayrılabilir.

İki tren aynı bağlantı üzerinden çoğullandı:

6.4. Öncelikler ve Bağımlılıklar

Her bir akış, sunucuyu öncelikle hangi akışların gönderileceğini seçmeye zorlayan kaynak kısıtlamaları olması durumunda, hangi akışın en önemli olduğunu söylemek için kullanılan bir öncelik(ağırlık olarak da bilinir) bilgisine sahiptir.

ÖNCELİK çerçevesini kullanarak bir istemci, sunucuya bir akışın bağlı olduğu diğer bir akışı da söyleyebilir. Bu istemciye öncelik agacı oluşturmasına izin verir ki bu ağacda "cocuk akışlar" bircok "ebeveyn akışa" bağlı olabilir.

Öncelik ağırlıkları ve bağımlılıkları çalışma zamanında etkin olarak değiştirilebilir, ki bu da, kullanıcılar görüntülerin bulunduğu bir sayfayı aşağıya kaydırdığında, tarayıcıların hangi görüntülerin en önemli olduğunu belirleyebilmesine veya sekmeleri değiştirirseniz, birdenbire odaklanacak yeni bir dizi akışın önceliğini oluşturabilmesine olanak tanır.

6.5. Başlık sıkıştırma

HTTP yurtsuz(stateless) bir protokoldür. Kısaca, bu, sunucunun önceki isteklerden çok fazla bilgi ve meta veri depolaması gerekmeden, her talebin sunucunun bu talebi sunması için gereken kadar ayrıntılı getirmesi gerektiği anlamına gelir. Http2 bu paradigmayı değiştirmediği için aynı şekilde çalışması gerekir.

Bu, HTTP'yi tekrarlı yapar. Bir müşteri bir web sayfasındaki, örneğin görüntüler gibi aynı sunucudan birçok kaynak istediğinde, hepsi için neredeyse aynı görünen bir istek dizisi talep edecektir. Sıklıkla aynı olan bu dizi sıkıştırma için yalvarır.

Web sayfası başına düşen nesne sayısı arttıkça (önceden belirtildiği gibi), çerezlerin kullanımı ve isteklerin boyutu da zamanla artmaya devam etti. Çerezlerin tüm isteklerde bulunması gerekir, çoğu zaman aynı istek çoklu istekler içindedir.

HTTP 1.1 istek boyutları o kadar büyük oluyor ki bazen ilk TCP penceresinden daha büyük olabiliyor, bu da istek gönderilmeden önce sunucudan ACK almak için tam bir gidiş dönüş zamanına ihtiyaç duyduklarından göndermenin çok yavaş olmasına sebep oluyor. Sıkıştırmanın bir başka kanıtı bu durumdur.

6.5.1. Sıkıştırma hileli bir konudur

HTTPS ve SPDY sıkıştırmasının BREACH ve CRIME saldırılarına karşı savunmasız olduğu tespit edildi. Bilinen metni akışa çıktıyı ekleyerek, nasıl değiştirdiğini öğrenebilir, böylece saldırgan şifreli bir yükte gönderileni anlamaya çalışabilir.

Bir protokol için etkin içeriğe sıkıştırma yapmak biraz düşünülmesi ve dikkatli olması gereken bir konudur(saldırılardan birine karşı savunmasız hale gelmeden yapılır). HTTPbis ekibi bunu yapmaya çalıştı.

HPACK bakın, HPACK(adından da anlaşılacağı gibi), HTTP/2 için Başlık Sıkıştırmasıdır. Bu sıkıştırma biçimi özellikle http2 başlıkları için hazırlanmış olup, ayrı bir internet taslağında belirtilmektedir. Yeni format, diğer karşı ölçümlerle(belirli bir üstbilgiyi ve çerçevelerin dolgusunu(adding) sıkıştırmamasını sağlayan bir bit gibi), sıkıştırmanın kullanılmasını zorlaştırır.

Roberto Peon'un (HPACK'in yaratıcılarından biri) sözleriyle:

> HPACK sızan bilgiyi önlemek, kodlama ve kod çözme işlemini hem hızlandırmak hem ucuzlaştırmak, sıkıştırılan içerik boyutu üzerinde alıcı adına kontrol sağlamak, proxy'nin yeniden indexlenmesine izin vermek(yani, bir proxy içindeki ön uç ve arka uç arasında paylaşılan durum) ve Huffman kodlu dizelerin çabuk karşılaştırmaları için tasarlandı.

6.6. Sıfırla - fikrini değiştir

HTTP 1.1'in getirdiği dezavantajlardan biri, belirli bir boyuta sahip bir İçerik-Uzunluğu ile bir HTTP mesajı gönderildiğinde, onu kolayca durdurmanızın mümkün olamayacağıdır. Tabi, TCP bağlantısını kesebilirsiniz (her zaman değil) ancak yine de yeni bir TCP el sıkışması yapmak zorunda kalmanız maliyetlidir.

Daha iyi bir çözüm, mesajı durdurup yeni bir başlangıç yapmak olacaktır. Bunu, boşa harcanmış bant genişliğini ve bağlantıları yıkma ihtiyacını önlemeye yardımcı olacak, http2'nin RST_STREAM çerçevesiyle yapabilirsiniz.

6.7. Sunucu İtme

Bu, "önbellek itme" olarak da bilinen özelliktir. Müşteri kaynak X sorduğunda, sunucu istemcinin istemcinin Z kaynağı da istediğini bilir ve sorulmadan istemciye gönderir. İstediğinde orada olacak şekilde Z'yi önbelleğine koyarak müşteriye yardımcı olur.

Sunucu itme, istemcinin sunucuya açıkça izin vermesi gereken bir özelliktir. İstemci belirli bir kaynağı istemiyorsa, itilen bir akışı RST_STREAM ile hızla sonlandırabilir.

6.8. Akış kontrolü

Her bir http2 akışının kendi akış penceresi vardır, bu pencerede diğer ucun veri göndermesine izin verilir. SSH'ın nasıl çalıştığını biliyorsanız, çok benzer olduğunu göreceksiniz.

Her bir akış için, iki uçun da gelen veriyi işlemek için yeterli alana sahip olup olmadığı veya diğer uçta yalnızca pencere genişletilinceye kadar belirli miktarda veri gönderilmesine izin verilebileceği gibi durumları bildirmeye hakları vardır. Sadece VERİ çerçeveleri akış kontrollüdür.

7. Uzantılar

Http2 protokolü, bir alıcının bilinmeyen tüm çerçeveleri (bilinmeyen bir çerçeve türüne sahip olanları) okuması ve yoksayması şartını getirir. İki taraf yeni çerçeve türlerinin kullanımını hop-by-hop esasına göre müzakere edebilir, ancak bu çerçeveler durum değişikliğine ve akış denetimine izin vermiyorlar.

Http2'nin uzantılara izin verip vermeyeceği konusunun tartışılması, protokolün gelişimi boyunca farklı ve zıt görüşlerle birlikte devam etti. Taslak-12'den sonra nihai olarak uzantılara izin verildi.

Uzantılar gerçek protokolün bir parçası değildir ve çekirdek protokolün dışında belgelendirilecektir. Protokole dahil olacak ve muhtemelen uzantılar olarak gönderilen ilk çerçeveler olacak olan, tartışılan iki çerçeve türü vardır:

7.1. Alternatif Hizmetler

Http2'nin kabulüyle, TCP bağlantılarının çok daha uzun olacağından ve HTTP 1.x bağlantılarından çok daha uzun süre canlı tutulacağından şüphelenmek için nedenler vardır. Bir istemci, her ana barındırıcıda/sitede tek bir bağlantıyla çok şey yapabilir ve bu bağlantı bir süre açık olabilir.

Bir site müşterisinin başka bir barındırıcıya bağlanmasını önerdiğinde bu durum HTTP yük dengeleyicilerin çalışmasının etkilenmesine yol acabilir. Bu durum, performans nedenleriyle veya bakım vb. için olabilir.

Sunucu, müşteriye alternatif bir servisAlt-Svc: header (yada http2 ile birlikte ALTSVC çerçevesi) hakkında bilgi vererek gönderir: aynı içeriğe başka bir rota, başka bir servis, ana makine ve port numarası kullanarak.

Bir istemci daha sonra bu hizmete eşzamansız olarak bağlanmaya çalışmalı ve yeni bağlantı başarılı olursa alternatifi kullanmalıdır.

7.1.1. Fırsatçı TLS

Alt-Svc başlığı, aynı içeriğin bir TLS bağlantısı üzerinden de erişilebilir olduğunu istemciye bildirmek için http:// üzerinden içerik sağlayan bir sunucuya izin verir.

Bu biraz tartışılabilir bir özelliktir. Böyle bir bağlantı kimliği doğrulanmamış TLS yapacak, herhangi bir yerde "güvenli" olarak ilan edilemeyecek, UI'da herhangi bir asma kilit(padlock) kullanamayacak, aslında kullanıcıya düz eski HTTP olmadığını gösteremeyecektir ancak bu hala fırsatçı TLS'dir ve bazı insanlar bu kavrama karşı fazlasıyla karşıdır.

7.2. Engellenmiş

Bu tür bir çerçeve, gönderilecek veriler olduğunda ancak akış denetimi herhangi bir veri göndermeyi yasakladığında dahi http2 tarafında tam olarak bir kez gönderilmesi amaçlanmıştır. Fikir şudur ki; eğer uygulamanız bu çerçeveyi alırsa, bir şeyi berbat ettiğinizi ve/veya mükemmel aktarım hızlarından daha azını aldığınızı biliyorsunuzdur.

Çerçeve bir uzantı haline gelmeden önce taslak-12'den bir alıntı:

> “Deneyi kolaylaştırmak için ENGELLİ çerçeve, bu taslak sürüme dahil edilmiştir. Deney sonuçları olumlu geri bildirim sağlamıyorsa kaldırılabilir.

8. http2 dünyası

Peki, http2 kabul edildiğinde işler nasıl gözükecek? Kabul edilecek mi?

8.1. Http2 sıradan insanları nasıl etkiler?

http2 henüz geniş ölçüde ne kullanıldı ne de uygulandı. Her şeyin nasıl gerçekleşeceğini kesin olarak bilemeyiz fakat SPDY'nin nasıl kullanıldığını gördük ve o andaki geçmiş ve güncel deneylere dayanan bazı tahminler ve hesaplamalar yapabiliriz.

http2, gerekli ağ gidiş-dönüş sayısını azaltır ve istenmeyen satır başını engelleme problemini çoğullama ve hızlı atılma tamamen önler.

Bugünün en yaygın kullanılan sitelerinde bile büyük miktarda paralel akışa izin verir.

Akışlar üzerinde doğru olarak kullanılan önceliklerle, istemcilerin daha az önemli olan veriden önce aslında daha önemli verileri alması ihtimali daha yüksektir.

Bütün bunlar bir araya getirildiğinde, şansın çok daha yüksek olduğunu ve bunun daha hızlı sayfa yüklemelerine ve daha duyarlı web sitelerine yol açacağını söyleyebilirim. Kısaca: daha iyi web deneyimi.

Ne kadar hızlı bir sürede ne kadar iyileştirme göreceğimizi, henüz söyleyebileceğimizi sanmıyorum. Birincisi, teknoloji hala çok erken ve istemcilerin ve sunucuların, bu yeni protokolün sunduğu tüm güçlerden gerçekten yararlanmak için uygulamaların düzeltildiğini görmeye bile başlamadık.

8.2. Http2 web gelişmelerini nasıl etkiler?

Yıllar içinde web geliştiricileri ve web geliştirme ortamları HTTP 1.1 ile ilgili sorunları çözmek için çeşitli araçlar(toolbox) oluşturmuşlardır; hatırlatmak gerekirse, http2'nin aleyhinde haklı neden olarak bunlardan bazılarını özetledim.

Araçlar ve geliştiricilerin düşünmeden kullandıkları bu geçici çözümlerin büyük bir kısmı büyük olasılıkla http2'nin performansını düşürür veya en azından yeni süper güçlerinden yararlanmaz. İçerilme ve püskürtme http2 ile yapılmamalıdır. Muhtemelen daha az bağlantı kullanacağı için, Sharding? http2 için zararlıdır.

web sitelerinin ve web geliştiricilerinin en azından kısa vadede geliştirmeye ve konuşlandırmaya ihtiyacı olması bir problemdir ki bu, hem HTTP1.1 hem de http2 istemcilerine sahip olması ve tüm kullanıcılar için maksimum performans sağlaması açısından iki farklı ön uç sunmak zorunda kalmadan zorlu olabilir.

Yalnız bu nedenlerden dolayı, http2'nin tam potansiyeline ulaşıldığına karar vermemizin biraz zaman alacağını düşünüyoruz.

8.3. http2 uygulamaları

Böyle bir belgede belirli uygulamaları belgelemeye çalışmak, tamamen boş bir çabaydı ve başarısızlığa mahkûm edildi ve yalnızca kısa sürede zaman aşımına uğramıştı. Bunun yerine durumu daha geniş anlamda açıklayacağım ve okuyucuları http2'nin web sitesine uygulamaların listesi yönlendireceğim.

Daha önce çok fazla sayıda uygulama vardı ve bu miktar http2 çalışması sırasında zamanla arttı. Yazı yazarken 40'dan fazla uygulama listelendi ve bunların çoğu son sürümü uyguluyor.

8.3.1 Tarayıcılar

Firefox yeni tasarimlarin ustunde bir internet tarayicisi olmustur, twitter uzak kalmistır ve servislerini http2 nin ustunde sunmuştur.? Google, hizmetlerini çalıştıran birkaç test sunucusunda http2 desteği sunmak için Nisan 2014'te başladı ve Mayıs 2014'ten bu yana Chrome'un geliştirme sürümlerinde http2 desteği sağladı. Microsoft, bir sonraki Internet Explorer sürümü için http2 desteğiyle bir teknik önizleme gösterdi. Hem Safari (iOS 9 ve Mac OS X El Capitan ile birlikte) hem Opera hem de http2'yi destekleyeceklerini söyledi.

8.3.2 Sunucular

Http2'nin zaten çok sayıda sunucu uygulaması var.

Popüler Nginx sunucusu 22 Eylül 2015'te piyasaya sürülen 1.9.5'den beri http2 desteği sunmaktadır(SPDY modülünün yerine geçer; böylece her ikisi de aynı sunucuda çalışamazlar).

Apache'nin httpd sunucusu, 9 Ekim 2015'te çıkan 2.4.17'den bu yana http2 modülü mod_http2 içeriyor.

H2O, Apache Traffic Server, nghttp2, Caddy and LiteSpeed have all released http2 capable servers.

H2O, Apache Traffic Server, nghttp2, Caddy ve LiteSpeed; bu yetenekli sunucuları hepsi http2'yi destekledi.

8.3.3 Diğerleri

curl ve libcurl, güvensiz http2'yi ve farklı TLS kütüphanelerini destekler.

Wireshark http2'yi destekliyor. Http2 ağ trafiğini analiz etmek için mükemmel bir araç.

8.4. Http2'nin ortak eleştirileri

Bu protokolün geliştirilmesi sırasında tartışmalar vardır ve tabi ki bu protokolün tamamen hatalı olduğuna inanılan belli bir miktarda insanlar var. Yaygın olan birkaç şikayetten bahsetmek ve onlara karşı oluşan argümanlardan bahsetmek istedim:

8.4.1. "Protokol, Google tarafından tasarlanmış veya üretilmiştir"

Dünyanın Google tarafından daha da bağımlı veya denetime tabi tutulduğunu ima eden önyargılar de vardır. Bu doğru değildir. Protokol, protokollerin 30 yılı aşkın bir süredir geliştirildiği şekilde IETF içinde geliştirildi. Bununla birlikte, hepimiz, Google'ın bu şekilde yeni bir protokol kurulmasının mümkün olduğunu kanıtlayan SPDY ile, yaptığı etkileyici çalışmaları tanıklık ettik ve bu kazanımlar ile neler yapılabileceğini gösteren numaralar sağladığını da kabul ediyoruz

Google has publicly announced that they will remove support for SPDY and NPN in Chrome in 2016 and they urge servers to migrate to HTTP/2 instead.

Google, 2016'da Chrome'da SPDY ve NPN'ye olan desteği kaldıracaklarını kamuoyuna ilan etti (https://blog.chromium.org/2015/02/hello-http2-goodbye-spdy.html) ve sunucuları geçiş yapılması için teşvik ediyorlar. Sunucularınızı HTTP / 2 olarak ayarlayın.

8.4.2. "Protokol yalnızca tarayıcılar için yararlıdır"

Bu doğru. Http2 gelişiminin ardındaki ana sürücülerden biri, HTTP boru hattının kurulumudur. Kullanım durumunuz başlangıçta boruhattı imkânına ihtiyaç duymuyorsa, http2 sizin için bir şey yapmaz. Protokoldeki tek gelişme kesinlikle kesinlikle bu değildir ama büyük bir gelişmedir.

Hizmetler, güç ve yetenekleri tam anlamıyla fark ettirmeye başladığında, tek bir bağlantı üzerinden birden çok akış getiriyor, http2'nin daha fazla uygulama kullanımını göreceğimizden şüphe yoktur.

Small REST APIs and simpler programmatic uses of HTTP 1.x may not find the step to http2 to offer very big benefits. But also, there should be very few downsides with http2 for most users.

Küçük REST API'leri ve HTTP 1.x'in daha basit programlı kullanımı, http2 için çok büyük yararlar sağlamak için bir adım bulamayabilir. Ancak aynı zamanda çoğu kullanıcı http2 ile ilgili çok az olumsuzluğa sahip olmalıdır.

8.4.3. "Protokol yalnızca büyük siteler için yararlıdır"

Not at all. The multiplexing capabilities will greatly help to improve the experience for high latency connections that smaller sites without wide geographical distributions often offer. Large sites are already very often faster and more distributed with shorter round-trip times to users.

Hepsi değil. Çoklama yetenekleri, geniş coğrafi dağılımlara sahip olmayan daha küçük sitelerin genellikle sundukları yüksek gecikme süresin ve bağlantı deneyimini iyileştirmeye yardımcı olacaktır. Büyük siteler zaten sıklıkla daha hızlıdır ve kullanıcılara daha kısa dolaşma süreleri ile daha fazla performans sağlarlar.

8.4.4. "TLS kullanımı yavaşlatıyor"

Bu, bir dereceye kadar doğru olabilir. TLS el sıkışması biraz ekstra para harcatabilir, ancak TLS için gereken gidişatları azaltmaya yönelik mevcut ve devam eden çabalar vardır. Düz metin yerine wire-hat? üzerinde TLS yapmak için yapılan ek yük çaba önemsiz değildir, daha fazla CPU ve güç, güvenli olmayan bir protokol ile aynı trafik modelinde harcanır. Ne kadar ve ne gibi bir etkiye sahip olacağı düşünülür. Bir bilgi kaynağı için istlsfastyet.com'a bakın.

Telekom ve diğer ağ operatörleri, örneğin ATIS Open Web Alliance'da, uydu, uçak ve benzeri ortamlarda hızlı bir web deneyimi sağlamak için önbellek, sıkıştırma ve diğer teknikler sunacaklarını söylüyorlarneed unencrypted traffic. http2, TLS kullanımını zorunlu kılmaz, bu nedenle şartları birleştirmemeliyiz.

Pek çok İnternet kullanıcısı, kullanıcıların gizliliğinin korunması gerekçesiyle TLS'in daha yaygın kullanılmasını tercih ettiğini belirtti.

Experiments have also shown that by using TLS, there is a higher degree of success than when implementing new plain-text protocols over port 80 as there are just too many middle boxes out in the world that interfere with what they would think is HTTP 1.1 if it goes over port 80 and might look like HTTP at times.

Deneyler ayrıca, TLS'yi kullanarak, bağlantı noktası 80 üzerinden yeni düz metin protokolleri uygularkenkinden daha yüksek bir başarı derecesine sahip olduğunu gösterdi; çünkü dünyada, HTTP1.1 olduğunu düşüneceklerine müdahale eden çok fazla orta kutu var 80 numaralı bağlantı noktasını aşıyor ve bazen HTTP gibi görünebilir.

TLS'i kullanan deneyler gosteriyor ki, 80 numaralı bağlantı noktasından yeni düz metin protokolleri uygularkenkinden daha yüksek bir başarı derecesi var çünkü 80 numaralı bağlantı noktasından geçerse HTTP1 olduğunu düşünerek çok fazla müdahale olabilir.?

Son olarak, tek bir bağlantı üzerinden HTTP2'nin çok katlı akışları sayesinde, normal tarayıcı kullanım örnekleri halen daha az TLS el sıkışmasıyla sonuçlanabilir ve böylece hala HTTP 1.1 kullanıldığında HTTPS'den daha hızlı performans gösterebilir.

8.4.5. "ASCII olmamak bir anlaşmazlıktır"

Evet, hata ayıklama ve izleme kolaylığı sağladığından açık bir şekilde protokolleri seviyoruz. Ancak metin tabanlı protokoller daha hataya eğilimli ve çok daha dönüştürme(parsing) problemlerine sahiptir.

If you really can't take a binary protocol, then you couldn't handle TLS and compression in HTTP 1.x either and its been there and used for a very long time.

8.4.6. "HTTP / 1.1'den daha hızlı değil"

Bu tabi ki daha hızlı olanı ölçmenin nasıl yapıldığı üzerine tartışmalara tabidir, ancak daha önce SPDY günlerinde daha fazla tarayıcı sayfası yüklendiğini kanıtlayan birçok test yapıldı (like "How Speedy is SPDY?" by people at University of Washington and "Evaluating the Performance of SPDY-enabled Web Servers" by Hervé Servy) ve bu tür denemeler de http2 ile tekrarlandı. Bu tür testlerin ve deneylerin yayınlanmaya başlaması için sabırsızlıkla bekliyorum. basic first test made by httpwatch.com, HTTP / 2'nin sözlerini tuttuğunu ima edebilir.

8.4.7. "Katman ihlalleri var"

Cidden, bu argümanınız bu mı? Katmanlar, küresel bir dinin kutsal dokunulmaz direkleri değildir ve eğer http2 yaparken birkaç gri alana girersek, verilen kısıtlamalar içinde iyi ve etkili bir protokol yapmakla ilgilenilmiştir.

8.4.8. "Birkaç HTTP / 1.1 eksikliğini gidermez"

Bu doğru. HTTP / 1.1 paradigmalarını sürdürmenin spesifik amacı ile birlikte, kalması gereken eski HTTP özellikleri de vardı. Sıklıkla korkulan çerezleri, yetkilendirme başlıklarını ve daha fazlasını içeren ortak başlıklar olduğu gibi. Fakat bu paradigmaların korunabilmesi le birlikte, protokol, temel parçaların bir yukseltmeye ihtiyaç duymdan tamamen değiştirilebilmesi veya yeniden yazılabilmesini mümkün kılar.

8.5. Http2 yaygın olarak kullanılacak mı?

Herhalde söylemek için henüz erken, fakat yine de tahmin edebilir ve burada yapacağım da budur.

On yıllarca süren ve yeni bir protokolün yaygınlaşması için örnek olarak naysayerler "IPv6'nın ne kadar iyi yapıldığına bakın" diyecek. http2 IPv6 değildir. Bu sıradan HTTP güncelleme mekanizmalarını ve port numaralarını ve TLS'yi kullanan TCP'nin üstünde bir protokoldür. Çoğu yönlendirici veya güvenlik duvarının hiç değişmesini gerektirmez.

Google, SPDY çalışmalarıyla dünyaya onu kanıtladı; bunun gibi yeni bir protokol tarayıcılar ve hizmetler tarafından çok sayıda uygulama ile oldukça kısa sürede kullanılabiliyor. Internet'te bugün SPDY'yi sunan sunucuların miktarı % 1 aralığında iken, bu sunucuların ele aldığı veri miktarı çok daha büyük. Bugün kesinlikle en popüler web sitelerinden bazıları SPDY'yi önriyor.

http2, SPDY ile aynı temel paradigmaları temel alarak, bir IETF protokolü olduğundan daha fazla konuşlandırılacağını söyleyebilirim. SPDY dağıtımı her zaman "bir Google protokolüdür" damgasının gerisinde kaldı.

Yayınlamanın arkasında birkaç büyük tarayıcı var. Firefox, Chrome, Safari, Internet Explorer ve Opera temsilcileri http2 özellikli tarayıcılar göndereceklerini ve çalışma uygulamalarını gösterdiklerini belirttiler.

Google, Twitter ve Facebook dahil olmak üzere yakında http2 sunacak pek çok büyük sunucu operatörü var ve http2'nin yakında Apache HTTP Sunucusu ve nginx gibi popüler sunucu uygulamalarına eklendiğini görmek istiyoruz. H2o, potansiyelini gösteren http2 desteğine sahip yeni ve inanılmaz hızlı bir HTTP sunucusudur.

HAProxy, Squid ve Varnish de dahil olmak üzere en büyük proxy sunucularından bazıları, http2'yi destekleme niyetlerini ifade ettiler.

2015 yılına kadar, http2 olan trafik miktarı artıyor. Google, yaklaşık% 18 oranında gelen HTTP / 2'yi görürken, Eylül ayının başında Firefox 40'ın kullanımı tüm HTTP trafiğinin% 13'ünde, tüm HTTPS trafiğinin% 27'sinde gerçekleşti. Google'ın, http2 kullanım düzeylerini başka türlü olabildiğince düşük kılan diğer yeni protokol deneylerini de çalıştırdığı unutulmamalıdır (bkz. QUIC 12.1).

9. Firefox'da http2

Firefox, taslakları yakından takip ediyor ve aylardır http2 test uygulamalarını sağlamıştır. Http2 protokolünün geliştirilmesi sırasında, istemciler ve sunucular, protokolün hangi taslak sürümü için testlerin uyguladıkları ile ilgili biraz can sıkıcı oldugu konusunda anlaşmışlardır.

9.1. Öncelikle açık olduğuna emin olun.

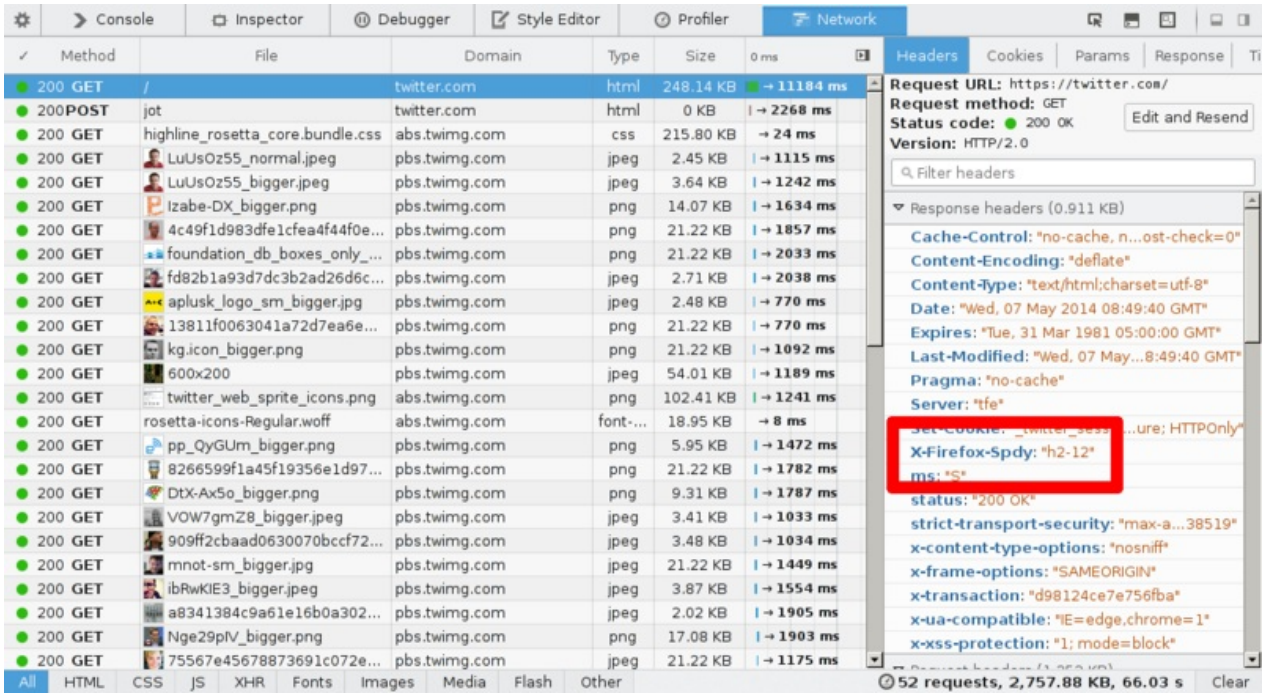
13 Ocak 2015 tarihinden itibaren piyasaya sürülen 35 sürümünden bu yana tüm Firefox sürümlerinde http2 desteği varsayılan olarak etkinleştirilmiştir.

Adres çubuğuna 'about: config' yazın ve "network.http.spdy.enabled.http2draft" adlı seçeneği arayın. *True* olarak ayarlandığından emin olun. Firefox 36, varsayılan olarak "true *" olarak ayarlanan "network.http.spdy.enabled.http2" adlı başka bir yapılandırma anahtarı ekledi. The latter one controls the “plain” http2 version while the first one enables and disables negotiation of http2-draft versions. Her ikisi de Firefox 36'dan beri varsayılan olarak geçerlidir.

9.2. TLS-only(Sadece-TLS)

Firefox'un yalnızca TLS üzerinden http2 uyguladığını unutmayın. Firefox'da yalnızca https: // sitelerine giderken http2'yi göreceksiniz.

9.3. Şeffaf!



Http2 kullanıldığını söyleyen herhangi bir UI öğesi yoktur. Bunu kolayca göremezsiniz. Bunu anlamanın bir yolu, "Web developer-> Network" özelliğini etkinleştirmek ve yanıt başlıklarını kontrol etmek ve sunucudan neyin geri geldiğini görmektir. Yanıt daha sonra "HTTP / 2.0" olur ve firefox, yukarıdaki ekran görüntüsünde gösterildiği gibi "X-Firefox-Spdy:" adlı kendi üstbilgisini ekler.

Http2 konuşurken Ağ aracında gördüğünüz başlıklar, http2'nin ikili biçiminden eski stil HTTP 1.x'e benzeyen başlıklara dönüştürülmüştür.

9.4. Http2 kullanımını görselleştir

Bir site http2 kullanıyorsa görselleştirmeye yardımcı olan Firefox eklentileri bulunur. Bunlardan biri “HTTP/2 and SPDY Indicator”.

10. Chromium'da http2

Chromium ekibi http2'yi uygulamaya uyguladı ve uzun süredir dev ve beta kanalında destek verdi. 27 Ocak 2015'te yayınlanan Chrome 40 ile başlayarak, http2 varsayılan olarak belirli bir kullanıcı grubu için etkinleştirilmiştir. Bu miktar zamanla kademeli olarak arttı.

SPDY desteği sonunda ortadan kaldırılacaktır. Bir blog yazısında, şu şekilde ifade edilmiştir: Şubat 2015:

> “"Chrome, Chrome 6'dan bu yana SPDY'yi destekledi, ancak avantajlarının çoğunun HTTP / 2'de mevcut olması nedeniyle, vedalaşma zamanı. 2016'nın başında SPDY'ye yönelik desteği kaldırmayı planlıyoruz "”

10.1. İlk olarak, etkinleştirildiğinden emin olun

Tarayıcınızın adres çubuğuna “chrome://flags/#enable-spdy4" yazın ve etkinleştirilmiş olarak göstermiyorsa "etkinleştir" seçeneğini tıklayın.

10.2. TLS-only(Sadece-TLS)

Firefox'un yalnızca TLS üzerinden http2 uyguladığını unutmayın. Firefox'da yalnızca https: // sitelerine giderken http2'yi göreceksiniz.

10.3. Http2 kullanımını görselleştir

Bir site http2 kullanıyorsa görselleştirmeye yardımcı olan Firefox eklentileri bulunur. Bunlardan biri “HTTP/2 and SPDY Indicator”.

10.4. QUIC

Chrome'un QUIC ile gerçekleştirdiği deneyler (see section 12.1) HTTP/2 çalışmalarını etkiler.

11. curl içerisinde http2

curl projesi, Eylül 2013'ten beri deneysel http2 desteği sağlıyor.

curl ruhu içinde, mümkün olduğunca http2'nin her yönünü desteklemeyi düşünüyoruz. curl sıklıkla bir test aracı ve web sitelerinde takla(poke on) atmanın yolu olarak kullanılır ve bunu http2 için de tutmak niyetindeyiz.

curl, http2 çerçeve katmanı işlevselliği için ayrı kütüphane nghttp2 kullanır. curl, nghttp2 1.0 veya üstünü gerektirir.

Şu anda linux'da curl ve libcurl her zaman HTTP / 2 protokol desteği etkin değildir.

11.1. HTTP 1.x benzerlikleri

Dahili olarak, curl gelen HTTP2 üstbilgilerini HTTP 1.x stil üstbilgilerine dönüştürür ve kullanıcıya sunar; böylece mevcut HTTP'ye çok benzer görünürler. Bu, curl ve HTTP'yi bugün kullananlar için daha kolay bir geçiş sağlar. Benzer şekilde curl, giden üstbilgileri aynı stilde dönüştürür. Onları HTTP 1.x tarzında curl'e verin ve http2 sunucularıyla konuşurken bunları anında dönüştüreceksiniz. Böylece, kullanıcıların hat(wire) üzerinde gerçekten kullanılan belirli HTTP sürümleriyle uğraşmasına veya bakım yapmasına gerek kalmamaktadır.

11.2. Düz metin, güvensiz

curl, HTTP2'yi Standart TCP üzerinden Upgrade: başlığı üzerinden destekler. Bir HTTP isteği yaparsanız ve HTTP 2'yi sorarsanız, curl, sunucudan mümkünse http2 bağlantısını güncellemesini isteyecektir.

11.3. TLS ve bazı kütüphaneler

curl, TLS arka uç için geniş bir yelpazede farklı TLS kütüphanelerini destekler ve bu hala http2 desteği için geçerlidir. Http2'nin uğruna TLS ile olan meydan okuma ALPN desteğidir ve bir ölçüde NPN desteğidir.

Hem ALPN hem de NPN desteği almak için curl'ü OpenSSL veya NSS'nin modern sürümlerine karşı oluşturun. GnuTLS veya PolarSSL'yi kullanarak NPN değil ALPN desteği elde edersiniz.

11.4. Komut satırı kullanımı

Curl'e http2'yi (düz metin veya TLS) kullanmasını söylemek için `--http2` seçeneğini (yani" tire çizgisi http2 ") kullanırsınız. curl hâlâ HTTP / 1.1 varsayılanıdır, bu nedenle HTTP2'yi istediğinizde ekstra seçenek gereklidir.

11.5. libcurl seçeneği

11.5.1 Etkin HTTP/2

Uygulamanız normal gibi https: // veya http: // URL'leri kullanıyor ancak libcurl'un http2 kullanmaya teşvik etmek için curl_easy_setopt'ın `CURLOPT_HTTP_VERSION` seçeneğini `CURL_HTTP_VERSION_2` olarak ayarlamalısınız. Daha sonra elinden gelen çabayı gösterebilir ve http2 yapabilir, aksi halde HTTP 1.1 ile çalışmaya devam eder.

11.5.2 Çoğullama

Libcurl mevcut davranışları büyük ölçüde korumaya çalıştığından, uygulamanız için CURLMOPT_PIPELINING seçeneği ile HTTP / 2 çoğullama özelliğini etkinleştirmeniz gerekir. Aksi takdirde, her bağlantı için bir defada bir istek kullanmaya devam edecektir.

Another little detail to keep in mind is that if you ask for several transfers at once with libcurl, using its multi interface, an applicaton can very well start any number of transfers at once and if you then rather have libcurl wait a little to add them all over the same connection rather than opening new connections for all of them at once, you use the CURLOPT_PIPEWAIT option for each individual transfer you rather wait.

Akılda tutulması gereken diğer bir küçük ayrıntı da, bir seferde libcurl ile çoklu aktarım isterseniz, çoklu arayüzü kullanmak, bir uygulama aynı anda herhangi bir sayıda aktarmaya başlayabilir ve daha sonra libcurl'yı eklemek için biraz beklemek zorunda kalırsanız hepsinin aynı anda birden fazla bağlantı kurmasına değil, aynı bağlantıya her seferinde beklediğiniz her bir aktarım için [CURLOPT_PIPEWAIT] CURLOPT_PIPEWAIT seçeneğini kullanabilirsiniz.

11.5.3 Sunucu itme

libcurl 7.44.0 ve sonrası, HTTP / 2 sunucu itme özelliğini destekler. CURLMOPT_PUSHFUNCTION seçeneği ile bir geri arama geri alma kurarak bu özelliğin avantajından yararlanabilirsiniz. Baskı uygulama tarafından kabul edilirse, başka herhangi bir aktarımda olduğu gibi, CURL kolay işleyici olarak yeni bir aktarım oluşturacak ve içeriği teslim edecektir.

12. http2 sonrası

Http2 için çok zor kararlar ve uzlaşmalar yapılmıştır. Http2'nin dağıtılmasıyla birlikte, ileride daha fazla protokol revizyonu yapmanın temelini oluşturan diğer protokol sürümlerine yükseltme için önceden belirlenmiş bir yol vardır. Aynı zamanda birden fazla farklı versiyonu paralel olarak işleyen bir kavram ve bir altyapı da getiriyor. Belki yeni tanıttığımızda eskilerini tamamen ortadan kaldırmamız gereklidir?

HTTP2, HTTP 1 ve http2 arasında ileri geri trafiği proxy vasıtasıyla tutma arzusundan dolayı, geleceğe getirilen bir sürü HTTP 1 "miras" içeriyor. Bu mirastan bazıları daha fazla gelişme ve icatlara engel oluyor. Belki de http3 bazılarından kurtulabilir.

Hâlâ http'de neyin eksik olduğunu düşünüyorsunuz?

12.1. QUIC

Google'ın [QUIC] (https://www.chromium.org/quic) (Hızlı UDP İnternet Bağlantıları) protokolü, SPDY ile aynı tarzda ve ruhta çok ilginç bir deneydir. QUIC, UDP kullanılarak gerçekleştirilen TCP + TLS + HTTP / 2 birleşimidir.

QUIC, çok daha az gecikme ile bağlantıların oluşturulmasına izin verir, sadece HTTP / 2 için olduğu gibi her biri için değil, bireysel akışları engellemek için de paket kaybını çözer ve farklı ağ arayüzleri üzerinden kolayca bağlantı yapılmasını sağlar, dolayısıyla MPTCP'nin çözeceği alanları da kapsar.

QUIC şimdiye kadar yalnızca Google tarafından Chrome'da uygulanmaktadır ve bu kod, bir [libquic] (https://github.com/devsisters/libquic) çabasıyla tam olarak çalışılsa bile başka yerlerde kolayca yeniden kullanılamaz. Protokol IETF ulaştırma çalışma grubuna [taslak] (https://tools.ietf.org/html/draft-tsvwg-quic-protocol-01) olarak getirildi.

13. Daha fazla bilgi için

Bu belgenin içeriği veya teknik ayrıntıları biraz aydınlattığını düşünüyorsanız merakınızı gidermenize yardımcı olacak ek kaynaklar aşağıda belirtilmiştir:

- HTTPbis posta listesi ve arşivleri: https://lists.w3.org/Archives/Public/ietf-http-wg/
- HTML2 sürümünde asıl http2 belirtimi: https://httpwg.github.io/specs/rfc7540.html
- Firefox http2 ağ ayrıntıları: https://wiki.mozilla.org/Networking/http2
- curl http2 uygulama ayrıntıları: https://curl.haxx.se/docs/http2.html
- Http2 web sitesi: https://http2.github.io/ and perhaps in particular the FAQ: https://http2.github.io/faq/
- Ilya Grigorik'in HTTP / 2 bölümü “High Performance Browser Networking”("Yüksek Performans Tarayıcı Ağı") kitabında: https://hpbn.co/http2/

14. Teşekkürler

Mark Nottingham'dan ilham ve paket formatı Lego görüntüsü.

HTTP eğilim verileri https://httparchive.org/'dan gelir.

RTT grafiği, Mike Belshe tarafından yapılan sunumlardan geliyor.

Çocuklarım Agnes ve Rex, satir başı engelleme için Legolarından ödünç almama izin verdikleri için.

İncelemeler ve geri bildirimler için aşağıdaki arkadaşlara teşekkür ederiz: Kjell Ericson, Bjorn Reese, Linus Swälas ve Anthony Bryan. Yardımınız çok değerlidir ve belgeyi gerçekten geliştirmiştir!

Çeşitli tekrarlamalar sırasında, aşağıdaki samimi insanlar belgede hata raporları ve geliştirmeler sağladı: Mikael Olsson, Remi Gacogne, Benjamin Kircher, saivlis, florin-andrei-tp, Brett Anthoine, Nick Parlante, Matthew King, Nicolas Peels, Jon Forrest, sbrickey, Marcin Olak, Gary Rowe, Ben Frain, Mats Linander, Raul Siles, Alex Lee, Richard Moore