Universidade do Minho
Escola de Engenharia

Comunicações e Redes

TP8: HTTP e Pipelining

João Miguel da Silva Alves (83624)

Paulo Jorge Alves (84480)

MESTRADO INTEGRADO EM ENGENHARIA BIOMÉDICA

INFORMÁTICA MÉDICA 2020/2021

PARTE I

Explique no que consiste HTTP com pipelining. Como funciona? Quando foi introduzido? Quais são as vantagens/desvantagens da sua utilização?

Normalmente, as solicitações HTTP são emitidas sequencialmente, com a próxima solicitação a ser emitida somente após a resposta à solicitação atual ter sido completamente recebida. Dependendo das latências da rede e das limitações de largura de banda, isto pode resultar num atraso significativo antes que a próxima solicitação seja vista pelo servidor.

O HTTP / 1.1 permite que várias solicitações HTTP sejam gravadas num *soquet* juntos, sem esperar pelas respostas correspondentes. O solicitante espera que as respostas cheguem na ordem em que foram solicitadas. O ato de *canalizar* as solicitações pode resultar numa melhoria dramática nos tempos de carregamento da página, especialmente em conexões de alta latência.

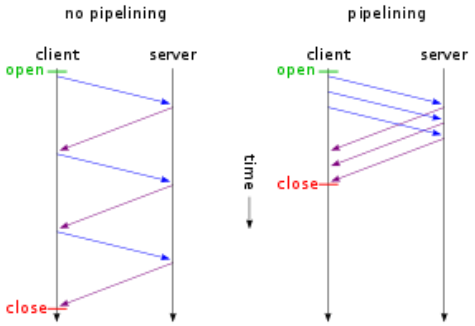

Figura 1 – sem pipelining vs com pipelining

Então, o HTTP com Pipelining é uma técnica na qual várias solicitações HTTP são enviadas numa única conexão TCP, sem esperar pelas respostas correspondentes. Assim, reduz-se drasticamente o número de pacotes TCP usados. Esta redução do número de pacotes necessários para carregar uma página beneficia a Internet como um todo, pois menos pacotes reduz naturalmente a carga sobre routers e redes IP.

Os servidores em conformidade com HTTP / 1.1 são necessários para suportar pipelining.

O HTTP/1.1 foi documentado pela primeira vez na RFC 2068 em 1997 e, a partir de 2020 tem uso minoritário. O HTTP/2 é a expressão mais eficiente da semântica do HTTP "on the wire", e foi publicado em 2015 e é usado por 50,0% dos sites; agora é suportado por praticamente todos os navegadores da web. O HTTP/3 é o sucessor proposto para HTTP / 2, que já é usado por mais de 4% dos sites, e é usado por mais de 5% dos computadores desktop (habilitado por padrão no macOS), usando UDP em vez de TCP para o protocolo de transporte subjacente.

Como já referido, a principal vantagem do uso do pipelining é a melhoria nos tempos de carregamento das páginas HTML, especialmente em conexões de alta latência (ex: Internet).

No entanto, existem desvantagens no uso do mesmo. Se várias solicitações forem enviadas em paralelo, e uma das solicitações demorar mais para processar, as respostas estarão fora de ordem. Uma vez que o HTTP é um protocolo sem estado, o cliente não tem como combinar as solicitações com as respostas, o que se torna num problema. Outra desvantagem está relacionada com o bloqueio *head-of-line* (bloqueio HOL), que corresponde a um fenómeno de limitação de desempenho que ocorre quando uma linha de pacotes é retida pelo primeiro pacote. [1][2][3]

Os browsers modernos utilizam este mecanismo?

Não.

O Pipelining foi substituído pela multiplexação via HTTP/2, que é suportada pela maioria dos navegadores modernos. Esta multiplexação permite que o navegador dispare várias solicitações ao mesmo tempo na mesma conexão e receba as solicitações de volta em qualquer ordem, ou até mesmo quebrar cada arquivo solicitado em pedaços e misturar os arquivos. O navegador da web está então encarregue de juntar todas as peças novamente. [4]

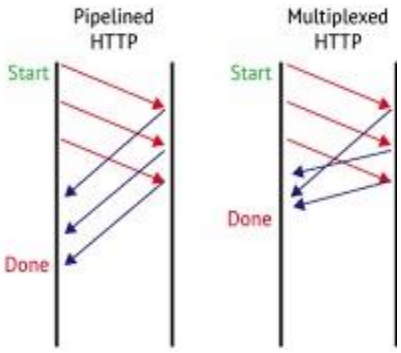

Figura 2 - HTTP/1 vs HTTP/2

Que soluções alternativas existem em versões mais recentes do HTTP?

No HTTP/1.1 com pipelining podem-se solicitar vários pedidos, um após o outro, sem esperar, mas todos eles chegam na ordem em que foram pedidos. Tem as desvantagens já abordadas anteriormente e, por isso, não é utilizado.

No HTTP/2 podem-se solicitar vários pedidos em qualquer ordem, sem atrasos. As respostas podem chegar numa ordem diferente, e pode-se até mesmo dividir os pedidos para que algumas partes desse pedido cheguem primeiro. Em última análise, isso deve significar que se obterá tudo mais rápido no geral e se poderá começar a trabalhar em cada item assim que chegar.

Este HTTP deixa toda a semântica de alto nível do HTTP/1.1, como métodos, códigos de status, e URLs iguais. A novidade aqui é como os dados são enquadrados e transportados entre o cliente e o servidor. É minimizado o número de solicitações necessárias para renderizar uma página inteira, sem reduzir a sua capacidade de funcionamento. O HTTP/2 permite que o servidor "empurre" o conteúdo, ou seja, responda com dados para mais consultas do que o

cliente solicitou. Isto permite que o servidor forneça dados que sabe que um navegador da web precisará para renderizar uma página da web, sem esperar que o navegador examine a primeira resposta.

Implementações de grandes navegadores tornaram um requisito para os websites implementar criptografia (SSL) para que fossem capazes de colher os benefícios do HTTP/2 e em algumas situações, isso incorreu numa sobrecarga computacional que deixou as melhorias na velocidade impercetíveis. Em alguns casos, os utilizadores relatavam lentidão após a transição para HTTP/2.

Embora esta versão tenha nos dado multiplexação e mitigado o bloqueio de head-of-line, o HTTP/2 é limitado por TCP. Pode-se usar uma conexão única de TCP para múltiplas correntes multiplexadas em conjunto para transferir dados, mas quando uma delas sofre a perda de um pacote, a conexão inteira é mantida refém, até que TCP faça seu papel (retransmitir o pacote perdido), ou seja, todos os pacotes, mesmo que já tenham sido transmitidos e estejam a aguardar no buffer do nó de destino, são bloqueados até que o pacote perdido seja retransmitido.

Daí surge a necessidade do HTTP/3, uma revisão importante que se baseia nos conceitos estabelecidos pelo HTTP/2. É um rascunho baseado num rascunho RFC anterior, denominado "HTTP sobre QUIC".

Com um foco em correntes UDP (User Datagram Protoco – protocolo simples da camada de transporte), QUIC alcança multiplexação sem ter de ficar sobre uma conexão TCP, ou seja, os pacotes não são forçados a aguardar até que os perdidos sejam retransmitidos. Estas conexões também tem a vantagem de eliminar a sobrecarga do handshake do TCP, o que reduz a latência.

Atualmente, 6.53% dos navegadores de computador suportam HTTP/3, enquanto nenhum navegador móvel o suporta. [1][5][6]

PARTE II

O Pale Moon é um programa que, por defeito, utiliza HTTP com pipelining. É um navegador web de código aberto com ênfase em customização, como expressado no seu lema "*Your browser, Your way*".

Após a instalação do *Pale Moon*, acederam-se às suas configurações. (figura 3)

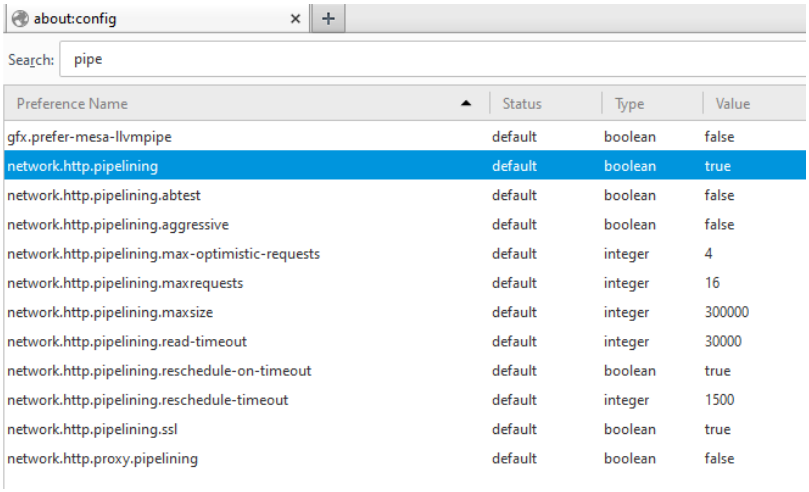

Figura 3 - about:config no Pale Moon

Desta forma, conseguiu-se verificar que este navegador usa, por defeito, pipelining em HTTP.

De seguida, com o auxílio do *Wireshark*, obtiveram-se capturas do acesso ao website: http://www.nyu.edu/, usando o browser habitual (*Chrome*) e usando o browser *Pale Moon*. Ambas as pesquisas foram realizadas em separadores anónimos de forma a não existir cache no momento das capturas. Para analisar e comparar ambas as pesquisas, e tal como sugerido, filtrou-se os pacotes obtidos pelo tipo HTTP e acedeu-se à representação gráfica dos mesmos. Os resultados obtidos, com o uso do *Chrome* e com o uso do *Pale Moon* estão representados nas figuras 4 e 5, respetivamente.

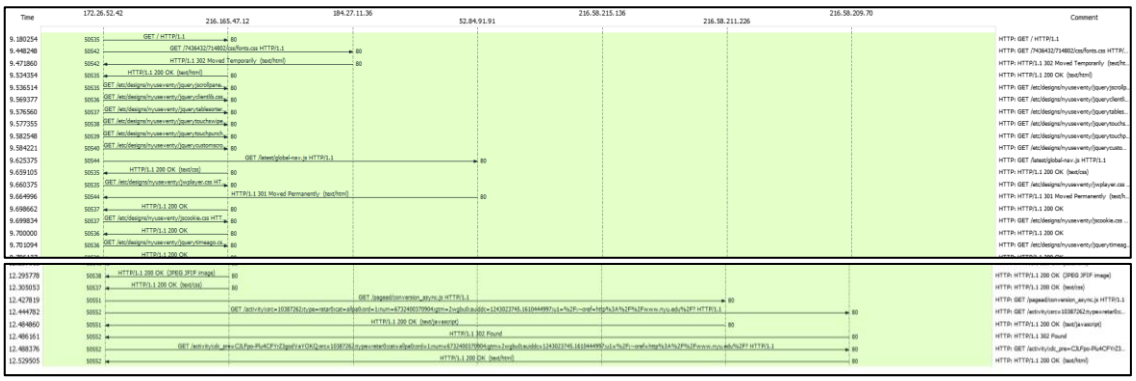

Figura 4 - Flow Graph com filtro "http" para captura do Chrome (apenas está representada a parte inicial e final)

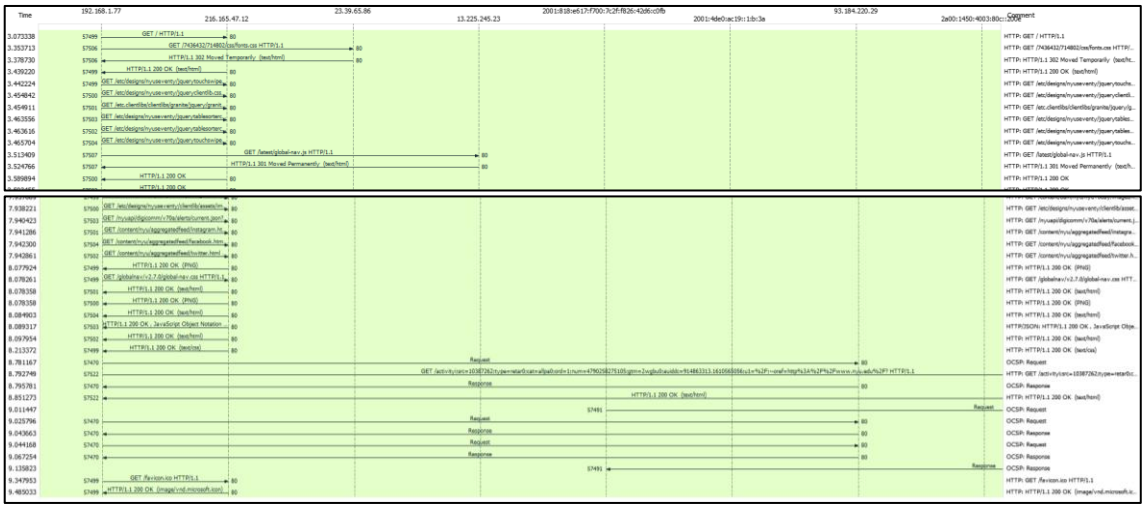

Figura 5 - Flow Graph com filtro "http" para captura do Pale Moon (apenas está representada a parte inicial e final)

Analisando as duas figuras acima, observa-se que para a captura obtida do Chrome (figura 4), existem, no início, sequências de vários pedidos a serem feitos. No entanto, verifica-se que na sua grande parte, as sequencias são do tipo "pedido-resposta". Isto porque, tal como mencionado acima, os browsers comuns não utilizam HTTP em pipelining. Em relação à figura 5 (captura obtida do *Pale Moon*), verifica-se que existem sequências de perguntas seguidas de sequências de respostas. Isto prova o que já foi dito anteriormente, que o Pale Moon faz uso do HTTP com pipelining.

De forma a analisar a interferência do aumento do número máximo de pedidos para 30, voltou a realizar-se a captura do Pale Moon (figura 7). Começou-se por alterar as definições do *Pale Moon* relativas ao número máximo de pedidos permitidos, de 16 (default) para 30 (figura 6).

about:config

Search: pipe

Preference Name	Status	Type	Value
gfx.prefer-mesa-llvmpipe	default	boolean	false
network.http.pipelining	default	boolean	true
network.http.pipelining.abtest	default	boolean	false
network.http.pipelining.aggressive	default	boolean	false
network.http.pipelining.max-optimistic-requests	default	integer	4
network.http.pipelining.maxrequests	**user set**	**integer**	**30**
network.http.pipelining.maxsize	default	integer	300000
network.http.pipelining.read-timeout	default	integer	30000
network.http.pipelining.reschedule-on-timeout	default	boolean	true
network.http.pipelining.reschedule-timeout	default	integer	1500
network.http.pipelining.ssl	default	boolean	true
network.http.proxy.pipelining	default	boolean	false

Figura 6 - about:config no Pale Moon

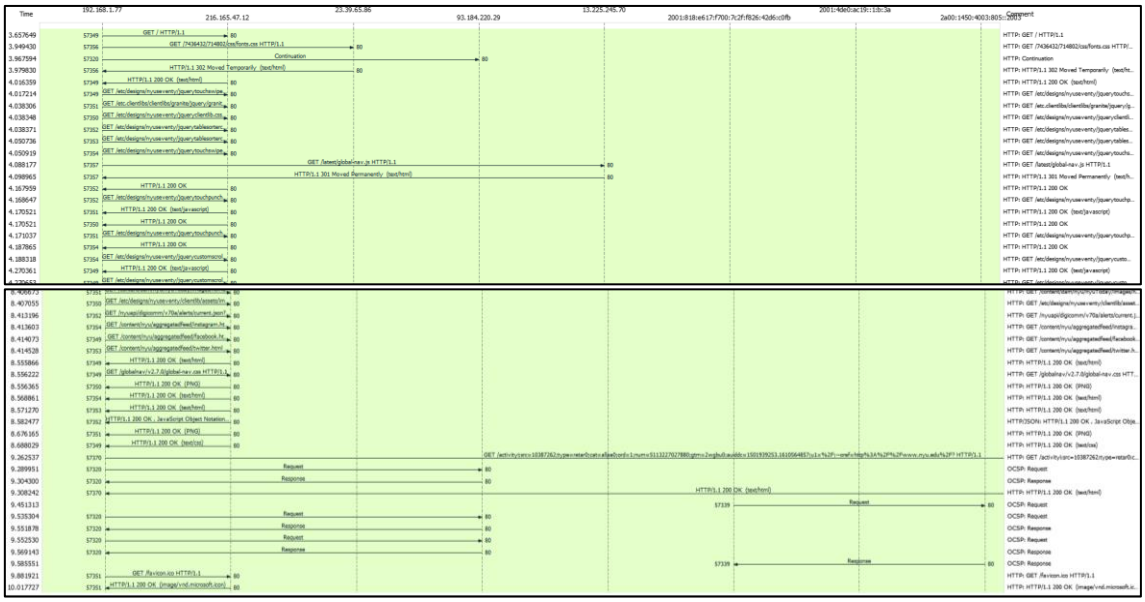

Figura 7 - Flow Graph para captura do Pale Moon - 30 pedidos (apenas está representada a parte inicial e final)

Ao alterar o valor do número máximo de pedidos permitidos para 30, estamos a permitir que se façam 30 pedidos de objetos consecutivos.

Nas condições ideais, os 30 pedidos serão respondidos também consecutivamente, sem falhas, e teremos eventualmente um tempo de descarga de objetos mais curto (e.g. pequenos objetos podem ser colocados num único pacote TCP/IP, aumentando a eficiência).

Mas, normalmente, colocar um valor elevado nesse parâmetro pode levar à existência de problemas. Mesmo que se descarregue os objetos rapidamente do servidor, se a quantidade destes for muito elevada, o browser vai demorar mais tempo a carregar a página, pois os objetos estão todos em espera no pipeline para se fazer display (ao invés de imprimir aos poucos no ecrã, parecendo mais fluído). Os próprios servidores, se receberem muitos pedidos consecutivos, podem ficar sobrecarregados e demorar mais tempo a responder, ou até mesmo fechar a conexão (podem, por exemplo, suspeitar que se trata de um ataque DoS).

Analisando, então, as diferenças temporais entre o tempo registado para o primeiro pedido de http e a última resposta obtém-se um menor intervalo de tempo com o número máximo de 30 pedidos, que corresponde ao esperado como descrito no segundo parágrafo a seguir à figura 7.

Bibliografia:

[1] https://en.wikipedia.org/wiki/HTTP_pipelining

[2] https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Gerenciamento_de_Conex%C3%A3o_em_HTTP_1.x

[3] https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-PT&prev=search&pto=aue&rurl=translate.google.com&sl=en&sp=nmt4&u=https://www-archive.mozilla.org/projects/netlib/http/pipelining-faq.html&usg=ALkJrhj5-IfmQvGSPbL4sW_2tUtEI8X21A

[4] https://qastack.com.br/programming/36517829/what-does-multiplexing-mean-in-http-2

[5] https://pt.wikipedia.org/wiki/HTTP/3

[6] https://kinsta.com/pt/blog/http3/