Daniel Q. Miranda

Monografia Preliminar - Implementação do Protocolo HTTP2 na linguagem Scala

São Paulo, Brasil

Outubro 2015

Daniel Q. Miranda

Monografia Preliminar - Implementação do Protocolo HTTP2 na linguagem Scala

Instituto de Matemática e Estatística - Universidade de Sâo Paulo

Orientador: Prof. Dr. Daniel Macêdo Batista

São Paulo, Brasil
Outubro 2015

Resumo

Resumo linha 1
Resumo linha 2
Palavras-chave: feio, chato, bobo.

Abstract

Abstract line 1
Abstract line 2
Keywords: ugly, boring, fool.

Sumário

Introdução

O protocolo de comunicação de rede HTTP/2 (HyperText Transfer Protocol, versão 2) foi definido e aceito pela Internet Engineering Task Force (IETF) em 2015 (RFC 7540). Incumbente sucessor do HTTP/1, formalizado em 1996, o HTTP/2 visa se adequar a demandas da Internet moderna, que conecta bilhões de dispositivos de todos os portes e conduz múltiplos petabytes de dados todos os dias.

Considerado um dos alicerces da Internet, o HTTP foi criado como método de transmissão para documentos, mas hoje carrega todo tipo de mídia - como aplicações complexas, áudio e vídeo - devido a sua flexibilidade. Observou-se porém que ineficiências obrigam desenvolvedores e administradores de infraestrutura a adotar soluções complexas para alcançar metas de desempenho e eficiência.

Visando sanar essas deficiências o HTTP/2, dentre outras melhorias:

- Introduz novos mecanismos para explorar ao máximo recursos de rede disponíveis, em especial através do paralelismo de transmissões em uma única conexão TCP. (vide Modelo de protocolo da Internet e Multiplexação).
- Substitui uma representação textual, na qual delimitadores de conteúdo e definições são sequencias especiais de caracteres, por uma binária, onde diversos tipos de mensagens são definidos com métodos de codificação individuais. Essa mudança permite a utilização de operações mais complexas, porém mais eficientes. (vide Formato Binário)
- Permite que servidores listem de antemão recursos relacionados a outros, para que clientes possam adquiri-los em simultâneo. Esse mecanismo visa acelerar em especial carregamento de páginas e aplicações Web complexas, que incluem dezenas de recursos externos.

Devido a participação de gigantes de várias áreas de atuação da Internet em sua confecção, e a base no protocolo SPDY, o HTTP/2 já dispõe de múltiplas implementações disponíveis tanto no âmbito de clientes quanto servidores. Navegadores (Google Chrome, Mozilla Firefox) e servidores (Apache, ngnix) altamente populares já o suportam, ao menos em caráter experimental. Existem, porém, oportunidades para novas implementações explorando diferentes metodologias e aspirações.

(TODO: links dos softwares acima)

Este trabalho descreverá uma implementação do protocolo HTTP/2, com facetas de cliente e servidor, na linguagem de programação orientada a objetos e funcional Scala, utilizando o modelo de atores (Hewitt 1977) implementado pelo conjunto de ferramentas Akka. Esta combinação foi escolhida devido a características como:

- Expressividade da linguagem, permitindo elegância e coesão de implementação
- Eliminação de compartilhamento de dados entre entidades concorrentes, facilitando a escalabilidade para múltiplos processadores e máquinas
- Maturidade da plataforma Java e seus ecossistema de software
- Facilidade de uso e eficiência da Akka para criação de sistemas concorrentes eficientes

(TODO: inserir citação do site da Akka e/ou exemplos de sistemas)

Denominar-se-a daqui para frente o protocolo original HTTP e suas revisões até 1.1 como "HTTP/1". A versão 2 definida em 2015 será mencionada como "HTTP/2", e aspectos comuns a ambas serão referidos genericamente como "HTTP"

O modelo de protocolo da Internet

(TODO: explicar stack de protocolos e papel do HTTP, comparar uso do TCP do HTTP/1 e HTTP/2, eficiência de 3-way handshake)

Motivação - O protocolo HTTP/2

Como sucessor do HTTP/1, o HTTP/2 foi desenvolvido para atender a diversos casos de uso de comunicação na Internet, dos quais um dos mais utilizados é a visualização de documentos e aplicações na Web. Para alcançar tais objetivos utiliza mecanismos mais complexos que a comunicação textual delimitada por linhas da primeira versão.

Formato Binário

O HTTP define objetos a serem transmitidos como conjuntos de *headers*, parâmetros e informações de um *pedido* ou *resposta* delimitados por linhas, seguidos de octetos de uma mensagem. Este modelo é suficiente para o caso original de transmissão de páginas Web simples, mas é falho para representar interações mais complexas que uma única transmissão de mensagem simultânea.

O HTTP/2 substitui a representação textual por um conjunto de mensagens de formato binario, que podem definir parâmetros de conexão, transmissão de headers, corpo

de mensagens, abertura de múltiplos canais, e ainda requisição de pré-carregamento de recursos além do sujeito atual.

(TODO: reescrever, expandir)

Compressão de Headers (HPACK)

O HTTP/2 define um sub-formato chamado HPACK para transmissão eficiente de metadados (*headers*) de mensagens, motivado por deficiências de segurança em métodos utilizados anteriormente em conjunto com encriptação. Tentativas anteriores de combinar compressão e privacidade mostraram-se vulneráveis a ataques, que alcançaram a extração de informações através da observação de padrões estatísticos nos dados comprimidos (CRIME (Kelsey 2002), BREACH (Prado, Harris, and Gluck 2013)).

O HPACK supera estas dificuldades utilizando técnicas e algoritmos mais simples, e talvez menos eficientes, mas que até o presente momento se mostram resistentes a ataques.

A necessidade de proteção de conteúdo de *headers* se deve principalmente ao uso dos *cookies* - conjuntos de dados enviados por um servidor para identificar um cliente, e retransmitidos pelo segundo em toda requisição.

(TODO: Explicação detalhada de cookies baseada no Kurose)

Três mecanismos são utilizados pelo HPACK para diminuir a banda de transmissão de *headers* sem comprometer a privacidade:

1. **Tabela estática de conteúdos pré-definidos** Uma lista de conteúdos mais comuns é compartilhada entre todos os programas que implementam o HTTP/2. Cada chave, valor, ou conjunto chave-valor que seja definido na tabela pode ser transmitido somente como um índice inteiro, substituindo a cadeia de carácteres usual.

2. **Tabela dinâmica de conteúdos transmitidos na conexão** Cada transmissão de *header*, caso uma exceção não seja explicitamente requisitada, é armazenada em uma *tabela dinâmica* de entradas. Repetições futuras de conteúdos equivalentes podem ser substituídas por um índice, explorando semelhanças entre mensagens de uma mesma conexão.

3. **Compressão de literais por codificação de Huffman** *Headers* que não possam fazer uso das regras anteriores podem ser comprimidas com um *Código de Huffman* simples, com símbolos de substituição pré-definidos, escolhidos para máxima compressão com base em amostras de tráfego real. Esse esquema não possui nenhuma fraqueza estatística conhecida.

Multiplexação e controle de fluxo

O HTTP/2 define um sistema de múltiplos fluxos de dados (*streams*) simultâneos. Eles representam canais de comunicação independentes e bidirecionais, pelos quais mensagens podem trafegar entre os interlocutores. Transmissões de vários fluxos podem transitar em uma única conexão TCP através da intercalação de *frames* entre eles.

Fluxos podem ser criados e destruídos de maneira individual e (fix: independente) do ciclo de vida da conexão como um todo. Cada um é atribuído uma *janela de controle de fluxo*, que pode ser usada por um interlocutor para restringir a velocidade de transmissão de dados em curso. Um remetente deve restringir suas transmissões caso a janela de um dado fluxo esteja totalmente utilizada, permitindo, por exemplo, que dispositivos com recursos escassos não sejam sobrecarregados.

Este mecanismo permite que diversos objetos sejam transmitidos simultaneamente, como ocorre muito comummente em páginas Web complexas, sem estressar a infraestrutura da camada de transmissão sobre a qual o HTTP existe.

(TODO: comparar com pipelining do HTTP/1.1, explicar control de fluxo e janela, comparar com conceitos similares do TCP)

Priorização

É possível atribuir dependências e níveis de prioridades a fluxos distintos, indicando maior importância ou urgência a mensagens neles enviadas.

Fluxos subordinados só podem receber recursos caso seus superiores estejam ociosos ou em espera. Num mesmo nível da hierarquia, recebem recursos proporcionais a um valor inteiro atribuído a cada um, representando uma fração do total disponível. (por exemplo, 3 fluxos de prioridades 3, 5 e 10, respectivamente receberiam 3/18, 5/18 e 10/18 da banda de transmissão disponível)

(TODO: expandir)

Conceitos e tecnologias

A Linguagem Scala

Scala é uma linguagem multi-paradigma, que tem como principal característica a combinação da orientação a objetos, compatível com a plataforma Java, e da programação funcional.

Possui um sistema de tipos poderoso, com funções de primeira-classes, tipos parametrizados e inferência local, e permite expressar sucintamente programas de diversos tipos.

Foi escolhida para elaboração desse trabalho devido a alta maturidade do ecossistema Java e da plataforma Akka, seu alto poder expressivo e foco em estruturas de dados imutáveis, concorrência e elegância.

A Plataforma Akka

Akka é uma plataforma que implementa o modelo de atores para criação de sistemas concorrentes, tolerantes a falhas e de alta escalabilidade em Scala ou Java. Permite programar fluxos de dados de maneira assíncrona e eficiente, incluindo servidores TCP como é necessário para utilização do HTTP.

Atores comunicam-se somente através de passagem de mensagens discretas, o que é conducente a diminuição de dependências e estado compartilhado, e a modularização dos componentes do software.

(TODO: Expandir)

(TODO: Incluir exemplos de código/fluxograma/modelagem, comparar com outros modelos)

O paradigma funcional e imutabilidade

TODO

O modelo de Atores

TODO

Implementação

Testes

TODO

Verificação de entradas e saídas

TODO

Testes de integração com aplicações reais

TODO

Aplicação da Codificação de Huffman sem árvores

O mapeamento estático de símbolos da Codificação de Huffman do HPACK correlaciona octetos com códigos que variam entre 5 e 31 bits. Ao contrário dos usos mais comuns desse algoritmo, onde o mapeamento é construído a partir de cada mensagem a ser comprimida, a tabela é pré-definida, igual para todas as mensagens, e de tamanho pequeno (256 entradas, suficientes para um octeto e um símbolo especial de fim de mensagem).

Estas particularidades permitem que seja adotada uma implementação mais simples e eficiente que a tradicional. Em vez de construir uma arvore de prefixos com nós correspondentes a um bit, e processar a entrada bit-a-bit, é possível particionar os códigos por seu tamanho, ler os dados octeto-a-octeto, e acessar um índice corresponde em uma tabela identificando o símbolo correspondente.

(falar O(n) vs O(log N), mas constante pequena))

(TODO: Expandir, incluir algoritmo)

Biblioteca de comunicação *Akka I/O*

TODO

Controle de fluxo de *streams*

TODO

Interface de programação

TODO

Referências

Hewitt, Carl. 1977. “Viewing Control Structures as Patterns of Passing Messages.” *Artificial Intelligence* 8 (3): 323–64. doi:http://dx.doi.org/10.1016/0004-3702(77)90033-9.

Kelsey, John. 2002. “Compression and Information Leakage of Plaintext.” In *Fast Software Encryption*, edited by Joan Daemen and Vincent Rijmen, 2365:263–76. Lecture Notes in Computer Science. Springer Berlin Heidelberg. doi:10.1007/3-540-45661-9_21.

Prado, Angelo, Neal Harris, and Yoel Gluck. 2013. “SSL, Gone in 30 Seconds.” In *Black Hat USA*. http://breachattack.com/resources/BREACH%20-%20BH%202013%20-%20PRESENTATION.pdf.