Lista 2 - Redes de Computadores

Mateus Carvalho Gonçalves - 201810245
Otávio Augusto Sousa Resende - 201810543

1. Na arquitetura P2P, todos os hospedeiros podem se comportar tanto como clientes, requisitando um arquivo, tanto como servidores, fornecendo arquivos para diversos outros hospedeiros. Assim, a noção de lados cliente-servidor está presente em processos diferentes, mas no uso geral, não há discernimento.

2. Quando há troca de mensagens, é necessário identificar o IP e a porta em que processo está rodando no hospedeiro origem e o IP e a porta do hospedeiro de destino.

3. O cache geralmente é utilizado em ambientes corporativos para diminuir o tempo de resposta em buscas em outros servidores por meio de um servidor proxy. O proxy guarda os objetos mais requisitados pela corporação. Caso o cache tenha o objeto, existe um mecanismo de get condicional que só requisita do servidor origem caso ele tenha sido modificado após uma data. Nem todos os objetos buscados podem estar no cache, assim, ele busca nos outros servidores.

4. a. Falso. O protocolo HTTP envia apenas um objeto por requisição.
   b. Verdadeiro.
   c. Falso. Para conexões não persistentes, o HTTP cria uma conexão para cada objeto requisitado, e a conexão é encerrada pelo servidor após o objeto ser recebido pelo cliente.
   d. Falso. O campo date indica a data e hora em que o objeto foi enviado.

5. O SSL opera na camada de aplicação. O socket da aplicação envia para o socket SSL dados não criptografados, este os criptografa e envia para o socket TCP. Para incluir o SSL no transporte, é necessário incluir o código SSL na aplicação.

6. Para traduzir o nome de hospedeiro para o IP é utilizado o protocolo DNS quFae, por sua vez, utiliza o protocolo UDP para transporte.

7. É necessário abrir uma conexão e depois fazer a requisição para cada servidor DNS visitado, além de mais uma requisição para o objeto. Assim,
   $$2 * (RTT_0) + RTT_1 + RTT_2 \ldots + RTT_n$$

8. a. São 2RTT para cada requisição e são 8 objetos + HTML, ou seja, são 9 requisições, além de mais 1RTT para cada servidor DNS visitado. Assim,
   $$9 * (2RTT) = 18RTT + RTT_1 + RTT_2 \ldots + RTT_n$$

b. Serão 3 operações de grupo, 1 para HTML, 5 requisições paralelas, e 3 requisições paralelas, além de mais 1RTT para cada servidor DNS visitado. Assim,

$$3 * (2RTT) = 6RTT + RTT_1 + RTT_2 \ldots + RTT_n$$

c. Será apenas 1 RTT para abrir conexão, 1 para HTML e 1 para cada objeto, totalizando 10RTT, além de mais 1RTT para cada servidor DNS visitado. Assim,

$$10RTT + RTT_1 + RTT_2 \ldots + RTT_n$$

9. No HTTP 1.1 habilitamos o GZIP para comprimir as informações que mandamos em nossas respostas. Uma boa prática que precisa ser habilitada explicitamente. No HTTP 2 GZIP é padrão e obrigatório.
   Para cada recurso que uma página possui, um request feito então para carrega-los mais rapidamente precisamos paralelizar essas requisições. O problema é que o HTTP 1.1 é um protocolo sequencial, só podemos fazer 1 request por vez. A solução é abrir mais de uma conexão ao mesmo tempo, paralelizando os requests em 4 a 8 requests (é o limite que temos). Uma forma comum de lidar com isso é usar vários hostnames na página (pag.com e img.pag.com), assim ganhamos mais conexões paralelas.
   No HTTP 2 as requisições e respostas são paralelas automaticamente em uma única conexão. É o chamado multiplexing.
   Fonte de pesquisa:
   https://pt.stackoverflow.com/questions/167014/quais-s%C3%A3o-as-diferen%C3%A7as-entre-http-2-e-http-1-1#:~:text=No%20HTTP%201.1%20os%20headers,headers%20para%20as%20requisi%C3%A7%C3%B5es%20seguintes.

10. (Professor cancelou pelo campus virtual)

11. O balanceamento de cargas consiste em criar várias entradas para o DNS no registro do DNS para o domínio, ou seja, o servidor autoritário pai contém vários registros do tipo A. Para definir como será o fluxo de acesso, várias técnicas podem ser utilizadas levando em conta aspectos diferentes como simples rotatividade e localização geográfica. Entre as técnicas podemos citar a Round Robin DNS (RRDNS) e a Network Load Balancing (NLB) da Microsoft.

12. Através do comando dig no terminal somado a url do site desejado, caso o tempo de resposta para acesso ao site seja muito pequeno, é bem provável que alguma outra pessoa já tenha acessado o mesmo site anteriormente em outra máquina.

13. a. esal.ufla.br e marakatu.ufla.br
    b. dig NS .
       b.root-servers.net
       d.root-servers.net
       m.root-servers.net

i.root-servers.net
j.root-servers.net
c.root-servers.net
g.root-servers.net
h.root-servers.net
k.root-servers.net
f.root-servers.net
l.root-servers.net
e.root-servers.net
a.root-servers.net

dig NS server ufla.br
esal.ufla.br
marakatu.ufla.br

dig NS ufla.br
esal.ufla.br
marakatu.ufla.br

dig A ufla.br esal.ufla.br
200.131.250.18

dig A ufla.br marakatu.ufla.br
200.131.250.5