UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
DEPARTAMENTO DE INFORMÁTICA APLICADA
INF01154 - Redes de Computadores N

Nível de aplicação: HTTP, DNS, Programação Ping.

1 Animação: HTTP delay estimation

Apresentar em aula a animação referente à estimativa de atraso no HTTP. Acesso em https://media.pearsoncmg.com/aw/ecs_kurose_compnetwork_7/cw/content/interactiveanimations/http-delay-estimation/index.html.

a) Escolher "non-persistent connections" com 4 objetos e atraso de transmissão para cada objeto de 1 RTT. Comentar.

b) Escolher "non-persistent connections with paralellism" com 2 conexões em paralelo. O restante mantém. Comentar.

c) Escolher "persistent connections without pipeline". O restante mantém. Comentar.

d) Escolher "persistent connections with pipeline". O restante mantém. Comentar.

2 Wireshark: HTTP

Quem tiver curiosidade, assistir o vídeo do Jim Kurose para wireshar e HTTP: https://mediaplayer.pearsoncmg.com/_ph_cc_ecs_set.title.Introduction_to_Wireshark_and_HTTP__/aw/streaming/ecs_kurose_compnetw_6/wireshark_videonotes.m4v

Abra o arquivo do laboratório de wireshark HTTP do Kurose, disponível na página da disciplina (*Cap2_Wireshark1_HTTP_v7.0.pdf*). **Execute todos os cinco exercícios de forma online**, ou seja, utilizando as URLs fornecidas no arquivo. Teste no navegador o uso de "F5" e "SHIFT-F5". Algumas vezes será útil.

a) No item 1 (*Basic HTTP GET/response*), observe que haverá um segundo GET para o arquivo "favicon.ico", que serve para mostrar um ícone ao lado da URL. Esse pode ser descartado.

b) No item 2 (*HTTP Conditional GET/response*), se você estiver usando Chrome, a limpeza de cache é feita em "*Configurações + Avançado + Privacidade e Segurança + Limpar dados de navegação*".

c) No item 3 intitulado (*Retrieving Long Documents*), uma possibilidade de filtro para facilitar a visualização é: *(http or tcp) and ip.addr==128.119.245.12 and tcp.port==62207*. Verifique a porta exata da conexão.

d) Uma forma gráfica de seguir um determinado fluxo no *wireshark* é utilizar "*statistics + flow-graph*". Escolha a opção de "*displayed packets*" após criar o filtro de visualização do item anterior.

e) **Curiosidade**: algumas vezes pode-se ver o tamanho do pacote no *wireshark* (sem CRC) sendo maior que o máximo (1514 bytes num quadro Ethernet). Isso acontece porque a *libpcap* do wireshark captura os pacotes ANTES do mesmo ir para a placa de rede. Existe uma opção que o SO envia pacotes maiores para serem fragmentados na placa. O nome é: *Large Receive Offload (LRO) or Receive Segment Coalescing (RSC)* quando se recebem pacotes maiores, e *TCP Segmentation Offload (TSO) aka Large Segment Offload (LSO) aka Generic Segment Offload (GSO)* quando se transmitem pacotes maiores. Um artigo explicando isso pode ser visto em: http://packetbomb.com/how-can-the-packet-size-be-greater-than-the-mtu/. Na verdade os pacotes **não são maiores**. Eles só não foram fragmentados ainda pela placa de rede.

f) **Dica**: caso você queira analisar com maiores detalhes cabeçalhos HTTP e outras singularidades, pode utilizar o ambiente de desenvolvimento de APIs chamado "Postman" (https://www.getpostman.com).

3 PROGRAMAÇÃO: Web Server

Abra o arquivo do programa de *web server* do Kurose, disponível na página da disciplina (*Cap2_Prog1_WebServer.pdf*). Copie o código em *python*, adicionando ao mesmo as linhas faltantes *(#Fill in start* e *#Fill in end*). Se quiser utilizar outra linguagem, OK. Seu código deve:

a) Abrir um arquivo html (index.html ou helloworld.html ou outro) a partir de um navegador Internet (usando a URL, por exemplo, *localhost:6789/index.html*).

b) Dicas: basicamente estão faltando as primitivas de sockets, que são para TCP:
   1. *Bind* e *Listen* (amarra o IP com a porta e determina fila de aceite de conexões)
   2. *Accept* (o servidor se prepara para receber a conexão)
   3. *Receive* (recebe o pedido HTTP GET dentro do laço)
   4. *Read* do arquivo solicitado (*f.read()*)
   5. *Send* da mensagem de resposta (200 OK)
   6. No caso de IOError (arquivo não encontrado), falta o *Send* de "*Not Found*" e fechar o socket (*Close*).

c) Para executar o servidor a partir de um terminal Linux, pode-se utilizar a linha de comando: "python3 nomearq.py".

4 Animação: DNS recursive/iterative queries

Apresentar em aula a animação referente à comparação entre DNS recursivo e iterativo. Acesso em https://media.pearsoncmg.com/aw/ecs_kurose_compnetwork_7/cw/content/interactiveanimations/recursive-iterative-queries-in-dns/index.html.

a) IMPORTANTE: para entender a simulação e o funcionamento do DNS (recursivo ou interativo), leve em consideração que a QUERY (pontilhado verde na simulação) é quem define se a consulta será interativa ou recursiva. Observe as *flags* "*recursion desired*" e "*recursion available*" do cabeçalho DNS quando fizer a atividade com o *sniffer*.

b) Escolher na primeira opção o primeiro (*Destination IP cached*). Testar todas outras opções.

c) Escolher na primeira opção (*Root name server has the Authoritative name server cached*). Testar todas outras opções.

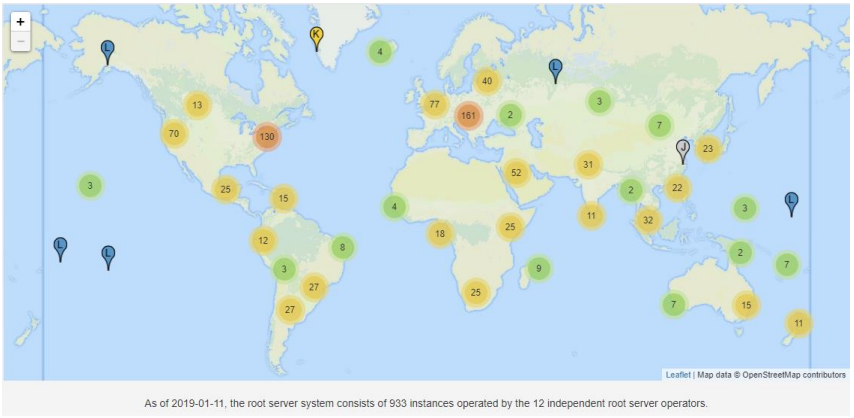
d) Escolher na primeira opção (*Root name server has na Intermediate name server cached*). Testar todas outras opções.

e) Escolher na primeira opção (*Local name server has the Destination IP cached*). Testar todas outras opções.

f) **OBS**: o que mais se assemelha ao comportamento observado seria o *Local Server* e o *Root Server* configurados como "*Interative*".

5 Wireshark: DNS

1. Abra o arquivo do laboratório de *wireshark* DNS do Kurose, disponível na página da disciplina (*Cap2_Wireshark2_DNS_v7.0.pdf*).
    a) No item 1, é sugerido capturar (com o *wireshark*) os comandos *nslookup* enquanto forem sendo efetuados. Pode ajudar no entendimento.
    b) Observe que os gerentes de rede já atualizaram bastante os servidores que estão no material do Kurose, e os resultados vão ser um pouco diferentes do que está no pdf.
    c) Tente comandos locais, como: a) >nslookup www.inf.ufrgs.br; b) >nslookup –type=ns inf.ufrgs.br; c) com o servidor de nome descoberto em "b", fazer uma consulta para mit.edu (>nslookup mit.edu xxxx); d) descobrir a resposta autoritativa ao IP de "smtp.inf.ufrgs.br", que pode ser obtida com o servidor descoberto em "b" (>nslookup smtp.inf.ufrgs.br xxxx).
    d) Ipconfig no Windows tem um equivalente no Linux que é o "ifconfig". Entretanto, para limpar a cache DNS no Linux utiliza-se o comando: "*sudo /etc/init.d/networking restart*". No Windows é "*ipconfig / flushdns*".
    e) Para ver o DNS da sua máquina, pode-se olhar o arquivo "/etc/resolv.conf" no Linux. No Windows, o próprio comando "ipconfig" fornece isso.
2. Acesse https://whois.registro.br/. Obtenha o nome de dois servidores DNS no Brasil. Por exemplo: www.ufrgs.br e www.senairs.com.br. Qual o contato do responsável em cada um deles?
3. Em http://www.root-servers.org/ tem o mapa de todos os servidores raiz (a.root-servers.net até m.root-servers.net). Cada letra é administrada por uma organização, como pode-se ver na figura a seguir. Responda:
    a) Quantos Root servers existem no Brasil?
    b) Cite dois operadores dos Root Servers no Brasil.
    c) Qual o total de Root Servers no Mundo?



6 PROGRAMAÇÃO: ping UDP

Abra o arquivo do programa do *servidor ping* do Kurose, disponível na página da disciplina (*Cap2_Prog2_UDPPinger.pdf*). Copie do pdf o código do servidor ping, colocando o mesmo para executar. **Desenvolva o código para o cliente UDP**. Se quiser utilizar outra linguagem, OK. Como é uma execução com parâmetros de linha de comando, o python deve estar no "PATH" da máquina, e a execução pode ser algo tipo *"> py UDP_Ping_cli.py localhost 12000*" ou, em Linux, *"> python3 UDP_Ping_cli.py localhost 12000*". Você deve:

    a) Comparar o tempo do ping numa execução local e entre máquinas diferentes no laboratório.
    b) Aferir (comparar) o resultado da saída do seu ping com o ping padrão de um sistema operacional existente (Windows, Linux, Mac).