CC4303 - Redes
Profesora Ivana Bachmann
Auxiliar Vicente Videla
Ayudantes Gonzalo Alarcón, Valentina Esteban, Pablo Villar

CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

Pauta Auxiliar 1

HTTP

22 de marzo, 2023

Conceptos nuevos pertinentes

- Conexiones persistentes: se reutiliza un socket de conexión para enviar varios datos entre el origen y el destino.
- Conexión no persistente: se cierra una vez fue utilizada.
- connection: header para indicar qué tipo de conexión utilizar. Sus valores son "keep-alive" para persistente, "closed" para no persistente.
- Round-Trip Time (RTT): tiempo en milisegundos de cuánto se demora una solicitud en ir de origen a destino, y luego de vuelta al origen.

P1. Servidores HTTP

Suponga que un web server corre en el host C en el puerto 80; este server usa conexiones persistentes y está recibiendo requests de dos hosts diferentes, A y B. ¿Se procesan todas las requests en el mismo socket de C? ¿Si se pasan a sockets distintos, tienen ambos sockets el puerto 80? Explique.
R: Los sockets se definen de acuerdo a la siguiente tupla: (IP origen, Puerto origen, IP destino, Puerto destino). Por cada conexión persistente, el web server crea un socket de conexión distinto pues esta tupla es distinta. De acuerdo a esto se tiene que tanto A como B se comunican con C a través de distintos sockets (IP de origen distinta), pero ambos tienen como puerto de destino el puerto 80.

P2. Conexión persistente

Considere que establecer conexión en un socket orientado a conexión se demora 3RTT y cerrar dicha conexión se demora 2RTT. Suponga una página HTML referencia 8 pequeños objetos en el mismo server. Asumiendo que el tiempo de descarga de los objetos es despreciable y que no se pueden establecer conexiones paralelas. ¿Cuánto tiempo toma descargar la página si:

1. se usan conexiones no persistentes?
   R: Suponiendo que cada objeto hace una conexión TCP y la cierra, tenemos 3+2RTT por objeto.
2. se usan conexiones persistentes?
   R: Si la conexión es persistente, solo hay que abrirla 1 vez = 3RTT del handshake inicial.

P3. Sockets

1. ¿Qué datos básicos necesita un socket orientado a conexión y cuáles necesita uno no orientado a conexión para mandar cosas?
   R: orientado a conexión: necesita saber entre qué direcciones (origen y destino) ocurre la comunicación
   no orientado a conexión: necesita saber la dirección de destino

2. ¿Cuál es la gran diferencia entre un socket orientado a conexión y uno no orientado a conexión?
   R: Un socket orientado a conexión establece un canal de comunicación, mientras que un socket no orientado a conexión no.

3. ¿Cómo funcionan los sockets no orientados a conexión? ¿Por qué alguien los preferiría?
   R: Los sockets no orientados a conexión mandan los datos a su dirección y esperan que en una de esas lleguen las cosas (no se asegura de que cosas lleguen ni nada). Son preferibles en situaciones donde la rapidez con que llega la información es más relevante que perder un poco de información (como en zoom).

P4. HTTP

1. ¿De qué forma viene la información en un mensaje HTTP?
   R: Viene en texto plano. Primero viene el HEAD que tiene como primera linea el start-line y luego los headers. El fin de los headers se marca con un doble salto de línea debajo del cual se encuentra el BODY.

2. ¿Si un agente malicioso bloquea el puerto 80 de su PC, podrá visitar páginas web?
   R: Sí, porque el puerto 80 está reservado para servidores HTTP (no clientes).

3. ¿Qué ocurre si envío un mensaje HTTP sin el doble salto de línea "\r \n \r \n" que separa el HEAD del BODY?
   R: En ese caso, quien recibe el mensaje cree que todo lo que le llegó son headers y puede quedarse pegado esperando el "fin del área de headers".

P5. Verdadero y Falso

1. HTTP es un lenguaje de marcado y HTML es un protocolo de comunicación.
   R: Falso, es al revés.

2. El HTTP response (la respuesta) se compone de headers o datos, no ambos.
   R: Falso, la respuesta contiene tanto headers, que contiene los metadatos e información necesaria sobre el mensaje, y los datos o BODY, que son lo datos que se quieren enviar.

3. El protocolo HTTP no guarda información de la conexión, para eso se usan los headers / cookies.
   R: Verdadero, el protocolo no tiene forma de guardar ningún tipo de información, por lo que se le solicita al cliente que lo haga, en forma de cookies.

4. El protocolo HTTP utiliza comunicación no orientada a conexión, para evitar delays innecesarios.
   R: Falso, HTTP utiliza comunicación orientada a conexión. Es importante que llegue toda la información y que llegue ordenada, para que así el cliente la pueda procesar correctamente.

5. Un servidor HTTP puede recibir una request y antes de responder, puede hacer consultas a otros servidores, formando una cadena de preguntas y respuestas.
   R: Verdadero, puede que un servidor no tenga todos los elementos solicitados, por ejemplo una imagen, un css, un javascript, etc, teniendo que solicitarlos a otros servidores.

6. Los headers siguen un orden determinado, por ejemplo 'Content-Type' siempre viene antes que 'Content-Length'.
   R: Falso, pueden estar en cualquier orden y es tarea del cliente el saber leerlos.

7. El header 'Content-Length' indica el largo del mensaje HTTP, incluyendo los headers.
   R: Falso, indica el largo del area BODY de la respuesta.

8. En caso de recibir de forma incompleta el BODY, puedo volver a hacer 'receive'.
   R: Verdadero, en caso de que el largo del BODY leído sea menor que el indicado por Content-Length se hace un nuevo Request hasta obtener todo el mensaje.

9. Usando mensajes HTTP puedo recibir tanto texto plano como imágenes.
   R: Verdadero, el protocolo sirve para enviar diversos tipos de contenido.