Trapy

Transport (Layer in) Python

Bienvenidos al proyecto de final de la asignatura de Redes de Computadoras, curso 2019-2020.

En este proyecto se debe implementar un protocolo de capa de transporte, que funcione sobre un medio inseguro, en este caso, internet en su totalidad. No tiene que ser compatible con TCP (*Transfer Control Protocol*), aunque es muy recomendable seguir las ideas utilizadas en este.

Para esto debe seguir la interfaz indicada en el archivo `trapy/trapy.py`. Se deben implementar las siguientes funciones:

1. `listen(address: str) -> Conn` prepara una conexión que acepta los paquetes enviados a address.
2. `accept(conn: Conn) -> Conn` espera por alguna petición de conexión utilizando un conn creado previamente con listen.
3. `dial(address: str) -> Conn` establece conexión con el otro extremo indicado en address y devuelve la conexión.
4. `send(conn: Conn, data: bytes) -> int` envía los datos por la conexión y devuelve la cantidad de bytes enviados.
5. `recv(conn: Conn, length: int) -> bytes` recibe a lo sumo length bytes almacenados en el buffer de la conexión.
6. `close(conn: Conn)` termina la conexión.

La clase `Conn` es la encargada de llevar el estado de la conexión.

En caso de alguna anomalía, lanzar una excepción del tipo `ConnException`.

Un ejemplo (trampa) de cómo podría quedar la solución funcional se brinda en el archivo `trapy/socket_trapy.py`, en el que se utiliza un socket TCP.

Para poder especificar los datos a enviar en capa de transporte se pueden utilizar los sockets de python. Por ejemplo, mirar el fichero `trapy/example_raw_send.py`. Nótese que también se tienen que especificar los encabezados IP. La información al respecto de estos encabezados se puede encontrar en el libro de texto.

Para crear un socket que prepara y envía el paquete especificando la información de capa de red y transporte, utilizar:

```
s = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_RAW)
ip_header  = b'\x45\x00\x00\x28'  # Version, IHL, Type of Service | Total Length
ip_header += b'\xab\xcd\x00\x00'  # Identification | Flags, Fragment Offset
ip_header += b'\x40\x06\xa6\xec'  # TTL, Protocol | Header Checksum
ip_header += b'\x0a\x00\x00\x01'  # Source Address
ip_header += b'\x0a\x00\x00\x02'  # Destination Address
```

```
packet = ip_header + transport_header


s.sendto(packet, ('10.0.0.2', 0))
```

Los encabezados de capa de red se deben especificar dado que el protocolo no tiene porqué ser TCP específicamente.

Una vez creado el mensaje de capa de red se le adiciona a continuación el de transporte y se envía. Hay que especificar la dirección IP del destinatario como parte del protocolo RPC y el puerto se deja como 0 dado que no tienen sentido los puertos en esta etapa del protocolo.

Para recibir mensajes enviados de forma genérica utilizar el ejemplo mostrado en `trapy/example_raw_recv.py`:

```
s = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_RAW)
s.bind(('10.0.0.2', 0))
print(s.recvfrom(65565))
```

Evaluación

3: Tolerante a fallas. Si hay alguna manera de que el mensaje llegue, entonces debe llegar, y se debe conservar el orden de los paquetes.

4: Optimizar el uso del ancho de banda. Para esto se podrían enviar más de un paquete a la vez, sin dejar de garantizar que en el otro extremo sean entregados al usuario en orden.

5: Control de flujo. Optimizar el uso del ancho de banda podría causar efectos contraproducentes, como inundar el buffer del enrutador. Se puede controlar la velocidad de envío utilizando alguna de las estrategias vistas en clase, como tener un tamaño de ventana dinámico.

Aspectos técnicos

Dependencias

Es necesario tener instalado **Python3**. La implementación de su código debe estar en Python3.

Es opcional instalar **Mininet**. Mininet es una herramienta para simular redes. Estas se pueden crear de forma programática, como se puede observar en la carpeta `tests/topos`. Se utiliza para automatizar el proceso de prueba y evaluación. Para correr los tests se necesita Mininet, así que es muy recomendable instalarlo. Esta herramienta solo funciona en linux.

Puede leer más sobre Mininet aquí.

Correctitud

Para ejecutar los tests, utilizar:

```
sudo python2 -m unittest discover tests
```

Es necesario utilizar permisos elevados porque Mininet lo requiere.

Para evaluar la solución el profesor utilizará un ambiente seguro con Docker, así que esto no será un problema de seguridad para automatizar la evaluación.