Servidor de Mensajería

Preguntas

1. ¿Qué es chat? ¿Para que sirve? ¿Cuál es su historia?

Chat, o más actualmente conocido como mensajería instantánea, se refiere a las comunicaciones por Internet que proporciona transmisiones de texto en tiempo real, entre remitente y destinatario. Dichos mensajes suelen ser cortos, y se responden rápidamente, lo que da la sensación de una conversación real (síncrona), a diferencia una conversación por correo electrónico o en un foro (asíncrona), donde los mensajes se leen y responden cuando al usuario le apetece.

El chat puede ser entre dos personas, o dentro de un grupo más numeroso, en lo que se llama una sala de chat.

Las conversaciones por chat inicialmente eran punto a punto entre clientes, después multicast entre clientes, y actualmente con un servidor que gestiona las comunicaciones. Nosotros trabajaremos chat utilizando aplicaciones cliente-servidor y protocolos abiertos, como siempre.

Chat, junto con ftp, los grupos de noticias, y el correo electrònico, fué una de las principales aplicaciones de Internet antes de que aparecieran las pàginas web.

A inicios de los 90 estaba extendidísimo el chat en grupo , primero mediante el uso del IRC, posteriormente a través del estándard XMPP (Jabber), y finalmente también a través de videoconferencia y telefonía IP utilizando los protocolos SIP y VOIP.

En los 00 hay un decline de los servicios de chat de texto. Por un lado la web 2.0 lleva a los usuarios a abandonar los estándares y clientes de chat para hacer chat des de páginas web de grupos sociales (por ej. chat de Facebook, etc.). Por otro lado cada vez más gente utiliza aplicaciones cerradas para videoconferencia (por ej. Skype, etc.).

En 2010 XMPP tenía 1200 millones de usuarios, y Skype 500 millones.

En la decada de los 10 llega el decline total. Los usuarios ahora utilizan aplicaciones cerradas de mensajería en smartphones (por ej. Whatsapp, Telegram, etc.). Aún así, todavía aparecen algunos protocolos abiertos para mensajería instantánea (por ej. el protocolo Matrix, usado en el cliente de mensajería Element).

En 2020 Whatsapp tiene 500 millones de usuarios que lo utilizan diariamente, y Telegram 500 millones de usuarios que lo utilizan mensualmente.

Los servidores de chat han sido sustituidos por mensajería instantánea a través del teléfono móvil, y por aplicaciones de videoconferencia en grupo. Aún así todavía tiene sentido montar servidores de mensajería para trabajo en grupo en el ámbito empresarial, ya que proporcionan una mayor privacidad, encriptación, y archivado de mensajes.

2. ¿Qué es IRC?

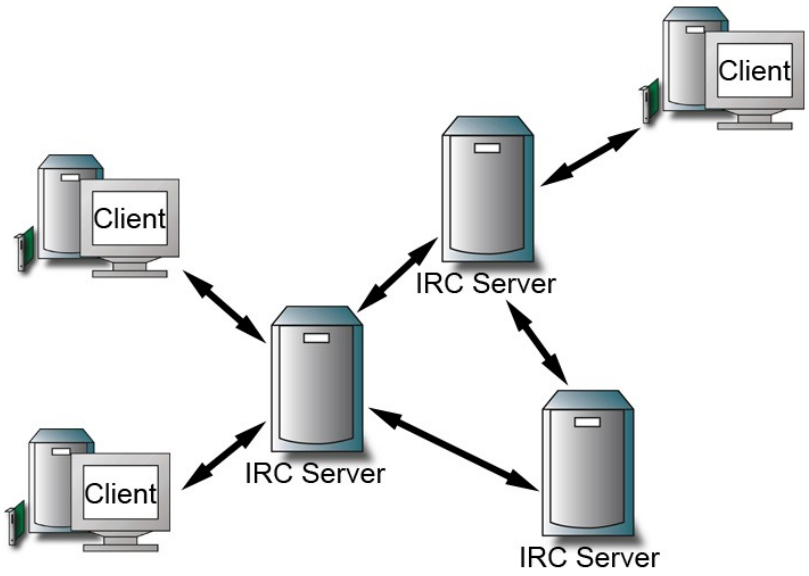
Internet Relay Chat (IRC) es un protocolo de la capa de aplicación para comunicación de texto entre clientes de chat y servidores de chat. Normalmente se asocia al puerto 6667 TCP.

El cliente de chat puede ser un programa específico o estar incrustado en una página web.

El chat normalmente es en grupo en lo que se llama salas o canales de chat, pero se pueden

enviar mensajes privados y ficheros.

El URI para un chat es: `irc://<host>[:<port>]/[<channel>[?<channel_keyword>]]`



Ver más en:

https://en.wikipedia.org/wiki/Comparison_of_Internet_Relay_Chat_clients

https://en.wikipedia.org/wiki/Comparison_of_Internet_Relay_Chat_daemons

https://en.wikipedia.org/wiki/List_of_Internet_Relay_Chat_commands

3. ¿En qué consiste XMPP??

Extensible Messaging and Presence Protocol (XMPP, pero originalmente llamado Jabber) es un protocolo de comunicación para el envío de mensajes, basado en XML. Normalmente está asociado al puerto 5222 TCP. Su desarrollo fué posterior al desarrollo del protocolo IRC.

Los mensajes pueden estar encriptados, y almacenarse en el servidor. También permite el envío de archivos de audio y vídeo.

Es un protocolo enormemente descentralizado, que además permite intercambio de mensajes XMPP con otros protocolos a través de las llamadas "pasarelas" -o "transports" en inglés-. Es decir, además de poder intercambiar mensajes entre usuarios de XMPP, se pueden enviar mensajes de un usuario de XMPP a otro que utiliza un protocolo diferente, como IRC, SMS o email. Para ello, el servidor de XMPP que tenga instalada una "pasarela" hacia el nuevo protocolo, transformará el mensaje.

Supongamos que julieta@capuleto.com desea chatear con romeo@montesco.net. Julieta y Romeo tienen sus respectivas cuentas en los servidores capuleto.com y montesco.net. Cuando Julieta escribe y envía su mensaje, entra en acción la siguiente secuencia de eventos:

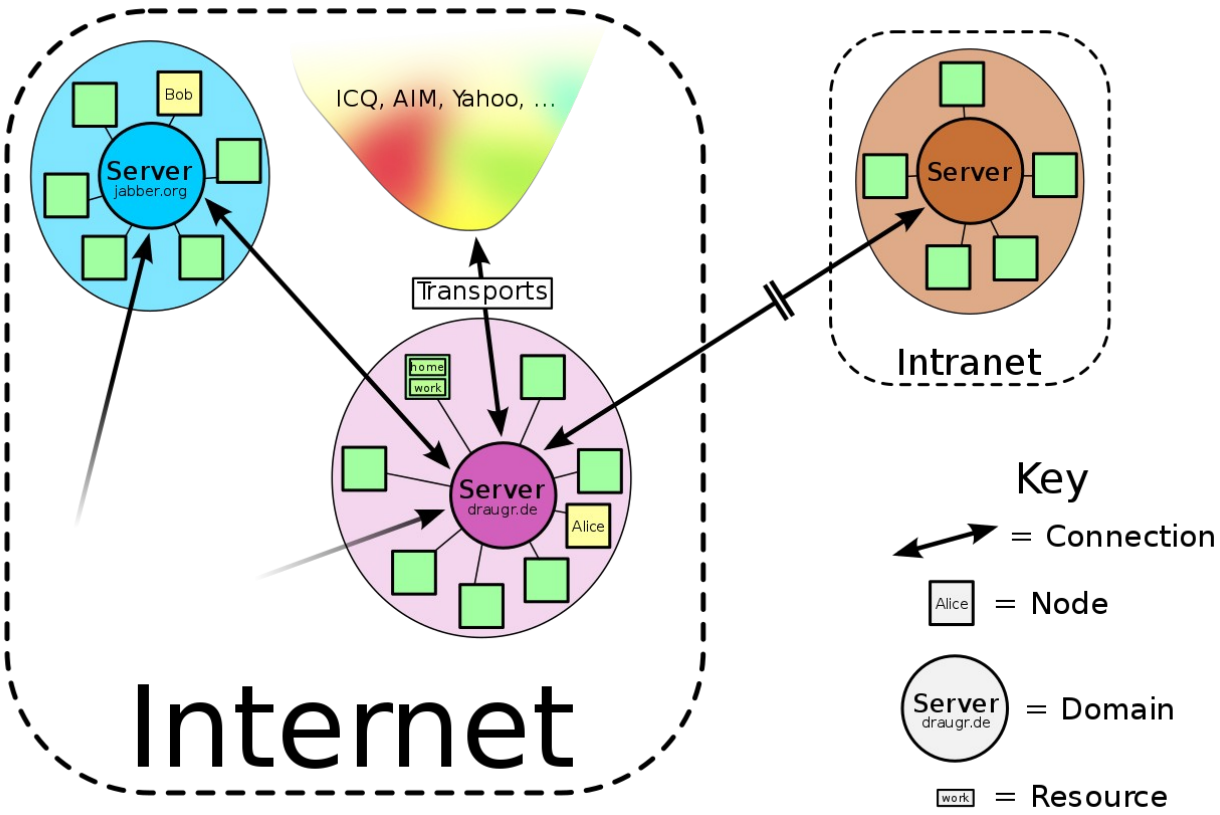
1) El cliente de Julieta envía su mensaje al servidor capuleto.com.

Si el servidor capuleto.com no tiene acceso al servidor montesco.net , el mensaje es desechado.

2) El servidor capuleto.com abre una conexión con el servidor montesco.net.

3) El servidor montesco.net entrega el mensaje a Romeo.

Si Romeo no está conectado, el mensaje es guardado para su posterior entrega.



Ver más en:

https://en.wikipedia.org/wiki/Category:XMPP_clients

https://en.wikipedia.org/wiki/Comparison_of_XMPP_server_software

4. ¿Qué es VoIP?

Voice over Internet Protocol (VoIP), también llamado telefonía IP, es un protocolo (-de hecho, es toda una familia de tecnologías-) para transportar comunicaciones de voz sobre paquetes IP en intranets y en Internet. Por lo tanto, la voz se envía digitalizada, en lugar de enviarse su homólogo analógico.

Ver más en:

https://en.wikipedia.org/wiki/Comparison_of_VoIP_software

https://en.wikipedia.org/wiki/Asterisk_(PBX)

5. ¿En qué consiste Matrix?

Matrix es un protocolo estándar y ligero para el envío de texto y voz en tiempo real. Se trata de un protocolo seguro, cifrado, moderno, que mantiene el historial de conversaciones, y completamente descentralizado.

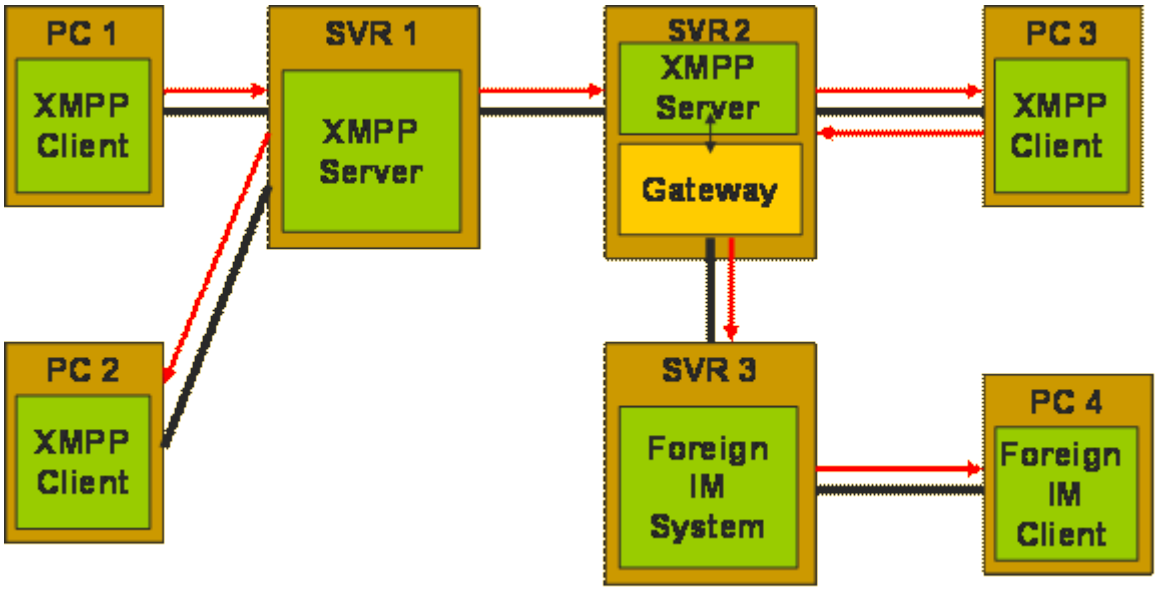
También dispone de "pasarelas" hacia otros protocolos, al igual que XMPP.

La aplicación cliente más famosa para Matrix es Element.

Ver más en:

https://www.matrix.org/faq

6. Dibuja el esquema de dos usuarios enviandose mensajes a través de sus servidores de XMPP.



7. ¿Viene algún software de mensajería con Windows Server? ¿Dónde? ¿Cómo se utiliza?

No viene ningún software de mensajería con Windows Server. El único software de mensajería actualmente conocido de Microsoft es Skype.

8. ¿Cuáles son los servidores de XMPP más conocidos para Unix/Linux? ¿Y los clientes de XMPP más conocidos para Unix/Linux?

Como clientes de mensajería para Linux que soportan XMPP son bastante conocidos Jitsi, Pidgin, Kopete, i Spark, aunque hay muchos más: https://xmpp.org/software/clients.html

Como servidores de XMPP los más usados son OpenFire y Prosody, pero hay más: https://xmpp.org/software/servers.html

9. Al instalar un servidor de mensajería, ¿Qué parámetros a configurar piensas que serán los más importantes?

Básicamente los usuarios y contraseñas.

10. Haz una comparativa de prestaciones entre estos programas de mensajería para Smartphones: Whatsapp, Telegram, Element, Signal

https://en.wikipedia.org/wiki/Comparison_of_cross-platform_instant_messaging_clients

https://www.securemessagingapps.com/

Datos de la práctica

Configuraremos nuestro servidor de XMPP para tener un par de usuarios que podrán enviar y recibir mensajes, tanto al interior como al exterior: ana@mired.org y bob@mired.org.

Práctica con Linux: servidor de mensajería XMPP con Openfire

Instalaremos servicio de mensajería XMPP *OpenFire* para Linux. Esta guía te puede ser de utilidad:

- https://www.tecmint.com/create-your-own-instant-messagingchat-server-using-openfire-in-linux/

Aunque en los repositorios de tu distribución preferida puedes encontrar servidores de XMPP como *ejabberd*, *jabberd2* o *prosody* , para una prueba rápida prefiero escoger *openfire* por su fácil configuración y administración vía interfaz grafica.

1. OpenFire es un aplicación Java, así que para que funcione debemos primero tener instalada la máquina virtual de Java (“Java Runtime Environment”):

   # **apt update**

   # **apt install default-jre**

   OpenFire además puede guardar los datos internamente, o puede guardarlos en un servidor de bases de datos externo, como MariaDB. Si esto último fuera el caso, debería instalar previamente dicho SGBDR. Pero para simplificar la prueba que vamos a hacer, no instalamos nada y le pediremos que guarde él los datos.

2. A continuación descargamos el programa de la web y lo instalamos:

   # **wget https://www.igniterealtime.org/downloadServlet?filename=openfire/openfire_4.7.1_all.deb**

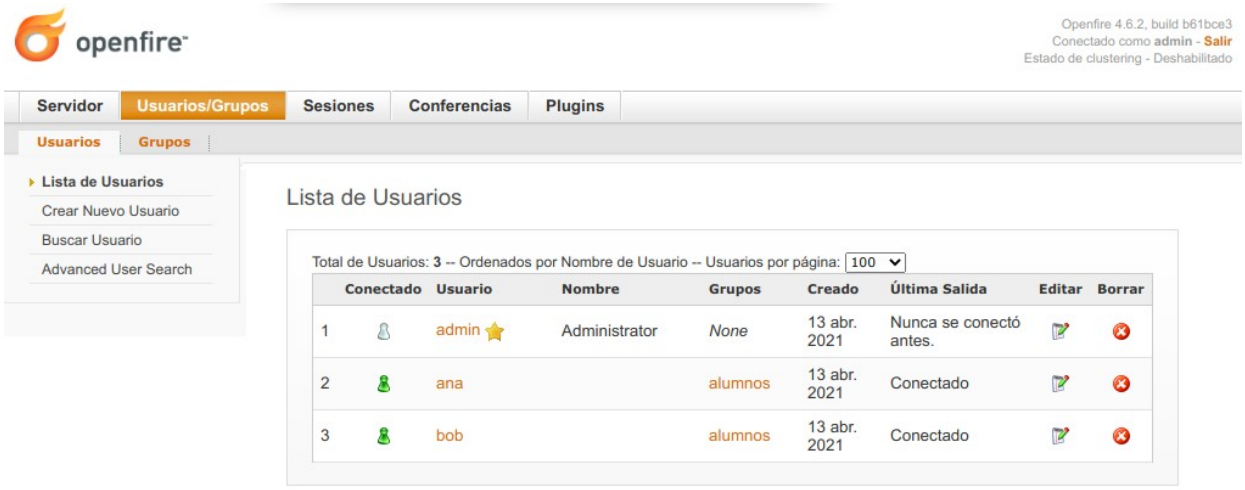
   # **apt install ./openfire_4.7.1_all.deb**

3. Por último accedemos a OpenFire vía navegador para completar el proceso de instalación. OpenFire abre el puerto 9090 para conectarse vía HTTP, y el puerto 9091 para connectarse vía HTTPS.

   http://servidor:9090/

   Pedirá: idioma, nombre de dominio y nombre de host, dónde guardar los datos, si tendrá una lista propia de usuarios o estarán en un servidor LDAP, y contraseña del usuario ‘admin’.

4. Ahora ya podemos acceder vía navegador al panel administrativo, entrando como usuario ‘admin’ creado en el último paso anterior. Una vez en el panel administrativo, podemos crear los usuarios de XMPP de dicho dominio:



5. Recuerda que en el caso de no tener servidor de DNS, y de solo querer hacer pruebas enviando mensajes de un usuario a otro en el mismo dominio, podemos hacer la chapucilla de cambiar la configuración del fichero /etc/hosts **en el cliente**:

   `# nano /etc/hosts`

   ```
   192.168.100.2 mired.org panoramix.mired.org
   ```

   `# ping panoramix.mired.org`

6. Con todo configurado en el servidor, sólo nos faltan algunos equipos en red con clientes de XMPP instalados para que ana y bob puedan chatear. De hecho, puedo crear usuarios para todos los alumnos de clase e invitarles a un chat des de sus clientes.

Práctica con Linux: servidor de mensajería Matrix con Synapse

- https://github.com/matrix-org/synapse/blob/master/INSTALL.md

Práctica con Linux: servidor de mensajería con Rocket.Chat

- https://www.howtoforge.com/how-to-install-rocketchat-server-with-nginx-on-ubuntu-20-04/

Práctica con Linux: recibir avisos del sistema vía Telegram

telegram-cli es un cliente de Telegram que funciona des de la línea de comandos. Permite chatear

en Telegram de manera interactiva, pero también permite enviar mensajes a usuarios y canales de Telegram des de scripts.

- https://blog.mypapit.net/2015/09/a-bash-script-for-sending-telegram-messages-in-linux.html

- https://geekland.eu/instalar-configurar-y-usar-telegram-cli/

En el servidor donde se ejecutará el script instala:

```
# apt update
# apt install telegram-cli
```

A continuación genera tu clave para que el cliente pueda enviar mensajes a Telegram:

```
# telegram-cli -k tg.pub
```

Te pedirá el número de teléfono asociado a Telegram (+34xxxxxxxxx) y un código que te habrán enviado instantáneamente al Telegram. Para pedir ayuda sobre un comando escribe `help` *`comando`* , y para salir escribe `safe_quit` , `quit` o pulsa CTRL+C

Para probar el cliente en modo interactivo ejecuta:

```
# telegram-cli -W -N
```

Y una vez dentro prueba el comando:

```
msg destino mensaje
```

En el destino, si un nombre de canal o usuario contiene espacios, sustituye cada espacio por _

En modo no interactivo, para ejecutar un comando que envíe un texto a Telegram escribimos:

```
# telegram-cli -W -e "msg destino mensaje"
```

Por ejemplo, para enviar un mensaje al canal llamado “mi primer bot”:

```
# telegram-cli -W -e "msg mi_primer_bot ola k ase"
# telegram-cli -W -e "msg mi_primer_bot `cat /etc/passwd | grep root`"
```

Sin embargo, para enviar un mensaje de varias líneas hay que insertar \n para cada salto de línea, lo cual es difícil si el texto resulta ser la salida de la ejecución de un comando.

Práctica con Linux: bot de Telegram interacciona con un ordenador

Se trata de crear un bot de Telegram el cual, cuando enviamos mensajes al chat asociado en Telegram, ejecuta acciones en el ordenador , como por ejemplo hacer un escaneo de alguna IP y reenviar el resultado al chat de Telegram.

- https://lliçons.jutge.org/python/telegram.html

- https://www.youtube.com/watch?v=YD7vtyLXu7U

Primero debemos visitar https://telegram.me/botfather en Telegram, y ejecutar el comando `/newbot` para generar nuestro bot. Anotad el “token” identificador del bot que generará.

A continuación, en el servidor donde se ejecutará el bot instala:

```
# apt update
# apt install python3-python-telegram-bot
```

Para el código en Python a ejecutar, aunque podría utilizar librerías de scapy o nmap para un

análisis fácil y profesional, implemento un análisis sencillo intentando abrir conexiones TCP a los diferentes puertos:

```
from telegram.ext import Updater, CommandHandler
import socket



# Escaneo cutre-TCP intentando abrir sockets

def tcp_scan(ip, port_ini=1, port_fin=1024):
    result=[]
    for p in range(port_ini, port_fin+1):
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        if s.connect_ex((ip, p)) == 0:
            result.append(p)
            s.close()
    return result



# Función con mensaje de ayuda que luego asociaré al comando /start

def start(update, context):
    context.bot.send_message(
        chat_id=update.effective_chat.id,
        text='¡Hola! Soy un bot con el comando /scan IP puerto_inicio
puerto_fin')


# Función que recupera los argumentos, llama al escaneo de puertos,
# y envía un mensaje con el resultado. Luego la asociaré al comando /scan

def scan(update, context):
    try:
        ip = context.args[0]
        p1 = int( context.args[1] )
        p2 = int( context.args[2] )
        s = tcp_scan(ip, p1, p2)
        context.bot.send_message(
            chat_id=update.effective_chat.id,
            text=str(s))
    except Exception as e:
        print(e)
        context.bot.send_message(
            chat_id=update.effective_chat.id,
            text='error')



# Inicializo el bot. El Token identificador está en un fichero 'token.txt'

TOKEN = open('token.txt').read().strip()
updater = Updater(token=TOKEN, use_context=True)
dispatcher = updater.dispatcher

# Asocio los comandos de Telegram a las funciones correspondientes

dispatcher.add_handler(CommandHandler('start', start))
dispatcher.add_handler(CommandHandler('scan', scan))
updater.start_polling()
```

Referencias

- http://en.wikipedia.org/wiki/Instant_messaging
- http://en.wikipedia.org/wiki/Comparison_of_instant_messaging_protocols
- http://en.wikipedia.org/wiki/Comparison_of_instant_messaging_clients#Protocol_support
- http://en.wikipedia.org/wiki/Internet_Relay_Chat
- http://en.wikipedia.org/wiki/Extensible_Messaging_and_Presence_Protocol
- http://en.wikipedia.org/wiki/Voice_over_IP
- https://en.wikipedia.org/wiki/Matrix_(protocol)
- https://en.wikipedia.org/wiki/Messaging_apps