**Utilización de sockets.**

En Java, las comunicaciones están asociadas al concepto de socket.

Cada programa lee desde o escribe en un socket de la misma manera que nosotros abrimos, leemos, escribimos, y cerramos un fichero. En esencia existen dos tipos de sockets:

Un tipo que es similar a un teléfono (una conexión orientada a servicio, esto es, Transmission Control Protocol (TCP).

Un tipo que es análogo a un buzón (un servicio de conexiones mediante “datagramas”, esto es, Unreliable (de poca confianza) Datagram Protocol (UDP)

Una diferencia importante entre los dos tipos de sockets es que el protocolo TCP aporta la seguridad de que todo lo que se envía se obtiene en la otra parte mientras que en el protocolo UDP no es así, de esta forma el emisor deberá chequear que ha sido recibido por el receptor.

**Protocolo UDP.**

En la familia de protocolos de Internet UDP proporciona una sencilla interfaz entre la capa de red y la capa de aplicación. UDP no otorga garantías para la entrega de sus mensajes y el origen UDP no retiene estados de los mensajes UDP que han sido enviados a la red. UDP sólo añade multiplexado de aplicación y suma de verificación de la cabecera y la carga útil. Cualquier tipo de garantías para la transmisión de la información deben ser implementadas en capas superiores.

La cabecera UDP consta de 4 campos de los cuales 2 son opcionales. Los campos de los puertos fuente y destino son campos de 16 bits que identifican el proceso de origen y recepción. Ya que UDP carece de un servidor de estado y el origen UDP no solicita respuestas, el puerto origen es opcional. En caso de no ser utilizado, el puerto origen debe ser puesto a cero. A los campos del puerto destino le sigue un campo obligatorio que indica el tamaño en bytes del datagrama UDP incluidos los datos. El valor mínimo es de 8 bytes. El campo de la cabecera restante es una suma de comprobación de 16 bits que abarca una pseudo-cabecera IP (con las IP origen y destino, el protocolo y la longitud del paquete UDP), la cabecera UDP, los datos y 0's hasta completar un múltiplo de 16. El checksum también es opcional en IPv4, aunque generalmente se utiliza en la práctica (en IPv6 su uso es obligatorio). A continuación se muestra los campos para el cálculo del checksum en IPv4, marcada en rojo la pseudo-cabecera IP.

El protocolo UDP se utiliza por ejemplo cuando se necesita transmitir voz o vídeo y resulta más importante transmitir con velocidad que garantizar el hecho de que lleguen absolutamente todos los bytes.

**Protocolo TCP.**

Transmission Control Protocol (en español 'Protocolo de Control de Transmisión') o TCP, es uno de los protocolos fundamentales en Internet. Fue creado entre los años 1973 y 1974 por Vint Cerf y Robert Kahn.

Muchos programas dentro de una red de datos compuesta por computadoras, pueden usar TCP para crear conexiones entre sí a través de las cuales puede enviarse un flujo de datos. El protocolo garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron. También proporciona un mecanismo para distinguir distintas aplicaciones dentro de una misma máquina, a través del concepto de puerto.

TCP da soporte a muchas de las aplicaciones más populares de Internet (navegadores, intercambio de ficheros, clientes FTP, etc.) y protocolos de aplicación HTTP, SMTP, SSH y FTP.

**Puertos.**

Los valores numéricos de los puertos 1-1023 se reservan a servicios de interés general, montados a menudo sobre protocolos de uso extendido:

El 80 para web con HTTP.

El 25 para correo aliente con SMTP.

El 110 para correo entrante con POP3.

El 119 para el servicio de noticias con NNTP, etc.

Los valores de puertos entre 1024 y 49151 se usan para servicios específicos de uso no general.

El resto (a partir de 49152) se emplean para designar servicios de uso esporádico.

**Utilización de TCP en Android:**

Su uso es muy parecido a java. Para realizar una comunicación es necesario un servidor y al menos un cliente.

Al iniciar el servidor este estará a la espera mediante el método accept() hasta que un cliente se conecte a él. Cuando se establece la comunicación empiezan las transferencias que se realizarán con flujos de datos (como DataInputStream o DataOutputStream).

Esto se realiza en los siguientes pasos:

1- Poner el ordenador que representa el servidor en estado de escucha (Listening).

2- Desde el ordenador cliente establecer la conexión indicando el Host o la IP del ordenador servidor.

3- Una vez establecida la conexión (sin producirse ningún error, el servidor acepta dicha conexión).

4- El servidor, en realidad, obtiene un nuevo socket sobre un puerto diferente, este puerto es el que realmente utiliza para la comunicación, porque necesita liberar el puerto donde atiendo las escuchas, para poder atender la solicitud, de otros nuevos clientes.

5- Se puede iniciar la comunicación enviando datos de cliente al servidor.

6- Procesar los datos del cliente en el servidor.

7- Dar respuesta al cliente.

8- Procesar los datos de la respuesta, y continuar con la comunicación de información repitiendo los pasos 3 a 6. Cuando se ha establecido la conexión, cualquier máquina cliente/servidor está capacitada para el envío de datos, por lo que ambas pueden iniciar una comunicación.

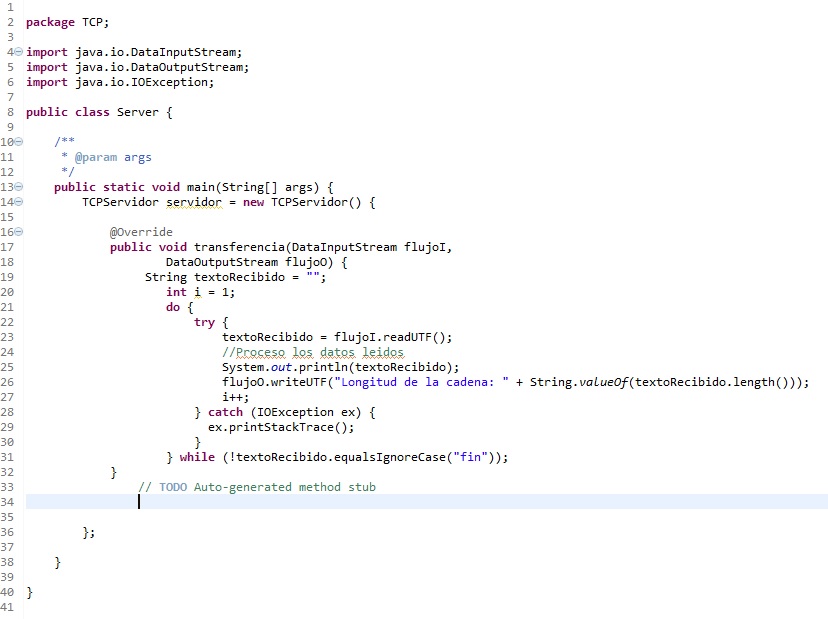
9- Para parar las comunicaciones, se cierra la conexión por parte del cliente.

10- Para parar el servidor, se sale del modo de escucha (Listening).

El ejemplo que he realizado está basado en el protocolo TCP, ya que es más seguro y de los más utilizados.

He utilizado las clases TCPcliente y TCPServidor del libro de Jesús bobadilla y después creado la clase principal java Server para crear la parte principal del servidor, y una aplicacion android para la parte principal del cliente. En las clases TCPCliente y TCPServidor están declarados e inicializados y lo único que habrá que hacer con ellos será definir el método transferencia al crear un objeto con él. En este método ira toda la comunicación entre servidor y cliente.

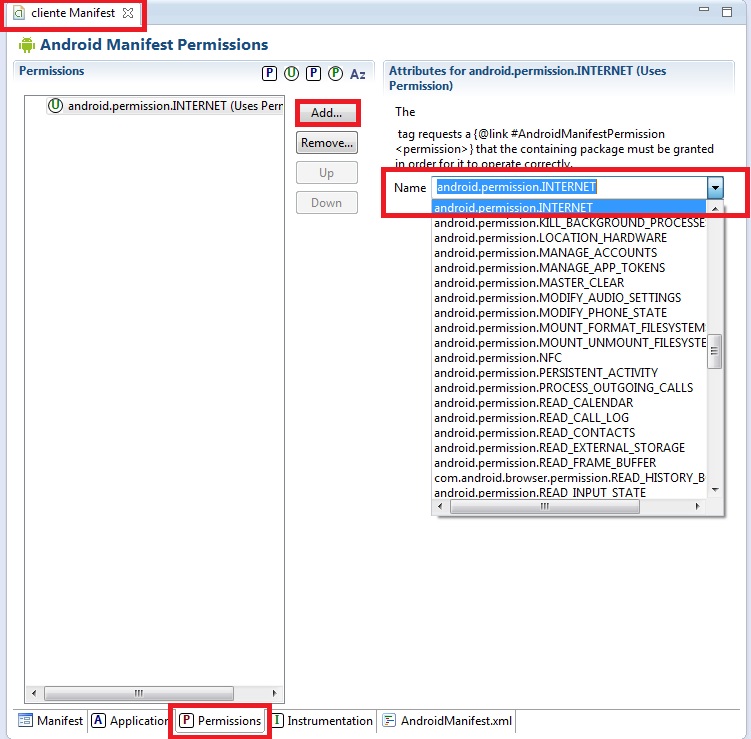
**Servidor:**

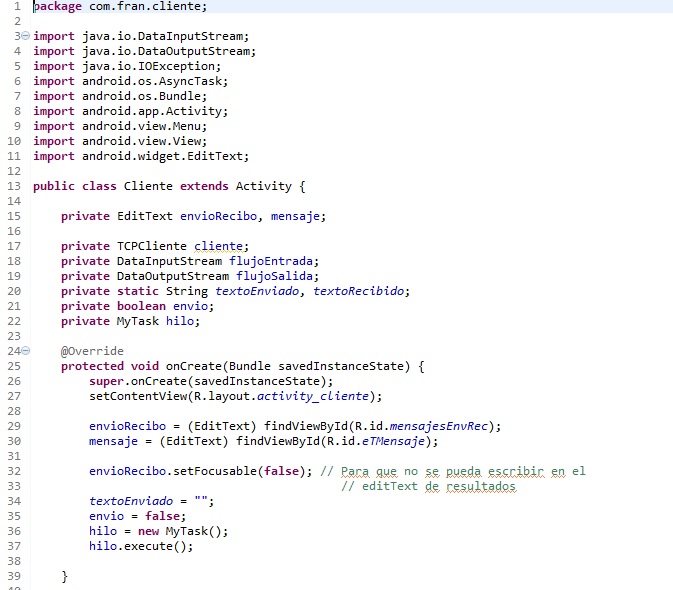


En el servidor lo único que hay que hacer es crear un objeto de la Clase TCPServidor y sobreescribir el método transferencia. En este método es donde se define lo que tiene que hacer el servidor con la información que recibe y lo que tiene que enviar.

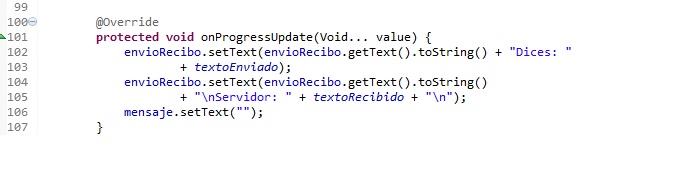
**Cliente:**

En android lo primero que debemos hacer es dar permisos de internet a la aplicación. Importante poner la IP del servidor en la clase TCPCliente.

A partir del nivel 10 de api (Honeycomb) cuando intentemos realizar la comunicación, saltara la excepcion android.os.NetworkOnMainThreadException, que quiere decir que no se pueden realizar conexiones a la red en el hilo principal de la aplicación, por lo que será necesario realizar un hilo secundario para realizar la acción de comunicación. En mi ejemplo use AsyncTask.

En el OnCreate de la aplicación habrá que inicializar las variables necesarias. 

En el AsyncTask en su método doInBackground se instanciara el objeto TCPCliente y habrá que definir el método transferencia. Para mostrar los datos en la interfaz habrá que llamar al método publishProgress(), que hará que se ejecute el onProgressUpdate, donde tendrás que indicar el View donde quieres que se muestre la información.



Todo esto solo funcionara si el dispositivo está conectado en la misma red que el servidor.

Bibliografía

<http://www.iessanandres.com/aulavirtual/mod/resource/view.php?id=3183>

Libro: "Comunicaciones y bases de datos con Java a través de ejemplos"

Autores: Jesús Bobadilla y Adela Sancho.- <http://eui.upm.es/~jbobi/jbobi/Libro2Java/>

UDP

<http://es.wikipedia.org/wiki/User_Datagram_Protocol>

TCP

<http://es.wikipedia.org/wiki/Transmission_Control_Protocol>

Socket

<http://docs.oracle.com/javase/7/docs/api/java/net/Socket.html>

android.os.NetworkOnMainThreadException

<http://android-er.blogspot.com.es/2012/04/androidosnetworkonmainthreadexception.html>

Más información para profundizar

<http://android-er.blogspot.com.ar/2011/01/simple-communication-using.html>

<http://androideity.com/2012/08/05/sockets-en-android/> (Al final de esta página hay varios enlaces relacionados)