1. Introdução geral

A função das aulas prácticas é:

- permitir-vos explorar alguns dos conceitos dados nas teóricas de forma mais práctica
- dar-vos um espaço onde poderão trabalhar, de uma forma acompanhada, nos dois projectos de desenvolvimento que terão de realizar

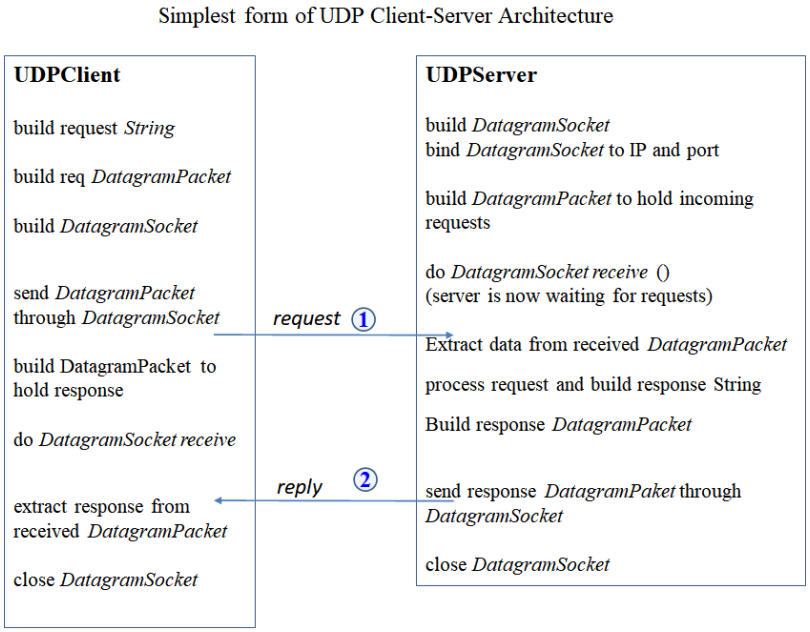
2. Aula de Hoje

Primeira aula TP será sobre a API de Java para UDP

O guião que está disponível no link que vos enviei por email.

Os detalhes básicos da aplicação a desenvolver:

- aplicação Cliente-Servidor simples (descrita, na sua forma mais simples na imagem abaixo)
- cliente envia uma mensagem com um pedido ao servidor, na forma de um datagrama UDP
- servidor retorna a resposta também como um datagrama UDP



Quanto à API Java que deverão usar para implementar este trabalho, podem encontrar informação bastante relevante no pdf com o título "*Java API for UDP communication*", para o qual têm um link no final do guião.

De uma maneira sumária a forma como deverão usar esta API na implementação é a seguinte:

- ***DatagramPacket*** – os datagramas UDP a trocar entre cliente e servidor deverão ser criados empregando instâncias da classe *DatagramPacket*;
- ***DatagramSocket*** – os pontos de envio e recepção de messagens, dos dois lados, deverão sem implementados com instâncias da classe *DatagramSocket*;
- ***InetAddress*** e as suas (subclasses Inet4A*ddr*ess e *Inet6Address*) – esta classe é importante para permitir o uso das duas anteriores;

Na classe ***DatagramPacket*** é de especial relevo o seguinte método:

- *getLength()* – é útil para usar com o construtor *String(byte[] buf, int len)*, para gerar uma "string" a partir dos bytes no campo de dados do pacote UDP
  (o primeiro passo no processamento duma mensagem, contida num datagrama UDP, será a transformação do "*byte array*" contido num *DatagramPacket* numa "*String*":

Outras classes úteis para o vosso trabalho são:
- *String* (manipulação de String):
  - *String.getBytes()* – converter uma String numa sequência de bytes a usar para construir um *DatagramPacket*
  - *String.string(bytes[],int len)* – para reconstruir a mensagem a partir da sequência de bytes extraída do *DatagramPacket* recebido (e.g. *String msg = new String(dgram.getData(), dgram.getLength)*)
  - *String.tim()* – para remover "white spaces" no ínicio e no fim duma "string"
  - *String.split()* – para partir uma mensagem nas suas componentes
  - *String.compareTo()* – comparar strings por exemplo para identificar o tipo de mensagem.

Voltando à API Java para UDP, têm também o método *DatagramSocket.setSoTimeout()* que é útil para permitir ao cliente recuperar duma forma fácil de erros na comunicação, e retransmitir a mensagem o nº de vezes que entender necessário.

Do lado servidor, o serviço descrito no guião pode facilmente ser implementado usando uma instância da classe ***java.util.Hastable***, não precisam de implementar um DNS resolver ne algo que se pareça.
Notem que, no entanto, o mais importante é conseguirem por cliente e servidor a comunicar.

A minha sugestão é que implementem primeiro o protocolo e provisões de comunicação, mesmo que o serviço não esteja correto (respostas pré-definidas) ou tenha sérias limitações, (suportando o registo duma única matrícula).

Anexo

Loopback address is a special IP number (127.0.0.1)

Both system and user ports are used by transport protocols (TCP, UDP, DCCP, SCTP) to indicate an application or service.
- Ports 0–1023 – system or well-known ports
- Ports 1024–49151 – user or registered ports
- Ports >49151 – dynamic / private ports

UDP Header

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Length																Checksum															

The field size sets a theoretical limit of 65,535 bytes (8 byte header + 65,527 bytes of data) for a UDP datagram. However the actual limit for the data length, which is imposed by the underlying IPv4 protocol, is 65,507 bytes (65,535 − 8 byte UDP header − 20 byte IP header).