***COMUNICAÇÃO POR SOCKETS***

***(UDP Datagram Communication)*** Um datagrama enviado por UDP é transmitido a partir de um *sending process* paraum *receiving process*. Se uam falha ocorre,a mesnagem pode não chegar ao destinatário. Um datagrama é transmitido quando entre processes quando um processo envia e um outro recebe esse. Para enviar ou receber mensagens datagramas, um processo deve primeiro criar um *socket* ligado a um endereço IP de Internet do host local e uma porta local. Um servidor ligará seu *socket* a uma porta no servidor. – uma porta que seja conhecida do cliente, assim ele pode enviar mensagens para o servidor. Um cliente também liga seus *socket* a qualquer porta local livre. O método *receive* (retorna o endereço IP e a porta do *sender*, em adição à mensagem, permitindo ao *recipient* enviar uma resposta. Para algumas aplicações, é aceitável usar um serviço que está sujeito a falhas ocasionais. DNS (*Domain Naming Serice*) , VOIP (*Voice Over IP*), *IP Multicast*, usam UDP. Datagramas UDP são algumas vezes um escolha atrativa, porque não padecem de *overheads* com garantia de entrega de mensagens. Existem duas fontes principais de overhead:

1. A necessidade de armazenar informação de estado na fonte e no destino, por exemplo, quando se utiliza conexão.
2. A transmissão de mensagens extras.

(***Comunicação TCP stream***) A API do protocolo TCP (*Transmission Control Protocol*) provê a abstração de um stream (fluxo) de bytes para os quais dados podem ser escritos e dos quais dados podem ser lidos. A API para *stream communication* assume que quando um par de processos está estabelecendo uma conexão, um dles exerce o papel de cliente e o outro o papel de servidor. O cliente envolve criar um *stream socket* ligado a uma porta e então, fazendo um *connect request*, solicita uma conexão a um servidor em sua porta. O servidor cria um *listing socket* ligado a uma porta do servidor e espera por clientes para requisitar conexões. O *listing socket* mantém um fila de requisições de conexões chegando. No modelo de *sockets*, quando o servidor aceita uma conexão, um novo *stream socket* é criado para o servidor se comunicar com o cliente, enquanto retendo seu *socket* à porta do servidor para ouvir requests de conexão a partir de outros clientes. O par de *sockets,* no cliente e servidor, são conectados por um par de *streams*, um em cada direção. Assim, cada *socket* tem um *input stream* e um *output stream*.

A API Java para *TCP streams* provê as classes *ServerSocket* e *Socket.*

***(IP Multicast)*** A troca de mensagens ponto-a-ponto não é o melhor modelo para comunicação de um processo para um grupo de outros processos, em diferentes computadores numa rede. *IP Multicast* é a transmissão de um datagrama **IP** para um "grupo de hosts", representado por um conjunto de zero ou mais hosts identificados por um único endereço **IP** de destino, que implementa uma comunicação de grupo, em que hosts podem entrar ou sair do grupo. IP multicast é um método de envio de datagramas do *Internet Protocol* (IP) para um grupo de receptores interessados em uma única transmissão. É uma forma de comunicação ponto-a-multiponto empregada para transmissão em aplicações com **streaming media** (fluxo de mídia) na Internet ou em redes privadas. O fluxo de mídia é uma forma de [distribuição digital](https://pt.wikipedia.org/wiki/Distribui%C3%A7%C3%A3o_digital), em oposição ao uso do *download* de dados. O fluxo dos dados é recebido e a mídia é reproduzida à medida que chega ao usuário, Nesta forma, as informações não são armazenadas pelo usuário em seu próprio computador. Assim não é ocupado espaço no [disco rígido](https://pt.wikipedia.org/wiki/Disco_r%C3%ADgido) (*HD*), para a posterior reprodução. Isso permite que um usuário reproduza conteúdos protegidos por [direitos de autor](https://pt.wikipedia.org/wiki/Direitos_autorais), na *Internet*, sem a violação desses direitos, similar ao [rádio](https://pt.wikipedia.org/wiki/Radiodifus%C3%A3o) ou [televisão](https://pt.wikipedia.org/wiki/Televis%C3%A3o) aberta, diferentemente do que ocorreria no caso do *download* do conteúdo, onde há o armazenamento da mídia no *HD*. A informação pode ser transmitida em diversas plataformas, na forma *IP* [*Multicast*](https://pt.wikipedia.org/wiki/Multicast)*.*

A API Java provê uma interface de datagrama para *IP Multicast* através da classe *MulticastSocket*, a qual é subclasse da classe *DatagramSocket (UDP = User Datagram Protocol)*, com a capacidade de juntar-se (*to join*) ou deixar (*to leave*) a um grupo de *multicast*. A classe *MulticastSocket* permite *sockets* ser criados para usar qualquer porta local especificada (por exemplo, 6789), ou qualquer porta local livre. Um processo pode se juntar a um grupo com um dados endereço *multicast*, por invocar o método *joinGroup de sua classe MulticastSocket.* O *multicast socket* junta-se a um grupo em uma dada porta e receberá datagramas, enviados por processos sobre outros computadores, ao grupo, naquela porta. Um processo pode deixar um grupo por invocar o método *leaveGroup* de sua classe *MulticastSocket*.

No cenário visto em termos de uma rede Internet ou privada, diversas instâncias do código da classe *MulticastPeer* (que implementa *MulticastSocket*) devem estar rodando simultaneamente sobre diferentes computadores, todos eles no mesmo grupo, e cada um deles deve receber sua própria mensagem e as mensagens daqueles que se juntaram ao grupo.