**Socket là gì?**

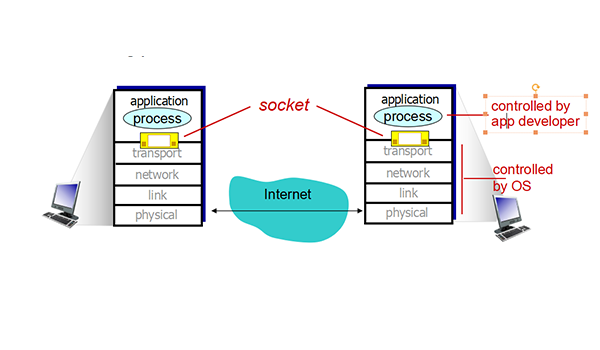
***Socket là điểm cuối end-point trong liên kết truyền thông hai chiều (two-way communication) biểu diễn kết nối giữa Client – Server***. Các lớp Socket được ràng buộc với một cổng port (thể hiện là một con số cụ thể) để các tầng TCP (TCP Layer) có thể định danh ứng dụng mà dữ liệu sẽ được gửi tới. Ứng dụng thực tiễn của Socket là gì? Cùng Mắt Bão tìm hiểu nhé!

Socket hoạt động thông qua các tầng TCP hoặc TCP Layer định danh ứng dụng, từ đó truyền dữ liệu thông qua sự ràng buộc với một cổng port

**Socket** là giao diện lập trình ứng dụng mạng được dùng để truyền và nhận dữ liệu trên internet. Giữa hai chương trình chạy trên mạng cần có một liên kết giao tiếp hai chiều, hay còn gọi là two-way communication để kết nối 2 process trò chuyện với nhau. Điểm cuối (endpoint) của liên kết này được gọi là socket.

Một chức năng khác của socket là giúp các tầng **TCP** hoặc **TCP Layer** định danh ứng dụng mà dữ liệu sẽ được gửi tới thông qua sự ràng buộc với một cổng port (thể hiện là một con số cụ thể), từ đó tiến hành kết nối giữa client và server.

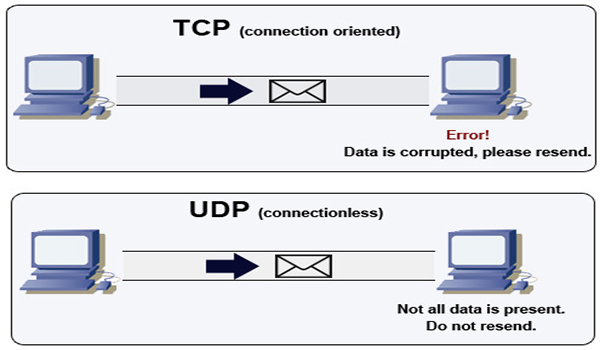
**Tại sao người dùng lại cần đến socket?**

Có thể sử dụng cùng lúc nhiều socket liên tục để tiết kiệm thời gian cũng như nâng cao năng suất làm việc

Ưu điểm lớn nhất của socket là hỗ trợ hầu hết các hệ điều hành bao gồm MS Windows, Linux,… Ngoài ra, socket cũng được sử dụng với nhiều ngôn ngữ lập trình, gồm C, C++, Java, Visual Basic, Visual C++,… nên nó có thể tương thích với hầu hết mọi đối tượng người dùng với những cấu hình máy khác nhau.

Đặc biệt, người dùng cũng có thể chạy cùng một lúc nhiều socket liên tục, giúp nâng cao hiệu suất làm việc, cũng như tiết kiệm thêm nhiều thời gian và công sức hơn.

**Socket hoạt động như thế nào?**

Chức năng của socket là kết nối giữa client và server thông qua TCP/IP và UDP để truyền và nhận giữ liệu qua Internet

Như đã đề cập trước đó, chức năng của **socket** là kết nối giữa client và server thông qua **TCP/IP** và**UDP** để truyền và nhận giữ liệu qua Internet. Giao diện lập trình ứng dụng mạng này chỉ có thể hoạt động khi đã có thông tin về thông số IP và số hiệu cổng của 2 ứng dụng cần trao đổi dữ liệu cho nhau.

2 ứng dụng cần truyền thông tin phải đáp ứng điều kiện sau thì socket mới có thể hoạt động:

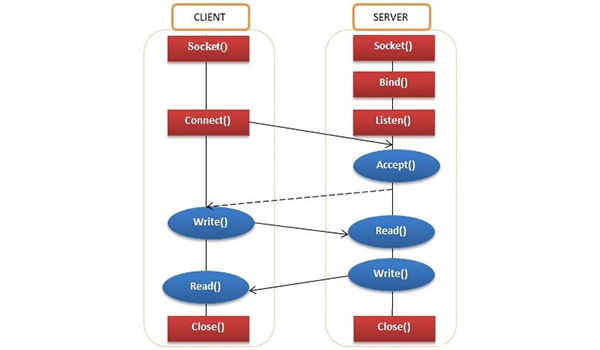
* 2 ứng dụng có thể nằm cùng trên một máy hoặc 2 máy khác nhau
* Trong trường hợp 2 ứng dụng cùng nằm trên một máy, số hiệu cổng không được trùng nhau.

**Phân loại Socket**

**Socket**được chia làm 3 phân loại khác nhau. Dưới đây, hãy cùng Mắt Bão tìm hiểu chi tiết về:

* Stream Socket
* Datagram Socket
* Websocket
* Unix socket

***Stream Socket là gì?***

Stream Socket chỉ hoạt động khi server và client đã kết nối với nhau

**Stream Socket** hay còn gọi là socket hướng kết nối, là socket hoạt động thông qua **giao thức TCP (Transmission Control Protocol)**. Stream Socket chỉ hoạt động khi server và client đã kết nối với nhau. Ngược lại đây chính là định nghĩa cho câu hỏi: “***Giao thức TCP là gì?***“

Ngoài ra, bạn cũng nên tham khảo qua bài viết: “[***TCP/IP là gì?***](https://wiki.matbao.net/kb/tcp-ip-la-gi-uu-va-nhuoc-diem-cua-tcp-ip-so-voi-mo-hinh-osi/)”

***TCP/IP viết tắt của Transmission Control Protocol (TCP) và Internet Protocol (IP) là giao thức cài đặt truyền thông, chồng giao thức mà hầu hết các mạng máy tính ngày nay đều sử dụng để kết nối*. TCP/IP được đặt theo tên của 2 giao thức là giao thức điều khiển giao vận và giao thức liên mạng. Đây là 2 giao thức đầu tiên trên thế giới được định nghĩa. TCP/IP được phát triển vào năm 1978 bởi Bob Kahn và Vint Cerf.**

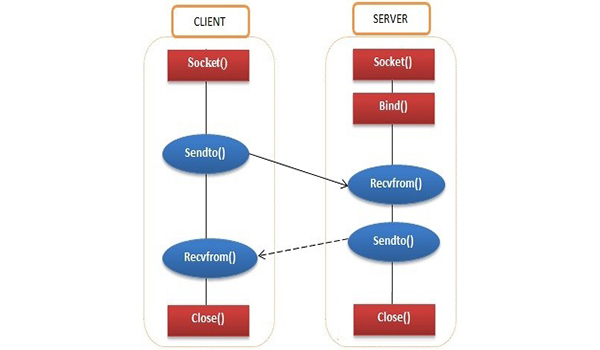
***Ưu điểm của Stream Socket là gì?***

* Dữ liệu truyền đi được đảm bảo truyền đến đúng nơi nhận, đúng thứ tự với thời gian nhanh chóng
* Mỗi thông điệp gửi đi đều có xác nhận trả về để thông báo cho người dùng thông tin về quá trình truyền tải.

***Nhược điểm của Stream Socket là gì?***

* Giữa máy chủ và máy nhận chỉ có 1 IP, nên khi kết nối, 1 máy phải chờ máy còn lại chấp nhận kết nối.

***Datagram Socket là gì?***

Datagram Socket có thể hoạt động kể cả khi không có sự thiết lập kết nối giữa 2 máy với nhau

Datagram **Socket** hay còn gọi là socket không hướng kết nối, là socket hoạt động thông qua**giao thức UDP ( User Datagram Protocol)**. Datagram Socket có thể hoạt động kể cả khi không có sự thiết lập kết nối giữa 2 máy với nhau. Nói cách khác, đây cũng chính là định nghĩa cho câu hỏi: “**Giao thức UDP là gì?**“

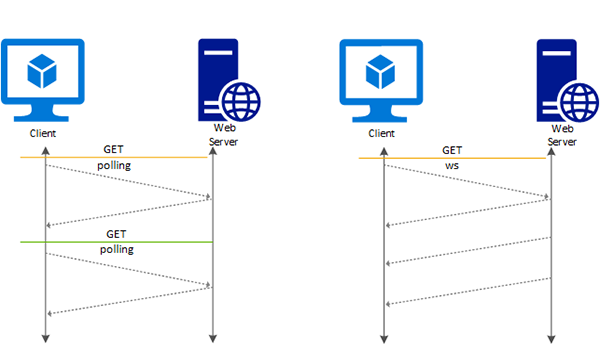
***Ưu điểm của Datagram Socket là gì?***

* Quá trình kết nối và truyền tải thông tin đơn giản, không cần thực hiện nhiều thao tác.
* Thời gian truyền tải dữ liệu cực nhanh.

***Nhược điểm của Datagram Socket là gì?***

* Quá trình truyền thông tin không đảm bảo tin cậy, thông tin có thể truyền sai thứ tự hoặc bị lặp.

***Websocket là gì?***

Websocket có chức năng giúp cho việc kết nối qua lại trên internet giữa client và server diễn ra một cách nhanh chóng và hiệu quả hơn

**Websocket** là công cụ hỗ trợ việc kết nối qua lại trên internet giữa client và server. Giúp diễn ra nhanh chóng và hiệu quả hơn thông qua việc sử dụng **TCP socket**. Không chỉ sử dụng riêng cho ứng dụng web, **Websocket** có thể áp dụng cho bất kì ứng dụng nào khác cần có sự trao đổi thông tin trên Internet.

***Ưu điểm của Websocket là gì?***

**Websocket** mang lại nhiều ưu điểm trong việc kết nối giữa client và server. Cụ thể như sau:

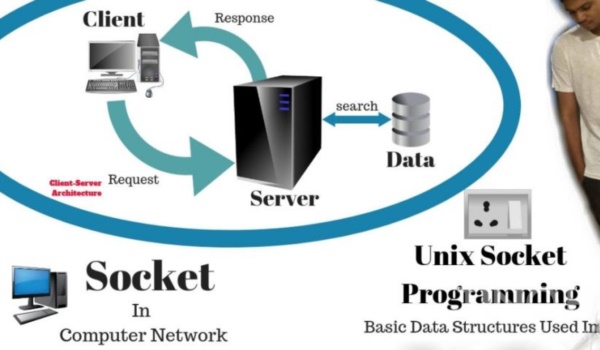
* Tăng tốc độ truyền tải thông tin giữa 2 chiều
* Dễ phát hiện và xử lý trong trường hợp có lỗi xảy ra
* Dễ dàng sử dụng, không cần cài đặt thêm các phần mềm bổ sung khác
* Không cần sử dụng nhiều phương pháp kết nối khác nhau

***Nhược điểm của Websocket là gì?***

Một số nhược điểm của **Websocket** mà bạn cần lưu ý khi sử dụng có thể kể đến như:

* Chưa hỗ trợ trên tất cả các trình duyệt
* Với các dịch vụ có phạm vi yêu cầu, Websocket chưa hỗ trợ hoàn toàn.

***Unix socket là gì?***

Unix socket giúp cho tốc độ kết nối và truyền tải dữ liệu giữa các ứng dụng trên cùng một máy tính diễn ra nhanh, nhẹ và hiệu quả hơn.

**Unix socket** là điểm giao tiếp hỗ trợ trao đổi giữa các ứng dụng khác nhau ngay trên cùng máy tính. Mọi hoạt động Unix socket diễn ra ngay ở nhân của hệ điều hành. Nhờ vậy, tốc độ kết nối và truyền tải giữa các ứng dụng nhanh, nhẹ và hiệu quả hơn.

**Unix socket** có thể tránh được các bước kiểm tra hoặc routing. Vì thế mà quá trình truyền tin sẽ đảm bảo và dễ dàng hơn.

***Ưu điểm Unix socket là gì?***

**Unix socket** có nhiều ưu điểm vượt trội. Nó hỗ trợ cho việc truyền tải nhanh hơn và hiệu quả hơn. Trong đó, tiêu biểu phải kể đến các ưu điểm như:

* Tăng tốc độ truy cập MySQL lên đến 30-50%
* Giảm thời gian latency xuống, từ 60ms còn 5ms
* Tăng PostgreSQL lên  hơn 30%
* Tăng Redis lên 50%
* …

***Nhược điểm Unix socket là gì?***

Bên cạnh các ưu điểm nổi bật phía trên, **Unix socket** cũng còn tồn tại một số nhược điểm như sau:

* Trong trường hợp các ứng dụng nằm trên những máy chủ khác nhau, sẽ không thể kết nối bằng Unix socket.
* Vấn đề phân quyền giữa các tệp tin trên Unix socket đôi khi vẫn xảy ra, ảnh hưởng đến việc sử dụng và thao tác.

**Code Java ví dụ giao thức TCP/IP**

***TCP/IP Socket Server***

package org.o7planning.tutorial.socket;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

import java.net.ServerSocket;

import java.net.Socket;

public class SimpleServerProgram {

public static void main(String args[]) {

ServerSocket listener = null;

String line;

BufferedReader is;

BufferedWriter os;

Socket socketOfServer = null;

// Mở một ServerSocket tại cổng 9999.

// Chú ý bạn không thể chọn cổng nhỏ hơn 1023 nếu không là người dùng

// đặc quyền (privileged users (root)).

try {

listener = new ServerSocket(9999);

} catch (IOException e) {

System.out.println(e);

System.exit(1);

}

try {

System.out.println("Server is waiting to accept user...");

// Chấp nhận một yêu cầu kết nối từ phía Client.

// Đồng thời nhận được một đối tượng Socket tại server.

socketOfServer = listener.accept();

System.out.println("Accept a client!");

// Mở luồng vào ra trên Socket tại Server.

is = new BufferedReader(new InputStreamReader(socketOfServer.getInputStream()));

os = new BufferedWriter(new OutputStreamWriter(socketOfServer.getOutputStream()));

// Nhận được dữ liệu từ người dùng và gửi lại trả lời.

while (true) {

// Đọc dữ liệu tới server (Do client gửi tới).

line = is.readLine();

// Ghi vào luồng đầu ra của Socket tại Server.

// (Nghĩa là gửi tới Client).

os.write(">> " + line);

// Kết thúc dòng

os.newLine();

// Đẩy dữ liệu đi

os.flush();

// Nếu người dùng gửi tới QUIT (Muốn kết thúc trò chuyện).

if (line.equals("QUIT")) {

os.write(">> OK");

os.newLine();

os.flush();

break;

}

}

} catch (IOException e) {

System.out.println(e);

e.printStackTrace();

}

System.out.println("Sever stopped!");

}

}

***TCP/IP Client Socket***

package org.o7planning.tutorial.socket;

import java.io.\*;

import java.net.\*;

public class SimpleClientDemo {

public static void main(String[] args) {

// Địa chỉ máy chủ.

final String serverHost = "localhost";

Socket socketOfClient = null;

BufferedWriter os = null;

BufferedReader is = null;

try {

// Gửi yêu cầu kết nối tới Server đang lắng nghe

// trên máy 'localhost' cổng 9999.

socketOfClient = new Socket(serverHost, 9999);

// Tạo luồng đầu ra tại client (Gửi dữ liệu tới server)

os = new BufferedWriter(new OutputStreamWriter(socketOfClient.getOutputStream()));

// Luồng đầu vào tại Client (Nhận dữ liệu từ server).

is = new BufferedReader(new InputStreamReader(socketOfClient.getInputStream()));

} catch (UnknownHostException e) {

System.err.println("Don't know about host " + serverHost);

return;

} catch (IOException e) {

System.err.println("Couldn't get I/O for the connection to " + serverHost);

return;

}

try {

// Ghi dữ liệu vào luồng đầu ra của Socket tại Client.

os.write("HELO");

os.newLine(); // kết thúc dòng

os.flush(); // đẩy dữ liệu đi.

os.write("I am Tom Cat");

os.newLine();

os.flush();

os.write("QUIT");

os.newLine();

os.flush();

// Đọc dữ liệu trả lời từ phía server

// Bằng cách đọc luồng đầu vào của Socket tại Client.

String responseLine;

while ((responseLine = is.readLine()) != null) {

System.out.println("Server: " + responseLine);

if (responseLine.indexOf("OK") != -1) {

break;

}

}

os.close();

is.close();

socketOfClient.close();

} catch (UnknownHostException e) {

System.err.println("Trying to connect to unknown host: " + e);

} catch (IOException e) {

System.err.println("IOException: " + e);

}

}

}

**Code Java ví dụ giao thức UDP**

***UDP Client Socket***

package com.mycompany.testmqtts;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

public class UDPServer {

public static void main(String args[]) throws Exception {

//khởi động udp server với port 8000

DatagramSocket socket = new DatagramSocket(8000);

System.out.println("server is running");

//tạo chuỗi byte

byte[] inServer = new byte[1024];

byte[] outServer = new byte[1024];

//tạo packet nhận dữ liệu

DatagramPacket rcvPkt = new DatagramPacket(inServer, inServer.length);

while (true) {

// chờ nhận dữ liệu từ client

socket.receive(rcvPkt);

System.out.println("Packet Received!");

System.out.println("ip Address!" + rcvPkt.getAddress());

System.out.println("port!" + rcvPkt.getPort());

System.out.println("message Received!" + new String(rcvPkt.getData()));

InetAddress IP = rcvPkt.getAddress();

int port = rcvPkt.getPort();

//lấy dữ liệu nhận và gửi dữ liệu lại cho client

String temp = new String(rcvPkt.getData());

temp = "server :" + temp.toUpperCase();

outServer = temp.getBytes();

//gửi dữ liệu lại cho client

DatagramPacket sndPkt = new DatagramPacket(outServer, outServer.length, IP, port);

socket.send(sndPkt);

}

}

***UDP Server Socket***

package com.mycompany.testmqtts;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

public class UDPClient {

public static void main(String args[]) {

try {

//tạo kết nối udp socket

DatagramSocket socket = new DatagramSocket();

//tạo các chuỗi byte

byte[] inData = new byte[1024];

byte[] outData = new byte[1024];

//ip or hostname của server udp

InetAddress IP = InetAddress.getByName("localhost");

//chuỗi dữ liệu gửi tới udp server

String data = "hello kaka";

outData = data.getBytes();

//gửi dữ liệu tới server udp

DatagramPacket sendPkt = new DatagramPacket(outData, outData.length, IP, 8000);

System.out.println("ready connect server");

socket.send(sendPkt);

socket.setSoTimeout(10000);

System.out.println("connect server success");

//chờ nhận dữ liệu từ udp server gửi về

DatagramPacket recievePkt = new DatagramPacket(inData, inData.length);

System.out.println("ready receive message from server)");

socket.receive(recievePkt);

System.out.println("receive messag");

System.out.println("Replay from Server: " + new String(recievePkt.getData()));

} catch (Exception e) {

System.out.println("error connect udp server");

}

}

}

**Tổng kết**

Việc tăng tốc độ tải trang là một yếu tố đặc biệt quan trọng. Nó sẽ ảnh hưởng lớn đến việc tăng trải nghiệm khách hàng. Một website có tốc độ tải mượt, nhanh chóng sẽ thu hút sự quan tâm của nhiều khách hàng hơn. Từ đó giúp tăng tỷ lệ khách hàng tiềm năng, cũng như lợi nhuận doanh nghiệp.

**Socket** cũng như các ứng dụng có liên quan là điều tuyệt đối không thể bỏ qua. Các nhà tạo trang web cần có sự lựa chọn. Ngoài ra bạn cũng có thể liên hệ với nhà cung cấp dịch vụ cho [***thuê Server***](https://www.matbao.net/cloud-server.html) để được họ tư vấn kỹ thuật. Đảm bảo sao cho phù hợp với cấu hình máy cũng như mục đích sử dụng để đạt được hiệu quả tối ưu nhất.