Fpt University

FPT POLYTECHNIC

ANDROID NETWORK

BÀI 7: SOCKET

www.poly.edu.vn

Mục tiêu

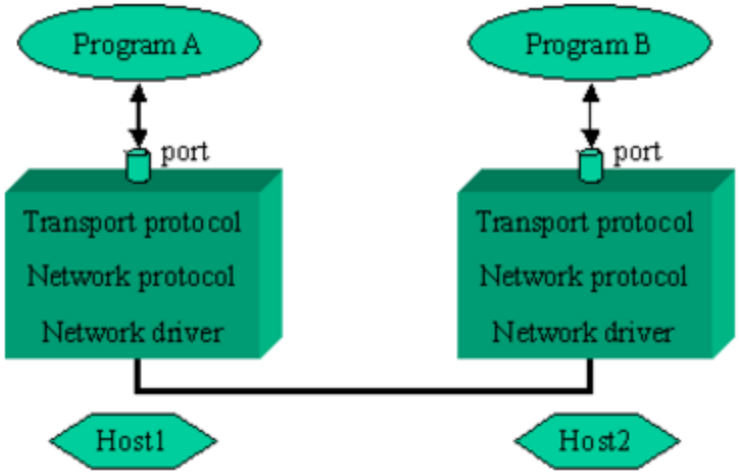
- Tìm hiểu giao tiếp Socket
- Triển khai app trên server kiểu Socket
- Triển khai được cách kết nối giữa android (client) với server bằng Socket.

GIỚI THIỆU VỀ SOCKET

- ✓ Socket là một giao diện lập trình ứng dụng (API-Application Programming Interface).
- ✓ Nó được giới thiệu lần đầu tiên trong ấn bản UNIX - BSD 4.2. dưới dạng các hàm hệ thống theo cú pháp ngôn ngữ C (socket(), bind(), connect(), send(), receive(), read(), write(), close(),..).
- ✓ Ngày nay, Socket được hỗ trợ trong hầu hết các hệ điều hành như MS Windows, Linux và được sử dụng trong nhiều ngôn ngữ lập trình khác nhau: như C, C++, Java, Visual Basic, Visual C++, . . .

GIỚI THIỆU VỀ SOCKET

- ✓ Socket cho phép thiết lập các kênh giao tiếp mà hai đầu kênh được đánh dấu bởi hai cổng (port). Thông qua các cổng này một quá trình có thể nhận và gởi dữ liệu với các quá trình khác.



Mô hình Socket

Port Number của Socket:

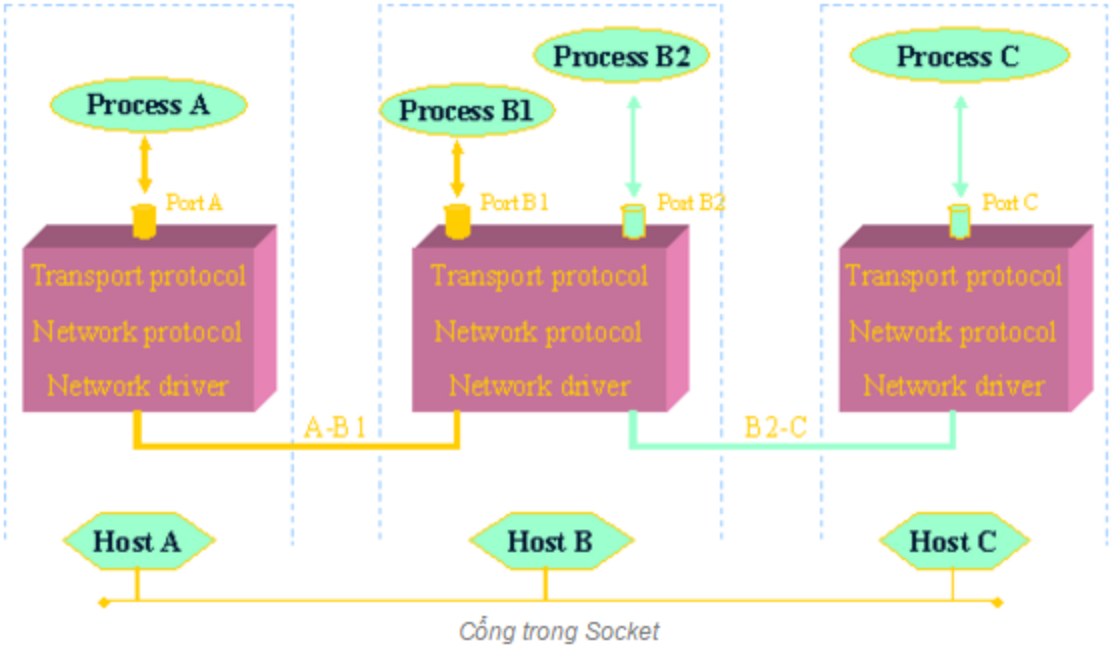
- ✓ Để có thể thực hiện các cuộc giao tiếp, một trong hai quá trình phải công bố số hiệu cổng của socket mà mình sử dụng.
- ✓ Mỗi cổng giao tiếp thể hiện một địa chỉ xác định trong hệ thống.
- ✓ Khi quá trình được gán một số hiệu cổng, nó có thể nhận dữ liệu gởi đến cổng này từ các quá trình khác. Quá trình còn lại cũng được yêu cầu tạo ra một socket.
- ✓ Ngoài số hiệu cổng, hai bên giao tiếp còn phải biết địa chỉ IP của nhau.

Địa chỉ IP giúp phân biệt máy tính này với máy tính kia trên mạng TCP/IP.

Trong khi số hiệu cổng dùng để phân biệt các quá trình khác nhau trên cùng một máy tính.

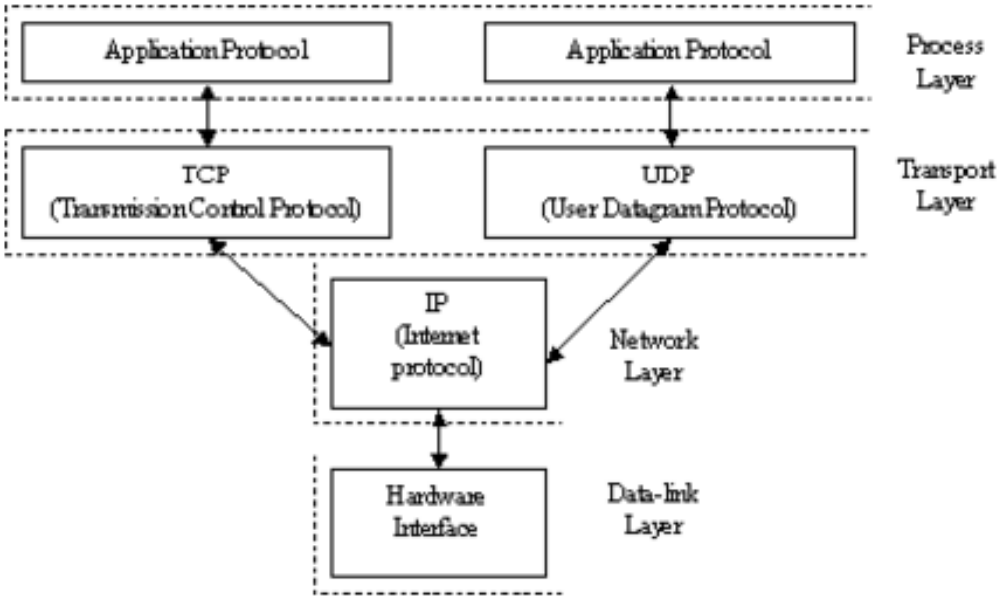
GIỚI THIỆU VỀ SOCKET

- Số hiệu cổng gán cho Socket phải duy nhất trên phạm vi máy tính đó, có giá trị trong khoảng từ 0 đến 65535 (16 bits). Trong đó, các cổng từ 1 đến 1023 được gọi là cổng hệ thống được dành riêng cho các quá trình của hệ thống.
- Ví dụ:
  - HTTP: 80
  - HTTPS: 443
  - FTP: 21
  - POP: 110



Giao thức giao tiếp:

- ✓ TCP: Giao tiếp có nối kết
- ✓ UPD: Giao tiếp không nối kết



Bộ giao thức TCP/IP

- ✓ Tầng vận chuyển giúp chuyển tiếp các thông điệp giữa các chương trình ứng dụng với nhau

GIỚI THIỆU VỀ SOCKET

- ✓ Socket là giao diện giữa chương trình ứng dụng với tầng vận chuyển.
- ✓ Nó cho phép ta chọn giao thức sử dụng ở tầng vận chuyển là TCP hay UDP cho chương trình ứng dụng của mình.

Chế độ có nối kết (TCP)	Chế độ không nối kết (UDP)
Tồn tại kênh giao tiếp ảo giữa hai bên giao tiếp	Không tồn tại kênh giao tiếp ảo giữa hai bên giao tiếp
Dữ liệu được gởi đi theo chế độ bảo đảm: có kiểm tra lỗi. truyền lại gói tin lỗi hay mất, bảo đảm thứ tự đến của các gói tin. . .	Dữ liệu được gởi đi theo chế độ không bảo đảm: Không kiểm tra lỗi, không phát hiện không truyền lại gói tin bị lỗi hay mất, không bảo đảm thứ tự đến của các gói tin . . .
Dữ liệu chính xác, Tốc độ truyền chậm.	Dữ liệu không chính xác, tốc độ truyền nhanh.
Thích hợp cho các ứng dụng cần độ chính xác.	Thích hợp cho các ứng dụng cần tốc độ, không cần chính xác cao: truyền âm thanh, hình ảnh . . .

ServerSocket class:

- ✓ Class dùng tạo đối tượng socket phía server

public ServerSocket(int port, int backlog, InetAddress address) throws IOException

- ✓ Port: số hiệu cổng.
- ✓ Backlog: định rõ có bao nhiêu client đến để lưu trữ trên hàng đợi.
- ✓ Address: chỉ rõ địa chỉ IP local sẽ được nối (bind) tới. Nó dùng cho các máy chủ có thể có nhiều địa chỉ IP, cho phép các máy chủ để xác định các địa chỉ IP của nó để chấp nhận yêu cầu của client.

TRIỂN KHAI APP TRÊN SERVER KIỂU SOCKET

ServerSocket class có các phương thức sau:

- ✓ **public int getLocalPort():** Trả về cổng mà Server Socket đang lắng nghe
- ✓ **public Socket accept() throws IOException:** Đợi một yêu cầu kết nối từ client. Phương thức này sẽ khóa ứng dụng cho tới khi có một yêu cầu kết nối từ client đến trên cổng cụ thể hoặc hết thời gian chờ (Time-out), giả sử rằng thời gian time-out được thiết lập sử dụng phương thức ***setSoTimeout()***. Ngược lại method sẽ bị khóa vô thời hạn.

TRIỂN KHAI APP TRÊN SERVER KIỂU SOCKET

ServerSocket class có các phương thức sau:

- ✓ **public void setSoTimeout(int timeout):** Thiết lập thời gian chờ tối đa (time-out) nghĩa là thời gian Server Socket sẽ chờ yêu cầu kết nối từ người dùng trong quá trình accept().
- ✓ **public void bind(SocketAddress host, int backlog):** Nối Server Socket tới một server cụ thể có cổng định sẵn trong đối tượng SocketAddress. Sử dụng phương thức này nếu bạn khởi tạo ServerSocket không tham số.

TRIỂN KHAI APP TRÊN SERVER KIỂU SOCKET

- ✓ Thông thường, một chương trình server chạy trên một máy tính cụ thể (Server Socket), được ràng buộc bởi cổng (Port number) cụ thể. Các chương trình phục vụ (Server program) chỉ chờ đợi, lắng nghe tại Server Socket các Client để thực hiện một yêu cầu kết nối.
- ✓ Một kết nối giữa chương trình chủ (Server) và 1 Client được tạo ra, bạn nên để chúng giao tiếp với nhau trên một luồng (Thread), như vậy mỗi khi có một kết nối mới một luồng mới lại được tạo ra.

```
private static class ServiceThread extends Thread {
```

TRIỂN KHAI APP TRÊN SERVER KIỂU SOCKET

```
11  public class ServerProgram {
12
13      public static void main(String args[]) throws IOException {
14
15          ServerSocket listener = null;
16
17          System.out.println("Server is waiting to accept user...");
18          int clientNumber = 0;
19
20
21          // Mở một ServerSocket tại cổng 7777.
22          // Chú ý bạn không thể chọn cổng nhỏ hơn 1023 nếu không là người dùng
23          // đặc quyền (privileged users (root)).
24          try {
25              listener = new ServerSocket(7777);
26          } catch (IOException e) {
27              System.out.println(e);
28              System.exit(1);
29          }
30
31          try {
32              while (true) {
33
34
35                  // Chấp nhận một yêu cầu kết nối từ phía Client.
36                  // Đồng thời nhận được một đối tượng Socket tại server.
37
38                  Socket socketOfServer = listener.accept();
39                  new ServiceThread(socketOfServer, clientNumber++).start();
40              }
41          } finally {
42              listener.close();
43          }
44
45      }
46
47      private static void log(String message) {
48          System.out.println(message);
49      }
50
```

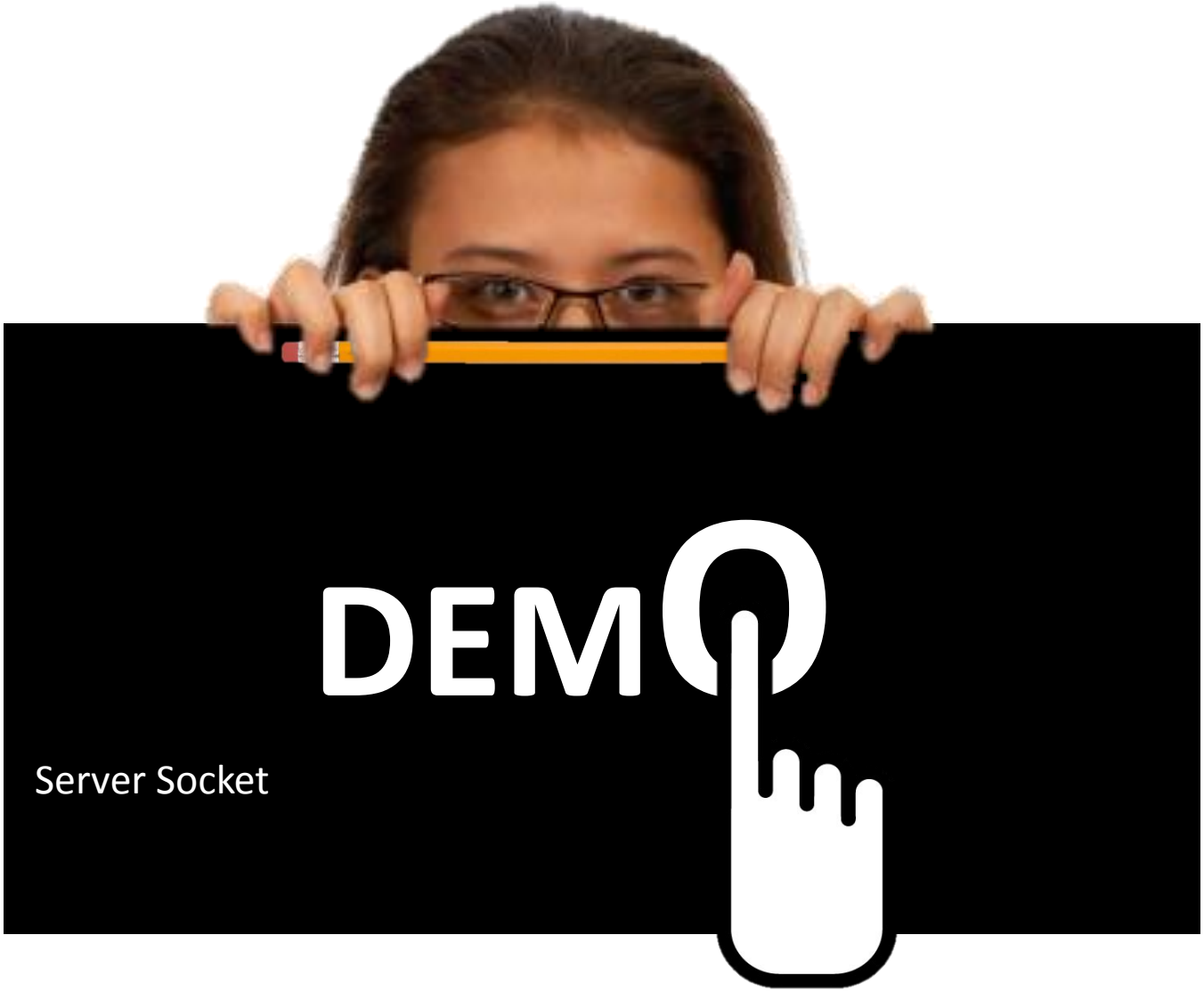
FPT POLYTECHNIC

TRIỂN KHAI APP TRÊN SERVER KIỂU SOCKET

```
51      private static class ServiceThread extends Thread {
52
53          private int clientNumber;
54          private Socket socketOfServer;
55
56          public ServiceThread(Socket socketOfServer, int clientNumber) {
57              this.clientNumber = clientNumber;
58              this.socketOfServer = socketOfServer;
59
60              // Log
61              log("New connection with client# " + this.clientNumber + " at " + socketOfServer);
62          }
63
64          @Override
65          public void run() {
66
67              try {
68
69
70                  // Mở luồng vào ra trên Socket tại Server.
71                  BufferedReader is = new BufferedReader(new InputStreamReader(socketOfServer.getInputStream()));
72                  BufferedWriter os = new BufferedWriter(new OutputStreamWriter(socketOfServer.getOutputStream()))
73
74                  while (true) {
75                      // Đọc dữ liệu tới server (Do client gửi tới).
76                      String line = is.readLine();
77
78                      // Ghi vào luồng đầu ra của Socket tại Server.
79                      // (Nghĩa là gửi tới Client).
80                      os.write(">> " + line);
81                      // Kết thúc dòng
82                      os.newLine();
83                      // Đẩy dữ liệu đi
84                      os.flush();
85
86                      // Nếu người dùng gửi tới QUIT (Muốn kết thúc trò chuyện).
87                      if (line.equals("QUIT")) {
88                          os.write(">> OK");
89                          os.newLine();
90                          os.flush();
91                          break;
92                      }
93                  }
```

TRIỂN KHAI APP TRÊN SERVER KIỂU SOCKET

```
94
95              } catch (IOException e) {
96                  System.out.println(e);
97                  e.printStackTrace();
98              }
99          }
00      }
01  }
```


DEMO
Server Socket

Socket class:

✓ Class dùng kết nối server

public Socket(InetAddress host, int port) throws IOException

Hoặc

public Socket(String host, int port) throws IOException

✓ Host: địa chỉ Server (IP hoặc tên miền)

✓ Port: số hiệu cổng.

TRIỂN KHAI KẾT NỐI GIỮA ANDROID (CLIENT) VỚI SERVER BẰNG SOCKET

Socket class có các phương thức sau:

- ✓ **public void connect(SocketAddress host, int timeout) throws IOException:** kết nối socket này tới host chỉ định, chỉ dung nếu khởi tạo Socket không có tham số kết nối.
- ✓ **public InetAddress getInetAddress():** trả về địa chỉ của máy tính mà socket này kết nối tới.
- ✓ **public int getPort():** Trả về cổng giàng buộc tới máy tính từ xa.
- ✓ **public void close() throws IOException:** Đóng socket

TRIỂN KHAI KẾT NỐI GIỮA ANDROID (CLIENT) VỚI SERVER BẰNG SOCKET

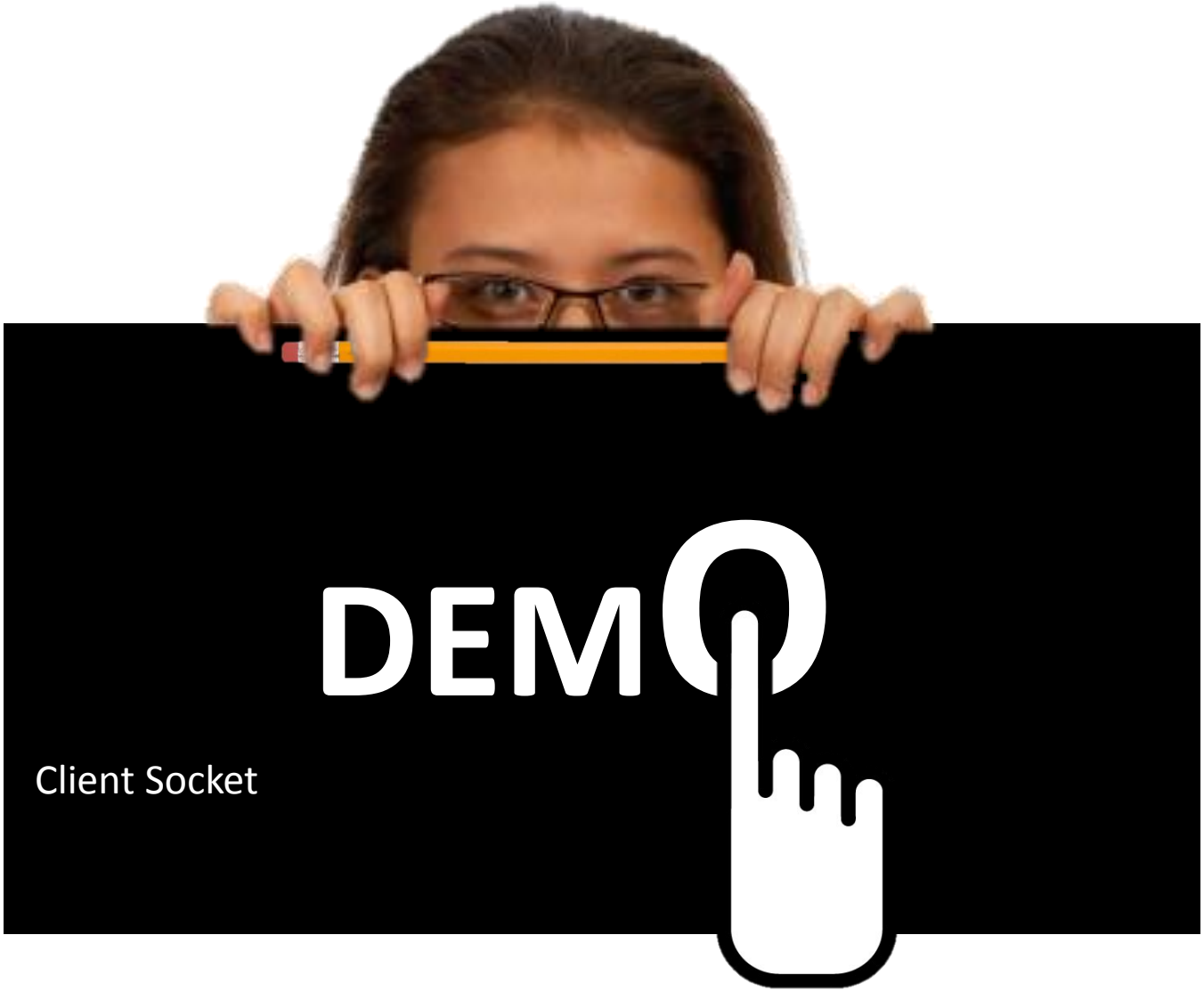
- ✓ **public int getLocalPort():** Trả về cổng socket là ràng buộc để trên máy local.
- ✓ **public SocketAddress getRemoteSocketAddress():** Trả về địa chỉ của socket từ xa (remote socket).
- ✓ **public InputStream getInputStream() throws IOException:** Trả về luồng đầu vào của Socket. Luồng đầu vào này được kết nối với luồng đầu ra của Socket từ xa (remote socket).
- ✓ **public OutputStream getOutputStream() throws IOException:** Trả về luồng đầu ra của Socket. Luồng đầu ra này kết nối với luồng đầu vào của Socket từ xa

TRIỂN KHAI KẾT NỐI GIỮA ANDROID (CLIENT) VỚI SERVER BẰNG SOCKET

```
7  public class ClientDemo {
8
9      public static void main(String[] args) {
10
11         // Địa chỉ máy chủ.
12         final String serverHost = "localhost";
13
14         Socket socketOfClient = null;
15         BufferedWriter os = null;
16         BufferedReader is = null;
17
18         try {
19             // Gửi yêu cầu kết nối tới Server đang lắng nghe
20             // trên máy 'localhost' cổng 7777.
21             socketOfClient = new Socket(serverHost, 7777);
22
23             // Tạo luồng đầu ra tại client (Gửi dữ liệu tới server)
24             os = new BufferedWriter(new OutputStreamWriter(socketOfClient.getOutputStream()));
25
26             // Luồng đầu vào tại Client (Nhận dữ liệu từ server).
27             is = new BufferedReader(new InputStreamReader(socketOfClient.getInputStream()));
28
29         } catch (UnknownHostException e) {
30             System.err.println("Don't know about host " + serverHost);
31             return;
32         } catch (IOException e) {
33             System.err.println("Couldn't get I/O for the connection to " + serverHost);
34             return;
35         }
36
```

TRIỂN KHAI KẾT NỐI GIỮA ANDROID (CLIENT) VỚI SERVER BẰNG SOCKET

```
37          try {
38              // Ghi dữ liệu vào luồng đầu ra của Socket tại Client.
39              os.write("HELO! now is " + new Date());
40              os.newLine(); // kết thúc dòng
41              os.flush();  // đẩy dữ liệu đi.
42              os.write("I am Tom Cat");
43              os.newLine();
44              os.flush();
45              os.write("QUIT");
46              os.newLine();
47              os.flush();
48
49              // Đọc dữ liệu trả lời từ phía server
50              // Bằng cách đọc luồng đầu vào của Socket tại Client.
51              String responseLine;
52              while ((responseLine = is.readLine()) != null) {
53                  System.out.println("Server: " + responseLine);
54                  if (responseLine.indexOf("OK") != -1) {
55                      break;
56                  }
57              }
58
59              os.close();
60              is.close();
61              socketOfClient.close();
62          } catch (UnknownHostException e) {
63              System.err.println("Trying to connect to unknown host: " + e);
64          } catch (IOException e) {
65              System.err.println("IOException:  " + e);
66          }
67      }
68
69  }
```


DEMO
Client Socket

TỔNG KẾT NỘI DUNG BÀI HỌC

- Tìm hiểu giao tiếp Socket
- Triển khai app trên server kiểu Socket
- Triển khai được cách kết nối giữa android (client) với server bằng Socket.