CHƯƠNG IV
LẬP TRÌNH VỚI GIAO THỨC DỊCH VỤ MẠNG PHÍA CLIENT

I. GIỚI THIỆU

Chương này sẽ hướng sinh viên sử dụng kỹ thuật lập trình socket đã được trang bị trong các chương trước để lập trình với một số giao thức dịch vụ mạng phổ biến trên internet như: DSN, Telnet, FTP, TFTP, SMTP, POP3, IMAP4, HTTP, RTP.

Để lập trình được với các giao thức truyền thông có sẵn, người lập trình phải:

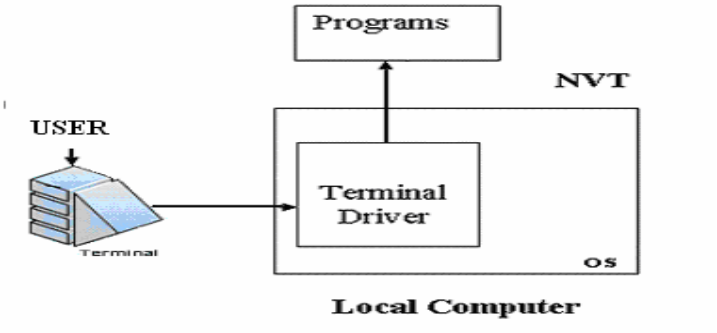
- Khảo sát kỹ đặc điểm, mô hình và cơ chế truyền thông của giao thức;
- Tập lệnh(command), tập đáp ứng(response) và tập tham số của các giao thức;
- Các chế độ hoạt động của giao thức
- Kỹ thuật cài đặt giao thức bằng các ngôn ngữ lập trình

Thông qua đó sinh viên nắm được kỹ thuật lập trình với các giao thức truyền thông có sẵn khác để phát triển các ứng dụng hoặc phát triển các modul tích hợp giải quyết các bài toán thực tế.

II. LẬP TRÌNH GIAO THỨC DỊCH VỤ TELNET

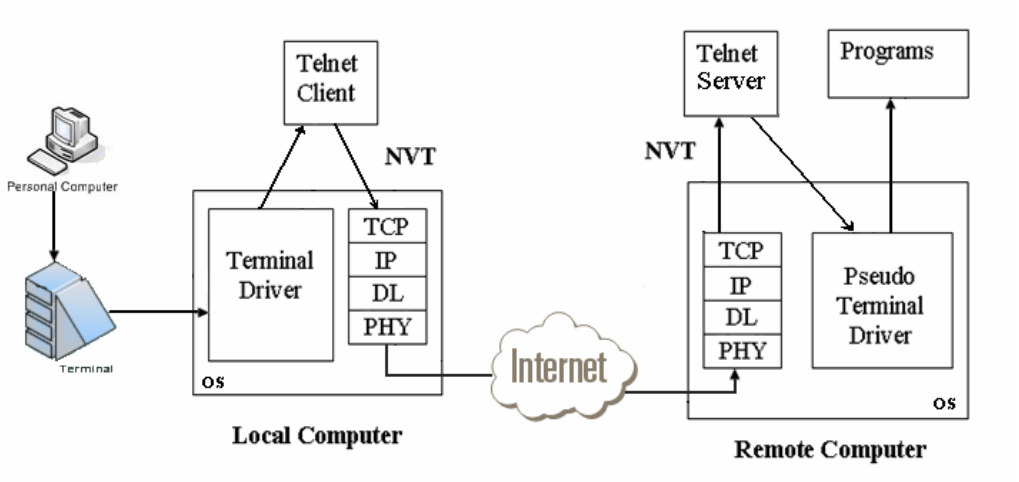
1. Một số khái niệm và đặc điểm của dịch vụ Telnet

- *Đầu cuối*: Trong dịch vụ Telnet, đầu cuối có thể coi là tổ hợp của bàn phím và màn hình. Thiết bị đầu cuối này cho phép người sử dụng nhập dữ liệu gửi tới trung tâm xử lý và nhận kết quả trả về.
- *Môi trường chia sẻ thời gian*: đây thực chất là một mạng các đầu cuối, các đầu cuối được kết nối với nhau thông qua trung tâm xử lý thường là một máy tính mạnh. Trong môi trường chia sẻ thời gian, các ký tự được người sử dụng nhập vào bàn phím đều được chuyển tới trung tâm xử lý. Sau khi xử lý xong kết quả được trả về màn hình người sử dụng.
- *Đầu cuối ảo*: khi một máy tính kết nối qua mạng Internet với máy tính từ xa với vai trò như một đầu cuối cục bộ trên máy tính từ xa đó gọi là đầu cuối ảo. Mạng gồm nhiều đầu cuối ảo được gọi là mạng đầu cuối ảo (Network Virtual Terminal).
- *Đăng nhập*: đây là quá trình người sử dụng mã tài khoản để truy nhập vào hệ thống từ xa. Có hai loại đăng nhập:
  - ❖ Đăng nhập cục bộ: là quá trình đăng nhập vào môi trường chia sẻ thời gian cục bộ.



Hình 4.1. Đăng nhập cục bộ

- Đăng nhập từ xa: máy tính cục bộ phải cài phần mềm Telnet client, máy tính từ xa phải cài phần mềm Telnet server.



Hình 4.2. Đăng nhập từ xa

Quá trình đăng nhập: Khi người sử dụng nhập các ký tự thông qua đầu cuối, ký tự đó sẽ được gửi tới Hệ điều hành của máy tính cục bộ (hệ điều hành không dịch ký tự đó mà nó gửi đến cho chương trình Telnet Client). Chương trình Telnet Client dịch ký tự đó ra dạng tập ký tự chung NVT-ASCII 7 bít và gửi đến các tầng TCP/IP để chuyển qua mạng Internet, tới các tầng TCP/IP của máy tính từ xa. Hệ điều hành gửi các ký tự đó đến chương trình Telnet Server, chương trình này sẽ dịch các ký tự đó ra dạng mà máy tính từ xa có thể hiểu được. Nhưng do hệ điều hành được thiết kế không cho phép gửi ký tự ngược lại hệ điều hành. Để giải quyết vấn đề này, trên máy tính từ xa bổ sung thêm modul phần mềm giả lập đầu cuối (Pseudo Terminal Driver). Từ đó Telnet Server gửi ký tự đó đến cho phần mềm này và chuyển tiếp đến hệ điều hành. Hệ điều hành sẽ gửi các ký tự đó đến chương trình phù hợp.

- Đặc điểm của dịch vụ Telnet:
  - TELNET= TErminaL NETwork
  - Telnet sử dụng kết nối TCP với số cổng mặc định là 23
  - Telnet gồm 2 phần mềm: Telnet client cài trên máy cục bộ, Telnet Server cài trên máy từ xa.
  - Telnet là dịch vụ đăng nhập từ xa. Sau khi đăng nhập thành công, máy cục bộ trở thành đầu cuối ảo của máy từ xa(màn hình , bàn phím... trở thành của máy từ xa). Dịch vụ cho phép truy cập và thao tác với tài nguyên trên máy từ xa.
  - Dịch vụ Telnet hiện đã được tích hợp vào hệ điều hành mạng và được coi như là giao thức chuẩn của TCP/IP.
- Đối với lập trình ứng dụng mạng, bài toán quan trọng nhất là xây dựng chương trình phần mềm phía client. Điều này cho phép người sử dụng có thể tạo ra được phần mềm với giao diện phù hợp và dễ dàng tích hợp với các dịch vụ khác. Để lập trình đươc dịch vụ Telnet phía người sử dụng,người lập trình phải nắm chắc tập ký tự NVT, các tuỳ chọn và các chính sách thoả thuận tuỳ chọn của Telnet, các lệnh điều khiển server và cấu trúc lệnh Telnet. Cuối cùng người sử dụng phải nắm được các chế độ hoạt động của Telnet trước khi cài đặt chương trình Telnet.

2. Một số kiến thức giao thức Telnet cơ bản

2.1. Tập ký tự chung NVT

Để tạo ra sự độc lập giữa máy tính cục bộ và máy tính từ xa trong các mạng không đồng nhất, telnet định nghĩa một giao diên chung gọi là tập kí tự mạng đầu cuối ảo NVT (Network Virtual Terminal). NVT gồm 2 tập kí tự:

- Tập ký tự dữ liệu: có bit cao nhất bằng 0 và có mã thuộc [0,127] .
- Tập ký tự điều khiển: có bit cao nhất bằng 1 và có mã thuộc [128,255] .

Name	**Code**	**Decimal Value**	**Function**
NULL	NUL	0	No operation
Line Feed	LF	10	Di chuyển máy in tới hàng in tiếp theo, định vị vị trí nằm ngang.
Carriage			Di chuyển máy in sang bên trái
Return	CR	13	Lề của hàng hiện thời
BELL	BEL	7	Sinh ra một tín hiệu nghe được hoặc rõ ràng (mà không di chuyển đầu in).
Back Space	BS	8	Di chuyển đầu in một ký tự định vị về phía lề trái (trên thiết bị in, mà thiết bị này thông thường được sử dụng tới mẫu văn bản ký tự hoàn chỉnh bằng cách in hai ký tự cơ bản trên phần đầu lẫn nhau).
Horizontal Tab	HT	9	Di chuyển máy in tới Horizontal Tab tiếp theo (Nó giữ nguyên không được chỉ rõ phải làm như thế nào để mỗi nhóm xác định hoặc thiết lập nơi được định vị).
Vertical Tab	VT	11	Tương tự như HT
Form Feed	FF	12	Di chuyển máy in tới phần đầu của trang tiếp theo và giữ vị trí nằm ngang (trên hiển thị trực quan, việc xóa màn hình và di chuyển con trỏ tới góc trái)

Một số kí tự dữ liệu quan trọng

Name	**Decimal Code**	**Meaning**
SE	240	End of subnegotiation parameters: Kết thúc của tham số thỏa thuận
NOP	241	No operation: không thao tác
		Data mark: Chỉ ra vị trí của sự kiện đồng bộ bên trong

DM	242	luồng dữ liệu. (Cái này luôn phải được kèm theo cảnh báo TCP).
BRK	243	Break: chỉ ra sự thoát
IP	244	Interupt Process: dùng để ngắt tiến trình đang chạy trên máy từ xa.
AO	245	Abort output: cho phép tiến trình hiện thời chạy hoàn thành nhưng không gửi đầu ra của nó cho người sử dụng
AYT	246	Are you there: gửi đến cho server và hỏi xem server còn hoạt động không.
EC	247	Erase character: người nhận nên xóa ký tự trước lần cuối từ luồng dữ liệu.
EL	248	Erase line: xóa ký tự từ luồng dữ liệu nhưng không bao gồm CRLF
GA	249	Go ahead: người dùng, dưới những hoàn cảnh nhất định có thể diễn tả kết thúc khác mà nó có thể truyền.
SB	250	SubOption Begin: chỉ thị bắt đầu một tùy chọn thành phần.
WILL	251	Chỉ ra sự mong muốn bắt đầu được thực hiện hoặc sự xác nhận mà bạn đang thực hiện.
WONT	252	Chỉ ra sự từ chối thực hiện hoặc tiếp tục thực hiện.
DO	253	Chỉ ra yêu cầu mà một nhóm thực hiện khác hoặc xác nhận điều bạn đang mong đợi của nhóm khác thực hiện.
DON'T	254	Chỉ ra sự yêu cầu mà nhóm khác ngừng thực hiện xác nhận điều mà bạn không mong chờ nhóm khác thực hiện.
IAC	255	Interpret as command: Đây là ký tự không dịch lệnh

Một số kí tự điều khiển quan trọng

2.2. Các tùy chọn

Các tùy chọn: được sử dụng để bổ sung thêm thông tin cho các lệnh:

Echo: hiển thị trả lời.

Terminal Type: tuỳ chọn kiểu đầu cuối.

Terminal Speed: thỏa thuận về tốc độ đầu cuối.

Binary : cho phép người nhận dịch mọi kí tự 8 bit như là dữ liệu nhị phân, trừ kí tụ IAC

Echo: cho phép Server phản hồi dữ liệu nhận được trở lại client để hiện lên màn hình

Suppress go head : loại bỏ kí tự CA

Timing: cho phép một thành viên phát sinh dấu hiệu định thời, để chỉ thị rằng tất cả dữ liệu nhận được trước đó đã được xử lý. Mã của các tuỳ chọn được thể hiện trong bảng sau:

Decimal Code	Name	RFC
1	Echo	857
3	Suppress go ahead	858
5	Status	859
6	Timing mark	860
24	Terminal type	1091
31	Window size	1073
32	Terminal speed	1079
33	Remote flow control	1372
34	Linemode	1184
36	Environment variables	1408

2.3. Sự thỏa thuận các tùy chọn

Trong Telnel trước khi sử dụng một tùy chọn nào đó thì giữa Client và Server phải có thỏa thuận về tùy chọn đó. Có hai phương thức thỏa thuận là: đề nghị và yêu cầu.

Với hai hình thức này thì có hai kiểu thỏa thuận:

- ✓ Cho phép một tùy chọn
- ✓ Làm mất hiệu lực một tùy chọn

Các lệnh dùng trong thỏa thuận tùy chọn:WILL, DO, WONT, DONT

2.4. Sự nhúng trong telnet

Trong telnet để gửi các lệnh và dữ liệu thì sử dụng một kết nối duy nhất, các lệnh được nhúng ở trong dòng dữ liệu để bên nhận phân biệt được lệnh với dữ liệu trước mỗi kí tự điều khiển đều có kí tự IAC. Trong trường hợp có 2 kí tự IAC đi liền nhau thì kí tự IAC thứ nhất sẽ bị bỏ qua và kí tự IAC thứ hai sẽ là dữ liệu.

2.5. Các chế độ làm việc của Telnet

- *Chế độ mặc định*: được sử dụng khi không có sự thỏa thuận dùng một chế độ khác. Trong chế độ này, khi các ký tự được nhập vào từ bàn phím, nó sẽ phản hồi ngay lên màn hình cục bộ và chỉ khi nhập hoàn chỉnh cả dòng ký tự thì dòng đó mới được gửi sang server và nó phải chờ tín hiệu GA (go Ahead) từ server trả về mới chấp nhận dòng mới (truyền theo kiểu half-duplex).
- *Chế độ Character*: trong chế độ này, mỗi khi có ký tự nhập vào từ bàn phím, trình Telnet Client gửi ký tự đó đến cho Server, Server sẽ gửi phản hồi ký tự đó lại trình Client để hiển thị lên màn hình cục bộ.

- *Chế độ Line Mode*: chế độ này bổ sung sự khiếm khuyết của hai chế độ trên. Mỗi khi Client nhận một dòng, nó gửi tới Server và nó sẽ nhận dòng mới mà không cần chờ tín hiệu GA gửi về từ Server (truyền thông theo kiểu full-duplex).

3. Cài đặt dịch vụ Telnet Client với Java

Chương trình Telnet phía người sử dụng phải thực hiện các công việc sau:

- Tạo một đối tượng Socket và thiết lập kết nối tới TelnetServer với địa chỉ máy mà trên đó trình Telnet Server đang chạy, và số cổng mà Telnet Server đang nghe.

Ví dụ: Giả sử telnet server chạy trên may tính có địa chỉ IP là 192.168.1.10, địa chỉ cổng là 23:

```
Socket telnetclient=new Socket("192.168.1.10",23);
```

- Tạo luồng nhập/xuất cho socket.
- Thực hiện gửi/ nhận các lệnh của Telnet thôing qua luồng nhập/xuất

ví dụ khi thoả thuận, client cần phải gửi lệnh WONT có mã là 252, IAC là 255 với lệnh:

```
if(c2==255)
{
    out.write(new byte[] {(byte)255, (byte)254, (byte)c2});
}
```

- Xây dựng giao diện GUI cho chương trình nếu muốn.

Sau đây là một chương trình ví dụ cài đặt dịch vụ Telnet đơn giản với giao thức Telnet:

```
// TelnetClient.java
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;
// Terminal hiển thị chữ trên cửa số
class Terminal extends Canvas
{
      // Kích cỡ font chữ
      private int charWidth, charHeight;
      // text[0] là dòng thao tác hiện tại
      private String[] text;
      // Khoảng cách với viền cửa sổ chính chương trình
      private final int margin=4;
      // Số dòng lệnh tối đa được lưu lại
      private final int lines=50;

      // Constructor, khởi tạo các giá trị ban đầu
      Terminal()
      {
            charHeight=12;
```

```
        setFont(new Font("Monospaced", Font.PLAIN, charHeight));
        charWidth=getFontMetrics(getFont()).stringWidth(" ");
        text=new String[lines];
        for (int i=0; i<lines; ++i)
        text[i]="";
        setSize(80*charWidth+margin*2, 25*charHeight+margin*2);
        requestFocus();
        // Lắng nghe sự kiện con trỏ chuột
        addMouseListener(new MouseAdapter() {
              public void mousePressed(MouseEvent e)
              {
                    requestFocus();
              }
        });
}
// In và lưu lại các kí tự người dùng nhập từ bàn phím
public void put(char c)
{
        Graphics g=getGraphics();
        if (c=='\r')
        {  // Return
              for (int i=lines-1; i>0; --i)
                    text[i]=text[i-1];
              text[0]="";
              update(g);  // Clear screen and paint
        }
        // Các kí tự điều khiển: backspace, delete, telnet EC
        else if (c==8 || c==127 || c==247)
        {
              int len=text[0].length();
              if (len>0)
              {
                    --len;
text[0]=text[0].substring(0, len);
g.setColor(getBackground());
g.fillRect(len*charWidth+margin, getSize().height-margin-charHeight,
             (len+1)*charWidth+margin, getSize().height-margin);
              }
        }
else if (c=='\t')
        {     // Tab với khoảng cách 8 space
        text[0]+="        ";
        text[0].substring(0, text[0].length()&-8);
```

```
            }
      else if (c>=32 && c<127)
            {  // Kí tự có thể in
       g.drawString(""+c, margin+text[0].length()*charWidth,
              getSize().height-margin);
      text[0]+=c;
            }
      g.dispose();
      }
      // Hiển thị những gì đã gõ từ bàn phím
      public void paint(Graphics g)
      {
            int height=getSize().height;
            for (int i=0; i<lines; ++i)
            g.drawString(text[i], margin, height-margin-i*charHeight);
      }
}
// luồng nhận sẽ chờ các kí tự đến từ một luồng vào (Input
// stream) và gửi đến Termial. Đàm phán các lựa chọn đầu cuối
class Receiver extends Thread
{
      private InputStream in;
      private OutputStream out;
      private Terminal terminal;
      public Receiver(InputStream in, OutputStream out, Terminal terminal)
      {
          this.in=in;
          this.out=out;
          this.terminal=terminal;
          start();
      }
      // Đọc các kí tự và gửi đến đầu cuối
      public void run()
      {
      while (true)
      {
      try {
      int c=in.read();
      if (c<0)
      {     // EOF
      System.out.println("Connection closed by remote host");
            return;
      }
```

```
      else if (c==255)
      {     // Đàm phán các lựa chọn đầu cuối
      int c1=in.read();  // 253=do, 251=will
      int c2=in.read();  // option
      if (c1==253)  // do option, send "won't do option"
      out.write(new byte[] {(byte)255, (byte)252, (byte)c2});
      else if (c1==251) // will do option, send "don't do option"
      out.write(new byte[] {(byte)255, (byte)254, (byte)c2});
      }
      else
      terminal.put((char)c);
      }
      catch (IOException x) {
            System.out.println("Receiver: "+x);
      }}}}
// TelnetWindow. Gửi dữ liệu bàn phím từ terminal đến một socket từ
// xa và bắt đầu nhận các kí tự từ socket và hiển thị các kí tự đó  trên
terminal
class TelnetWindow extends Frame
{
      Terminal terminal;
      InputStream in;
      OutputStream out;
      // Constructor
      TelnetWindow(String hostname, int port)
      {
            super("telnet "+hostname+" "+port);  // Set title\
            // Thiết lập cửa sổ
            add(terminal=new Terminal());
            // Xử lý việc đóng cửa sổ
            addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e)
            {
            dispose();
            try {
            out.close();
               }
            catch (IOException x) {
            System.out.println("Closing connection: "+x);
            }
                  }
                  public void windowClosed(WindowEvent e) {
                        System.exit(0);
                  }
```

```
            });
            // Xử lý các thao tác với bàn phím
            terminal.addKeyListener(new KeyAdapter() {
            public void keyTyped(KeyEvent e) {
                  char k=e.getKeyChar();
                  try {
                        terminal.put(k);
                        out.write((int)k);
                        if (k=='\r')
                        {
                              out.write('\n');  // Convert CR to CR-LF
                              out.flush();
                        }
                        }
                        catch (IOException x) {
                        System.out.println("Send: "+x);
                        }}});
          try {
            // Mở một connection
            System.out.println("Opening connection to "+hostname+" on port
"+port);
            Socket socket=new Socket(hostname, port);
            InetAddress addr=socket.getInetAddress();
            System.out.println("Connected to "+addr.getHostAddress());
            in=socket.getInputStream();
            out=socket.getOutputStream();
            // Hiển thị cửa sổ
            pack();
            setVisible(true);
            // Bắt đầu nhận dữ liệu từ server
            new Receiver(in, out, terminal);
            System.out.println("Ready");
          }
          catch (UnknownHostException x) {
            System.out.println("Unknown host: "+hostname+" "+x);
            System.exit(1);
            }
          catch (IOException x) {
                  System.out.println(x);
                  System.exit(1);
            }}}
// Chương trình chính
public class TelnetClient
{
```

```
public static void main(String[] argv)
{
    // Phân tách các đối số: telnet hostname port
    String hostname="";
    int port=23;
    try {
            hostname=argv[0];
            if (argv.length>1)
            port=Integer.parseInt(argv[1]);
    }   catch (ArrayIndexOutOfBoundsException x) {
      System.out.println("Usage: java telnet hostname [port]");
      System.exit(1);
    }
    catch (NumberFormatException x) {}
    TelnetWindow t1=new TelnetWindow(hostname, port);
   }
}
```

4. Chạy thử chương trình

Bước 1: Dịch chương trình TelnetClient.java

Bước 2: Kiểm tra xem trên máy từ xa, trình Telnet server đã được khởi tạo chạy chưa, nếu chưa thì chạy nó và dùng trình quản trị Telnet Server, thiết lập các tham số phù hợp.

Bước 3: Chạy chương trình Telnet Client từ máy cục bộ.

III. LẬP TRÌNH DỊCH VỤ TRUYỀN TỆP VỚI GIAO THỨC FTP

1. Dịch vụ truyền tệp FTP

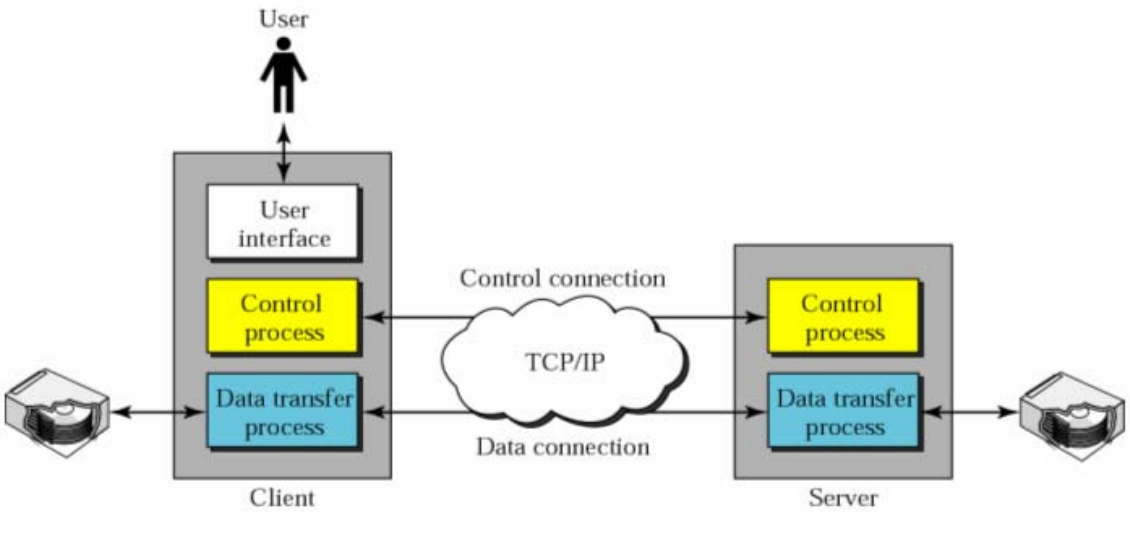
1.1. Giao thức FTP

1.1.1. Đặc điểm

- FTP là giao thức chuẩn của TCP/IP
- FTP sử dụng kết nối TCP, là kết nối truyền thông tin cậy
- FTP gồm 2 phần mềm: Phần mềm FTPClient cài trên máy cục bộ và FTPServer cài trên máy từ xa(File Server).
- FTP sử dựng 2 kết nối truyền thông đồng thời để tăng hiệu quả của việc truyền tệp qua mạng:
  - ❖ Kết nối điều khiển: Sử dụng phương thức truyền thông đơn giản và dữ liệu truyền dưới dạng text(NVT-ASCII 7bít). Kết nối này cho phép truyền lệnh từ client tới server và truyền đáp ứng từ server về client. Kết nối này sử dụng số cổng mặc định là 21 phía server.
  - ❖ Kết nối dữ liệu: Kết nối này sử dụng các phương thức truyền thông phức tạp vì phải truyền nhiều kiểu dữ liệu khác nhau. Kết nối này được thiết lập mỗi khi truyền một tệp và huỷ sau khi truyền xong tệp đó. Kết nối này bao giờ cũng được khởi tạo sau kết nối điều khiển và kết thúc trước khi huỷ bỏ kết nối điều khiển(kết nối điều khiển duy trì trong suốt phiên làm việc). Kết nối dữ liệu sử dụng số cổng

mặc định phía server là 20. Có 2 cách thiết lập kết nối dữ liệu: dùng lệnh PORT và lệnh PASV.

- FTP có 3 chế độ truyền tệp:
  - ❖ Cất tệp trên máy cục bộ lên máy tính từ xa dưới sự giám sát của lệnh STOR.
  - ❖ Lấy một tệp trên máy tính từ xa về máy tính cục bộ dưới sự giám sát của lệnh RETR.
  - ❖ Lấy danh sách các mục trong một thư mục trên máy từ xa về máy cục bộ dưới sự giám sát của lệnh LIST.
- Mô hình hoạt động của FTP thể hiện như hình vẽ



Hình 4.3. Mô hình FTP

1.1.2. *Tập lệnh và đáp ứng của FTP*

1.1.2.1. Tập lệnh:

Tập lệnh FTP chỉ được thi hành phía FTP Server, không dùng cho người sử dụng. Khi client gửi một lệnh FTP đến FTPServer, lệnh đó sẽ được FTPServer thi hành và trả đáp ứng về cho client. Cú pháp lệnh FTP có dạng:

<COMMAND> <SPACE> [PARAMS]

FTP có hơn ba mươi lệnh được chia làm sáu nhóm và được liệt kê trong bảng sau:

Nhóm lệnh truy cập:

Command	*Argument(s)*	*Description*
USER	User id	User information
PASS	User password	Password
ACCT	Account to be charged	Account information
REIN		Reinitialize
QUIT		Log out of the system
ABOR		Abort the previous command

Nhóm lệnh quản lý tệp:

Command	*Argument(s)*	*Description*
CWD	Directory name	Change to another directory
CDUP		Change to the parent directory
DELE	File name	Delete a file
LIST	Directory name	List subdirectories or files
NLIST	Directory name	List the names of subdirectories or files without other attributes
MKD	Directory name	Create a new directory
PWD		Display name of current directory
RMD	Directory name	Delete a directory
RNFR	File name (old file name)	Identify a file to be renamed
RNTO	File name (new file name)	Rename the file
SMNT	File system name	Mount a file system

Nhóm lệnh định dạng dữ liệu:

Command	*Argument(s)*	*Description*
TYPE	A (ASCII), E (EBCDIC), I (Image), N (Nonprint), or T (TELNET)	Define the file type and if necessary the print format
STRU	F (File), R (Record), or P (Page)	Define the organization of the data
MODE	S (Stream), B (Block), or C (Compressed)	Define the transmission mode

Nhóm lệnh định nghĩa cổng:

Command	*Argument(s)*	*Description*
PORT	6-digit identifier	Client chooses a port
PASV		Server chooses a port

Nhóm lệnh truyền tệp:

Command	*Argument(s)*	*Description*
ALLO	File name(s)	Allocate storage space for the files at the server
REST	File name(s)	Position the file marker at a specified data point
STAT	File name(s)	Return the status of files

Nhóm lệnh còn lại:

Command	*Argument(s)*	*Description*
HELP		Ask information about the server
NOOP		Check if server is alive
SITE	Commands	Specify the site-specific commands
SYST		Ask about operating system used by the server

1.2.1.2. Tập đáp ứng(response)

Đáp ứng FTP được gửi từ FTP server về client sau mỗi khi FTP server thực thi một lệnh FTP gửi từ client đến server. Cú pháp của một đáp ứng của FTP có dạng sau:

<XYZ> <SPACE> <TEXT>

Với XYZ là phần mã gồm 3 số nguyên, mỗi chữ số và giá trị số được ấn định với một ý nghĩa xác định:

Code	*Description*
Positive Preliminary Reply	
120	Service will be ready shortly
125	Data connection open; data transfer will start shortly
150	File status is OK; data connection will be open shortly

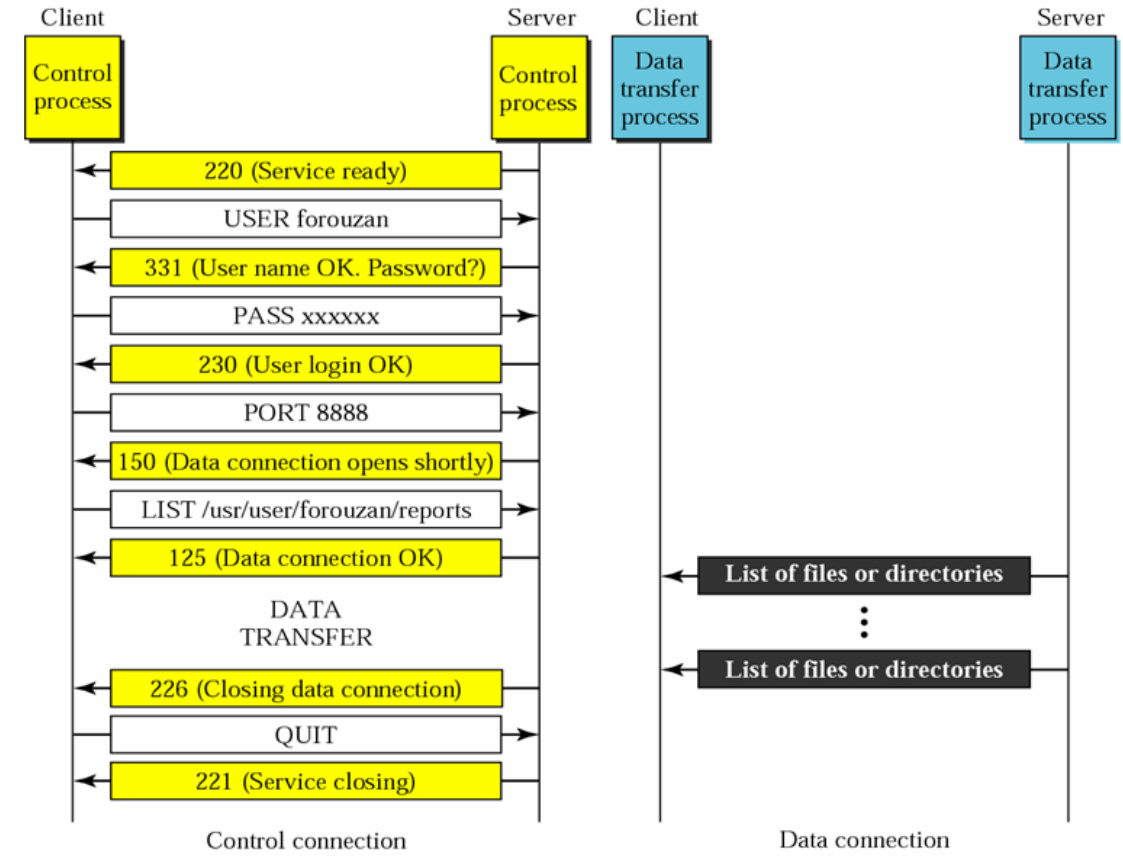
Positive Completion Reply	
200	Command OK
211	System status or help reply
212	Directory status
213	File status
214	Help message
215	Naming the system type (operating system)
220	Service ready
221	Service closing
225	Data connection open
226	Closing data connection
227	Entering passive mode; server sends its IP address and port number
230	User login OK
250	Request file action OK

Positive Intermediate Reply	
331	User name OK; password is needed
332	Need account for logging
350	The file action is pending; more information needed

Transient Negative Completion Reply	
425	Cannot open data connection
426	Connection closed; transfer aborted
450	File action not taken; file not available
451	Action aborted; local error
452	Action aborted; insufficient storage
Permanent Negative Completion Reply	
500	Syntax error; unrecognized command

501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command parameter not implemented
530	User not logged in
532	Need account for storing file
550	Action is not done; file unavailable
552	Requested action aborted; exceeded storage allocation
553	Requested action not taken; file name not allowed

1.1.3. Ví dụ quá trình truyền tệp giữa FTPclient và FTPserver



Hình 4.4. Ví dụ quá trình truyền tệp FTP

2. Kỹ thuật cài đặt giao thức FTP với java

2.1. Các bước cài đặt:

Để có thể truyền tệp với máy chủ truyền tệp với giao thức FTP, chương trình phải:

- Thiết lập và huỷ bỏ kết nối điều khiển.
- Thiết lập và huỷ bỏ kết nối dữ liệu sử dụng lệnh PORT hoặc PASV

- Gửi các lệnh từ client tới server và nhận đáp ứng từ server trả về. Tốt nhất là viết các phương thức bao lấy các lệnh của FTP và phương thức xử lý đáp ứng trả về.
- Nắm chắc trình tự để có thể thực hiện download hoặc upload tệp sử dụng giao thức FTP.

2.2.Chương trình truyền tệp FTP

Trong chương trình này, chúng tôi thực hiện các công việc sau:

- Khai báo tạo đối tượng Socket và thiết lập kết nối tới FTPServer để tạo kết nối điều khiển và tạo luồng nhập xuất cho socket:

Ví du: Giả sử FTPServer nằm trên máy cục bộ và sử dụng số cổng mặc định 21

Socket clientFTP=new Socket("localhost",21);

Hoặc viết phương thức kết nối như ví dụ sau:

```
public boolean connect(String host, int port)
        throws UnknownHostException, IOException
    {
        connectionSocket = new Socket(host, port);
        outputStream = new
PrintStream(connectionSocket.getOutputStream());
        inputStream = new BufferedReader(new

InputStreamReader(connectionSocket.getInputStream()));

        if (!isPositiveCompleteResponse(getServerReply())){
            disconnect();
            return false;
        }

        return true;
    }
```

Hoặc hàm giải phóng kết nối:

```
 public void disconnect()
     {
         if (outputStream != null) {
             try {
                    if (loggedIn) { logout(); };
                 outputStream.close();
                 inputStream.close();
                 connectionSocket.close();
             } catch (IOException e) {}

             outputStream = null;
             inputStream = null;
             connectionSocket = null;
         }
     }
```

- Khai báo các phương thức để thực hiện gửi các lệnh của FTP tới FTPServer:

Ví dụ:

❖ Phương thức thực hiện đăng nhập với lệnh USER và PASS

```
public boolean login(String username, String password)
        throws IOException
    {
        int response = executeCommand("user " + username);
```

```
            if (!isPositiveIntermediateResponse(response)) return
        false;
            response = executeCommand("pass " + password);
            loggedIn = isPositiveCompleteResponse(response);
            return loggedIn;
        }
```

Trong đó phương thức *executeCommand()* để thực thi một lệnh FTP bất kỳ:

```
    public int executeCommand(String command)
            throws IOException
        {
            outputStream.println(command);
            return getServerReply();
        }
```

- **Phương thức đọc/ghi dữ liệu:**

```
public boolean readDataToFile(String command, String fileName)
        throws IOException
    {
        // Open the local file
        RandomAccessFile outfile = new RandomAccessFile(fileName, "rw");
        // Do restart if desired
        if (restartPoint != 0) {
            debugPrint("Seeking to " + restartPoint);
            outfile.seek(restartPoint);
        }
        // Convert the RandomAccessFile to an OutputStream
        FileOutputStream fileStream = new
FileOutputStream(outfile.getFD());
        boolean success = executeDataCommand(command, fileStream);
        outfile.close();
        return success;
    }
}
public boolean writeDataFromFile(String command, String fileName)
        throws IOException
    {
        // Open the local file
        RandomAccessFile infile = new RandomAccessFile(fileName, "r");
        // Do restart if desired
        if (restartPoint != 0) {
            debugPrint("Seeking to " + restartPoint);
            infile.seek(restartPoint);
        }
        // Convert the RandomAccessFile to an InputStream
        FileInputStream fileStream = new FileInputStream(infile.getFD());
        boolean success = executeDataCommand(command, fileStream);
        infile.close();
        return success;
    }
```

- **Phương thức download và Upload tệp:**

```
   public boolean downloadFile(String fileName)
           throws IOException
       {
           return readDataToFile("retr " + fileName, fileName);
       }

public boolean downloadFile(String serverPath, String localPath)
        throws IOException
    {
        return readDataToFile("retr " + serverPath, localPath);
```

```
}
```

❖ **Một số phương thức thực hiện các lệnh FTP được liệt kê trong bảng sau:**

STT	Phương thức cài đặt	Lênh FTP
1	*public boolean changeDirectory(String directory) throws IOException*	**CD**
2	*public boolean renameFile(String oldName, String newName) throws IOException*	**RNFR, RNTO**
3	*public boolean removeDirectory(String directory) throws IOException*	**RMD**
4	*public boolean deleteFile(String fileName) throws IOException*	**DELE**
5	*public String getCurrentDirectory() throws IOException*	**PWD**

2.3. Chương trình ví dụ

Đoạn chương trình sau là ví dụ minh hoạ các phương thức đã cài đặt trên:

```
try {
    if (connection.connect(host)) {
         if (connection.login(username, password)) {
             connection.downloadFile(serverFileName);
              connection.uploadFile(localFileName);
          }
          connection.disconnect();
      }
  } catch (UnknownHostException e) {
      // handle unknown host
  } catch (IOException e) {
      // handle I/O exception
  }
```

IV. LẬP TRÌNH GỬI/NHẬN THƯ VỚI GIAO THỨC SMTP và POP3

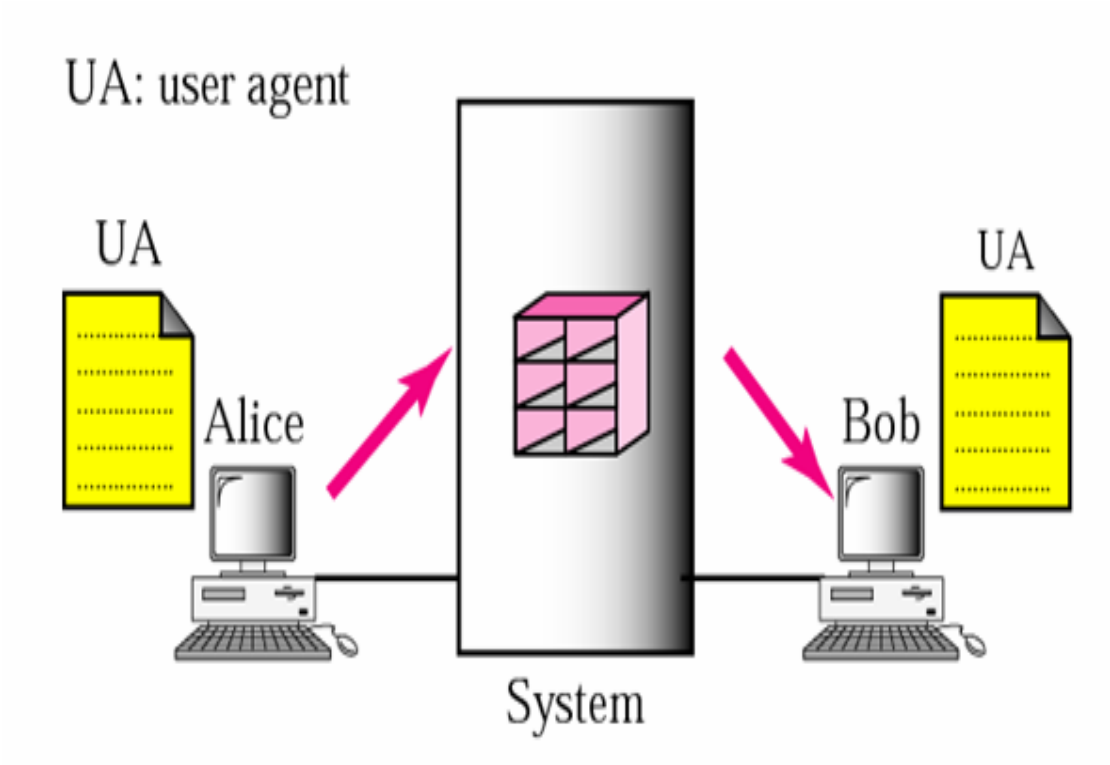
1. Giao thức SMTP

1.1 Giới thiệu

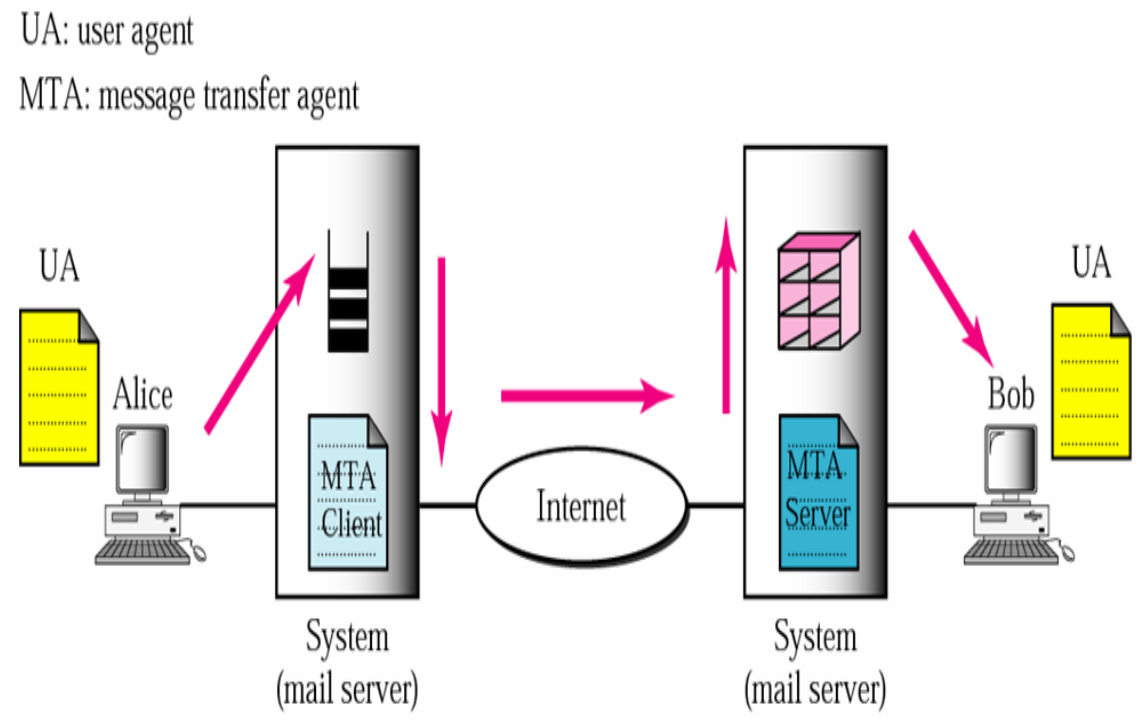
Mục đích của giao thức SMTP là truyền mail một cách tin cậy và hiệu quả. Giao thức SMTP không phụ thuộc vào bất kỳ hệ thống đặc biệt nào và nó chỉ yêu cầu trật tự của dữ liệu truyền trên kênh đảm bảo tin cậy.

1.2 Mô hình của giao thức SMTP

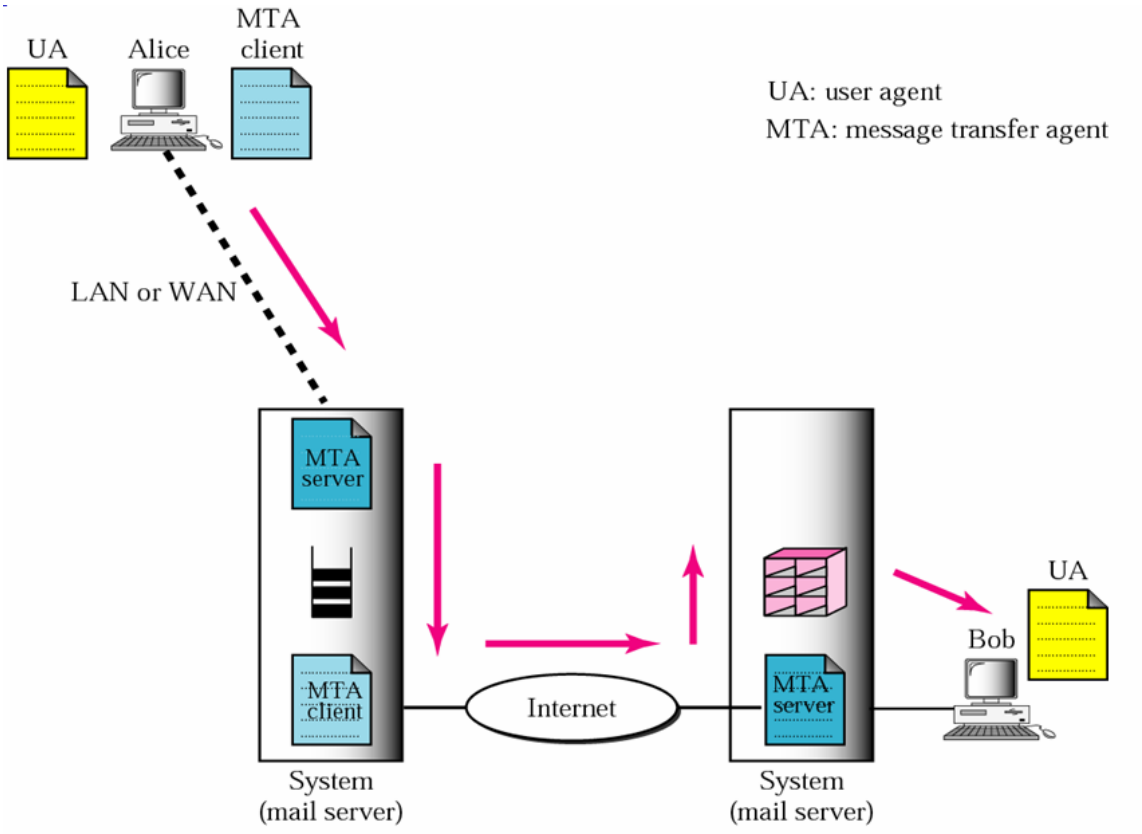
Giao thức SMTP được thiết kế dựa vào mô hình giao tiếp sau: khi có yêu cầu từ user về dịch vụ mail, bên gửi Sender-SMTP thiết lập một kênh truyền hai chiều tới bên nhận Receiver-SMTP và Receiver-SMTP gởi đáp ứng trở lại cho Sender-SMTP



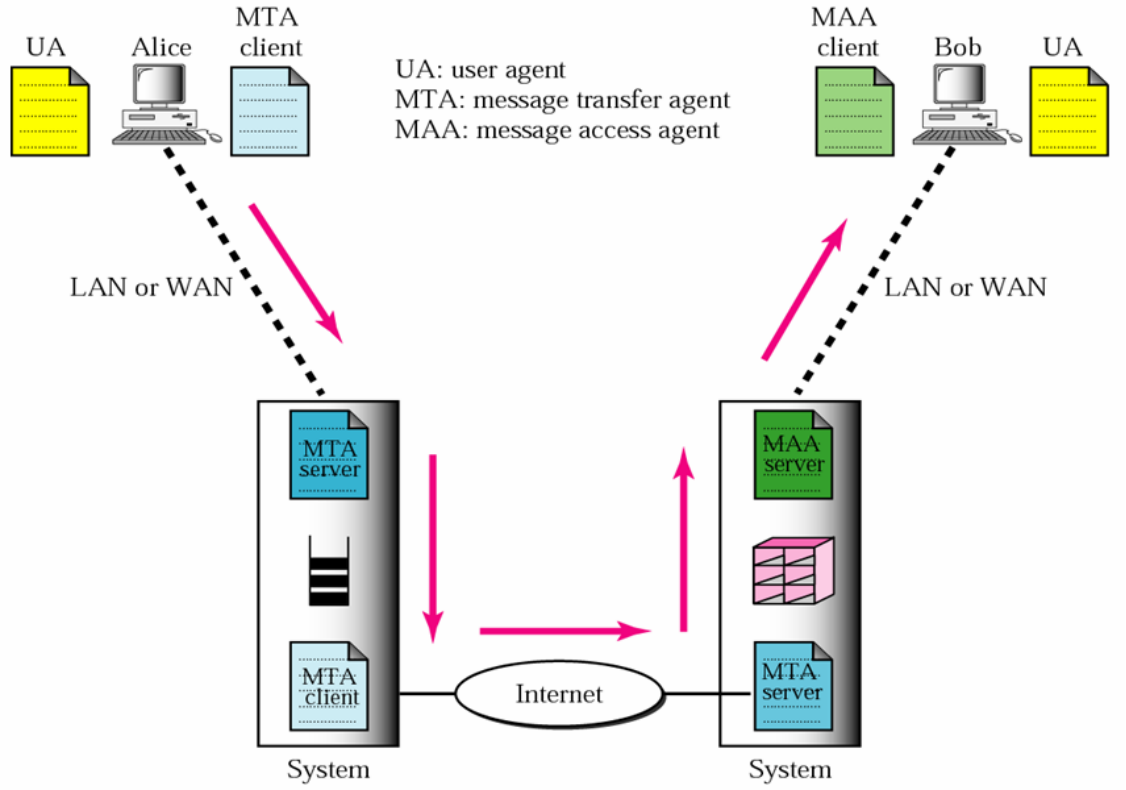
Hình 4.6. . Mô hình người gửi và nhận trên cùng hệ thống



Hình 4.7.Mô hình gửi thư qua hệ thống khác nhau



Hình 4.8.Mô hình gửi thư cả 2 phía qua mạng LAN/WAN



Hình 4.9. Mô hình người gửi/nhận kết nối mail server qua LAN/WAN

1.3. Tập lệnh và đáp ứng của SMTP

Những lệnh SMTP định nghĩa sự truyền mail hay chức năng của hệ thống mail được yêu cầu bởi user. Những lệnh SMTP là những chuỗi ký tự kết thúc bằng <CRLF>. Bản thân mã lệnh là những ký tự chữ (alphabetic) kết thúc bởi <SP> nếu có những tham số theo sau và nếu không có thì <CRLF>. Cú pháp của những mailbox phải tuân theo những quy ước của receiver.

Một phiên giao dịch mail chứa đựng một vài đối tượng dữ liệu, được truyền như là những đối số cho các lệnh khác nhau. Receiver-path là đối số của lệnh MAIL. Forward-path là đối số của những lệnh RCPT. Và mail data là đối số của lệnh DATA. Nhưng đối số hay những đối tượng dữ liệu này được truyền đi và duy trì cho đến khi truyền xong bởi sự chỉ định kết thúc của mail data. Mô hình hiện thực cho cách làm này là những buffer riêng biệt được cung cấp để lưu trữ kiểu của đối tượng dữ liệu, đó là các buffer: reverse-path, forward-path, và mail data buffer. Nhưng lệnh xác định tạo ra thông tin được gắn vào một buffer xác định , hoặc xóa bớt đi một hay một số buffer nào đó

Keyword	*Argument(s)*
HELO	Sender's host name
MAIL FROM	Sender of the message
RCPT TO	Intended recipient of the message
DATA	Body of the mail
QUIT	
RSET	
VRFY	Name of recipient to be verified
NOOP	
TURN	
EXPN	Mailing list to be expanded
HELP	Command name
SEND FROM	Intended recipient of the message
SMOL FROM	Intended recipient of the message
SMAL FROM	Intended recipient of the message

Còn các đáp ứng của SMTP tương tự gần giống như của FTP nhưng giá trị của x chỉ lấy từ 2 đến 5.

1.4. Cài đặt chương trình gửi thư với SMTP

Để gửi thư, chương trình ứng dụng phải thực hiện các thao tác cơ bản sau đây:

- Đầu tiên phải tạo đối tượng socket và kết nối tới mail server bằng cách chỉ ra tên miền hoặc địa chỉ IP của máy chủ mail server và sử dụng số cổng mặc định 25.
- Khai báo tạo luồng nhập xuất cho socket
- Thực hiện lần lượt gửi các lệnh và tham số của SMTP tới mail server theo trận tự sau:
  - HELLO
  - MAIL FROM
  - RCPT TO
  - DATA
  - QUIT

Sau mỗi lệnh gửi, phải thực hiện đọc các đáp ứng trả về.

Ví dụ về một giao dịch gửi thư của SMTP:

```
R: 220 BBN-UNIX.ARPA Simple Mail Transfer Service Ready.
S: HELO USC-ISIF.ARPA
R: 250 BBN-UNIX.ARPA
S: MAIL FROM:<Smith@USC-ISIF.ARPA>
R: 250 OK
S: RCPT TO:<Green@BBN-UNIX.ARPA>
R: 250 OK
S: DATA
R: 354 Start mail input; end with <CRLF>.<CRLF>
S: ...
S: ...
S: ...
...
R: 250 OK
S: QUIT
R: 221 BBN-UNIX.ARPA Service closing transmission channel.
```

Sau đây là mã cài đặt của chương trình ví dụ:

```
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.Socket;
import java.util.StringTokenizer;

public class SendMail
{
Object mailLock             = null;  //In case we want a multi-threaded
mailer
public String mailServerHost = "";
public String from           = "";
public String to             = "";
public String replyTo        = "";
public String subject        = "Java is Fun";
public String mailData       =
   "HyperSendMail";
public String errorMsg = "";
public Socket mailSendSock = null;
public  BufferedReader inputStream = null;
public PrintStream outputStream    =  null;
public String serverReply          = "";
SendMail()
{
   // Doesn't do anything but we need this for extension purposes.
}

// Server, from,to,subject, data

SendMail(String server,String tFrom,String tTo,String sub,String
sendData)
{
   mailServerHost = server;
   mailLock=this; //    from = tFrom;
   to   = tTo;
   if(sendData != null)
      mailData = sendData;
}

SendMail()
{
   if(mailLock != null)
   {
      if(mailLock instanceof Applet)
      {
         Applet app = (Applet)
      }
   }
}
*/

public void send()
{
   if(!open())          //Yikes! get out of here.
      return;
   try
   {
      outputStream.println("HELO sendMail");
      serverReply = inputStream.readLine();
   }
   catch(Exception e0)
   {
```

```
            e0.printStackTrace();
        }
        try
        {
            outputStream.println("MAIL FROM: "+from);
            serverReply = inputStream.readLine();
            if(serverReply.startsWith("5"))
            {
                close("FROM: Server error :"+serverReply);
                return;
            }

            if(replyTo == null)
                replyTo = from;
            outputStream.println("RCPT TO: <"+to+">");
            serverReply = inputStream.readLine();
            if(serverReply.startsWith("5"))
            {
                close("Reply error:"+serverReply);
                return;
            }
            outputStream.println("DATA");
            serverReply = inputStream.readLine();
            if(serverReply.startsWith("5"))
            {
                close("DATA Server error : "+serverReply);
                return;
            }
            outputStream.println("From: "+from);
            outputStream.println("To: "+to);
            if(subject != null)
                outputStream.println("Subject: "+subject);
            if(replyTo != null)
                outputStream.println("Reply-to: "+replyTo);
            outputStream.println("");
            outputStream.println(mailData);
            outputStream.print("\r\n.\r\n");
            outputStream.flush();
            serverReply = inputStream.readLine();
            if(serverReply.startsWith("5"))
            {
                close("DATA finish server error: "+serverReply);
                return;
            }
            outputStream.println("quit");
            serverReply = inputStream.readLine();
            if(serverReply.startsWith("5"))
            {
                close("Server error on QUIT: "+serverReply);
                return;
            }
            inputStream.close();
            outputStream.close();
            mailSendSock.close();
        }
        catch(Exception any)
        {
            any.printStackTrace();
            close("send() Exception");
        }
        close("Mail sent");
    }
```

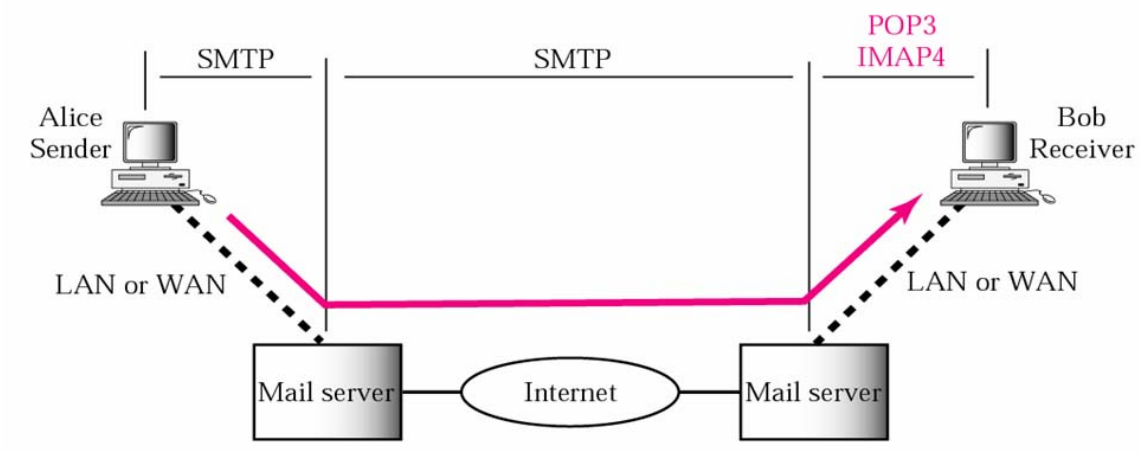
```
public boolean open()
{
   synchronized(mailLock)
   {
      try
      {
         mailSendSock = new Socket(mailServerHost, 25);
         outputStream = new PrintStream(mailSendSock.getOutputStream());
         inputStream = new BufferedReader(new InputStreamReader(
          mailSendSock.getInputStream()));
         serverReply = inputStream.readLine();
         if(serverReply.startsWith("4"))
         {
            errorMsg = "Server refused the connect message :
"+serverReply;
            return false;
         }
      }
      catch(Exception openError)
      {
         openError.printStackTrace();
         close("Mail Socket Error");
         return false;
      }
      System.out.println("Connected to "+mailServerHost);
      return true;
   }

}

public void close(String msg)
{
          //try to close the sockets
   System.out.println("Close("+msg+")");
   try
   {
      outputStream.println("quit");
      inputStream.close();
      outputStream.close();
      mailSendSock.close();
   }
   catch(Exception e)
   {
      System.out.println("Close() Exception");
     // We are closing so see ya later anyway
   }
}

// What do you know the damned thing works :)
/*
public static void main(String Args[])
{
SendMail sm = new
SendMail(
          "mail.hyperbyte.ab.ca",         //Mail Server
          "tswain@hyperbyte.ab.ca",       // sender
          "tswain@hyperbyte.ab.ca",       // Recipient
          "Java mail test",               // Subject
          "test test test!");             // Message Data
          sm.send();                      // Send it!
}
*/
```

```
/*
// Going a be an applet/thread safe version of readLine()
public void readLine(DataInputStream in,)
{
}
*/
}
```

2. Giao thức POP3

2.1. Giới thiệu

POP3 (Post Office Protocol Version 3) là một giao thức truy cập hộp thư. Nó gồm 2 phần mềm: POP3 Server cài trên máy chủ có chứa hộp thư; POP3 Client cài đặt trên máy cục bộ. Để truy cập được thư, người sử dụng dùng phần mềm truy cập hộp thư thiết lập kết nối tới POP3 Server tại số cổng mặc định là 110. POP3 server sẽ gửi trả về cho client một danh sách các mục thư chứa trong hộp thư người sử dụng. Giai đoạn sử dụng giao thức truy cập thư được thể hiện như hình vẽ.



2.2. Một số lệnh và đáp ứng của POP3

Một số lệnh quan trọng của POP3 được miêu tả sau đây. Còn các đáp ứng của POP3 tương tự như của giao thức FTP.

- **USER username:** đối số username là một chuỗi định danh mailbox, chỉ có ý nghĩa đối với server. Nó trả lời "+OK" nếu tên mailbox có hiệu lực và "-ERR" nếu không chấp nhận tên mailbox
- **PASS string:** đối số là một password cho mailbox hay server. Nó trả lời "+OK" đã khóa maildrop và sẵn sàng và "-ERR" nếu password không hiệu lực hoặc không được phép khóa maildrop.
- **QUIT:** Không có đối số và trả lời "+OK".

- **STAT:** không có đối số. Trả lời "+OK nn mm" với nn là số message, mm là kích thước maildrop tính bằng byte. Các message được đánh dấu xóa không được đếm theo tổng số.
- **LIST [msg]:** đối số là số thứ tự của message, có thể không liên quan tới các message đã được đánh dấu xóa. Trả lời "+OK scan listing flow" với scan listing là số thứ tự của message đó, theo sau là khoảng trống và kích thước chính xác của message đó tính theo byte; hoặc trả lời "-ERR no such message".
- **RETR msg:** đối số là số thứ tự message, có thể không liên quan tới các message đã được đánh dấu xóa. Trả lời "+OK message flows" hoặc "-ERR no such message".
- **DELE msg:** đối số là số thứ tự message, có thể không liên quan tới các message đã được đánh dấu xóa. Trả lời "+OK message deleted", POP3 sẽ đánh dấu xóa message này hoặc "-ERR no such message".
- **NOOP:** không có đối số và trả lời "+OK". POP3 server không làm gì hết, chỉ hồi âm lại cho client với trả lời "+OK".
- **RSET:** không có đối số, trả lời "+OK" để phục hồi lại các message đã bị đánh dấu xóa bởi POP3 server.

2.3. Các thao tác truy cập thư

Để thực hiện truy cập lấy thư, chương trình lấy thư phải thực hiện các thao tác cơ bản sau:

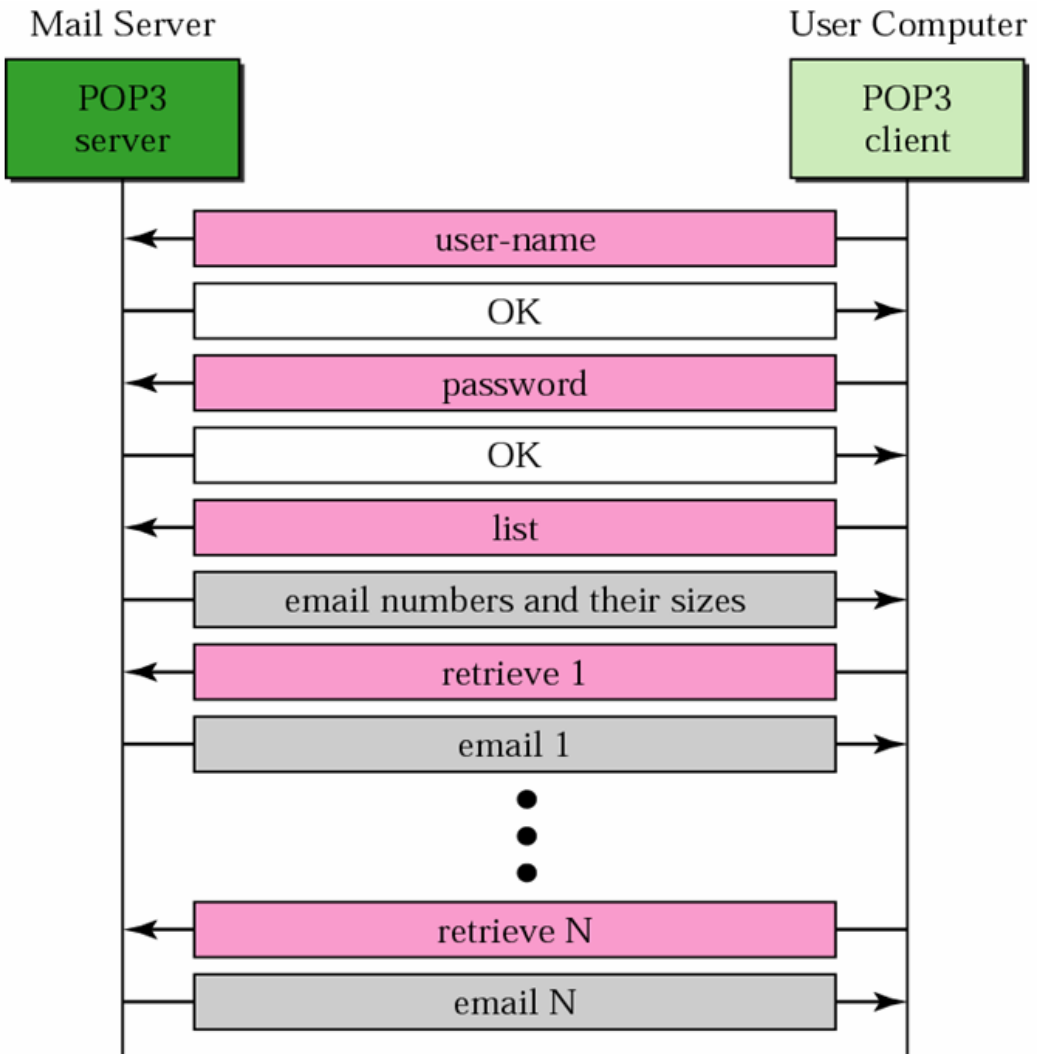
- Đăng nhập bằng lệnh USER, PASS với tài khoản hợp lệ
- Gửi lệnh thao tác với hộp thư.

Ví dụ quá trình thực hiện truy cập hộp thư lấy thư thể hiện như hình 4.10

2.4. Xây dựng chương trình truy cập hộp thư với giao thức POP3

Các thao tác cơ bản:

- Tạo đối tượng Socket và thiết lập với Mail Server tại số cổng 110.
- Tạo luồng nhập/xuất
- Thực hiện gửi lệnh tới mail server, sau mỗi lệnh gửi, nó thực hiện đọc đáp ứng trả về
- Kết thúc chương trình



Hình 4.10. Ví dụ quá trình lấy thư với giao thức POP3

Chương trình ví dụ sau minh hoạ cách cài đặt chương trình nhận thư với giao thức POP3.

```
//CheckMail.java
import java.net.*;
import java.io.*;
public class CheckMail {
public static void main(String s[]) {
   //  CheckMail [mailServer] [user] [password]
   try {
     CheckMail t = new CheckMail();
     int i = t.checkMyMail(s[0], s[1], s[2]);
     if (i==0) {
       System.out.println("No mail waiting.");
     }
     else {
       System.out.println
        ("There " + (i==1?"is " :"are ") + i +
         " message" +(i==1?"":"s")+ " waiting.");
     }
   }
   catch (Exception e) {
     e.printStackTrace();
   }
```

```
        }
        private void send(BufferedWriter out, String s) throws IOException {
           out.write(s+"\n");
           out.flush();
        }
        private String receive(BufferedReader in) throws IOException {
           return in.readLine();
        }
        private int checkMyMail
              (String server, String user, String pass) throws IOException {
           Socket s = new Socket(server, 110);
           BufferedReader in  = new BufferedReader(
               new InputStreamReader(s.getInputStream()));
           BufferedWriter out = new BufferedWriter(
               new OutputStreamWriter(s.getOutputStream()));
           receive(in);
           send(out, "USER " + user);
           receive(in);
           send(out, "PASS " + pass);
           receive(in);
           return getNumberOfMessages(in, out);
        }
        public int getNumberOfMessages
           (BufferedReader in, BufferedWriter out) throws IOException {
           int i = 0;
           String s;
           send(out, "LIST");
           receive(in);
           while((s = receive(in)) != null) {
              if (!(s.equals("."))) {
                i++;
              }
              else
                return i;
              }
           return 0;
        }
}
```

V. KẾT LUẬN

Như vậy trong chương này đã bước đầu cung cấp cho người lập trình cách lập trình với các giao thức truyền thông đã phát triển sẵn có thông qua kỹ thuật socket. Đây là chương quan trọng, nó vừa củng cố cho sinh viên kiến thức mạng, vừa trang bị cho sinh viên biết cách cài đặt các giao thức đó bằng một ngôn ngữ lập trình cụ thể. Trên cơ sở đó sinh viên có thể hoàn thiện một dịch vụ mạng hoàn chỉnh hoặc phát triển các modul chương trình để tích hợp vào các chương trình uúng dụng khác nhau. Ngoài các giao thức trên, sinh viên nên lập trình với một số giao thức Internet phổ biến khác như DNS, TFTP, HTTP, RTP hoặc cài đặt các giao thức, gói tin của các giao thức TCP, UDP, ICMP, ARP, IP, ICMP hoặc khảo sát phát triển các ứng dụng với họ giao thức Hxxx, SIP...Cuỗi cùng một điều nhấn mạnh với người học khi phát triển các ứng dụng mạng với các giao thức: Phải nắm chắc mô hình, cấu trúc, cơ chế truyền thông của các giao thức thì mới lập trình được. Một vấn đề khác, thông qua chương này người lập trình có thể phát triển các giao thức truyền thông riêng của mình để giải quyết bài toán cụ thể.