LỜI NÓI ĐẦU

Việc quản lý tiến trình và điều khiển máy tính từ xa, hay quản lý tiến trình từ một máy tính khác đã từng, đang và sẽ luôn là một nhu cầu cần thiết trong sử dụng máy tính hàng ngày cũng như công việc. Từ đó, những thuật toán, chương trình về quản lý, điều khiển máy tính từ xa ngày càng được hoàn thiện và phát triển.

Do đó, dựa trên những kiến thức tích lũy được từ các môn học và sự hướng dẫn của giáo viên, và những kiến thức được tra cứu trên Internet em đã xây dựng chương trình lấy thông tin từ tiến trình và điều khiển desktop trong mạng LAN dựa vào mô hình Client – Server. Bài báo cáo Đồ Án Cơ Sở Ngành Mạng này sẽ trình bày các cơ sở lý thuyết , thuật toán giải quyết vấn đề và các kết quả đạt được.

Em xin chân thành cảm ơn thầy PGS.TS. Huỳnh Công Pháp đã hướng dẫn hoàn thành môn đồ án này. Vì còn những thiếu sót trong các kỹ năng lập trình cũng như trình bày của bản thân nên bài báo cáo sẽ không tránh khỏi nhiều sai sót. Do đó, em mong các thầy cô sẽ đưa ra những đánh giá khách quan để em có thể rút kinh nghiệm về sau.

NHẬN XÉT

……..……..……

……..……

……..……

……..…

……..……

……..……

……..……

……..……

……..……

……..……

……..……

……..……

……..……

……..……

……..……

……..……

……..……

……..……

……..……

……..……

……..……

……..……

……..……

Phần Hệ điều hành

MỤC LỤC

CHƯƠNG I. CƠ SỞ LÝ THUYẾT

1.1 Giới thiệu về tiến trình

1.1.1 Khái niệm

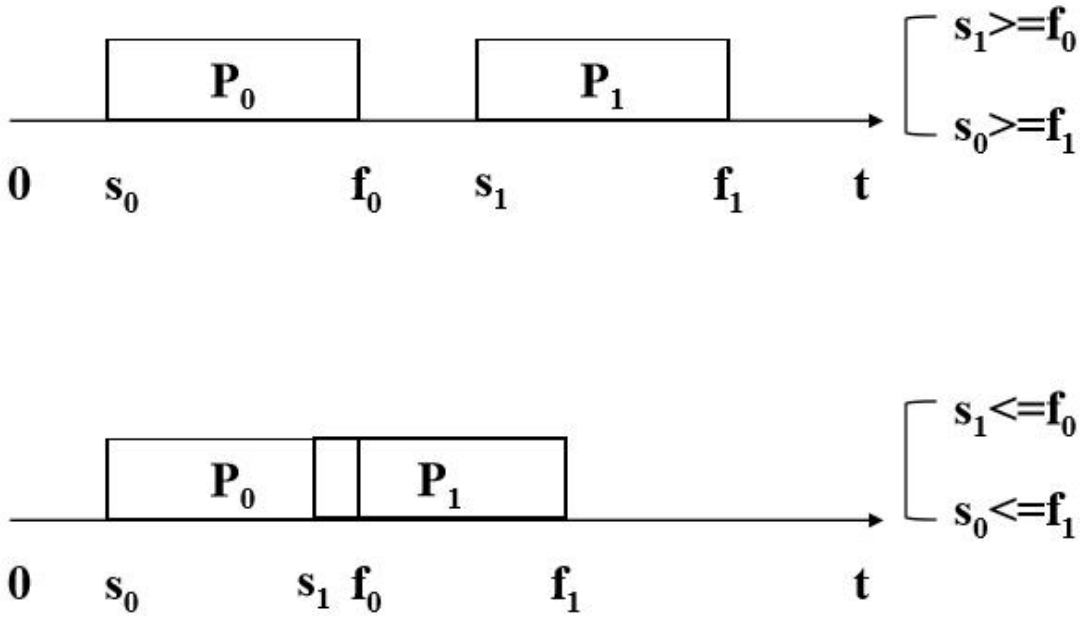
Trong khoa học máy tính, tiến trình là một thực thể điều khiển đoạn mã lệnh có riêng một không gian địa chỉ, có ngăn xếp (*stack*) riêng rẽ, có bảng chứa các số miêu tả file (*file descriptor*) được mở cùng tiến trình và đặc biệt là có một định danh PID (*process identifier*) duy nhất trong toàn bộ hệ thống vào thời điểm tiến trình đang chạy.

1.1.2 Phân loại

Có 2 loại tiến trình:

- Tiến trình kế tiếp: thời điểm bắt đầu của tiến trình này nằm sau thời điểm kết thúc của tiến trình kia

- Tiến trình song song: thời điểm bắt đầu của tiến trình này nằm trước thời điểm kết thúc của tiến trình kia



Hình 1: Hai loại tiến trình

1.1.3 Process Control Block

PCB: vùng nhớ lưu trữ các thông tin mô tả cho tiến trình như:

* Định danh của tiến trình: phân biệt giữa các tiến trình.

* Trạng thái tiến trình: hoạt động hiện hành của tiến trình.

* Ngữcảnh của tiến trình:

- Trạng thái CPU: nội dung các thanh ghi(IP). Lưu trữ nội dung thanh ghi khi xảy ra ngắt.

- Bộ xử lý: xác định sốhiệu CPU mà tiến trình đang sử dụng(máy có cấu hình nhiều CPU).

- Bộ nhớ chính: danh sách các vùng nhớ được cấp cho tiến trình.

- Tài nguyên sử dụng: danh sách các tài nguyên hệ thống màtiến trình đang sửdụng.

- Tài nguyên tạo lập: danh sách các tài nguyên được tiến trình tạo lập.

* Thôngtin giao tiếp:

- Tiến trình cha: tiến trình tạo lập tiến trình này

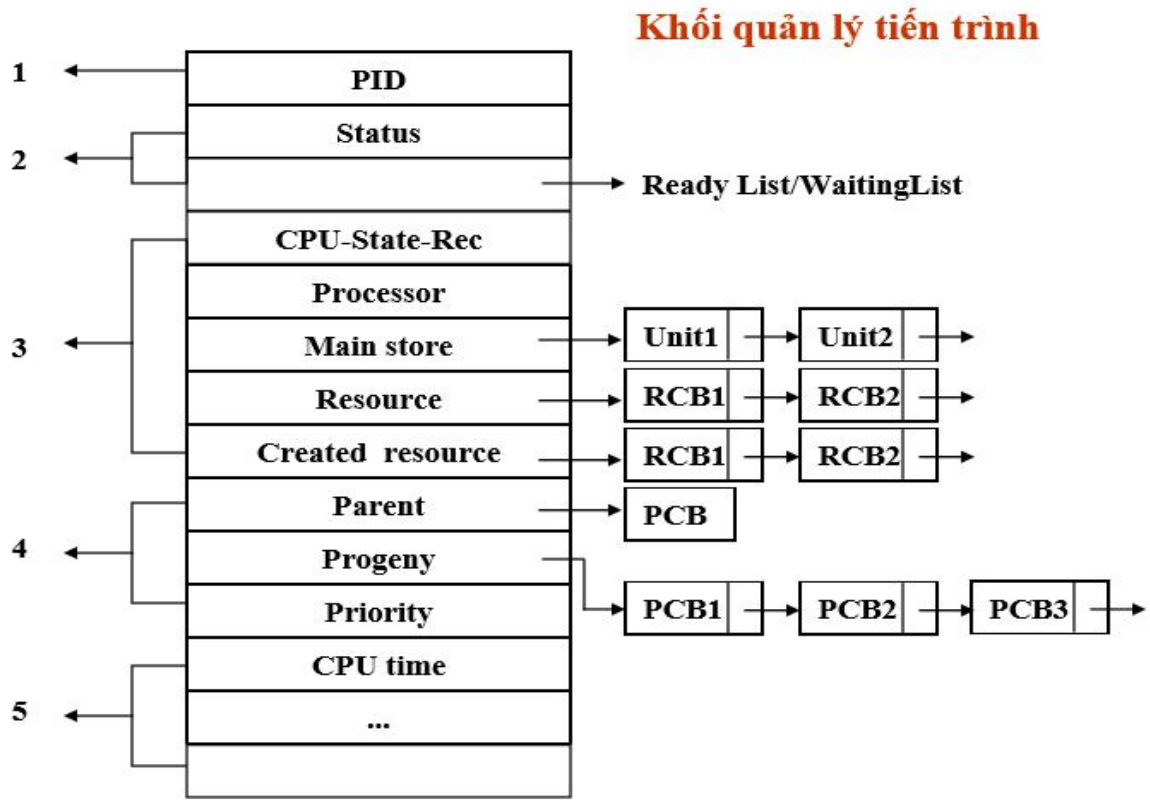
- Tiến trình con: các tiến trình do tiến trình này tạo ra

- Độ ưu tiên: thông tin giúp bộ điều phối lựa chọn tiến trình được cấp CPU

* Thông tin thống kê về hoạt động của tiến trình:

- Thời gian sửdụng CPU

- Thời gian chờ



Hình 2: Khối quản lý tiến trình

1.1.4 Mô hình trạng thái

Các trạng thái của tiến trình

•Mới tạo: tiến trình đang được tạo lập.

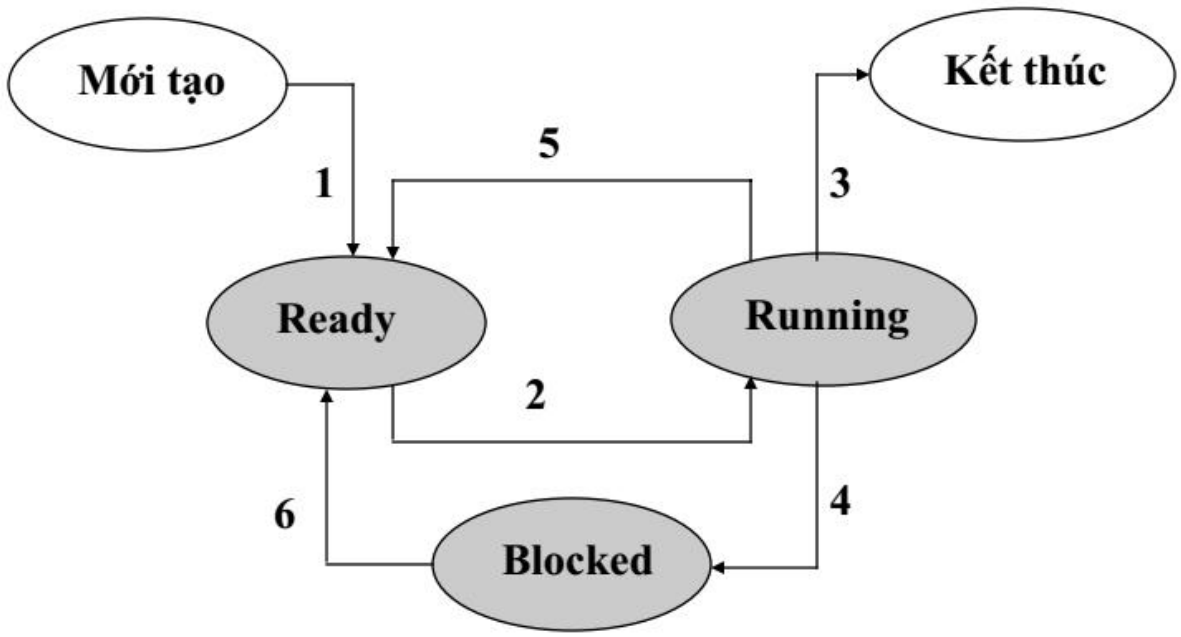
•Running: tiến trình đang được xử lý.

•Ready: tiến trình đang sẵn sàng,chờ cấp CPU để xử lý

•Blocked: tiến trình bị chặn,không thể tiếp tục.

•Kết thúc: tiến trình hoàn tất xử lý

❖ Sơ đồ chuyển trạng thái của tiến trình



Hình 3: Khối quản lý tiến trình

1.2 Điều phối tiến trình

1.2.1 Mục tiêu điều phối

- Sự công bằng giữa các tiến trình

- Tính hiệu quả (tậndụng100% thời gian sử dụng CPU)

- Cực tiểu hoáthời gian lưu lại trong hệthống

- Thời gian đáp ứng hợp lý (cựcti ểuhoáthờigian hồiđápchocáctươngtáccủaNSD)

- Thông lượng tối đa (cực đại hoá số công việc được xử lý trong một thời gian cố định)

1.2.2 Điều phối độc quyền

- Tiến trình khi nhận được CPU thì có độc quyền sử dụng cho đến khi tiến trình hoàn tất hay tự nguyện giải phóng CPU

- Quyết định điều phối CPU xảy ra khi:

+ Tiến trình chuyển từtrạng thái Running sang Blocked

+ Tiến trình kết thúc

- Giải thuật đơn giản, dễcài đặt nhưng ngăn cản các tiến trình còn lại trong hệ thống có cơ hội để xử lý

1.2.3 Điều phối không độc quyền

- Tiến trình có thể bị tạm dừng hoạt động bất cứ lúc nào mà không được báo trước, để tiến trình khác xử lý. (khi có một tiến trình khác có độ ưu tiên cao hơn về quyền dành sử dụng CPU)

- Quyết định điều phối CPU xảy ra khi:

+ Tiến trình chuyển từ trạng thái Running sang Blocked

+Tiến trình chuyển từ trạng thái Running sang Ready.

+ Tiến trình chuyển từ trạng thái blocked sang Ready

+ Tiến trình kết thúc

- Ngăn cản được tình trạng các tiến trình độc chiếm CPU, nhưng việc tam dừng một tiến trình dẫn đến các mâu thuẫn trong truy xuất. Đòi hỏi phương pháp đồng bộ hoá thích hợp

1.2.4 Đồng hồ ngắt thời gian

-Bộ đếm thời gian qui định một thông số thời gian tthích hợp ứng với một lượt cấp CPU cho một tiến trình

-Sau một khoảng thời gian t sẽ xảy ra một ngắt báo hiệu hết thời gian sử dụng CPU của tiến trình hiện hành. HĐH sẽ thu hồi CPU và bộ điều phối sẽ quyết định tiến trình nào sẽ được cấp phát

1.2.5 Độ ưu tiên của tiến trình

-Độ ưu tiên của tiến trình: giá trị giúp phân định tầm quan trọng của các tiến trình

-Độ ưu tiên tĩnh:

+ Được gán sẵn cho tiến trình khi mới được ta ra

+ Không thay đổi

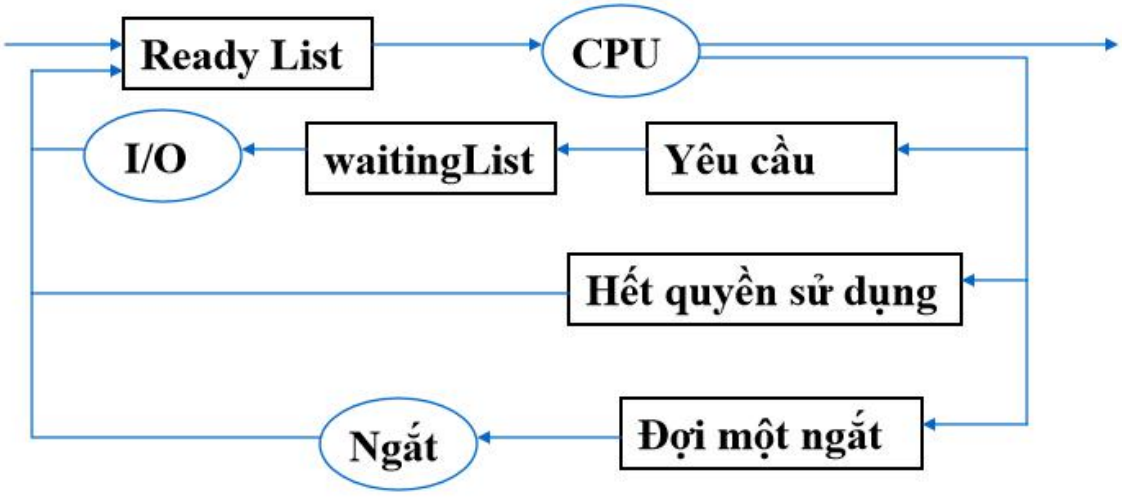
-Độ ưu tiên động: thay đổi theo thời gian và môi trường xử lý của tiến trình

1.2.6 Tổ chức điều phối

- Danh sách sẵn sàng (Ready List)

- Danh sách chờ đợi (Waiting List)

- Các danh sách chờ đợi riêng cho từng tài nguyên (thiết bị ngoại vi)



Sơ đồ chuyển đổi giữa các danh sách điều phối

Hình 4: Tổ chức điều phối

1.3 Đồng bộ hoá tiến trình

Sử dụng biến khoá

- Giải thuật sử dụng biến khoá để đồng bộ

```
while (1) { while (lock==1);// wait lock=1;
critical_section();
lock=0;
Noncritical_section(); }
```

- Giải thuật sử dụng biến khoá để đồng bộ

```
while (1) { while (lock==1);// wait lock=1;
critical_section();
lock=0;
non_critical_section(); }
```

*Kiểm tra luân phiên

- Các tiến trình muốn đi vào miền găng thì được gắn nhãn 0|1
- Sử dụng biến turn để chỉ thứ tự luân phiên.
- Nếu turn==0: tiến trình có nhãn 0 được vào miền găng
- Nếu turn==1: tiến trình có nhãn 1 được vào miền găng

*Giải pháp Sleep and Wakeup

- Sử dụng 2 thủ tục: sleep và wakeup

- Khi tiến trình chưa đủ điều kiện để vào miền găng, nó gọi sleep để tự khoá đến khi một tiến trình khác gọi wakeup để đánh thức nó.

- Tiến trình khi ra khỏi miền găng sẽ gọi wakeup để đánh thức tiến trình khác.

- int busy;// 1: nếu miền găng đang bận, 0: không bận

- int blocked;//đếm số lượng tiến trình đang bị khoá

Giải thuật:

```
while (1) {
  if (busy) {
    blocked=blocked+1;
    sleep();
  }
  else busy=1;
  critical_section();
```

```
  busy=0;
  if (blocked) {
    wakeup(process);
    blocked=blocked-1;
  }
  noncritical_section();
}
```

Hình 5: Giải thuật sleep & wakeup

1.4 Xác định trạng thái an toàn

Sử dụng các cấu trúc dữ liệu sau:

Int allocation[numprocs,numresources]; //allocation[p,r] số lượng tài nguyên r thực sự cấp phát cho p

Int max[numprocs,numresources]; // max[p,r] nhu cầu tối đa của tiến trình p về tài nguyên r

Int need[numprocs,numresources]; //need[p,r]=max[p,r]-allocation[p,r]

int available[numresources] //available[r]số lượng tài nguyên r còn tự do

Int word[numresouces]=available;

Int finish[numproces]=false;

1.Tìm i sao cho

a. finish[i]==false;

b. need[i,j]<=word[j];với mọi tài nguyên j nếu không có i như thế, đến bước 3
2.Word[j]=word[j]+allocation[i,j];

finish[i]=true;

đến bước 1;

3.Nếu finish[i]==true với mọi i thì hệthống ở trạng thái an toàn. Ngược lại hệ thống bị tắc nghẽn

CHƯƠNG II. PHÂN TÍCH VÀ THIẾT KẾ

2.1 Khái niệm và ký hiệu dùng

* Lấy môi trường Runtime của hệ thống

```
runTime = Runtime.getRuntime();
```

* Tạo luồng vào để đọc tiến trình

```
InputStream inputStream = p.getInputStream();
InputStreamReader inputStreamReader = new InputStreamReader(inputStream);
BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
```

* Đọc tiến trình hệ thống và thêm vào dấu & như là dấu cách trong đầu ra của dữ liệu

```
String line = bufferedReader.readLine();
process = "&";
while (line != null) {
line = bufferedReader.readLine();
process += line + "&";
```

* Trỏ đường dẫn để lưu trữ file được ghi

```
OutputStreamWriter outputStreamWriter = new OutputStreamWriter(new
FileOutputStream("ProcessList.txt"));
BufferedWriter bufferedWriter = new BufferedWriter(outputStreamWriter);
```

2.2 Môi trường và ngôn ngữ lập trình

Chương trình được viết trên phần mêm Java với môi trường JDK 8.131

CHƯƠNG III. TRIỂN KHAI VÀ ĐÁNH GIÁ

4.1 Cấu trúc chương trình

Chương trình chỉ có một file duy nhất GetProcessList.java

4.2 Chạy thử Chương Trình

3.2.1 Dữ liệu vào

Đầu vào là các thông số tiến trình đang chạy trên máy tính

3.2.2 Quá trình tính toán

Chương trình tự động lấy thông tin tiến trình và ghi vào file ProcessList.txt

3.2.3 Dữ liệu ra

ProcessList - Notepad

File Edit Format View Help

```
Image Name                     PID Session Name        Session#    Mem Usage
========================= ======== ================ =========== ============
System Idle Process              0 Services                   0          4 K
System                           4 Services                   0     24,332 K
smss.exe                       380 Services                   0        512 K
csrss.exe                      496 Services                   0      2,192 K
wininit.exe                    604 Services                   0      3,368 K
services.exe                   780 Services                   0      5,192 K
lsass.exe                      788 Services                   0     10,640 K
svchost.exe                    888 Services                   0     16,100 K
svchost.exe                    968 Services                   0     12,600 K
svchost.exe                    744 Services                   0    102,568 K
svchost.exe                    776 Services                   0     55,152 K
svchost.exe                    868 Services                   0     17,208 K
WUDFHost.exe                  1172 Services                   0      2,256 K
nvSCPAPISvr.exe               1280 Services                   0      2,264 K
nvwmi64.exe                   1288 Services                   0      1,676 K
nvvsvc.exe                    1296 Services                   0      5,404 K
svchost.exe                   1608 Services                   0     24,536 K
svchost.exe                   1616 Services                   0     20,312 K
svchost.exe                   1624 Services                   0     23,332 K
svchost.exe                   1636 Services                   0     11,148 K
svchost.exe                   2008 Services                   0      8,176 K
svchost.exe                   1480 Services                   0      4,660 K
svchost.exe                   1424 Services                   0      3,780 K
spoolsv.exe                   2204 Services                   0      4,768 K
svchost.exe                   2484 Services                   0     17,956 K
dasHost.exe                   2524 Services                   0      9,716 K
HidMonitorSvc.exe             2608 Services                   0      2,324 K
svchost.exe                   2668 Services                   0     14,500 K
MsMpEng.exe                   2676 Services                   0    140,312 K
sqlwriter.exe                 2884 Services                   0      2,196 K
Memory Compression            3004 Services                   0    149,512 K
sqlservr.exe                  3172 Services                   0     23,256 K
sqlservr.exe                  3216 Services                   0     24,460 K
NisSrv.exe                    4720 Services                   0      8,980 K
SearchIndexer.exe             1900 Services                   0     40,856 K
csrss.exe                    12548 Console                   52      4,568 K
winlogon.exe                  4844 Console                   52     10,768 K
dwm.exe                       6968 Console                   52     30,996 K
nvxdsync.exe                  9384 Console                   52     19,132 K
nvvsvc.exe                    1008 Console                   52     12,628 K
nvwmi64.exe                   5520 Console                   52     11,836 K
Apoint.exe                    4920 Console                   52     13,684 K
sihost.exe                    4512 Console                   52     19,032 K
svchost.exe                   5096 Console                   52     28,180 K
taskhostw.exe                 9640 Console                   52     16,748 K
ggdllhost.exe                 7176 Console                   52      1,156 K
RuntimeBroker.exe            10264 Console                   52     23,776 K
explorer.exe                  7720 Console                   52    118,008 K
ShellExperienceHost.exe       2748 Console                   52     60,780 K
SearchUI.exe                  7976 Console                   52     81,924 K
ApMsgFwd.exe                  5956 Console                   52      6,040 K
hidfind.exe                   8536 Console                   52      4,936 K
```

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết quả đạt được

Đồ án đã tìm hiểu được các khái niệm của tiến trình nói chung.

Đồng thời, trong đồ án cũng đã xây dựng thành công chương trình demo đọc và hiển thị thông tin của các tiến trình hiện hoạt động của Window

Bên cạnh những mặc đã đạt được ở trên, trong quá trình làm đồ án em đã học được thêm rất nhiều kiến thức bổ ích, những kinh nghiệm quý giá về lý thuyết tiến trình, xử lý…, cũng như về ngôn ngữ để thực hiện mà cụ thể đó là ngôn ngữ lập trình Java với sự hỗ trợ mạnh vẽ của môi trường phát triển JDK.

2. Những vấn đề tồn tại

Bên cạnh một số vấn đề mà em đã đạt được, thì vẫn còn tồn tại một số vấn đề mà chúng em gặp khó khăn và chưa giải quyết được tiêu biểu như:

- Chưa xây dựng được giao diện trược quan.

- Chưa tác động trực tiếp lên các tiến trình được (như tắt, bật..)

…

Và một số vấn đề tồn tại khác trong quá trình xây dựng, tìm hiểu và trình bày báo cáo, xây dựng chương trình mà em chưa thể hoàn chỉnh được.

3. Hướng phát triển

Về phần tìm hiểu và cơ sở lý thuyết, em sẽ tiếp tục tìm hiểu và nghiên cứu sâu hơn đề tài để phục vụ tốt cho những môn học sau này.

Về phần chương trình, sẽ tiếp tục xây dựng, khắc phục và hoàn thiện một số vấn đề bất cập, bổ sung thên một số tính năng như: xây dựng giao diện làm việc với tiến trình tương tự như TaskManage… Và nếu có thời gian chúng em sẽ hoàn chỉnh thêm một số chức năng, thông tin cần thiết như: tác động trực tiếp lên tiến trình, như ngưng, hoặc tắt hẳn tiến trình…

Cuối cùng chúng em xin chân thành cám ơn sự hướng dẫn chu đáo, nhiệt tình của thầy PGS.TS. Huỳnh Công Pháp đã hướng dẫn và nhiệt tình giúp đỡ cho em trong quá trình làm đồ án để em hoàn thành đồ án một cách tốt nhất.

TÀI LIỆU THAM KHẢO

[1] *Giáo trình Cấu Trúc Máy Tính.*

[2] Đặng Vũ Tùng, *Giáo trình Nguyên lý hệ điều hành* - Đặng Vũ Tùng.

[3] Trần Hồ Thủy Tiên, *Bài giảng Nguyên lý hệ điều hành*, khoa CNTT, trường Đại học Bách khoa, Đại học Đà Nẵng 2015.

[4] *Principles Of Computer Architecture* - Class test edition-Augst 1999.

[5] http://www.ntfs.com

[6] http://vi.wikipedia.org/wiki/Microsoft_Windows

[7] http://vi.wikipedia.org/wiki/NTFS

PHỤ LỤC

GetProcessList.java

```
package OS;

/**
 * Created by tungv on 5/12/2017.
 */

import java.io.*;
import java.util.StringTokenizer;


public class GetProcessList
{

    private String GetProcessListData()
    {
        Process p;
        Runtime runTime;
        String process = null;
        try {
            System.out.println("Processes Reading is started...");

            //Get Runtime environment of System
            runTime = Runtime.getRuntime();

            //Execute command throws Runtime
            p = runTime.exec("tasklist");       // For Windows
            //p=r.exec("ps ux");                //For Linux

            //Create Inputstream for Read Processes
            InputStream inputStream = p.getInputStream();
            InputStreamReader inputStreamReader = new
InputStreamReader(inputStream);
            BufferedReader bufferedReader = new BufferedReader(inputStreamReader);

            //Read the processes from system and add & as delimiter for tokenize
the output
            String line = bufferedReader.readLine();
            process = "&";
            while (line != null) {
                line = bufferedReader.readLine();
                process += line + "&";
            }

            //Close the Streams
            bufferedReader.close();
            inputStreamReader.close();
            inputStream.close();

            System.out.println("Processes are read.");
        } catch (IOException e) {
            System.out.println("Exception arise during the read Processes");
```

```
            e.printStackTrace();
        }
        return process;
    }

    private void showProcessData()
    {
        try {

            //Call the method For Read the process
            String proc = GetProcessListData();

            //Create Streams for write processes
            //Given the filepath which you need.Its store the file at where your
java file.
            OutputStreamWriter outputStreamWriter = new OutputStreamWriter(new
FileOutputStream("ProcessList.txt"));
            BufferedWriter bufferedWriter = new BufferedWriter(outputStreamWriter);

            //Tokenize the output for write the processes
            StringTokenizer st = new StringTokenizer(proc, "&");

            while (st.hasMoreTokens()) {
                bufferedWriter.write(st.nextToken());  //Write the data in file
                bufferedWriter.newLine();               //Allocate new line for
next line
            }

            //Close the outputStreams
            bufferedWriter.close();
            outputStreamWriter.close();

        } catch (IOException ioe) {
            ioe.printStackTrace();
        }

    }

    public static void main(String[] args)
    {
        GetProcessList gpl = new GetProcessList();
        gpl.showProcessData();

    }
}
```

Phần Mạng

MỤC LỤC

CHƯƠNG I - CƠ SỞ LÝ THUYẾT

1.1. Giới thiệu giao thức TCP/IP

Giao thức TCP/IP được phát triển từ mạng ARPANET và Internet và được dùng như giao thức mạng và vận chuyển trên mạng Internet. TCP (Transmission Control Protocol) là giao thức thuộc tầng vận chuyển và IP (Internet Protocol) là giao thức thuộc tầng mạng của mô hình OSI. Họ giao thức TCP/IP hiện nay là giao thức được sử dụng rộng rãi nhất để liên kết các máy tính và các mạng.

1.1.1. Giao thức IP

Nhiệm vụ chính của giao thức IP là cung cấp khả năng kết nối các mạng con thành liên kết mạng để truyền dữ liệu, vai trò của IP là vai trò của giao thức tầng mạng trong mô hình OSI. Giao thức IP là một giao thức kiểu không liên kết (connectionlees) có nghĩa là không cần có giai đoạn thiết lập liên kết trước khi truyền dữ liệu.

Sơ đồ địa chỉ hóa để định danh các trạm (host) trong liên mạng được gọi là địa chỉ IP 32 bits (32 bit IP address). Mỗi giao diện trong 1 máy có hỗ trợ giao thức IP đều phải được gán 1 địa chỉ IP (một máy tính có thể gắn với nhiều mạng do vậy có thể có nhiều địa chỉ IP). Mục đích của địa chỉ IP là để định danh duy nhất cho một máy tính bất kỳ trên liên mạng.

1.1.2. Giao thức TCP

TCP là viết tắt của *Transmission Control Protocol*, cho phép sự giao tiếp đáng tin cậy giữa hai ứng dụng. TCP đặc trưng được sử dụng qua Internet Protocol, được xem như là TCP/IP. TCP là một giao thức "có liên kết" (connection - oriented), nghĩa là cần phải thiết lập liên kết giữa hai thực thể TCP trước khi chúng trao đổi dữ liệu với nhau. Một tiến trình ứng dụng trong một máy tính truy nhập vào các dịch vụ của giao thức TCP thông qua một cổng (port) của TCP. Số hiệu cổng TCP được thể hiện bởi 2 bytes.

Một cổng TCP kết hợp với địa chỉ IP tạo thành một đầu nối TCP/IP (socket) duy nhất trong liên mạng. Dịch vụ TCP được cung cấp nhờ một liên kết logic giữa một

cặp đầu nối TCP/IP. Một đầu nối TCP/IP có thể tham gia nhiều liên kết với các đầu nối TCP/IP ở xa khác nhau. Trước khi truyền dữ liệu giữa 2 trạm cần phải thiết lập một liên kết TCP giữa chúng và khi không còn nhu cầu truyền dữ liệu thì liên kết đó sẽ được giải phóng.

Các thực thể của tầng trên sử dụng giao thức TCP thông qua các hàm gọi (function calls) trong đó có các hàm yêu cầu để yêu cầu, để trả lời. Trong mỗi hàm còn có các tham số dành cho việc trao đổi dữ liệu.

1.2. Socket và cổng

Các Socket cung cấp kỹ thuật giao tiếp giữa hai máy tính sử dụng TCP. Một chương trình Client tạo một socket trên đầu cuối của giao tiếp và cố gắng để kết nối socket đó tới một Server.

Khi kết nối được tạo, Server tạo một đối tượng Socket trên đầu cuối của giao tiếp. Client và Server bây giờ có thể giao tiếp bằng việc đọc và ghi từ Socket.

Lớp java.net.Socket biểu diễn một Socket, và lớp java.net.ServerSocket cung cấp một kỹ thuật cho chương trình Server để nghe thông tin từ các Client và thành lập các kết nối với chúng.

Các bước sau xảy ra khi thành lập một kết nối TCP giữa hai máy tính sử dụng Socket:

- Server khởi tạo một đối tượng ServerSocket, biểu thị số hiệu cổng (port) nào để xuất hiện giao tiếp.
- Server gọi phương thức accept() của lớp ServerSocket. Phương thức này đợi tới khi một Client kết nối với Server trên cổng đã cho.
- Sau khi Server đang đợi, một Client khởi tạo một đối tượng Socket, xác định tên Server và số hiệu cổng để kết nối tới.
- Constructor của lớp Socket cố gắng để kết nối Client tới Server và số hiệu cổng đã xác định. Nếu giao tiếp được thành lập, bây giờ Client có một đối tượng Socket có khả năng giao tiếp với Server.

- Trên Server-side, phương thức accept() trả về một tham chiếu tới một socket mới trên Server mà được kết nối với socket của Client.

Sau khi các kết nối được thành lập, giao tiếp có thể xảy ra bởi sử dụng I/O stream. Mỗi Socket có cả một OutputStream và InputStream. OutputStream của Client được kết nối với InputStream của Server, và InputStream của Client được kết nối với OutputStream của Server.

TCP là một giao thức giao tiếp hai chiều, vì thế dữ liệu có thể được gửi qua cả hai luồng tại cùng một thời điểm. Các lớp hữu ích sau đây cung cấp đầy đủ các phương thức để triển khai các Socket.

1.3. Mô hình Client / Server

Đây là mô hình phổ biến nhất và được chấp nhận rộng rãi trong các hệ thống phân tán. Trong mô hình này sẽ có một tập các tiến trình server mà mỗi tiến trình đóng vai trò như là một trình quản lý tài nguyên cho một tập hợp các tài nguyên cho trước và một tập hợp các tiến trình client trong đó mỗi tiến trình thực hiện một tác vụ nào đó cần truy xuất tới tài nguyên phần cứng hoặc phần mềm dùng chung. Bản thân các trình quản lý tài nguyên cần phải truy xuất tới các tài nguyên dùng chung được quản lý bởi một tiến trình khác, vì vậy một số tiến trình vừa là tiến trình client vừa là tiến trình server.

Các tiến trình client phát ra các yêu cầu tới các server bất kỳ khi nào chúng cần truy xuất tới một trong các tài nguyên của các server. Nếu yêu cầu là đúng đắn thì server sẽ thực hiện hành động được yêu cầu và gửi một đáp ứng trả lời tới tiến trình client.

Mô hình client/server cung cấp một cách tiếp cận tổng quát để chia sẻ tài nguyên trong các hệ thống phân tán. Mô hình này có thể được cài đặt bằng rất nhiều môi trường phần cứng và phần mềm khác nhau. Các máy tính được sử dụng để chạy các tiến trình client/server có nhiều kiểu khác nhau và không cần thiết phải phân biệt giữa chúng; cả tiến trình client và tiến trình server đều có thể chạy trên cùng một máy tính. Một tiến trình server có thể sử dụng dịch vụ của một server khác.

Mô hình truyền tin client/server hướng tới việc cung cấp dịch vụ. Quá trình trao đổi dữ liệu bao gồm:

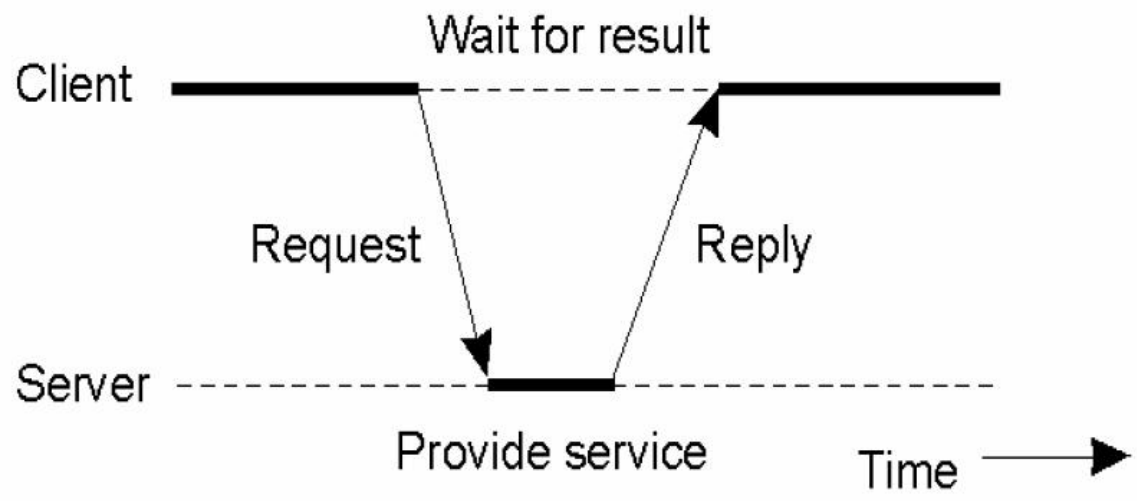
(1) Truyền một yêu cầu từ tiến trình client tới tiến trình server

(2) Yêu cầu được server xử lý

(3) Truyền đáp ứng cho client

Mô hình truyền tin này liên quan đến việc truyền hai thông điệp và một dạng đồng bộ hóa cụ thể giữa client và server. Tiến trình server phải nhận thức được thông điệp được yêu cầu ở bước một ngay khi nó đến và hành động phát ra yêu cầu trong client phải được tạm dừng (bị phong tỏa) và buộc tiến trình client ở trạng thái chờ cho tớ khi nó nhận được đáp ứng do server gửi về ở bước 3.

Mô hình client/server thường được cài đặt dựa trên các thao tác cơ bản là gửi. (send)



và nhận (receive).

Hình 1-1 Mô hình biểu diễn cơ chế Client-Server

CHƯƠNG II -TRÌNH BÀY VẤN ĐỀ

2.1. Phân tích vấn đề

Trong quá trình sử dụng máy tính, có thể phát sinh các sự cố, vấn đề liên quan đến máy tính mà bản thân người sử dụng không thể chủ động xử lý mà cần nhờ đến một người khác thông thạo hơn về vấn đề đó giải quyết giúp. Thế nhưng như vậy cũng không tránh khỏi những vấn đề khác nảy sinh:

- Yêu cầu di chuyển : người gặp vấn đề hoặc người giải quyết vấn đề phải di chuyển, hoặc cả 2 cùng phải di chuyển để xử lý trực tiếp vấn đề (đây là vấn đề lớn nhất nảy sinh).
- Mất công di chuyển là vậy nhưng cũng chưa hẳn là vấn đề của máy tính được giải quyết triệt để, có thể phát sinh thêm lỗi sau này, nếu vậy phải yêu cầu di chuyển nhiều lần, rất bất tiện.
-

Một trong những phương pháp đơn giản để giải quyết bài toán này là xây dựng một chương trình, ứng dụng có thể giúp điều khiển máy tính từ xa, hỗ trợ việc giải quyết các vấn đề mà không cần thiết phải gặp trực tiếp để xử lý. Việc này sẽ giúp tiết kiệm chi phí đi lại cũng như tăng cao hiệu suất làm việc cho cả 2 phía.

2.2. Thiết kế giải pháp - thuật toán chương trình

Thuật toán được xây dựng dựa trên mô hình client / server trong đó client sẽ là máy trạm, gửi màn hình điều khiển lên server và server sẽ là người nhận, tiếp nhận xử lý màn hình của máy trạm. Do đó, thuật toán chính sẽ gồm 2 phần : Client và Server

2.2.1. Phía Client

- Tạo *DataOutputStream OutToServer* - để gửi dữ liệu cho server
- Phương thức getInputStream() - trả về một luồng nhập để đọc dữ liệu từ một socket vào chương trình.

Bước 1 : Thiết lập kết nối với máy chủ

- Nhập địa chỉ IP của máy chủ
- Nhập mật khẩu của máy chủ

Bước 2 : Thực hiện quản lý máy tính từ xa.

- Xác thực kết nối - *Authenticate*
- Gửi thông tin sự kiện từ máy trạm lên máy chủ thông qua class *SendEvents*
- Nhận màn hình điều khiển từ máy chủ qua *receiveScreen*
- Thực hiện các thao tác và chuột sẽ được class *Robot* ghi nhận và xử lý
-
- Kết thúc

2.2.2. Phía Server

- Tạo *DataInputStream InFromServer* - để nhận dữ liệu từ client
- Tạo *FileOutputStream OutToFile* - để ghi dữ liệu ra file

Bước 1 : Tạo kết nối

- Đặt mật khẩu - *SetPassword*

Bước 2 : Thực hiện quá trình quản lý từ xa

- Tạo kết nối cho máy trạm truy cập - *InitConnection*
- Gửi màn hình điều khiển cho máy trạm qua *SendScreen*
- Nhận thông tin sự kiện từ máy trạm thông qua class *ReceiveEvents*
-
- Kết thúc

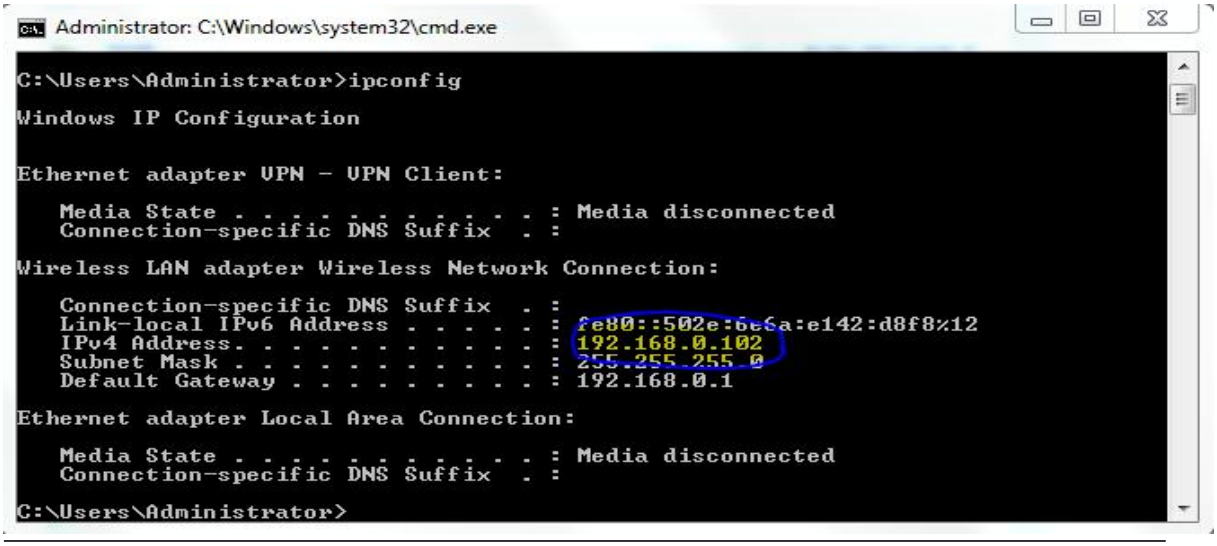
CHƯƠNG III – TRIỂN KHAI VÀ ĐÁNH GIÁ

3.1. Giao diện chương trình

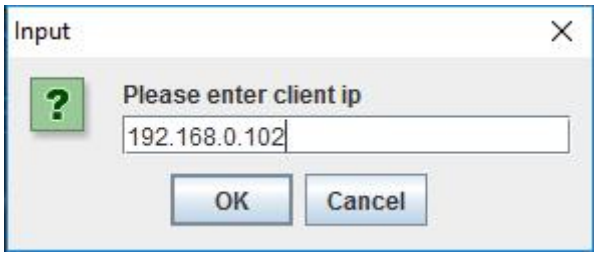
Danh sách các thành phần giao diện chủ yếu:

- TextBox "Set Password" đặt mật khẩu cho máy chủ
- TextBox "Password" nhập mật khẩu của máy chủ
- Button "Submit" Xác thực mật khẩu
- TextBox "Please enter Server IP" nhập địa chỉ IP của máy chủ
- Button "OK" Xác nhận địa chỉ IP
- Button "Cancel" hủy lệnh kết nối

3.1.1. Lấy địa chỉ IP

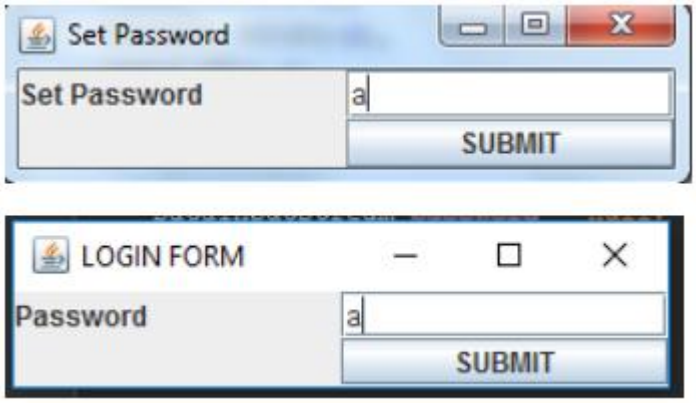


Hình 3-1 Lấy địa chỉ IP của máy trạm



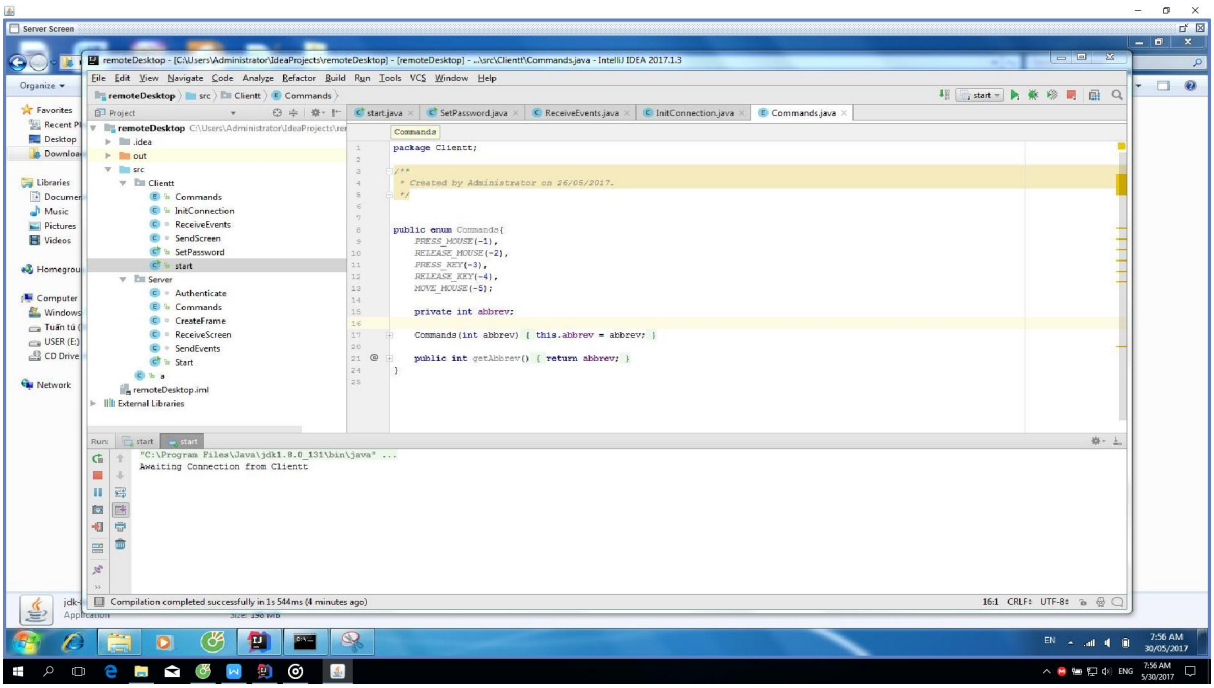
Hình 3-2 Nhập địa chỉ IP của Client tại máy Server

3.1.2. Mật khẩu

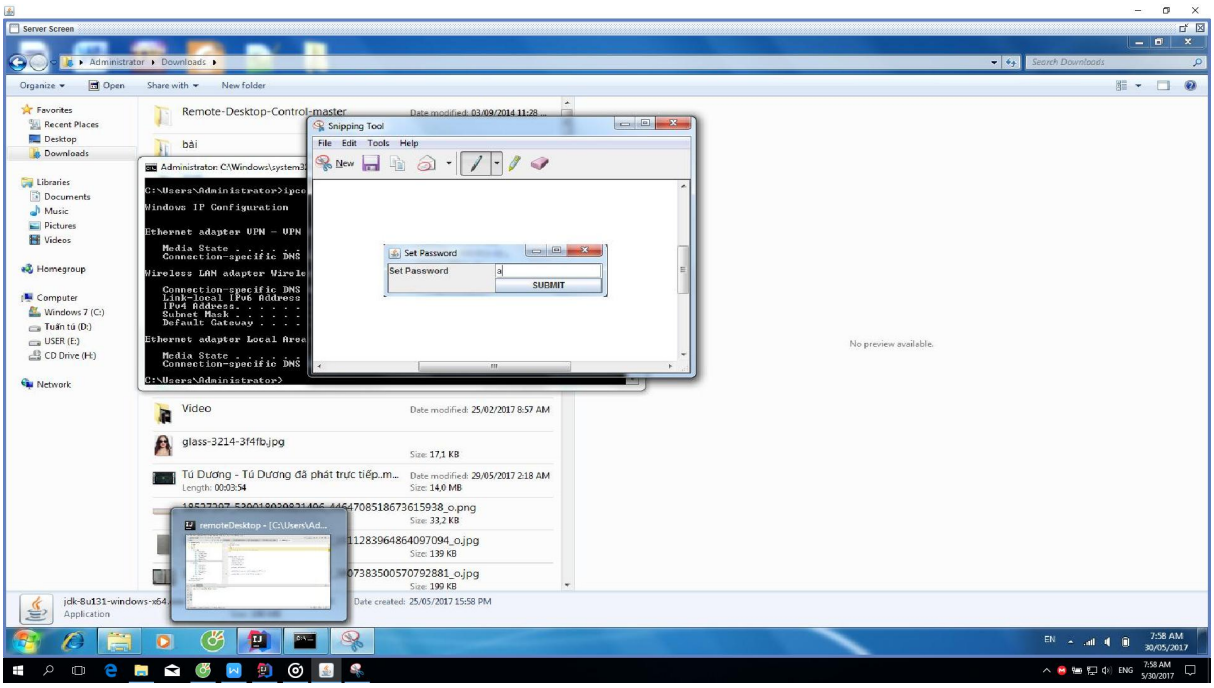


Hình 3-3 Mật khẩu

3.1.3. Quá trình điều khiển



Hình 3-4 Quá trình điều khiển



Hình 3-5 Quá trình điều khiển

3.2. Hoạt động của chương trình

Sau khi khởi động chương trình, người sử dụng thực hiện

Bước 1. Người sử dụng nhập kiểm tra địa chỉ IP máy chủ và nhập vào máy trạm để tạo kết nối

Bước 2. Người sử dụng đặt mật khẩu ở máy chủ và nhập mật khẩu ở máy trạm để thực hiện truy cập

Bước 3. Thực hiện thao tác với máy chủ từ máy trạm.

Sau đó, người dùng có thể tiếp tục hoặc thoát chương trình bằng cách ngắt kết nối chương trình từ phía máy chủ..

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết quả đạt được

Chương trình đã giải quyết được yêu cầu cơ bản của đề tài là điều khiển, quản lý các tác vụ từ xa.

Trong toàn bộ quá trình thao tác đã có giao diện trực quan, dễ sử dụng.

2. Những vấn đề tồn tại

+ Khi thao tác thì vẫn có độ trễ và độ lệch của việc bắt tọa độ chuột.

+ Chương trình phụ thuộc nhiều vào tốc độ xử lý cả máy và băng thông của mạng.

+

3. Hướng phát triển

Về phần tìm hiểu và cơ sở lý thuyết, em sẽ tiếp tục tìm hiểu và nghiên cứu sâu hơn về đề tài để phục vụ sau này.

Về phần chương trình sẽ tiếp tục xây dựng và khắc phục và hoàn thiện một số vấn đề bất cập, bổ sung một số chức năng tốt hơn để cải tiến chương trình.

TÀI LIỆU THAM KHẢO

[1] - Phạm Minh Tuấn,*Bài giảng Lập trình mạng*

[2] - *Lập trình Socket với giao thức TCP*

http://laptrinh.vn/d/4517-lap-trinh-socket-giao-thuc tcp.html?s=2f914b9aed6268f30cb34e0fac489f6b

[3] - *Giao thức TCP/IP*

http://sinhvienit.net/tut/mang-may-tinh/mang-can-ban/131-chuong-7-giao-thuc-tcp-ip.html

[4] - *Lập trình mạng socket trong java*

http://vietjack.com/java/lap_trinh_mang_trong_java.jsp

PHỤ LỤC

Client package

* receivescreen.java

```
package Network.Client;

/**
 * Created by tungv on 5/24/2017.
 */

import java.awt.Graphics;
import java.awt.Image;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectInputStream;
import javax.imageio.ImageIO;
import javax.swing.JPanel;

class ReceiveScreen extends Thread{
    private ObjectInputStream cObjectInputStream = null;
    private JPanel cPanel = null;
    private boolean continueLoop = true;
    InputStream oin = null;
    Image image1 = null;

    public ReceiveScreen(InputStream in,JPanel p){
        oin = in;
        cPanel = p;
        start();
    }

    public void run(){
        try{
            //Read screenshots of the client and then draw them
            while(continueLoop){
                byte[] bytes = new byte[1024*1024];
                int count = 0;
                do{
                    count+=oin.read(bytes,count,bytes.length-count);
                }while(!(count>4 && bytes[count-2]==(byte)-1 && bytes[count-
1]==(byte)-39));

                image1 = ImageIO.read(new ByteArrayInputStream(bytes));
                image1 =
image1.getScaledInstance(cPanel.getWidth(),cPanel.getHeight(),Image.SCALE_FAST);

                //Draw the received screenshots

                Graphics graphics = cPanel.getGraphics();
                graphics.drawImage(image1, 0, 0, cPanel.getWidth(),
cPanel.getHeight(), cPanel);
            }
```

```
        } catch(IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

* sendEvents.java

```
package Network.Client;

/**
 * Created by tungv on 5/24/2017.
 */


import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
import java.io.IOException;
import java.io.PrintWriter;
import java.net.Socket;
import javax.swing.JPanel;


class SendEvents implements KeyListener, MouseMotionListener, MouseListener{
    private Socket cSocket = null;
    private JPanel cPanel = null;
    private PrintWriter writer = null;
    String width = "", height = "";
    double w;
    double h;

    SendEvents(Socket s, JPanel p, String width, String height){
        cSocket = s;
        cPanel = p;
        this.width = width;
        this.height = height;
        w = Double.valueOf(width.trim()).doubleValue();
        h = Double.valueOf(width.trim()).doubleValue();

        //Associate event listeners to the panel

        cPanel.addKeyListener(this);
        cPanel.addMouseListener(this);
        cPanel.addMouseMotionListener(this);

        try{
            //Prepare PrintWriter which will be used to send commands to the client
            writer = new PrintWriter(cSocket.getOutputStream());
        } catch(IOException ex) {
```

```
            ex.printStackTrace();
        }
    }

    public void mouseDragged(MouseEvent e){
    }

    public void mouseMoved(MouseEvent e){
        double xScale = (double)w/cPanel.getWidth();
        double yScale = (double)h/cPanel.getHeight();
        writer.println(Commands.MOVE_MOUSE.getAbbrev());
        writer.println((int)(e.getX()*xScale));
        writer.println((int)(e.getY()*yScale));
        writer.flush();
    }

    public void mouseClicked(MouseEvent e){
    }

    public void mousePressed(MouseEvent e){
        writer.println(Commands.PRESS_MOUSE.getAbbrev());
        int button = e.getButton();
        int xButton = 16;
        if(button==3){
            xButton = 4;
        }
        writer.println(xButton);
        writer.flush();
    }

    public void mouseReleased(MouseEvent e){
        writer.println(Commands.RELEASE_MOUSE.getAbbrev());
        int button = e.getButton();
        int xButton = 16;
        if(button==3){
            xButton = 4;
        }
        writer.println(xButton);
        writer.flush();
    }

    public void mouseEntered(MouseEvent e){
    }

    public void mouseExited(MouseEvent e){
    }

    public void keyTyped(KeyEvent e){
    }

    public void keyPressed(KeyEvent e){
        writer.println(Commands.PRESS_KEY.getAbbrev());
        writer.println(e.getKeyCode());
        writer.flush();
    }
```

```
    public void keyReleased(KeyEvent e){
        writer.println(Commands.RELEASE_KEY.getAbbrev());
        writer.println(e.getKeyCode());
        writer.flush();
    }
}
```

* authenticate.java

```
package Network.Client;

/**
 * Created by tungv on 5/24/2017.
 */


import javax.swing.*;

import java.awt.*;

import java.awt.event.*;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.net.Socket;

class Authenticate extends JFrame implements ActionListener{
    private Socket cSocket = null;
    DataOutputStream psswrchk = null;
    DataInputStream verification = null;
    String verify ="";
    JButton SUBMIT;
    JPanel panel;
    JLabel label, label1;
    String width="",height="";
    final JTextField text1;

    Authenticate(Socket cSocket){
        label1=new JLabel();
        label1.setText("Password");
        text1 = new JTextField(15);
        this.cSocket = cSocket;

        label=new JLabel();
        label.setText("");
        this.setLayout(new BorderLayout());

        SUBMIT = new JButton("SUBMIT");

        panel=new JPanel(new GridLayout(2,1));
        panel.add(label1);
        panel.add(text1);
        panel.add(label);
        panel.add(SUBMIT);
        add(panel,BorderLayout.CENTER);
        SUBMIT.addActionListener(this);
        setTitle("LOGIN FORM");
    }
```

```
    public void actionPerformed(ActionEvent ae){


        String value1=text1.getText();

        try{
            psswrchk= new DataOutputStream(cSocket.getOutputStream());
            verification= new DataInputStream(cSocket.getInputStream());
            psswrchk.writeUTF(value1);
            verify=verification.readUTF();

        }catch (IOException e){
            e.printStackTrace();
        }

        if(verify.equals("valid")){
            try{
                width = verification.readUTF();
                height = verification.readUTF();

            }catch (IOException e){
                e.printStackTrace();
            }
            CreateFrame abc= new CreateFrame(cSocket,width,height);
            dispose();
        }
        else {
            System.out.println("enter the valid password");
            JOptionPane.showMessageDialog(this, "Incorrect  password", "Error",
JOptionPane.ERROR_MESSAGE);
            dispose();
        }

    }

}
```

* start.java

```
package Network.Client;

/**
 * Created by tungv on 5/24/2017.
 */

import java.net.Socket;

import javax.swing.JOptionPane;

public class Start{
    static String port = "4907";

    public static void main(String args[]){
        String ip = JOptionPane.showInputDialog("Please enter Server ip");
        new Start().initialize(ip, Integer.parseInt(port));
    }

    public void initialize(String ip, int port){
        try{

            Socket sc = new Socket(ip,port);
            System.out.println("Connecting to the Server");
            //Authenticate class is responsible for security purposes
            Authenticate frame1= new Authenticate(sc);

            frame1.setSize(300,80);
            frame1.setLocation(500,300);
            frame1.setVisible(true);
        } catch (Exception ex){
            ex.printStackTrace();
        }
    }
}
```

Server package

* Init Connection.java

```
package Network.Server;

/**
 * Created by tungv on 5/24/2017.
 */
import java.awt.Dimension;
import java.awt.GraphicsDevice;
import java.awt.GraphicsEnvironment;
//import java.awt.GraphicsDevice;
import java.awt.Rectangle;
import java.awt.Robot;
import java.awt.Toolkit;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
//import javax.swing.*;

public class InitConnection{

    ServerSocket socket = null;
    DataInputStream password = null;
    DataOutputStream verify = null;
    String width="";
    String height="";

    InitConnection(int port,String value1){
        Robot robot = null;
        Rectangle rectangle = null;
        try{
            System.out.println("Awaiting Connection from Server");
            socket=new ServerSocket(port);

            GraphicsEnvironment gEnv =
GraphicsEnvironment.getLocalGraphicsEnvironment();
            GraphicsDevice gDev = gEnv.getDefaultScreenDevice();

            Dimension dim=Toolkit.getDefaultToolkit().getScreenSize();
            String width=""+dim.getWidth();
            String height=""+dim.getHeight();
            rectangle=new Rectangle(dim);
            robot=new Robot(gDev);

            drawGUI();

            while(true){
                Socket sc=socket.accept();
                password=new DataInputStream(sc.getInputStream());
                verify=new DataOutputStream(sc.getOutputStream());
                //String username=password.readUTF();
```

```
                String pssword=password.readUTF();

                if(pssword.equals(value1)){
                    verify.writeUTF("valid");
                    verify.writeUTF(width);
                    verify.writeUTF(height);
                    new SendScreen(sc,robot,rectangle);
                    new ReceiveEvents(sc,robot);}
                else{
                    verify.writeUTF("Invalid");
                }
            }
        }catch (Exception ex){
            ex.printStackTrace();
        }
    }

    private void drawGUI(){
    }

}
```

* receiveEvents.java

```
package Network.Server;

/**
 * Created by tungv on 5/24/2017.
 */
import java.awt.Robot;
import java.io.IOException;
import java.net.Socket;
import java.util.Scanner;
/* Used to recieve server commands then execute them at the client side*/

class ReceiveEvents extends Thread{
    Socket socket= null;
    Robot robot = null;
    boolean continueLoop = true;

    public ReceiveEvents(Socket socket, Robot robot){

        this.socket = socket;
        this.robot = robot;
        start(); //Start the thread and hence calling run method
    }



    public void run(){
        Scanner scanner = null;
        try {
            scanner = new Scanner(socket.getInputStream());
            while(continueLoop){
                //recieve commands and respond accordingly

                int command = scanner.nextInt();
                switch(command){
                    case-1:
                        robot.mousePress(scanner.nextInt());
                        break;
                    case-2:
                        robot.mouseRelease(scanner.nextInt());
                        break;
                    case-3:
                        robot.keyPress(scanner.nextInt());
                        break;
                    case-4:
                        robot.keyRelease(scanner.nextInt());
                        break;
                    case-5:
                        robot.mouseMove(scanner.nextInt(),scanner.nextInt());
                        break;
                }

            }
        }catch(IOException ex){
```

```
            ex.printStackTrace();
        }
    }//end function

}//end class
```

*sendScreen.java

```
package Network.Server;

/**
 * Created by tungv on 5/24/2017.
 */
import java.awt.Rectangle;
import java.awt.Robot;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.io.OutputStream;
import java.net.Socket;
import javax.imageio.ImageIO;


class SendScreen extends Thread{

    Socket socket=null;
    Robot robot=null;
    Rectangle rectangle=null;
    boolean continueLoop=true;

    OutputStream oos=null;

    public SendScreen(Socket socket,Robot robot,Rectangle rect) {
        this.socket=socket;
        this.robot=robot;
        rectangle=rect;
        start();
    }

    public void run(){

        try{
            oos=socket.getOutputStream();

        }catch(IOException ex){
            ex.printStackTrace();
        }

        while(continueLoop){
            BufferedImage image=robot.createScreenCapture(rectangle);
```

```
            try{
                ImageIO.write(image,"jpeg",oos);
            }catch(IOException ex){
                ex.printStackTrace();
            }

            try{
                Thread.sleep(10);
            }catch(InterruptedException e){
                e.printStackTrace();
            }
        }
    }
}
```

*setPassword.java

```
package Network.Server;

/**
 * Created by tungv on 5/24/2017.
 */
import javax.swing.*;

import java.awt.event.*;
import java.awt.*;

public class SetPassword extends JFrame implements ActionListener{
    static String port="4907";
    JButton SUBMIT;
    JPanel panel;
    JLabel label1,label2;
    JTextField text1,text2;
    String value1;
    String value2;
    JLabel label;

    SetPassword(){
        label1=new JLabel();
        label1.setText("Set Password");
        text1 = new JTextField(15);

        label=new JLabel();
        label.setText("");


        this.setLayout(new BorderLayout());

        SUBMIT = new JButton("SUBMIT");
```

```
        panel=new JPanel(new GridLayout(2,1));
        panel.add(label1);
        panel.add(text1);

        panel.add(label);
        panel.add(SUBMIT);
        add(panel,BorderLayout.CENTER);
        SUBMIT.addActionListener(this);
        setTitle("Set Password to connect to the Server");

    }

    public void actionPerformed(ActionEvent ae){


        value1=text1.getText();
        dispose();
        new InitConnection(Integer.parseInt(port),value1);
    }

    public String getValue1(){

        return value1;
    }


    public static void main(String[] args){

        SetPassword frame1= new SetPassword();
        frame1.setSize(300,80);
        frame1.setLocation(500,300);
        frame1.setVisible(true);

    }

}
```

*start.java

```
package Network.Server;

/**
 * Created by tungv on 5/24/2017.
 */

public class start{
    public static void main(String[] args){
        SetPassword frame1= new SetPassword();
        frame1.setSize(300,80);
        frame1.setLocation(500,300);
        frame1.setVisible(true);
    }
}
```