TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

──────── \* ───────

BÁO CÁO

**Bài tập lớn Lập trình Mạng**

**Xây dựng máy chủ mã hóa file sử dụng nén zlib**

Giảng viên hướng dẫn: **Thầy Hải Anh**

Nhóm sinh viên thực hiện:

**Đào Duy Nhẫn – 2011 1939**

**Hoàng Như Tám – 2013 3411**

# **TÓM TẮT NỘI DUNG BÁO CÁO**

Với mục đích môn học là sử dụng giao tiếp giữa các ứng dụng sử dụng socket trong mạng, nhóm chúng em đã chọn đề tài: Xây dựng dịch vụ nén và giải nén file văn bản sử dụng công nghệ nén zlib.

Với một đề tài như thế, nhóm chúng em có rất nhiều ý tưởng để mở rộng đề tài, tuy nhiên, do thời gian có hạn, nên chúng em chỉ xây dựng một dịch vụ đơn giản nhất có thể, sử dụng socket trên ngôn ngữ lập trình c++ để truyền thông giữa client và server. Cũng do thời gian hạn hẹp nên chúng em chỉ xây dựng server(và client) trên nền tảng Linux(cụ thể là Fedora 23). Hệ thống vẫn có thể vận hành trên nền tảng Linux khác, tuy nhiên chúng em chưa kiểm chứng.

Bài báo cáo được chia thành các phần như sau:

* Chương I: Giới thiệu triển khai hệ thống.
* Chương II: Giới thiệu về thiết kế ngoài.
* Chương III: Giới thiệu về thiết kế bên trong.
* Chương IV: Phân tích một số điểm yếu của hệ thống.
* Kết luận của nhóm.

# **LỜI CẢM ƠN**

Em xin được gửi lời cám ơn chân thành tới thầy giáo Hải Anh. Thầy đã hướng dẫn, giảng dạy chúng em môn học “Lập trình mạng”. Với môn học này, chúng em đã hiểu sâu thêm về socket, các trường hợp đứt kết nối có thể sảy ra trong thực tế. Đồng thời hiểu được cách để sử dụng soket. Trong quá trình học tập, chúng em rất cám ơn thầy đã nhiệt tình giải đáp những câu hỏi, chỉ dạy chúng em trong suốt kì học.

Hưng Yên, tháng 12 năm 2016

Nhóm sinh viên thực hiện

Đào Duy Nhẫn.

Hoàng Như Tám.

# **MỤC LỤC**

Contents

[**TÓM TẮT NỘI DUNG BÁO CÁO** 2](#_Toc469478326)

[**LỜI CẢM ƠN** 3](#_Toc469478327)

[**MỤC LỤC** 4](#_Toc469478328)

[Chương I: Giới thiệu triển khai hệ thống 5](#_Toc469478329)

[1. Yêu cầu hệ thống 5](#_Toc469478330)

[2. Biên dịch và cài đặt hệ thống 5](#_Toc469478331)

[1.2. Biên dịch và cài đặt 5](#_Toc469478332)

[1.3. Sử dụng hệ thống 2](#_Toc469478333)

[Chương II: Giới thiệu thiết kế ngoài 3](#_Toc469478334)

[Chương III: Giới thiệu về thiết kế bên trong 3](#_Toc469478335)

[1. Thiết kết chung. 3](#_Toc469478336)

[1.2. Server 3](#_Toc469478337)

[1.3. Client 3](#_Toc469478338)

[1.4. TCP-Stream 3](#_Toc469478339)

[1.5. TCP-Acceptor 4](#_Toc469478340)

[1.6. TCP-Connector 4](#_Toc469478341)

[1.7. Zpipe 4](#_Toc469478342)

[1. Thiết kế chi tiết 4](#_Toc469478343)

[1.8. Server 4](#_Toc469478344)

[1.9. Client 5](#_Toc469478345)

[1.10. TCP-Stream 5](#_Toc469478346)

[Chương IV: Phân tích một số điểm yếu của hệ thống 6](#_Toc469478347)

[1. Server 6](#_Toc469478348)

[2. TCP-Stream 6](#_Toc469478349)

[3. Tổng thể project 6](#_Toc469478350)

[Kết Luận Của Nhóm 7](#_Toc469478351)

# Giới thiệu triển khai hệ thống

## Yêu cầu hệ thống

Hệ thống này chúng em thiết kế cho nó chạy trên nền tảng Linux, vì vậy chỉ hệ thống linux mới có thể chạy được nó.

Tiếp đến project chúng em sử dụng trình hỗ trợ biên dịch là autoconf và automake. Trên những distro linux khác nhau có thể tên gọi của chúng khác nhau, nhưng đó là tên phổ biến của chúng.

Về ngôn ngữ lập trình, chúng em lập trình bằng ngôn ngữ c++. Tuy vậy, nhưng thực chất các đoạn mã và các chức năng thì thuần C. Lập trình trên c++ cho phép nhóm chúng em dễ quản lí các chức năng hơn. Ngoài ra chúng em không hề sử dụng bất cứ tính năng lạ nào. Vì vậy, hệ thống cần có trình biên dịch c++.

## Biên dịch và cài đặt hệ thống

### Biên dịch và cài đặt

Các bước cài đặt hệ thống và sử dụng nhóm đã ghi các bước vào file README và file INSTALL.

Trước tiên là thư viện nén file zlib. Được đặt trong đường dẫn src/lib/zlib. Phiên bản zlib 1.2.8 là phiên bản mới nhất tại thời điểm nhóm tiến hành làm project. Nhóm tải trực tiếp từ trang chủ về, vì vậy cần phải biên dịch lại. Cụ thể, ta di chuyển vào thư mục zlib, sau đó thực hiện biên dịch tự động bằng hai lệnh là: make clean và make.

Sau khi biên dịch thư viện zlib xong, ta tiến hành biên dịch lõi hệ thống. Di chuyển vào thư mục gốc (của project). Chạy những lệnh sau:

* Autoreconf –fiv
* ./configure
* Make

Sau khi biên dịch xong, những file lõi được biên dịch thành công sẽ được đặt ở thư mục bin (nằm trong thư mục gốc của project). Tiếp theo, tiến hành cài đặt hệ thống. Thực hiện hai thao tác sau để hoàn thành việc cài đặt hệ thống. (Đứng tại thư mục gốc của project):

* Chạy với quyền quản trị hệ thống (root): make install
* Chạy với quyền quản trị hệ thống (root): ./bin/installinstall\_server

Đến lúc này, hệ thống đã được cài đặt. Chúng ta sẽ chỉ cần phải mở file cấu hình hệ thống được đặt ở những vị trí: /andromeda/server/config.conf và /andromeda/client/config.conf. Trong hai file này sẽ chỉ định cổng mà server sẽ lắng nghe để phục vụ người dùng. Đồng thời nếu là client sẽ chỉ định địa chỉ server và cổng mà client sẽ kết nối tới.

Cụ thể, tại file /andromeda/server/config.conf chúng ta chỉ ghi cổng mà server sẽ lắng nghe. File /andromeda/client/config.conf dòng đầu sẽ ghi cổng, dòng thứ hai sẽ ghi địa chỉ của server, bao gồm cả địa chỉ ip hoặc địa chỉ name server cũng được.

Đến đây thì việc cài đặt hệ thống đã hoàn tất. Tất cả các thành phần của hệ thống đều có thể gọi được từ bất kì vị trí nào trong thống, vì chúng đều được cài đặt là lệnh thường trú của hệ thống.

### Sử dụng hệ thống

Về phía server, đứng tại một vị trí bất kì, ta chạy lệnh: server\_dev\_b\_plant. Khi này, trên màn hình terminal không xuất hiện thông báo gì, chỉ khi nào có client kết nối tới thì server sẽ thông báo một số thông tin cần thiết.

Về phí client, người dùng muốn mã hóa một file nào đó thì sử dụng cú pháp:

client\_dev\_b\_plant Encrypt <đường dẫn tuyệt đối của file muốn mã> <đường dẫn tuyệt đối của file muốn lưu trữ>

Lưu ý rằng, tham số “Encrypt” là phải chính xác.

Khi muốn giải mã một file nào đó thì người dùng sử dụng cú pháp:

client\_dev\_b\_plant Decrypt <đường dẫn tuyệt đối của file muốn mã> <đường dẫn tuyệt đối của file muốn lưu trữ>

Tham số “Decrypt” cũng phải chính xác.

Khi chạy chương trình, trên giao diện dòng lệnh sẽ thông báo một số sự kiện quan trọng. Khi thực hiện xong tác vụ, chương trình sẽ tự thoát.

# Giới thiệu thiết kế ngoài

Tổng thể, hệ thống được chia thành hai thành phần: server và client.

Phía server luôn chạy để phục vụ người dùng bất cứ khi nào muốn. Khi người dùng yêu cầu, server sẽ tiến hành fork tiến trình, sau đó tải file từ người dùng lên server, tiếp đó sử dụng lời gọi lênh zpipe(chương trình nén/giải nén) để tiến hành xử lí file. Cuối cùng, gửi trả lại file đã xử lí về cho client.

Phía người dùng khi cần thiết sẽ tiến hành sử dụng chương trình để tải file lên máy chủ sử dụng socket, sau khi tải lên máy chủ thì chờ đợi máy chủ nén/giải nén file, xong sẽ tải về máy người dùng dựa vào các tham số đầu vào của chương trình.

Tất cả các thành phần server và client đều là sử dụng dòng lệnh, tuy vậy, chúng có thể được gọi từ bất cứ đâu trong hệ thống. Không cần phải chỉ đường dẫn tuyệt đối của file thực thi.

# Chương III: Giới thiệu về thiết kế bên trong

## Thiết kết chung.

Cả server và client đều sử dụng thành phần chung là TCP-Stream, đó là là một đối tượng bao gồm một socket và những phương thức đi kèm để thuận tiện cho việc thao tác với socket đó. Đó là phương tiện để truyền tải dữ liệu giữa client và server.

### Server

Server có một thành phần là TCP-Acceptor, thành phần này cho phép máy chủ lắng nghe, chờ đợi client. Khi có client kết nối tới, nó sẽ trả về một TCP-Stream để làm việc và truyền tải dữ liệu cần thiết.

Server sử dụng TCP-Acceptor để đợi client kết nối tới. Khi có client kết nối tới thì TCP-Acceptor trả về một TCP-Stream. Server tiến hành fork tiến trình. Tiến trình cha tiếp tục dùng TCP-Acceptor để đợi client, trong khi đó tiến trình con sẽ phục vụ client, sử dụng TCP-Stream hiện có. Server sẽ hỏi client muốn thực hiện tác vụ gì? Sau đó sẽ kéo file từ client lên server và lưu trữ tại thư mục /andromeda/server. Tên file sẽ được tạo ra bằng cách ghép kí tự F và giờ của hệ thống. Tùy thuộc mã hóa hay giải nén, tiến trình con sẽ gọi một lệnh thống: zpipe để thực hiện tác vụ. Sau đó thì đẩy file đã xử lí về client rồi thoát.

### Client

Client cũng có một thành phần chủ yếu là TCP-Connector, thành phần này cho phép client kết nối tới server. Khi kết nối thành công thì sẽ trả về một TCP-Stream, đó sẽ là kênh giao tiếp với server. Tùy thuộc tham số đầu vào mà client sẽ thực hiện một số động tác như là gửi yêu cầu lên server, hoặc kéo, tải file từ client lên server hoặc ngược lại.

### TCP-Stream

TCP-Stream thực chất là một đối tượng gồm một socket, đồng thời kèm một số thông tin như là port, host. Gói kèm đó là những method như là gửi một đoạn dữ liệu, nhận một đoạn dữ liệu, gửi file, nhận file.

### TCP-Acceptor

TCP-Acceptor là một đối tượng, gồm một thuộc tính là cổng. Đối tượng này sẽ tiến hành bind vào cổng của nó, sau đó đợi client. Khi có client kết nối tới, nó trả về một TCP-Stream.

### TCP-Connector

Đối tượng này cũng bao gồm hai thuộc tính: cổng và địa chỉ của server. Khi thực hiện phương thức connect, nó sẽ tiến hành kết nối tới server và cổng của nó. Nếu thành công thì sẽ trả về một đối tượng TCP-Stream.

### Zpipe

Chương trình này sử dụng thư viện zlib và các hàm của nó để tiến hành nén/giải nén file. Chương trình nhận từ tham số dòng lệnh vào các thông số cần thiết và làm việc một cách độc lập hoàn toàn so với client và server. Đây là một phần gần như tách biệt, độc lập so với server và client.

## Thiết kế chi tiết

### Server

Ngay khi bắt đầu chương trình, server sẽ tìm kiếm file cấu hình là file /andromeda/server/config.conf để đọc thông số: cổng lắng nghe để phục vụ client.

Nếu không thể mở được file này, server đưa ra thông báo cho người dùng và kết thúc chương trình. Nếu mở được thì sẽ tiến hành đọc cổng.

Tiếp sau đó, server tạo một đối tượng TCP-Acceptor với tham số là cổng đã đọc.

Từ giờ trở đi, server đi vào vòng loop: gọi TCP-Acceptor->accept() rồi fork tiến trình, phục vụ client (TCP-Stream đó).

Tiến tình con phục vụ client đầu tiên sẽ nhận một chuỗi kí tự từ client. Nếu chuỗi kí tự đó là “Encrypt” hoặc “Decrypt” thì server sẽ phản hồi lại client một chuỗi “Oke”, ngược lại thì gửi cho client một message “Unknow request” rồi kết thúc tiến trình con.

Nếu request từ client là “Encrypt” thì server sẽ bật cờ isEncrypt = 1 để sau này thuận tiện cho việc gọi lệnh zpipe.

Tiếp theo là kéo file từ client lên server. Trước hết là lấy giờ của hệ thống, sau đó chuyển nó sang dạng kí tự. Thêm chữ F vào đầu chuỗi rồi lấy đó làm file tạm mà client sẽ up lên. File tạm này được lưu ở thư mục /andromeda/server/

Sau khi kéo file xong, server tùy thuộc trạng thái của isEncrypt mà gọi zpipe với tham số phù hợp để tiến hành nén/giải nén file.

Tiếp theo, server gửi trả file về cho client bằng method của TCP-Stream.

Cuối cùng thì thực hiện xóa các file tạm rồi thoát tiến trình con.

Tham khảo mã code trong file src/server/server\_dev\_b\_plant.cpp

### Client

Ngay khi bắt đầu chương trình, client cũng sẽ tìm kiếm file cấu hình /andromeda/server/config.conf để đọc thông tin về cổng và địa chỉ của server. Nếu không thể mở được file thì client sẽ thông báo tới người dùng rồi thoát chương trình.

Sau khi đọc cấu hình, tùy thuộc tham số dòng lệnh, tạo một đối tượng TCP-Connector với hai thông số đã đọc trên rồi tiến hành gọi method connect để kết nối đến server. Nếu kết nối thành công thì client sẽ nhận được một đối tượng TCP-Stream.

Client sẽ tiến hành gửi request đến server. Đó là chuỗi “Encrypt” hoặc “Decrypt”. Sau đó đợi phản hồi từ server. Nếu phản hồi từ server mà không phải là chuỗi “Oke” thì client tự hiểu rằng đã sảy ra lỗi ở đâu đó, vì vậy client sẽ thông báo cho người dùng về sự kiện này rồi thoát chương trình.

Nếu thuận lợi, nhận được phải hồi “Oke” từ người dùng thì client sẽ gọi method sendFile của đối tượng TCP-Stream để gửi file lên server. Sau khi hoàn tất thì gọi method receiveFile của đối tượng TCP-Stream để tải file đã xử lí về.

Tham khảo mã code trong file src/client/client\_dev\_b\_plant.cpp

### TCP-Stream

Đối tượng này có một số thuộc tính cơ bản là m\_sd – đó là file descriptor của socket nếu kết nối thành công. Thuộc tính thứ 2 là m\_peerIP là thuộc tính chứa địa chỉ của server – khi cần thiết. Ngoài ra còn có cả m\_peerPort.

Đối tượng có một số method quan trọng là send, receive để thao tác với dữ liệu như là thao tác với socket thông thường. Mở rộng ra nhóm chúng em bổ xung thêm 4 method được xây dựng dựa trên 2 method send và receive:

* sendFile
* receiveFile
* sendDataSegment
* receiveDataSegment

Trước hết nói về method sendDataSegment. Method sẽ nhận vào 2 tham số là địa chỉ của segment data cần gửi, thứ 2 là độ dài của đoạn dữ liệu đó. Trước hết, ta sẽ gửi một biến long – chỉ độ dài của đoạn dữ liệu. Kế sau đó ta sẽ gửi dữ liệu theo sau. Trong vòng lặp for, mỗi khi gửi được một đoạn dữ liệu k thì ta sẽ giảm dữ liệu cần gửi đi k. Cứ thế cho đến khi nào đoạn dữ liệu cần gửi bằng 0 thì có nghĩa là dữ liệu đã được gửi đi hết.

Đối với method receiveDataSegment. Method sẽ nhận vào địa chỉ của đoạn không gian sẽ lưu trữ dữ liệu sẽ nhận. Đầu tiên nó sẽ nhận một biến long – chỉ độ dài mà đoạn dữ liệu sẽ nhận. Rồi trong vòng lặp while, ta cứ đọc từng đoạn dữ liệu một, đọc được bao nhiêu thì lại công thêm vào biến “dữ liệu đã đọc” bấy nhiêu. Khi nào biến “dữ liệu đã đọc” bằng với biến long ban đầu thì method sẽ kết thúc.

Phân tích thiết kế method sendFile. Method nhận vào tham số là đường dẫn của file sẽ gửi. Sau khi kiểm tra tính hợp lệ của đường dẫn, ta tiến hành mở file. Nếu mở thành công thì sẽ gửi message “Open file: Success” đến phía nhận file. Còn nếu không mở được file thì sẽ gửi tin nhắn “Open file: Failed” đến bên kia.

Trong trường hợp mở file thành công, ta đợi tín hiệu “Ready!” từ phía đối tác. Nếu không nhận được tin nhắn này thì method sẽ kết thúc với giá trị trả về là FAIL. Nếu nhận được tin nhắn xác nhận sẵn sàng nhận file từ phí đối tác, ta tiến hành ước lượng độ dài của file. Sau khi ước lượng độ dài này, ta sẽ gửi số liệu này qua cho đối tác. Sau đó, lần lượt đọc dữ liệu từ file và gửi. Đọc được bao nhiêu thì gửi bấy nhiêu, đồng thời có ghi nhận. Khi nào truyền đủ kích cỡ như đã thông báo với đối tác thì method kết thúc.

Đối với method receiveDataSegment. Method cũng nhận vào một tham số chính là đường dẫn của file sẽ được lưu trữ. Method cũng sẽ kiểm tra tính hợp lệ của đường dẫn, đồng thời mở file để lưu trữ dữ liệu. Nếu mở thành công thì sẽ gửi cho phía gửi một tin nhắn là “Ready!”, ngược lại sẽ gửi cho đối tác một tin nhắn “Can't create file” để bên gửi tự hiểu mà hủy quá trình gửi file. Khi đã sẵn sàng nhận file, method sẽ nhận biến độ dài của file đầu tiên. Rồi tiến hành nhận dữ liệu và ghi nhận lại số liệu. Khi nào đã nhận đủ dữ liệu rồi thì đóng file lại. Kết thúc quá trình.

Tham khảo code trong thư mục src/lib/tcp\_stream/

# Chương IV: Phân tích một số điểm yếu của hệ thống

## Server

Mặc dù server đã có bước chào hỏi qua client thông qua câu hỏi “Encrypt” hoặc “Decrypt” nhưng vẫn chưa đủ nếu bị tấn công, kẻ địch có thể chào hỏi đúng khuôn phép, nhưng rồi sau đó lại dùng một quy trình khác đi, rất có thể server sẽ bị treo cho client đó rất lâu. Server cũng không có cơ chế từ chối client khi cần thiết.

## TCP-Stream

Theo như thiết kế của đối tượng này, ta nhận thấy rằng, khi kết nối giữa client và server mà bị đứt đột ngột thì phía còn lại sẽ không nhận biết được tình trạng này, dẫn tới phải đợi hết thời gian chờ đợi hoặc có thể cứ chờ đợi đối tác vô thời hạn.

Lợi dụng đặc điểm này, kẻ tấn công hoàn toàn có thể phát động là treo server trong một nốt nhạc, vì cạn kiệt tài nguyên của server.

## Tổng thể project

Hệ thống thiết kế rất sơ khai, không có các file log để ghi nhận lịch sử. Khi đứt kết nối sảy ra, cả client và server đều bị khựng lại. Không có cơ chế khắc phục lỗi. Phía client còn phải dùng dòng lệnh chứ không có giao diện. Hệ thống còn rất nhiều lỗ hổng, rất dễ dàng bị sập đổ nếu đưa vào sử dụng thực tế.

# Kết Luận Của Nhóm

Trước hết là về kiến thức trong môn học. Nhóm đã tìm hiểu và ứng dụng được trong thực tế. Dù chưa thể sử dụng kiến thức ở mức độ sâu (phát hiện và xử lí khi sảy ra đứt kết nối), tuy nhiên các thành viên trong nhóm đã biết về những kiến thức đó, có thể sử dụng được khi có đủ thời gian.

Thứ hai là về project. Nhóm chúng em ban đầu có dự định làm một hệ thống đủ sức phát hiện được đứt gãy kết nối, hạn chế tấn công từ kẻ địch. Hỗ trợ tính tăng resume-tiếp tục công việc sau khi kết nối được thiết lập trở lại. Ngoài ra còn hỗ trợ một số tính năng như là: nếu file của người dùng đã có trên server thì không cần phải truyền tải file đó nữa, chỉ cần truyền tải file cần thiết. Hoặc khả năng cân tải nếu có 2 client cùng yêu cầu một tác vụ với file giống nhau. Hỗ trợ khả năng truyền, tải song song luồng(tối đa 16 luồng song song). Nhóm cũng có dự định làm giao diện cho client, lưu nhớ được lịch sử. Phía server cũng có thể giám sát tình trạng server. Mặc dù như vậy nhưng nhóm không có đủ thời gian, vì vậy nhóm bắt buộc phải làm một phiên bản cơ bản nhất để đạt được yêu cầu tối thiểu của môn học. Code dự định ban đầu còn giang dở vì project chưa hoàn thành, nhưng nó được thiết kế theo c++ nên khi có cơ hội, nhóm sẽ kế thừa, tận dụng nó cho một project khác.

Nhóm chân thành cám ơn thầy giáo đã nhiệt tình giảng giải và giải đáp thắc mắc của chúng em trong quá trình làm project.

Hưng Yên, 14/12/2016

Nhóm thực hiện:

Đào Duy Nhẫn

Hoàng Như Tám