**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BỘ MÔN MẠNG MÁY TÍNH & VIỄN THÔNG**

**- - 12HCA2 - -**

**MẠNG MÁY TÍNH NÂNG CAO**

**ỨNG DỤNG BITTORRENT**

**Giáo viên:** **Trần Trung Dũng**

**Nhóm sinh viên:**

Bùi Bá Lộc – 1241363

Dương Diệu Pháp – 1241378

Tp HCM – 2013.

**MỤC LỤC**

[1. Thành viên nhóm 3](#_Toc370763124)

[1.1 Đánh giá tổng quan 3](#_Toc370763125)

[1.2 Mức độ hoàn thành chức năng 3](#_Toc370763126)

[1.3 Phân công 3](#_Toc370763127)

[2. Báo cáo chức năng 5](#_Toc370763128)

[2.1 Cấu trúc gói tin 5](#_Toc370763129)

[2.2 Cấu trúc file / thư mục lưu trữ 5](#_Toc370763130)

[2.3 Cấu trúc file log 6](#_Toc370763131)

[2.4 Kịch bản gửi nhận gói tin 7](#_Toc370763132)

[2.5 Kỹ thuật chọn peer để download 1 chunk 7](#_Toc370763133)

[2.6 Cơ chế đảm bảo tính tin cậy 8](#_Toc370763134)

[2.7 Kỹ thuật tạm dừng / phục hồi dữ liệu khi download 8](#_Toc370763135)

[3. Hướng dẫn sử dụng Bittorrent 9](#_Toc370763136)

[3.1 Giao diện chương trình 9](#_Toc370763137)

[3.2 Chức năng Cắt tập tin 10](#_Toc370763138)

[3.3 Chức năng nối tập tin 10](#_Toc370763139)

[3.4 Chức năng Tải tập tin 10](#_Toc370763140)

[3.4.1 Tải từng chunk 11](#_Toc370763141)

[3.4.2 Tải tập tin (tất cả các chunk) 11](#_Toc370763142)

[3.5 Xem nhật ký chương trình 11](#_Toc370763143)

[4. Tham khảo 12](#_Toc370763144)

# Thành viên nhóm

## Đánh giá tổng quan

|  |  |  |
| --- | --- | --- |
| **Thành viên** | **MSSV** | **Đánh giá (dựa trên mức độ hoàn thành các chức năng bên dưới)** |
| Bùi Bá Lộc | 1241363 | 96.7% |
| Dương Diệu Pháp | 1241378 | 97.5% |

## Mức độ hoàn thành chức năng

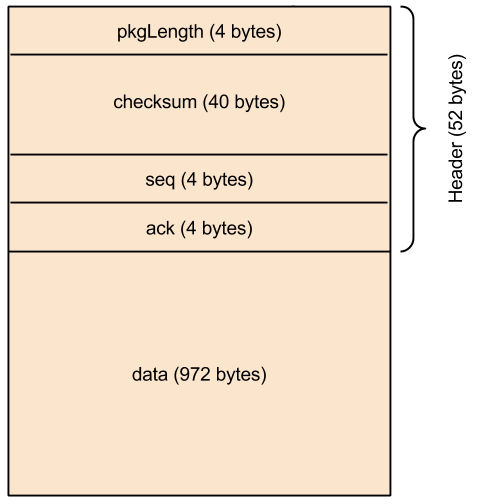
|  |  |  |
| --- | --- | --- |
| **STT** | **Tên kết quả** | **Mức độ hoàn thành** |
| 1 | Tìm vị trí lưu các chunks bằng cách gởi broadcast | Gởi n gói tin |
| 2 | Download các chunks | Mức 2: download đồng thời các chunks |
| 3 | Download 1 file | Mức 2: có optimize khi chọn peer để download 1 chunk (download rared chunk trước, dùng nguyên lý Bernoulli) |
| 4 | Đảm bảo tính tin cậy | Dùng stop and wait |
| 5 | Chia sẻ 1 file | Mức 2: 1 file từ máy local |
| 6 | Ghi log | Ghi log chi tiết |
| 7 | Báo cáo | Báo cáo chi tiết |
| 8 | Chức năng thêm: Tạm dừng / phục hồi tiến trình download | Hoàn thàn tốt |

## Phân công

|  |  |  |
| --- | --- | --- |
| **Nội dung** | **Phân công** | **Mức độ hoàn thành** |
| Kịch bản gửi nhận gói tin | Pháp | 100% - Thiết lập được kịch bản gửi nhận hợp lý |
| Cấu trúc gói tin | Pháp | 90% - Thiết lập cấu trúc cơ bản gồm 2 phần chính gói gọn trong 1024 bytes dữ liệu. Còn một số thành phần dư thừa, chưa áp dụng trong quá trình xử lý gói tin |
| Cấu trúc file lưu trữ | Lộc | 100% - File chứa thông tin các peer trong mạng Nodes.map, cấu trúc của chunk, của file torrent. |
| Cắt tập tin thành các chunk | Lộc | 100% - Cắt thành công tập tin thành các chunk dữ liệu |
| Ghép tập tin từ các chunk | Lộc | 100% - Nối thành công tập tin từ các chunk dữ liệu và file torrent |
| Thiết kế giao diện | Pháp | 100% - Giao diện trực quan với thanh tiến trình và trạng thái hiển thị quá trình cắt, ghép, tải tập tin |
| Ghi nhật ký file log | Lộc | 100% - Lưu từng nhật ký gửi nhận gói tin |
| Phát sinh mã hash | Lộc | 100% - Hỗ trợ phát sinh mã hash với thuật toán SHA1 |
| Kỹ thuật tải 1 chunk | Pháp | 100% - Tải chunk thành công |
| Kỹ thuật tải đồng thời nhiều chunk | Pháp + Lộc | 90% - Tải đồng thời nhiều chunk, nhưng vẫn chưa kiểm soát tắt nghẽn nên thỉnh thoảng gây nên tình trạng time out |
| Kỹ thuật tìm vị trí các chunk bằng cách gởi broadcast | Lộc | 100% - Lấy được danh sách các chunk của từng peer |
| Kỹ thuật tối ưu hoá việc lựa chọn peer để tải 1 chunk (nguyên lý Bernoulli) | Pháp | 100% - Lựa chọn peer có số thread download ít nhất để tải |
| Kỹ thuật sử dụng multi-thread | Lộc | 100% - Phần lớn các chức năng thực hiện đa luồng |
| Kỹ thuật phát sinh sự kiện tuỳ chỉnh và tạm dừng / phục hồi dữ liệu download | Pháp | 100% - Tại mỗi chức năng đều tạo sự kiện để theo dõi, can thiệp hoặc thông báo cho người dùng khi hoàn tất |
| Cơ chế đảm bảo tính tin cậy | Pháp | 100% - Sử dụng thành phần header trong gói tin để kiểm tra vị trí byte bắt đầu, vị trí byte đã nhận, checksum,…  Thực hiện theo cơ chế: gửi phản hồi nhận gói tin thành công rồi mới gửi tiếp. |
| Chia sẻ chunk dữ liệu | Lộc | 80% - Tại cùng một thời điểm cho phép các peer download chunk đã tải từ peer khác. Bị tình trạng tắt nghẽn |
| Viết báo cáo | Pháp | Báo cáo chi tiết |

# Báo cáo chức năng

## Cấu trúc gói tin



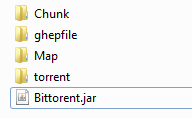
Mỗi gói tin có kích thước 1024 bytes, trong đó phần header chiếm 52 bytes. Các thành phần gói tin:

* pkgLength: Kích thước gói tin, là phần dữ liệu được lưu trữ thực sự trong gói tin
* checksum: Lưu trữ mã hashcode với thuật toán SHA1, giúp kiểm tra tính đúng đắn của gói tin.
* Seq: Vị trí của byte dữ liệu đầu tiên.
* Ack: Vị trí byte dữ liệu cuối cùng đã nhận được.
* Data: Dữ liệu của gói tin (tối đa 972 bytes).

Ngoài ra còn có những gói tin là chuỗi ký tự để kiểm tra trạng thái, thông tin của peer nhanh chóng.

## Cấu trúc file / thư mục lưu trữ

Chương trình đi kèm với 4 thư mục, 2 tập tin:



Thư mục Chunk: chứa các file .chunk  
Thư mục ghepfile: chứa tập tin hoàn chỉnh sau khi ghép các file .chunk lại với nhau  
Thư mục Map: chứa tập tin Nodes.map, nội dung gồm N + 1 dòng, lưu thông tin các node

* Dòng đầu tiên lưu số N: số node trong mạng
* N dòng tiếp theo: mỗi dòng là thông tin của 1 peer theo cấu trúc [id] [ip address] [port]

Thư mục torrent: Lưu tập tin .torrent dùng để tải hoặc ghép file .chunk, nội dung gồm N + 1 dòng:

* Dòng đầu tiên lưu số N: số file chunks.
* N dòng tiếp theo: mỗi dùng là thông tin của 1 fiel .chunk thứ i và giá trị băm của nó theo cấu trúc [i] [hashcode].

Tập tin Bittorrent.jar: tập tin thực thi chương trình.

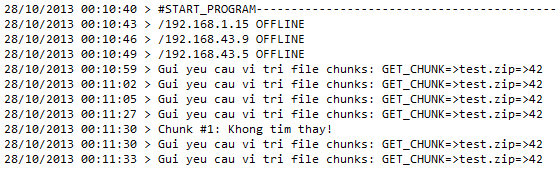
## Cấu trúc file log

Tập tin log.txt được phát sinh ra sau khi thoát chương trình giúp ghi lại nhật ký gửi / nhận các gói tin truyền dữ liệu và gói báo nhận khi truyền các chunk tại nơi gửi và nhận.

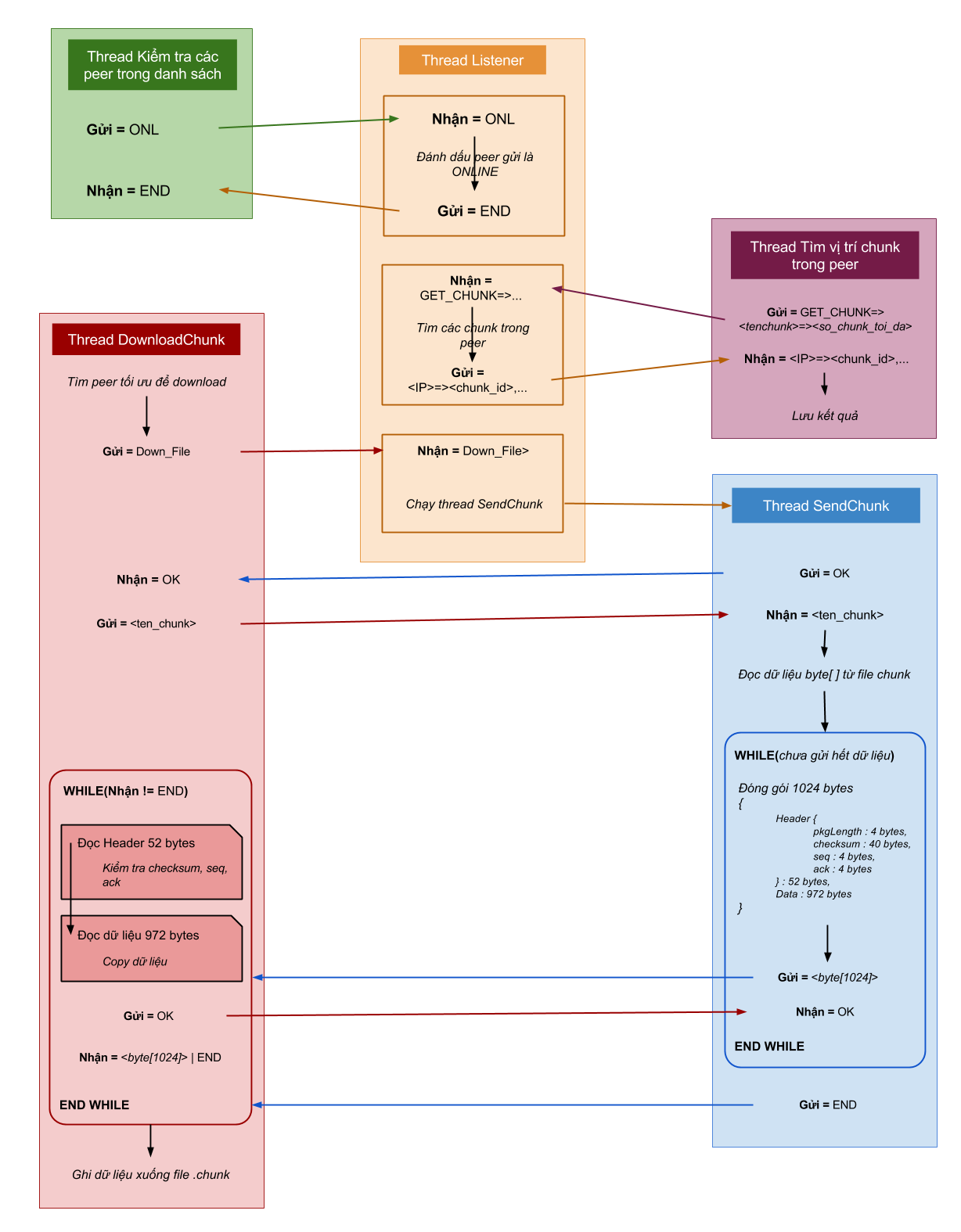
Cấu trúc gồm nhiều dòng [*time\_span*] > [*content*], trong đó:

* *time\_span*: thời gian thao tác.
* *content*: nội dung thao tác.

Ví dụ 1 mẩu nhật ký:



## Kịch bản gửi nhận gói tin



## Kỹ thuật chọn peer để download 1 chunk

Tối ưu hoá việc chọn peer bằng cách dùng nguyên lý Bernoulli. Theo đó, ta sẽ có các đối tượng như sau:

* class *PeerInfo*: lưu thông tin của 1 peer. Thuộc tính *danhSachChunk* lưu danh sách chunk của peer, *danhSachChunkDangDown* lưu danh sách các chunk đang được tải.
* biến toàn cục *danhSachPeer*: lưu danh sách các peer đọc lên trong file Nodes.map

Thuật toán:

* Tìm danh sách peer mà chứa chunk cần download.
* Nếu không có peer nào thì thông báo lỗi, dừng chương trình.
* Tìm trong danh sách kết quả peer có số chunk được tải nhỏ nhất và chọn làm peer đó để tải chunk

## Cơ chế đảm bảo tính tin cậy

Thực hiện theo cơ chế gửi gói tin dữ liệu và chờ gói tin phản hồi từ peer nhận. Trong đó, Peer gửi đóng gói các thông số của gói tin vào 52 bytes header trước khi gửi đi. Sau khi peer nhận nhận gói tin từ peer gửi, peer nhận đọc dữ liệu các trường trong phần header của gói tin để thực hiện việc kiểm tra tính chính xác của dữ liệu:

* pkgLength: Kích thước gói tin gửi / nhận.
* checksum: Lưu trữ mã hashcode với thuật toán SHA1, giúp kiểm tra tính đúng đắn của gói tin.
* Seq: Vị trí của byte dữ liệu đầu tiên.
* Ack: Vị trí byte dữ liệu cuối cùng đã nhận được.

Peer nhận sẽ phát sinh mã hash SHA1 để so sánh với checksum được lưu trữ trong gói tin nhận được, nếu không trùng khớp sẽ tiến hành huỷ gói tin. Ngược lại, peer nhận thực hiện copy byte dữ liệu từ vị trí seq đến hết chiều dài của phần Data của gói tin, lưu trữ và ghi xuống tập tin khi hoàn tất.

## Kỹ thuật tạm dừng / phục hồi dữ liệu khi download

Sử dụng một biến toàn cục lưu giá trị true / false để kiểm tra điều kiện tạm dừng / phục hồi tiến trình tải dữ liệu.

private volatile boolean isRunning = true;

Khởi tạo phương thức điều khiển tiến trình:

public void pauseThread() {

isRunning = false;

}

public void resumeThread() {

isRunning = true;

}

Trong phương thức tải dữ liệu, đặt vòng lặp while để tạm dừng có hiệu lực

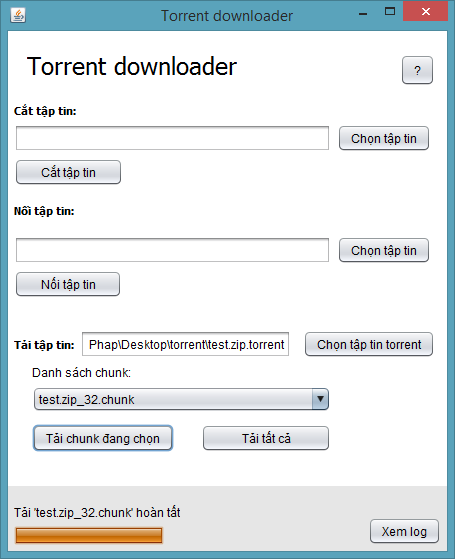
while (!isRunning) {

//Lặp cho đến khi nhận được lệnh isRunning = true

}

# Hướng dẫn sử dụng Bittorrent

## Giao diện chương trình

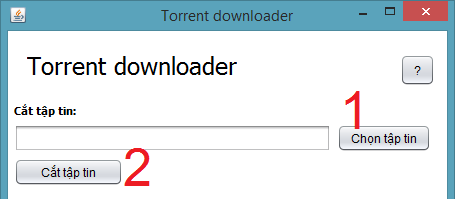


Chương trình chia làm 4 phần chính: Cắt tập tin, Nối tập tin, Tải tập tin và thanh trạng thái.

Để bắt đầu, bạn mở tập tin \Map\Nodes.map (tạo mới nếu không tồn tại) và thêm thông tin các máy peer theo cấu trúc ở Mục 2.2

Khởi động chương trình Bittorrent tại các máy peer đã thêm vào file Nodes.map

## Chức năng Cắt tập tin



Đầu tiên chọn tập tin cần cắt, định dạng bất kỳ, sau đó nhấn nút *Cắt tập tin* để bắt đầu. Mỗi tập tin cắt ra có dung lượng 512 KB sẽ được lưu tại \Chunk\ten\_tap\_tin\ten\_tap\_tin\_[i].chunk, với [i] là thứ tự của file cắt.

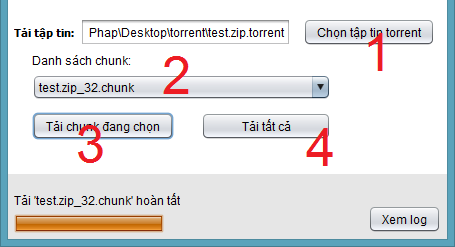
Đồng thời, 1 file torrent\ten\_tap\_tin.torrent được tạo ra, nội dung gồm danh sách các chunk của tập tin vừa cắt kèm theo mã hashcode SHA1 nhằm mục đích kiểm tra tính tin cậy.

## Chức năng nối tập tin



Đầu tiên chọn tập tin ten\_tap\_tin.torrent, sau đó nhấn nút *Nối tập tin* để bắt đầu. Chương trình sẽ tìm các file .chunk trong thư mục \Chunk\ten\_tap\_tin\ để nối lại. Tập tin hoàn chỉnh sẽ được lưu tại ghepfile\ten\_tap\_tin

## Chức năng Tải tập tin



Đầu tiên chọn tập tin \*.torrent cần tải về. Chương trình sẽ tự kiểm tra danh sách các file chunk thiếu của file torrent, kết quả được liệt kê thành danh sách trong ComboBox

### Tải từng chunk

Chọn chunk cần tải trong ComboBox phía trên, sau đó nhấn nút *Tải chunk đang chọn*. Bạn có thể tải nhiều chunk đồng thời bằng cách lặp lại bước trên.

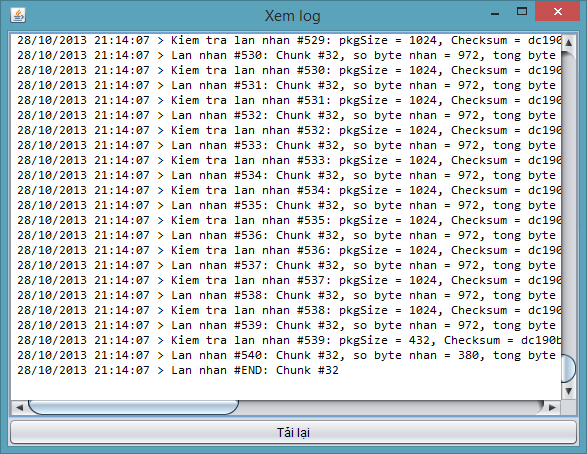
### Tải tập tin (tất cả các chunk)

Để tải tất cả file chunk, bạn nhấn nút *Tải tất cả*. Trong quá trình tải dữ liệu, bạn có thể tạm dừng hoặc khôi phục tiến trình tải bằng cách nhấn nút *Tạm dừng* / *Phục hồi*

## Xem nhật ký chương trình



Nhấn nút *Xem log* tại thanh trạng thái để hiểu thị cửa sổ nhật ký chương trình:



Chi tiết cấu trúc của nhật ký thì xem tại Mục 2.3 của báo cáo.

# Tham khảo

MessageDigest class trong Java Doc: <http://docs.oracle.com/javase/6/docs/api/java/security/MessageDigest.html>

How do I create my own events to pass between objects? <http://www.jguru.com/faq/view.jsp?EID=98547>