Projekt TIN - Dokumentacja

Wiktor Michalski
Przemysław Stawczyk
Maciej Szulik
Kamil Zacharczuk

January 6, 2020

Contents

Part I

Dokumentacja Koncepcji

Chapter 1

Zadanie

1.1 Treść Zadania

Napisać program obsługujący prosty protokół P2P

1. Zasób to plik identyfikowany pewną nazwą, za takie same zasoby uważa się zasoby o takich samych nazwach i takiej samej wielkości pliku w bajtach.
2. Początkowo dany zasób znajduje się w jednym węźle sieci, następnie może być propagowany do innych węzłów w ramach inicjowanego przez użytkownika ręcznie transferu - raz pobrany zasób zostaje zachowany jako kopia.
3. Po pewnym czasie działania systemu ten sam zasób może się znajdować w kilku węzłach sieci (na kilku maszynach).
4. System ma informować o posiadanych lokalnie (tj. w danym węźle) zasobach i umożliwiać ich pobranie.
5. Program powinien umożliwiać współbieżne:
   - wprowadzanie przez użytkownika (poprzez interfejs tekstowy):
     - nowych zasobów - z lokalnego systemu plików
     - poleceń pobrania nazwanego zasobu ze zdalnego węzła
   - pobieranie zasobów (także kilku jednocześnie)
   - rozgłaszanie informacji o posiadanych lokalne zasobach
6. W przypadku pobierania zdalnego zasobu system sam (nie użytkownik) decyduje skąd zostanie on pobrany.
7. Powinno być możliwe pobranie zasobu z kilku węzłów na raz (tj. "w kawałkach").
8. Zasób pobrany do lokalnego węzła jest kopią oryginału, kopia jest traktowana tak samo jak oryginał (są nierozróżnialne). Istnienie kopii jest rozgłaszane tak samo jak oryginału.
9. Właściciel zasobu może go unieważnić wysyłając odpowiedni komunikat rozgłaszany. Wszystkie kopie zasobu powinny przestać być rozgłaszane. W przypadku trwających transferów zasobów powinny się one poprawnie zakończyć, dopiero wtedy informacja o zasobie może zostać usunięta.

1.2 Wariant zadania dla zespołu

4. Opóźnienia dla wybranego węzła - węzeł reaguje, ale (czasami) z dużym opóźnieniem.

1.3 Interpretacja Zadania

Doprecyzowanie treści i dodatkowe założenia

- W związku z tym, że kopia i oryginał są nierozróżnialne, zasób może być unieważniony przez dowolnego użytkownika, który go posiada.
- Każdy węzeł okresowo rozgłasza informację o posiadanych zasobach. Unieważnienie pliku oznacza, że żaden z węzłów nie będzie już rozgłaszał faktu posiadania tego pliku.
- Unieważnienie wysyłane jest asynchronicznie poprzez broadcast UDP.
- W przypadku unieważnienia pliku w trakcie trwającego przesyłu tego pliku przesyłanie kończy się sukcesem, o ile nie wystąpią inne błędy. Nowy posiadacz pliku nie będzie jednak nigdy rozgłaszał o nim informacji.
- Każdy węzeł przechowuje listy dostępnych zasobów każdego innego węzła. Po odebraniu rozgłoszenia listy zasobów od innego węzła lista ta jest nadpisywana w pamięci węzła odbierającego. Informacje o węźle, w tym lista jego zasobów, są usuwane w przypadku braku, przez ustalony czas, nadchodzącego rozgłoszenia jego listy zasobów.
- W przypadku połączenia z innym węzłem w celu pobrania od niego pliku oczekiwanie na odpowiedź tego węzła ma pewien timeout. Ponadto, jeżeli węzeł przekracza pewien ustalony czas odpowiedzi (nawet jeżeli nie dochodzi do timeout'u), to inkrementujemy licznik "opóźnień" tego węzła (każdy węzeł przechowuje takie liczniki dla każdego innego węzła). Po osiągnięciu ustalonej wartości licznik ten jest zerowany, a węzeł zliczający nie będzie próbował łączyć się z "opóźnionym" węzłem przez pewien określony czas.
- Jeśli węzeł niespodziewanie zakończy połączenie TCP i przestanie rozgłaszać swoją tablicę, to po pewnym czasie pozostałe węzły uznają to za opuszczenie przez niego sieci.
- W przypadku gdy pojawi się błąd w trakcie transferu TCP, usuwamy pobrane dane (segmenty) i kończymy wątek pobierający. Ponowne pobieranie od tego węzła będzie odbywać się po ponownym połączeniu z węzłem.

Chapter 2

Opis Funkcjonalny Projektu

Użytkownik systemu ma wgląd w dwie listy

- lokalny rejestr zasobów - pliki, które użytkownik dodał lub pobrał od innych,
- pliki obecne w systemie - pliki posiadane w lokalnym rejestrze zasobów przez innych użytkowników, które nie zostały unieważnione.

Dla każdego użytkownika generowana jest, oczywiście, odrębna para list.

Użytkownik może wprowadzać tekstowe komendy, aby za ich pomocą

- *wyświetlić listę lokalnych zasobów,*
- *wyświetlić listę zasobów obecnych w systemie,*
- *wyświetlić listę dostępnych komend,*
- *opuścić system,*

a także wykonywać operacje na plikach, wśród których rozróżniamy:

- *dodanie pliku,*
  można dodać do zasobów plik, którego nazwa nie wystąpiła jeszcze wśród plików w lokalnym rejestrze zasobów i reszcie sieci.
  Zakładamy, że nie wystąpi sytuacja, gdy więcej niż jeden użytkownik doda plik o tej samej nazwie “jednocześnie” - to znaczy przed “zauważeniem” przez całą sieć dodania pliku o takiej nazwie przez któregokolwiek z nich.
- *usunięcie pliku,*
  można usunąć plik z własnego rejestru zasobów.
- *unieważnienie pliku,*
  można unieważnić plik, który mamy we własnym rejestrze zasobów. Oznacza to, że zasób nie będzie już widoczny na liście plików dostępnych w systemie, ale dotychczasowi posiadacze nadal będą go posiadali w swoim lokalnym systemie plików.
- *pobranie pliku,*
  można pobrać plik, którego nie mamy jeszcze w rejestrze zasobów, a który jest obecny w systemie.

Chapter 3

Protokół

3.1 Składnia :

```
<Command 1 octet> ::= <REVOKE> | <FILE_LIST> | <REQ_CONN> | <REQ_SEGMENT>

<ResourceHeader> ::=
        <nazwa pliku: 256 octets, NULL terminated> <rozmiar pliku: 64bit>

<Message> ::= <Command> | <Command><ResourceHeader> |
        <Command><ResourceHeader><Options> |
        <Command><No_Of_Files 16bit><ResourceHeader><ResourceHeader>...

<Resource> ::= <1 KB of file>
```

3.2 Komendy :

- *Unieważnienie pliku:*
  Broadcast po UDP:
  `<Command = REVOKE><ResourceHeader = Revoked File>`
- *Rozgłaszanie dostępnych plików:*
  Broadcast po UDP:
  `<Command = FILE_LIST><No_Of_Files = liczba dostępnych plików>`
  `<ResourceHeader = plik1>....`
- *Żądanie utworzenia połączenia TCP:*
  Wysyłane do węzła po TCP: `<Command = REQ_CONN>`
- *Żądanie pobrania segmentu:*
  Wysyłane do węzła po TCP:
  `<Command = REQ_SEGMENT><ResourceHeader = plik><Options = segment number>`

Chapter 4

Organizacja Projektu

4.1 Moduły

1. Moduł CLI odpowiedzialny za komunikację z użytkownikiem.
2. Moduł obsługi sieci.
3. Moduł dispatchera obsługujący protokół.

4.2 Współbieżność

Ogólna koncepcja zakłada istnienie następujących bazowych, działających w pętli wątków:

1. *Obsługa przychodzących żądań przesłania posiadanego zasobu.*
   Wątek ten nasłuchuje na porcie TCP. W przypadku nawiązania połączenia na tym porcie tworzony jest watek potomny. Wątek ten odbiera żądanie przesłania lokalnie posiadanego pliku i nadzoruje to przesłanie.
2. *Odbiór komunikatów broadcast UDP.*
   Komunikaty te obejmują okresowe rozgłaszanie przez każdego użytkownika listy lokalnie posiadanych zasobów, rozgłaszanie unieważnienia zasobu. Po odebraniu komunikatu wątek ten przekazuje otrzymane informacje do wątku synchronizującego dane.
3. *Okresowe rozgłaszanie lokalnej listy zasobów.*
   Wątek, który co pewien czas rozgłasza przez UDP listę zasobów, które udostępnia do pobrania. Wątek blokuje się pomiędzy kolejnymi broadcastami.
4. *Wątek synchronizujący dane.*
   Wątek manipulujący danymi przechowywanymi przez program takimi jak: lista lokalnych zasobów, informacje o pozostałych węzłach - dla każdego z nich lista zasobów, które udostępnia, liczniki opóźnień itp.
5. *Obsługa interfejsu użytkownika.*
   Interakcja z użytkownikiem przez CLI. Odbieranie komend od użytkownika i odpowiednie reagowanie - powoływanie nowych wątków, które mają zająć się realizacją komendy, między innymi:
   - W przypadku chęci pobrania pliku tworzony jest wątek nadzorujący to pobieranie. Na potrzeby połączenia z wieloma węzłami może on stworzyć kilka kolejnych wątków przypisanych do połączeń z węzłami.

W zależności od wyniku pobierania przekaże odpowiednie informacje do wątku synchronizującego dane.

- W przypadku chęci dodania, usunięcia lub unieważnienia pliku tworzony jest wątek, który zajmie się wprowadzeniem tej zmiany i nadzorowaniem wszystkich jej następstw, np.: fizycznie doda plik do systemu i przekaże informację o nowym pliku do wątku synchronizującego dane. W przypadku unieważnienia stworzy nowy wątek, który rozgłosi odpowiednią informację w systemie.
- W przypadku chęci wyświetlenia którejś z list zasobów lub listy dostępnych komend tworzony jest wątek, który odczyta odpowiednie dane i przygotuje je w odpowiedniej formie do wyświetlenia użytkownikowi.

Wyniki działań powyższych wątków przekazywane są z powrotem do wątku obsługującego CLI, który wyświetla je użytkownikowi.

Chapter 5

Implementacja

- Jezyki : *C++*
- Biblioteki : *Boost:Asio, std::thread*

Part II

Dokumentacja kodu źródłowego - *English*

Description: TODO

Build: Project Uses CMake based config. 'create_configs.sh' shell script will generate ninja files for compilation. Boost and thearding libaries are required

Chapter 6

Class Index

6.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 7

Class Documentation

7.1 simpleP2P::CLI Class Reference

Public Member Functions

- **CLI** (Resource_Database &res_db_, Logging_Module &Logger_, boost::asio::io_service &io_service_, FileManager &fm_, Host &localhost_, Printer &printer_)
- std::thread **init** ()

The documentation for this class was generated from the following files:

- include/CLI.h
- src/CLI.cpp

7.2 simpleP2P::CLICommand Class Reference

Public Member Functions

- **CLICommand** (std::string, std::string, std::function< Int32(const std::string &)>)
- void **operator()** (std::string) const
- std::string **getName** () const
- std::string **getDesc** () const

The documentation for this class was generated from the following files:

- include/CLICommand.h
- src/CLICommand.cpp

7.3 simpleP2P::CLIResourceService Class Reference

Public Member Functions

- void **add** (std::string resource)
- void **remove** (std::string resource)

- void **revoke** (std::string resource)
- void **get_local** ()
- void **get_remote** ()

The documentation for this class was generated from the following files:

- include/CLIResourceService.h
- src/CLIResourceService.cpp

7.4 simpleP2P::CompleteResource Class Reference

Class representing a complete resource (resource and full its data)

`#include <CompleteResource.h>`

Public Member Functions

- CompleteResource (std::shared_ptr< Resource > resource_c)

  Construct a new Complete Resource object based on base Resource object.
- std::shared_ptr< Resource > get_resource () const

  Get the underlaying resource object.
- Segment get_segment ()

  Synchronised method returning Segment object representing the first unbusy and incomplete segment.
- void set_segment (Segment &segment)

  Synchronised method marking the given segment as completed.
- bool is_completed ()

  Synchronised method returning true if all segments have been downloaded, false otherwise.
- void unset_busy (SegmentId id)

  Synchronised method unmarking busy segment and notifying one of waithing worker threads.

7.4.1 Detailed Description

Class representing a complete resource (resource and full its data)

7.4.2 Constructor & Destructor Documentation

7.4.2.1 CompleteResource()

```
simpleP2P::CompleteResource::CompleteResource (
            std::shared_ptr< Resource > resource_c )
```

Construct a new Complete Resource object based on base Resource object.

Parameters

resource_c	base resource object

7.4.3 Member Function Documentation

7.4.3.1 get_resource()

```
std::shared_ptr< Resource > simpleP2P::CompleteResource::get_resource ( ) const
```

Get the underlaying resource object.

Returns

std::shared_ptr<Resource>

7.4.3.2 get_segment()

```
Segment simpleP2P::CompleteResource::get_segment ( )
```

Synchronised method returning Segment object representing the first unbusy and incomplete segment.

If resource downlaoding has been completed, special Segment object with id set to NO_SEGMENT_ID is returned. If there is no appropriate segment found and downloading is not completed, method suspends calling worker thread pending some segment release.

Returns

Segment

7.4.3.3 is_completed()

```
bool simpleP2P::CompleteResource::is_completed ( )
```

Synchronised method returning true if all segments have been downloaded, false otherwise.

Returns

true if downloading is completed
false otherwise

7.4.3.4 set_segment()

```
void simpleP2P::CompleteResource::set_segment (
            Segment & segment )
```

Synchronised method marking the given segment as completed.

If all segments have been downloaded, method notifies all waiting worker threads so that they can join.

Parameters

segment	

7.4.3.5 unset_busy()

```
void simpleP2P::CompleteResource::unset_busy (
            SegmentId id )
```

Synchronised method unmarking busy segment and notifying one of waithing worker threads.

Parameters

id	

The documentation for this class was generated from the following files:

- include/CompleteResource.h
- src/CompleteResource.cpp

7.5 simpleP2P::DownloadService Class Reference

Used to download resource.

`#include <DownloadService.h>`

Public Member Functions

- DownloadService (Logging_Module &logging_module_c, boost::asio::io_service &io_service_↩ c, FileManager &file_manager_c, Resource_Database &resource_database_c, std::shared_ptr< Resource > resource_c)

  Construct a new Download Service object.

- void init ()

  Method initiating downloading in the current thread.

- std::thread init_thread ()

  Method initiating downloading in a new thread.

7.5.1 Detailed Description

Used to download resource.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 DownloadService()

```
simpleP2P::DownloadService::DownloadService (
            Logging_Module & logging_module_c,
            boost::asio::io_service & io_service_c,
            FileManager & file_manager_c,
            Resource_Database & resource_database_c,
            std::shared_ptr< Resource > resource_c )
```

Construct a new Download Service object.

Parameters

logging_module_c	
io_service_c	
file_manager_c	
resource_database_c	
resource_c	

7.5.3 Member Function Documentation

7.5.3.1 init_thread()

```
std::thread simpleP2P::DownloadService::init_thread ( )
```

Method initiating downloading in a new thread.

Returns

std::thread

The documentation for this class was generated from the following files:

- include/DownloadService.h
- src/DownloadService.cpp

7.6 simpleP2P::DownloadWorker Class Reference

Class used to download resource segments from one host.

`#include <DownloadWorker.h>`

Public Member Functions

- DownloadWorker (Logging_Module &logging_module_c, boost::asio::io_service &io_service_c, std::shared_ptr< Host > host_c, std::shared_ptr< CompleteResource > complete_resource_c)

  Construct a new Download Worker object.
- std::thread init ()

  Method initiating connection to host and downloading resource segments in a new thread.
- void check_timeout ()

  Method checking worker timeout and taking appropriate actions if it exeeds:
- void close ()

  Synchronised method closing worker politely.
- bool is_closed ()

  Synchronised method checking if worker is closed.

7.6.1 Detailed Description

Class used to download resource segments from one host.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 DownloadWorker()

```
simpleP2P::DownloadWorker::DownloadWorker (
            Logging_Module & logging_module_c,
            boost::asio::io_service & io_service_c,
            std::shared_ptr< Host > host_c,
            std::shared_ptr< CompleteResource > complete_resource_c )
```

Construct a new Download Worker object.

Parameters

logging_module_c	
io_service_c	
host_c	
complete_resource_c	

7.6.3 Member Function Documentation

7.6.3.1 check_timeout()

```
void simpleP2P::DownloadWorker::check_timeout ( )
```

Method checking worker timeout and taking appropriate actions if it exeeds:

- the host timeout counter is increased, what may cause banning host for certain time,
- currently downloaded segment is marked as unbusy and may be downloaded by another worker.

7.6.3.2 init()

```
std::thread simpleP2P::DownloadWorker::init ( )
```

Method initiating connection to host and downloading resource segments in a new thread.

Returns

std::thread

The documentation for this class was generated from the following files:

- include/DownloadWorker.h
- src/DownloadWorker.cpp

7.7 simpleP2P::FileManager Class Reference

Handles read/write to the files on disc.

```
#include <FileManager.h>
```

Public Member Functions

- void get_file (FileRequest request, char *result, std::size_t size)

  Buffers specificated segments of the specificated file in the char array.
- void store_resource (CompleteResource &resource)

  Stores the file contents in the physical file on disc.

7.7.1 Detailed Description

Handles read/write to the files on disc.

An API which provides:

- buffering contents of requested segments of a specificated local file,

- storing a complete, downloaded file physically on the local disc. Ensures synchronization of those operations.

7.7.2 Member Function Documentation

7.7.2.1 get_file()

```
void simpleP2P::FileManager::get_file (
            FileRequest request,
            char * result,
            std::size_t size )
```

Buffers specificated segments of the specificated file in the char array.

(!) All segments will be returned concatenated in a single char array, provided in the 'result' parameter. They will be put to the array in the order as provided in the 'request' param. Keep this in mind if you requested the last segment of the file, size of which may vary.

Parameters

request	Specifies file and its segments to buffer.
result	The array to buffer the file contents in.
size	Size of the char array.

7.7.2.2 store_resource()

```
void simpleP2P::FileManager::store_resource (
            CompleteResource & resource )
```

Stores the file contents in the physical file on disc.

Parameters

resource	File to store on the disc. The data will not be interpreted, so make sure it's complete and ready to store.

The documentation for this class was generated from the following files:

- include/FileManager.h
- src/FileManager.cpp

7.8 simpleP2P::FileRequest Class Reference

Carries info about a single file transfer request - resource header and numbers of wanted segments.

`#include <FileRequest.h>`

Public Member Functions

- FileRequest (std::vector< Int8 > rh, std::initializer_list< Uint32 > s)

  Constructor allows specificating the resource and segments.
- std::vector< Int8 > get_resource_header () const

  Get the resource header.
- std::vector< Uint32 > get_segments () const

  Get the segments' numbers.

7.8.1 Detailed Description

Carries info about a single file transfer request - resource header and numbers of wanted segments.

An instance of this class is created by TCP client, sent to TCP server, which passes it to the FileManager in order to get the requested segments of the requested file and send them to the TCP client.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 FileRequest()

```
simpleP2P::FileRequest::FileRequest (
            std::vector< Int8 > rh,
            std::initializer_list< Uint32 > s )
```

Constructor allows specificating the resource and segments.

The fields then can't be modified, only get.

The documentation for this class was generated from the following files:

- include/FileRequest.h
- src/FileRequest.cpp

7.9 simpleP2P::Host Class Reference

Class contains node information and points to files it possess.

`#include <host.h>`

Public Member Functions

- Host (boost::asio::ip::address ip)

  Constructor.
- bool has_resource (Resource res)

  Determines if host has resource.
- bool operator== (const Host &other) const

  Operator == checks host_ip for equality.
- bool operator!= (const Host &other) const

  Operator != checks host_ip for equality.
- boost::asio::ip::tcp::endpoint get_endpoint () const

  Method returning boost tcp endpoint (ip address and port) of the host.
- bool is_retarded ()

  Method returning true if program considers a given host to be retarded.
- void increase_timeout_counter ()

  Method increasing a host timeout counter.
- std::chrono::system_clock::time_point get_ban_time_point () const

  Method returning a time point to which a host is considered to be retarded.
- const std::vector< std::weak_ptr< Resource > > & **get_possesed** () const

Friends

- class Resource_Database

  friendship to manage Host's Resources timeouts etc
- class **Udp_Server**

7.9.1 Detailed Description

Class contains node information and points to files it possess.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 Host()

```
simpleP2P::Host::Host (
            boost::asio::ip::address ip )
```

Constructor.

Parameters

ip	Ip of the Host

7.9.3 Member Function Documentation

7.9.3.1 get_ban_time_point()

```
std::chrono::system_clock::time_point simpleP2P::Host::get_ban_time_point ( ) const
```

Method returning a time point to which a host is considered to be retarded.

Returns

std::chrono::system_clock::time_point

7.9.3.2 get_endpoint()

```
boost::asio::ip::tcp::endpoint simpleP2P::Host::get_endpoint ( ) const
```

Method returning boost tcp endpoint (ip address and port) of the host.

Returns

boost::asio::ip::tcp::endpoint

7.9.3.3 has_resource()

```
bool simpleP2P::Host::has_resource (
            Resource res )
```

Determines if host has resource.

Parameters

res	Resource to be checked

Returns

true if Host has Resource res

7.9.3.4 increase_timeout_counter()

```
void simpleP2P::Host::increase_timeout_counter ( )
```

Method increasing a host timeout counter.

After exeeding timeout limit, a host is considered to be retarded up to some point in time.

7.9.3.5 is_retarded()

```
bool simpleP2P::Host::is_retarded ( )
```

Method returning true if program considers a given host to be retarded.

Returns

true if a host is retarded
false otherwise

7.9.3.6 operator"!=()

```
bool simpleP2P::Host::operator!= (
            const Host & other ) const
```

Operator != checks host_ip for equality.

Parameters

other	other

Returns

true if not equal

7.9.3.7 operator==()

```
bool simpleP2P::Host::operator== (
            const Host & other ) const
```

Operator == checks host_ip for equality.

Parameters

other	other

Returns

true if equal

The documentation for this class was generated from the following files:

- include/host.h
- src/host.cpp

7.10 simpleP2P::Logging_Module Class Reference

class Providing logging support based on text logs

`#include <logging_module.h>`

Public Member Functions

- **Logging_Module** (std::ostream &output_c=std::cerr)
- std::thread init ()

  Constructor for the logging thread.
- void add_log_line (std::string line, const std::time_t time)

  Synchronised method for logging output.

7.10.1 Detailed Description

class Providing logging support based on text logs

7.10.2 Member Function Documentation

7.10.2.1 add_log_line()

```
void simpleP2P::Logging_Module::add_log_line (
            std::string line,
            const std::time_t time )
```

Synchronised method for logging output.

Parameters

line	

7.10.2.2 init()

```
std::thread simpleP2P::Logging_Module::init ( )
```

Constructor for the logging thread.

Parameters

output_c	Output stream for the logs

Note

if output stream is a file you must explicitly close it

The documentation for this class was generated from the following files:

- include/logging_module.h
- src/logging_module.cpp

7.11 simpleP2P::Printer Class Reference

class printing outputs, using queue in order to avoid races

`#include <printer.h>`

Public Member Functions

- Printer (std::ostream &output_c=std::cout)

  Constructor for the printer.
- std::thread init ()

  Init methods run worker in thread and returns it.
- void print (std::string line)

  Synchronised method for printing output.

7.11.1 Detailed Description

class printing outputs, using queue in order to avoid races

7.11.2 Constructor & Destructor Documentation

7.11.2.1 Printer()

```
simpleP2P::Printer::Printer (
            std::ostream & output_c = std::cout )
```

Constructor for the printer.

Parameters

output_c	Output stream for couts

7.11.3 Member Function Documentation

7.11.3.1 init()

```
std::thread simpleP2P::Printer::init ( )
```

Init methods run worker in thread and returns it.

Returns

printing thread

7.11.3.2 print()

```
void simpleP2P::Printer::print (
            std::string line )
```

Synchronised method for printing output.

Parameters

line	

The documentation for this class was generated from the following files:

- include/printer.h
- src/printer.cpp

7.12 simpleP2P::RequestServer Class Reference

Asynchronous TCP server.

```
#include <RequestServer.h>
```

Public Member Functions

- RequestServer (boost::asio::io_service &io_service, Uint16 port)

  Constructor allows setting the parameters for the connnetion acceptor.
- std::thread init ()

  Turns on the listening and accepting connections and returns the thread in which it works.

7.12.1 Detailed Description

Asynchronous TCP server.

It accepts connections asynchronously and for each of them creates a worker object to handle it.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 RequestServer()

```
simpleP2P::RequestServer::RequestServer (
            boost::asio::io_service & io_service,
            Uint16 port )
```

Constructor allows setting the parameters for the connnetion acceptor.

Parameters

io_service	boost::asio::io_service for the acceptor.
port	Port for the acceptor to listen on.

The documentation for this class was generated from the following files:

- include/RequestServer.h
- src/RequestServer.cpp

7.13 simpleP2P::RequestServerModule Class Reference

Module of the TCP server receiving file requests and sending the requested files' segments.

```
#include <RequestServerModule.h>
```

Public Member Functions

- RequestServerModule (Uint16 port_)

  Constructor, allows setting the port for the server.
- std::thread init ()

  Returns the thread object for the module.

7.13.1 Detailed Description

Module of the TCP server receiving file requests and sending the requested files' segments.

7.13.2 Member Function Documentation

7.13.2.1 init()

```
std::thread simpleP2P::RequestServerModule::init ( )
```

Returns the thread object for the module.

Starts the server and returns the thread in which the server works. The thread of the server.

The documentation for this class was generated from the following files:

- include/RequestServerModule.h
- src/RequestServerModule.cpp

7.14 simpleP2P::RequestWorker Class Reference

TCP connection handler, created by the TCP server.

`#include <RequestWorker.h>`

Inherits enable_shared_from_this< RequestWorker >.

Public Member Functions

- RequestWorker (boost::asio::io_service &io_service)

  Constructor allows setting the socket on which the connection is established.
- void start ()

  Start handling the request.
- tcp::socket & socket ()

  Get socket.

7.14.1 Detailed Description

TCP connection handler, created by the TCP server.

Receives the file request, buffers requested segments and sends them to the client.

The documentation for this class was generated from the following files:

- include/RequestWorker.h
- src/RequestWorker.cpp

7.15 simpleP2P::Resource Class Reference

Forward declaration.

`#include <resource.h>`

Public Member Functions

- Resource (std::string name, Uint64 size, std::string path="./")

  Constructor.
- Resource (std::vector< Uint8 > resource_header)

  Constructor makes resource from header.
- std::vector< Uint8 > generate_resource_header ()

  Generates Resource header.

- bool has_host (Host host)

  Determines if resource is possesed by Host.
- Uint16 calc_segments_count () const

  Calculates and returns segment count.
- void set_revoked ()

  Function used to set invalidated flag.
- bool **isInvalidated** ()
- Uint64 getSize () const

  Getter for file size.
- const std::string & getName () const

  Getter for file name.
- const std::string & getPath () const

  Getter for file path.
- bool operator== (const Resource &other) const

  Operator == checks file size and name for equality.
- bool operator!= (const Resource &other) const

  Operator != checks file size and name for equality.
- const tbb::concurrent_vector< std::weak_ptr< Host > > & **get_hosts** () const

Friends

- class Resource_Database

  friendship to manage Resource Hosts, path etc

7.15.1 Detailed Description

Forward declaration.

Class contains file information and points to nodes with file possesion

7.15.2 Constructor & Destructor Documentation

7.15.2.1 Resource() [1/2]

```
simpleP2P::Resource::Resource (
            std::string name,
            Uint64 size,
            std::string path = "./" )
```

Constructor.

Parameters

name	filename
size	filesize
path	filepath, default is "./"

7.15.2.2 Resource() [2/2]

```
simpleP2P::Resource::Resource (
            std::vector< Uint8 > resource_header )
```

Constructor makes resource from header.

Parameters

resource_header	Resource header

7.15.3 Member Function Documentation

7.15.3.1 calc_segments_count()

```
Uint16 simpleP2P::Resource::calc_segments_count ( ) const  [inline]
```

Calculates and returns segment count.

Returns

segment count

7.15.3.2 generate_resource_header()

```
std::vector< Uint8 > simpleP2P::Resource::generate_resource_header ( )
```

Generates Resource header.

Returns

Resource header

7.15.3.3 getName()

```
const std::string & simpleP2P::Resource::getName ( ) const
```

Getter for file name.

Returns

file name

7.15.3.4 getPath()

```
const std::string & simpleP2P::Resource::getPath ( ) const
```

Getter for file path.

Returns

file path

7.15.3.5 getSize()

```
Uint64 simpleP2P::Resource::getSize ( ) const
```

Getter for file size.

Returns

file size

7.15.3.6 has_host()

```
bool simpleP2P::Resource::has_host (
            simpleP2P::Host host )
```

Determines if resource is possesed by Host.

Parameters

host	Host

Returns

true if resource is possessed by host

7.15.3.7 operator"!=()

```
bool simpleP2P::Resource::operator!= (
            const Resource & other ) const
```

Operator != checks file size and name for equality.

Parameters

other	other

Returns

true if not equal

7.15.3.8 operator==()

```
bool simpleP2P::Resource::operator== (
            const Resource & other ) const
```

Operator == checks file size and name for equality.

Parameters

other	other

Returns

true if equal

7.15.3.9 set_revoked()

```
void simpleP2P::Resource::set_revoked ( )  [inline]
```

Function used to set invalidated flag.

To allow references on resource outside database to gather information about revoke

The documentation for this class was generated from the following files:

- include/resource.h
- src/resource.cpp

7.16 simpleP2P::Resource_Database Class Reference

Class holding information about files in network and on localhost.

```
#include <resource_database.h>
```

Public Member Functions

- Resource_Database (Host localhost)

  Constructor.
- bool has_file (const Resource &res)

  Check if localhost has certain file.
- void add_file (const Resource &res, const Host &host)

  Adds connection between file and resource, adn creates them if they do not exist.

- bool remove_file (const Resource &res, const Host &host)

  Removes connection between file and resource.
- void update_host (const Host &host)

  Updates the list of resources aviable from host Triggered after receive of full Beacon Packet.
- void revoke_resource (const Resource &resource)

  Revokes resource and disconnects it from Hosts in database and database itself Resource will still point to Hosts that possess it.
- void add_file (const Resource &res)

  same as add_file(Resource, Host) but host is localhost
- bool remove_file (const Resource &res)

  same as remove_file(Resource, Host) but host is localhost
- std::shared_ptr< Resource > who_has_file (std::vector< Uint8 > resource_header)

  Returns shared pointer to resource to allow access to information about file owners.
- std::shared_ptr< Resource > who_has_file (const Resource &res)

  Returns shared pointer to resource to allow access to information about file owners.
- std::vector< std::vector< Uint8 > > generate_database_headers ()

  Generates listing of localhost content in a header.
- std::shared_ptr< Host > getHost () const

  Get localhost information.
- const std::vector< std::shared_ptr< Resource > > **getResources** () const

7.16.1 Detailed Description

Class holding information about files in network and on localhost.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 Resource_Database()

```
simpleP2P::Resource_Database::Resource_Database (
            Host localhost )
```

Constructor.

Parameters

localhost	localhost

7.16.3 Member Function Documentation

7.16.3.1 add_file() [1/2]

```
void simpleP2P::Resource_Database::add_file (
            const Resource & res,
            const Host & host )
```

Adds connection between file and resource, adn creates them if they do not exist.

Parameters

res	Resource to be added
host	Host which possess Resource res

7.16.3.2 add_file() [2/2]

```
void simpleP2P::Resource_Database::add_file (
            const Resource & res )
```

same as add_file(Resource, Host) but host is localhost

Parameters

res	Resource to be added

7.16.3.3 generate_database_headers()

```
std::vector< std::vector< Uint8 > > simpleP2P::Resource_Database::generate_database↩
_headers ( )
```

Generates listing of localhost content in a header.

Returns

listing header of localhost resources

7.16.3.4 getHost()

```
std::shared_ptr< Host > simpleP2P::Resource_Database::getHost ( ) const
```

Get localhost information.

Returns

localhost

7.16.3.5 has_file()

```
bool simpleP2P::Resource_Database::has_file (
            const Resource & res )
```

Check if localhost has certain file.

Parameters

res	Resource to be checked

Returns

true if host already has some resource

7.16.3.6 remove_file() [1/2]

```
bool simpleP2P::Resource_Database::remove_file (
            const Resource & res,
            const Host & host )
```

Removes connection between file and resource.

Parameters

res	Resource to be removed from host list
host	Host which resource will be removed

Returns

returns false if file did not existed or was not possesed

7.16.3.7 remove_file() [2/2]

```
bool simpleP2P::Resource_Database::remove_file (
            const Resource & res )
```

same as remove_file(Resource, Host) but host is localhost

Parameters

res	Resource to be removed from localhost list

Returns

returns false if file did not existed or was not possesed

7.16.3.8 revoke_resource()

```
void simpleP2P::Resource_Database::revoke_resource (
            const Resource & resource )
```

Revokes resource and disconnects it from Hosts in database and database itself Resource will still point to Hosts that possess it.

Parameters

resource	Resource to be revoked

7.16.3.9 update_host()

```
void simpleP2P::Resource_Database::update_host (
            const Host & host )
```

Updates the list of resources aviable from host Triggered after receive of full Beacon Packet.

Parameters

host	Host and possesed resources in a struct

7.16.3.10 who_has_file() [1/2]

```
shared_ptr< Resource > simpleP2P::Resource_Database::who_has_file (
            std::vector< Uint8 > resource_header ) [inline]
```

Returns shared pointer to resource to allow access to information about file owners.

Parameters

res	Resource about which information is gathered

Returns

shared pointer to res

7.16.3.11 who_has_file() [2/2]

```
shared_ptr< Resource > simpleP2P::Resource_Database::who_has_file (
            const Resource & res )
```

Returns shared pointer to resource to allow access to information about file owners.

Parameters

res	Resource about which information is gathered

Returns

shared pointer to res

The documentation for this class was generated from the following files:

- include/resource_database.h
- src/resource_database.cpp

7.17 simpleP2P::Segment Class Reference

Class representing a segment of a resource.

`#include <Segment.h>`

Public Member Functions

- Segment (SegmentId id_c, Uint8 *data_c)

  Construct a new Segment object.
- SegmentId get_id () const

  Get the segment id.
- Uint8 * get_data_ptr () const

  Get the pointer to the segment data.
- std::vector< Uint8 > serialize_id ()

Static Public Member Functions

- static Segment no_segment_left ()

  Special method returning Segment object with id set to NO_SEGMENT_ID indicating no segment to download in complete resource object.

Static Public Attributes

- static const SegmentId NO_SEGMENT_ID = static_cast<SegmentId>(-1)

  Special id number indicating no segment to download in complete resource object.

7.17.1 Detailed Description

Class representing a segment of a resource.

It contains an id of the segment and a pointer to the segment data in complete resource object.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 Segment()

```
simpleP2P::Segment::Segment (
            SegmentId id_c,
            Uint8 * data_c )
```

Construct a new Segment object.

Parameters

id_c	segment id
data_c	pointer to the segment data in complete resource object

7.17.3 Member Function Documentation

7.17.3.1 get_data_ptr()

```
Uint8 * simpleP2P::Segment::get_data_ptr ( ) const
```

Get the pointer to the segment data.

Returns

Uint8*

7.17.3.2 get_id()

```
SegmentId simpleP2P::Segment::get_id ( ) const
```

Get the segment id.

Returns

SegmentId

7.17.3.3 no_segment_left()

```
Segment simpleP2P::Segment::no_segment_left ( )  [static]
```

Special method returning Segment object with id set to NO_SEGMENT_ID indicating no segment to download in complete resource object.

Returns

Segment

7.17.3.4 serialize_id()

```
std::vector<Uint8> simpleP2P::Segment::serialize_id ( )
```

Returns

std::vector<Uint8>

The documentation for this class was generated from the following files:

- include/Segment.h
- src/Segment.cpp

7.18 simpleP2P::UDP_Beacon Class Reference

Public Member Functions

- **UDP_Beacon** (boost::asio::ip::udp::socket socket_c)
- std::thread **init** ()
- void revoke_file (std::string resource)

  to sent aproperiate revoke header TODO: construct resource struct and create command in place

The documentation for this class was generated from the following files:

- include/udp_beacon.h
- src/udp_beacon.cpp

7.19 simpleP2P::Udp_Client Class Reference

class UDP Client to handle all outgoing packets

```
#include <udp_client.h>
```

Inherits enable_shared_from_this< Udp_Client >.

Public Member Functions

- Udp_Client (boost::asio::io_service &io_service, Resource_Database &database, Logging_Module &logger, const boost::asio::ip::address &broadcast_address, Uint16 broadcast_port, Uint32 timeout=5 ∗60)

  Constructor of UDP Client.

- ∼Udp_Client ()

  Destructor closes socket.

- void revoke_file (Resource resource)

  Constructs revoke header sends it.

7.19.1 Detailed Description

class UDP Client to handle all outgoing packets

7.19.2 Constructor & Destructor Documentation

7.19.2.1 Udp_Client()

```
simpleP2P::Udp_Client::Udp_Client (
            boost::asio::io_service & io_service,
            Resource_Database & database,
            Logging_Module & logger,
            const boost::asio::ip::address & broadcast_address,
            Uint16 broadcast_port,
            Uint32 timeout = 5 * 60 )
```

Constructor of UDP Client.

Parameters

io_service	asio Io Service
database	Database
logger	Logger
broadcast_address	address on which packets will be sent
broadcast_port	port on which packets will be sent
timeout	beacon interval

7.19.3 Member Function Documentation

7.19.3.1 revoke_file()

```
void simpleP2P::Udp_Client::revoke_file (
            simpleP2P::Resource resource )
```

Constructs revoke header sends it.

Parameters

resource	Resource to be revoked

The documentation for this class was generated from the following files:

- include/udp_client.h
- src/udp_client.cpp

7.20 simpleP2P::UDP_Listener Class Reference

Public Member Functions

- **UDP_Listener** (boost::asio::ip::udp::socket socket_c)
- std::thread **init** ()

The documentation for this class was generated from the following files:

- include/udp_listener.h
- src/udp_listener.cpp

7.21 simpleP2P::Udp_Module Class Reference

Class containing all UDP related resources and logic.

`#include <udp_module.h>`

Public Member Functions

- Udp_Module (Resource_Database &database_c, Logging_Module &logger_c, boost::asio::ip::address broadcast_address, Uint16 port, Uint32 beacon_interval)

  Constructor.
- std::thread init ()

  Init methods run worker in thread and returns it.
- void revoke_file (const Resource &resource)

  Sends revoke datagram.

7.21.1 Detailed Description

Class containing all UDP related resources and logic.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 Udp_Module()

```
simpleP2P::Udp_Module::Udp_Module (
            Resource_Database & database_c,
            Logging_Module & logger_c,
            boost::asio::ip::address broadcast_address,
            Uint16 port,
            Uint32 beacon_interval )
```

Constructor.

Parameters

broadcast_address	address on which packets will be sent
port	port on which packets will be sent
beacon_interval	beacon interval

7.21.3 Member Function Documentation

7.21.3.1 init()

```
std::thread simpleP2P::Udp_Module::init ( )
```

Init methods run worker in thread and returns it.

Returns

logging thread

7.21.3.2 revoke_file()

```
void simpleP2P::Udp_Module::revoke_file (
            const Resource & resource )
```

Sends revoke datagram.

Parameters

resource	Resource t obe revoked

The documentation for this class was generated from the following files:

- include/udp_module.h

- src/udp_module.cpp

7.22 simpleP2P::Udp_Server Class Reference

class UDP Server to handle all incoming packets

`#include <udp_server.h>`

Inherits enable_shared_from_this< Udp_Server >.

Public Member Functions

- Udp_Server (boost::asio::io_service &io_service, Resource_Database &database, Logging_Module &logger, const boost::asio::ip::address &broadcast_address, Uint16 broadcast_port)

  Constructor of UDP Server.
- ~Udp_Server ()

  Destructor closes socket.

7.22.1 Detailed Description

class UDP Server to handle all incoming packets

7.22.2 Constructor & Destructor Documentation

7.22.2.1 Udp_Server()

```
simpleP2P::Udp_Server::Udp_Server (
            boost::asio::io_service & io_service,
            Resource_Database & database,
            Logging_Module & logger,
            const boost::asio::ip::address & broadcast_address,
            Uint16 broadcast_port )
```

Constructor of UDP Server.

Parameters

io_service	asio Io Service
database	Database
logger	Logger
broadcast_address	address on which Server will listen
broadcast_port	port on which Server will listen

The documentation for this class was generated from the following files:

- include/udp_server.h
- src/udp_server.cpp