Funky Torrents

©The Group

Software Architecture Description

Version 1.0
IT University of Göteborg
TIG029 – Software Architecture for Distributed Systems

1. Revision history

Version no.	Updated	Description	Name
0.1	'2010-10-18'	First Draft	M.Bergqvist, B.Eriksson, A.Issa, S.Aydin
0.2	'2010-10-27'	Logical View updated after discussions	M.Bergqvist, B.Eriksson, S.Aydin
0.3	'2010-11-04'	Use cases updated	M.Bergqvist, B.Eriksson
0.4	'2010-11-06'	Use cases updated again	M.Bergqvist, B.Eriksson
0.5	'2010-11-07'	Minor updates to logical view	M.Bergqvist
0.6	‘2010-11-27’	Updated logical view	M.Bergqvist
0.7	‘2010-12-09’	Updated Scenario View	M.Bergqvist
0.8	‘2010-12-16’	Updated Logical View	M.Bergqvist
1.0	‘2010-12-21’	Last updates to conform to delivered prototype	M.Bergqvist

Table 1. Revision history

2. Background and Scope:

Funky Torrents is a system ordered by the local library. Its purpose is to take burden off from the library's own web server when it comes to distributing electronic publications such as e-books and audio books. Funky Torrents will help in distributing these using peer to peer technologies, letting the library customers take part in the distribution. Funky Torrents main parts are developed using Erlang programming language.

Funky Torrents is developed by The Group, a young company whose employees are heavily addicted to funky rhythms and groovy beats. But even though the music interest has its' roots in the 70's the technology points in to the future.

3. About the SAD document:

For the Software Architecture Description we have looked at, and based our model on, Kruchten's 4+1 view model. Though this is a small project during a short period of time, this SAD will not follow it exactly, just be used as a source of inspiration. The views selected are, in order:

- Scenario View
- Logical View
- Process View

4. Stakeholders and their QA concerns

Users: Library Customers and Library Staff
QA concerns: Usability, Reliability, Scalability
Acquirer: The Local Library owned by the Municipal, represented by Hans Svensson
QA concerns: Reliability, Lifetime, Scalability
Developers: The Group
QA concerns: Testability, Maintainability, Extensibility
Maintainers: The Local Library IT department, represented by Ulrik Eklund
QA concerns: Maintainability

5. Glossary:

- Electronic publication - A publication in electronic format such as a .pdf file or a sound-file for example.
- torrent file - A file containing information about an electronic publication that is shared via peer to peer technology.
- Piece - A piece of an electronic publication file, size depends on the .torrent-file's specifications
- Chunk - The smallest part of an electronic publication file. A piece consists of several chunks. It is the chunks that gets downloaded one by one from another peer and then put together to a full electronic publication file.
- Peer - Another library customer sharing electronic publications via a bittorrent software (not the Funky Torrents bittorrent software client)
- Tracker - A server that keeps track of available peers and what electronic publications they have to share.
- QA - Quality Attribute(s)

6. Scenario View:

Intended for Users, Acquirer, Developers and Maintainers

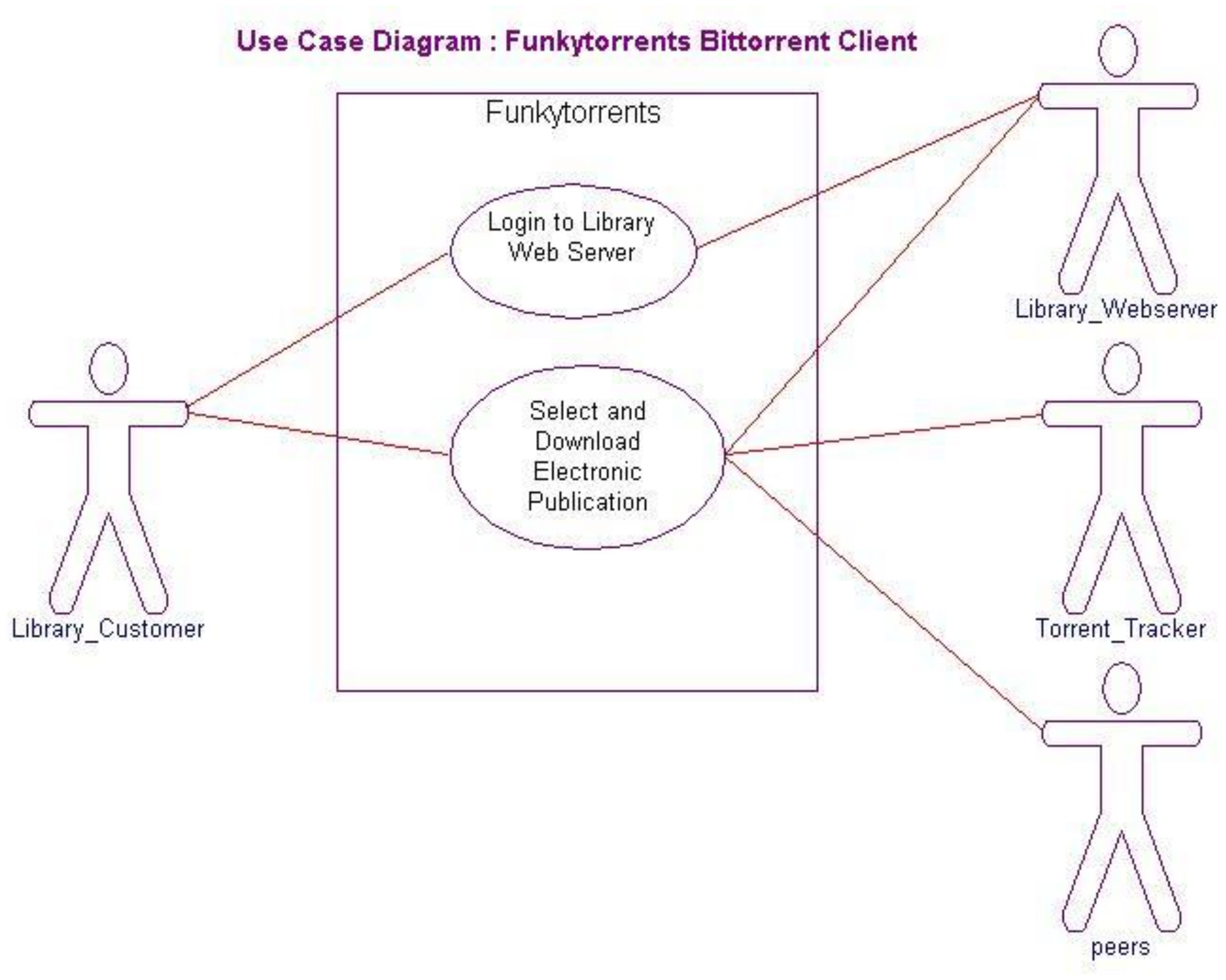

Use Case Diagram : Funkytorrents Bittorrent Client
Funkytorrents
Login to Library Web Server
Select and Download Electronic Publication
Library_Customer
Library_Webserver
Torrent_Tracker
peers

Figure 1. Use case diagram of Funky Torrents

When reading this part of the SAD it is important to keep in mind how Funky Torrents is defined. The system as a whole consists of a Library Web Server, a Torrent Tracker and the Funky Torrents software itself. Funky Torrents is the main part that The Group focus on to develop. The GUI is web based and is thereby displayed in an web browser to the Library Customer.

6.1.UC1 - Login to Library Web Server

Level: User Goal

Primary Actor: Library Customer

Stakeholders and Interests:

1. **Library Customer:** Wants to search for available electronic publications that the local library supplies.
2. **Library:** Wants Library customer to log in to web server to make sure that any unauthorized people do not access the system. Wants to make electronic publications available to the library customers via a web page hosted on a web server.

Preconditions:

Library customer has a computer with an Internet connection. Funky Torrents, Erlang R14B and a web browser are installed on the computer. Library customer has started Funky Torrents and the UI appears in web browser.

Post conditions:

Web browser shows Library starting web page where Library customer can see available electronic publications.

Basic Flow:

1. Library customer selects login option from web page.
2. Web browser shows library login web page in browser.
3. Library customer enters login details.
4. Web server controls login details. If correct, web browser is redirected to library web page where customer can select among available electronic publications.

Use case ends.

Alternative Flow:

A1. Library customer enters faulty login details

Starts at Basic Flow 3

1. Library customer enters faulty login details
2. Web server detects faulty login and presents info message via Library customer's browser.

Continues from Basic Flow step 3

Scenarios for UC1

S1: Basic Flow: 1, 2, 3, 4

S2: Basic Flow: 1, 2
Alternative Flow A1: 1, 2
Basic Flow: 3, 4

6.2.UC2 – Search and download electronic publication

Level: User Goal

Primary Actor: Library Customer

Stakeholders and Interests:

- **Library Customer:** Wants to browse through the available electronic publications and download items of interest.
- **Library:** Wants customers to download electronic publications via peer to peer technology.

Preconditions:

Library customer has completed "UC1 – Login to Library Web Server" basic flow.

Post conditions:

Library customer has found an interesting electronic publication and downloaded it with Funky Torrents. Funky Torrents is still running.

Basic Flow:

1. Library customer browses library web page to find interesting electronic publications and selects wanted publication.
2. Funky Torrents connects to available peers and starts to download electronic publication from those.
3. When download is complete an info message is shown.
4. Library customer acknowledges message and lets Funky Torrents run.
5. Use case ends.

A1. Library customer cancels download.

Start after Basic Flow 2

1. Library customer selects to abort download session in Funky Torrents.

Continues at Basic Flow 1 or 5

A2. Download gets interrupted for some external reason e.g. loss of connection.

Starts after Basic Flow 2

1. Funky Torrents stops downloading because of external reasons.
2. Library customer makes sure that external problem is resolved.

Continues from Basic Flow 1.

Scenarios for UC2

S1: Basic Flow: 1, 2, 3, 4, 5

S2: Basic Flow: 1, 2

Alternative Flow A1: 1

Basic Flow: 1, 2, 3, 4, 5

S3: Basic Flow: 1, 2

Alternative Flow A1: 1

Basic Flow: 5

S4: Basic Flow: 1, 2

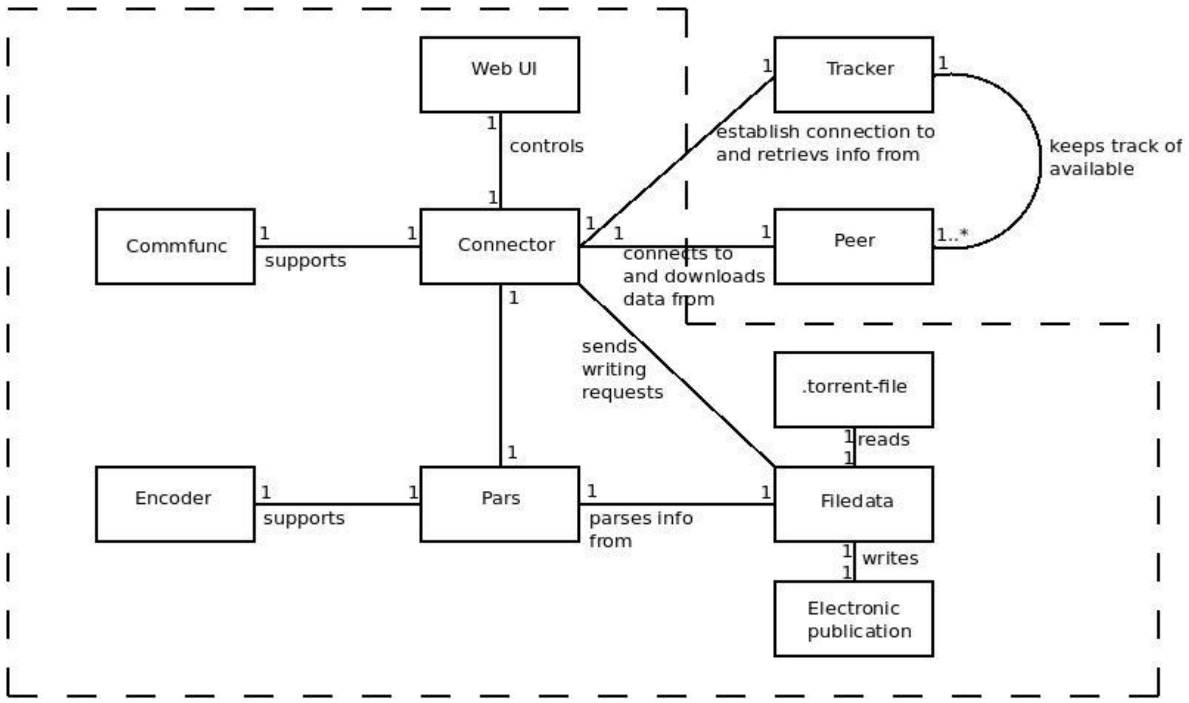
Alternative Flow A2: 1, 2

Basic Flow: 1, 2, 3, 4, 5

7. Logical View:

Intended for Developers, Maintainers and Acquirer

In the diagram below it is easy to get an overview of how the Funky Torrents bittorrent client works. Everything starts with the **Web UI** from where the library customer selects an electronic publication to download. This request is sent to the **Connector** module which in turn asks the **Pars** module to extract information from the electronic publication's .torrent-file. This is done via the file reading & writing module **Filedata**. Once the information has been retrieved the **Connector** can send a request to the right **Tracker** asking for available **Peers**. When a list of **Peers** has been received, the **Connector** module can connect to the first available **Peer** which holds a copy of the desired electronic publication. All communication with the **Peer** is handled with support of pattern matching functions in the **Commfunc** module. As soon as a chunk of a file has been downloaded it is passed on to the **Filedata** module which writes it to the local hard drive. Connector is the module holding the absolute majority of the logic in the Funky Torrents bittorrent client.

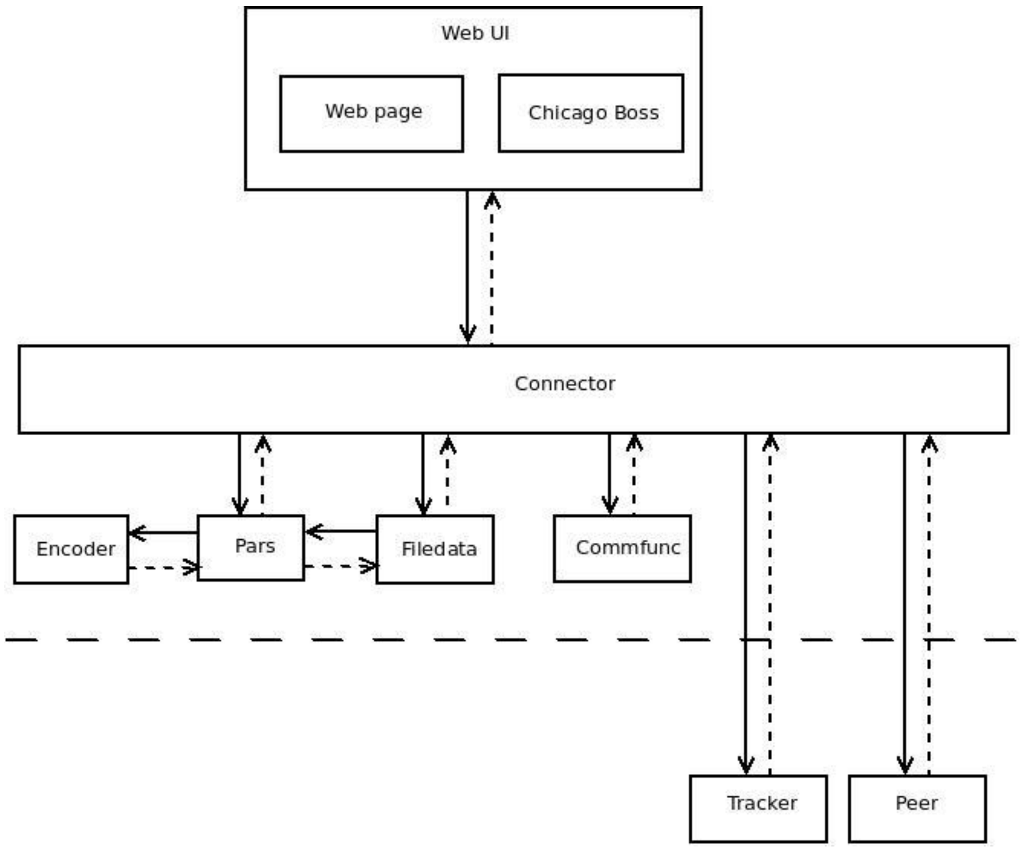


- - - - - - - = system boundaries, everything inside box is running on the same computer.

________ = connections between different entities

numbers denotes the number of instances of each entity

It can be seen in the diagram that we have strived for a good Testability and Maintainability. The supporting modules makes it easy to test different functions separately. Most of the modules inside Funkytorrents have a single interface except for Connector and Filedata. The Filedata module should not be too hard to make more independent in future releases. Also see the next diagram.



———————> = message

- - - - - > = reply

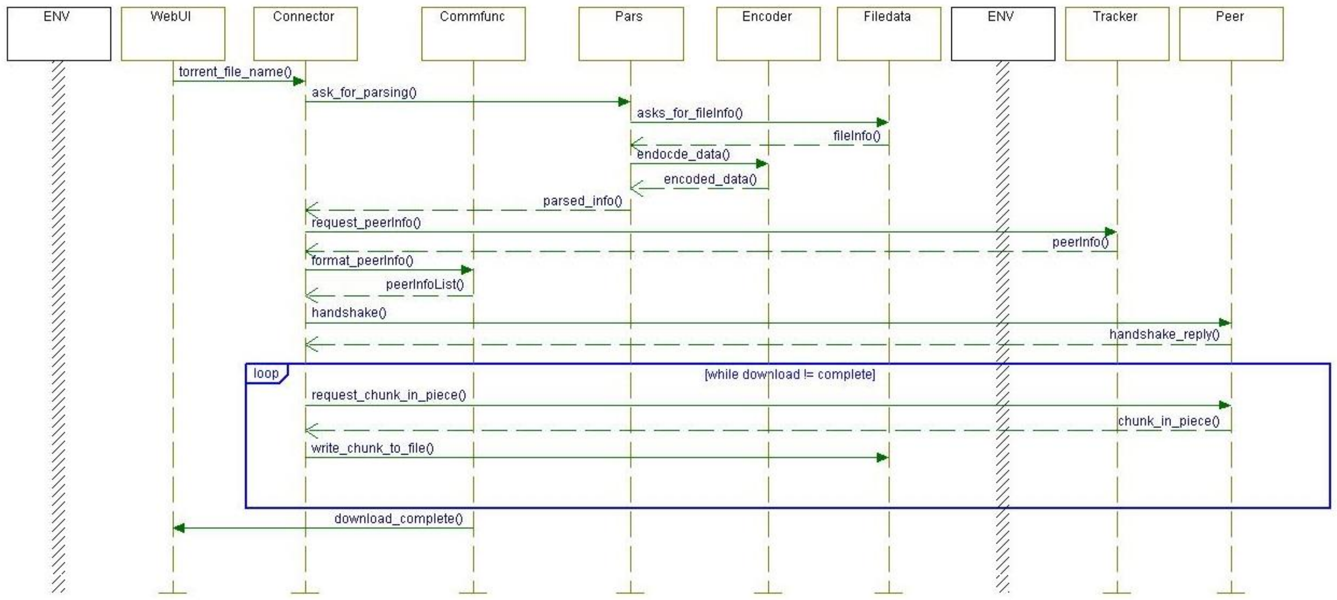
The overall architecture of Funky Torrents is layered in n-tier style to emphasize maintainability and usability. It makes it easier to go in and maintain or rebuild the software layer wise. At the moment it is a 2-tier architecture, but as the Web UI with its webpage interface and its supporting framework, Chicago Boss, is supposed to be located at a web server the architecture can easily be modified to suite this by just adding a dashed horizontal line between Web UI and Connector to create a 3-tier diagram, but thats more of a physical view question.

For future updates, to be able to get a distributed download, the peerconnecting part of Connector should be organized in a master-slave style and the filewriting part of filedata should be implemented with an message due to handle the calls from these slaves.

8. Process View

Intended for Developers and Maintainers

In the following sequence diagram it is possible to see how the different modules inside the system communicates. It does not show anything else than the communicating parts of the system, so entities like .torrent-files and electronic publication files has been left out.



> = **message**

- - - - - > = **reply**

ENV with striped lines under defines the environmental limits for Funky Torrents Bittorrent Client

The above diagram can be used during implementation of the software and during maintenance of the same to be able to follow the data flow and communication within the software as well as with outside.

9. Summary

Architectural trade-offs

Getting the prototype that exists today to work the master slave architecture had to be postponed to later implementations. Instead things are handled in a more sequential way. (See the sequence diagram in process view.) This means that the Users and the Acquirer, as stakeholders, unfortunately can not be met when it comes to their wish of scalability and reliability. This has to wait at the moment.