

Distributed Downloading mechanism for Firefox browser using Bit-torrent concept

SOFTWARE DESIGN SPECIFICATION

Prepared on:

09/03/2006

Prepared by:

Akhil Jayaraj

Arun George Mathew

Cherian Thomas

Jitesh Nambiar

Contents

1. Introduction.
 - 1.1 Purpose of Requirements Document
 - 1.2 Scope of development project
 - 1.3 Definitions, Acronyms, and Abbreviations
 - 1.4 References
 - 1.5 Document overview
2. System architecture description
 - 2.1 Overview of modules / components
 - 2.2 Structure and relationships
 - 2.3 User interface issues
3. Detailed description of components
 - 3.1 Client Module
 - 3.1.1 Browser Interaction Module
 - 3.1.2 Execution Module
 - 3.2 Tracker Module
4. Reuse and relationships to other products
5. Design decisions and tradeoffs
6. Pseudocode for components
7. Appendices
 - 7.1 Usecase Diagram
 - 7.2 Activity Diagram
 - 7.3 Sequence Diagram
 - 7.3.1 Client Sequence Diagram
 - 7.3.2 Peer Sequence Diagram

1. Introduction

1.1 Purpose of Requirements Document

The goal of our project – Distributed Downloading mechanism for Firefox browser using Bit-torrent concept is to build an architecture extension for firefox such that effective download speed for an individual accessing a server is proportional to the number of users. The extension is meant for firefox users using public static domains in the World Wide Web. This document will ensure that all specification mentioned in it are dealt with and verified.

1.2 Scope of development project

The scope of this project extends to all users connecting to the World Wide Web using the Mozilla Firefox browser. This extension software that we are developing can be easily downloaded from the internet and installed with the Mozilla Firefox browser. It is being designed to be used only for HTTP protocol hence will not interfere in the working of other protocols. In case it encounters FTP or any other kind of download mechanism, it will not take any action and the default proceedings of the Mozilla Firefox browser will come into effect.

1.3 Definition, Acronyms, Abbreviations

1. P2P system

A peer-to-peer (P2P) network is one where each workstation has both server and client capabilities and users can initiate communication between any two (or more) computers. P2P is an alternative to the traditional client-server model of networking, and is especially handy for trading files across the Internet.

2. Client

A software program, that is used to contact and obtain data from another computer, often across a great distance.

3. GUI

A GUI (pronounced "goeey") is a graphics-based interface that lets you access programs by pointing to icons, buttons, and windows rather than by typing a string of commands at a command prompt.

4. Swarm

Together, all users connected to a server is called a *swarm*. Six *peers* and two *seeds* make a *swarm* of eight.

5. Peer

A *peer* is one instance of a p2p client running on a computer (browser) on the Internet that connect to and transfer data.

6. Seed

A *seed* is a *peer* that has a complete copy of the page and still offers it for upload. The more *seeds* there are, the better the chances are for completion of the file.

7. Leech

A *leech* is usually a *peer* who has a negative effect on the swarm by having a very poor share ratio - in other words, downloading much more than they upload.

8. Tracker

A *tracker* is a server that keeps track of which seeds and peers are in the swarm. Clients report information to the tracker periodically and in exchange receive information about other clients that they can connect to. The tracker is not directly involved in the data transfer and does not have a copy of the file.

9. Availability

(*also distributed copies*) The number of full copies of the file available to the client. Each *seed* adds 1.0 to this number, as they have one complete copy of the file. A connected peer with a fraction of the file available adds that fraction to the availability (i.e. a peer with 65.3% of the file downloaded increases the availability by 0.653).

10. Interested

Describes a downloader who wishes to obtain pieces of a file the client has. For example, the uploading client would flag a downloading client as 'interested' if that client did not possess a piece that it did, and wished to obtain it.

11. Snubbed

An uploading client is flagged as *snubbed* if the downloading client has not received any data from it in over 60 seconds.

1.4 References:

Internet:

<http://www.bittorrent.com/bittorrentecon.pdf>

http://en.wikipedia.org/wiki/Bit_torrent

<http://www.bittorrent.com/protocol.html>

<http://www.mozilla.org/developer/>

http://groups.csail.mit.edu/uid/chickenfoot/wiki/index.php?title=Firefox_Development

ment

Books referred:

BitTorrent For Dummies by Susannah Gardner and Kris Krug

Firefox by Dimitra Arliss, Bernard Behrens, Warren Clarke, et al

The lists of references are subject to change

1.5 Document Overview

The purpose of this document is to write down formally the requirements considered to be necessary for building the for the Mozilla firefox extension. The first section provides a brief idea about the working of the project along with various acronyms, definitions, abbreviations, and reference materials. Section 2 provides an overview of the system, and a brief description of all the system functions.

2. System architecture description

2.1 Overview of modules / components

The extension can be broadly divided into two modules:

1. Tracker Module
2. Extension Module

2.1.1. Tracker Module:

This module will run on our central server. The function of this module is to keep track of all the peers who have the BITAACJ extension. It will keep a record of all the webpage that are being visited by the peers. For each peer it will record the following data:

- (i) IP address of the peer
- (ii) Time Stamp

2.1.2. Extension Module

This module will run in the different peers who will install the BITAACJ extension. This module will act as an interaction between the peer and the tractor. The main functions of this module will be done by two subordinate modules:

2.1.2.1. Browser Interaction Module: This module will take will care of the different messages going to and fro from the browser and The main functions of this module will be done by the following modules:

- (i) URL extraction module: will take care
- (ii) HTML parsing module
- (iii) Cache Retrieval

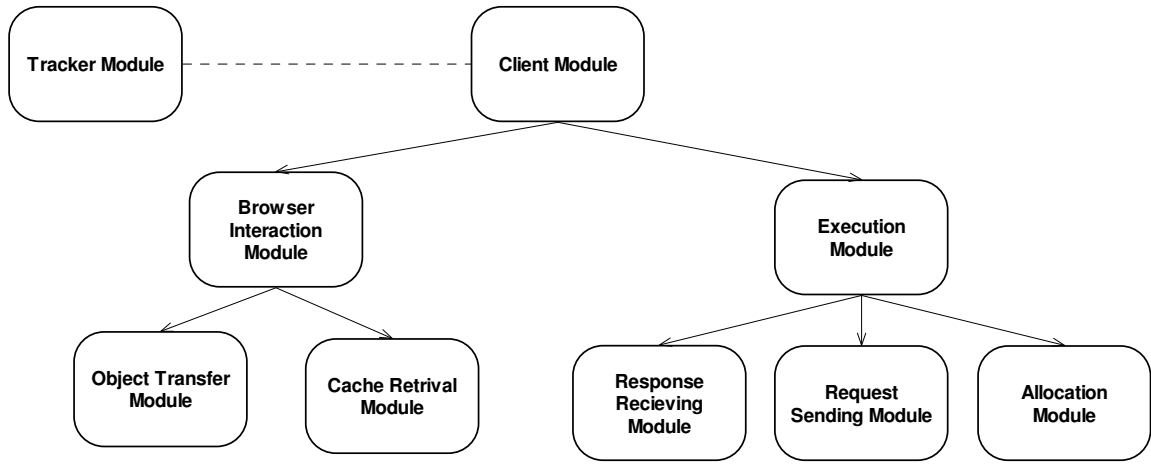
2.1.2.2. Execution Module: This module can be further divided into three modules:

- (i) Response Sending Module
- (ii) Request Receiving Module
- (iii) Allocation Module

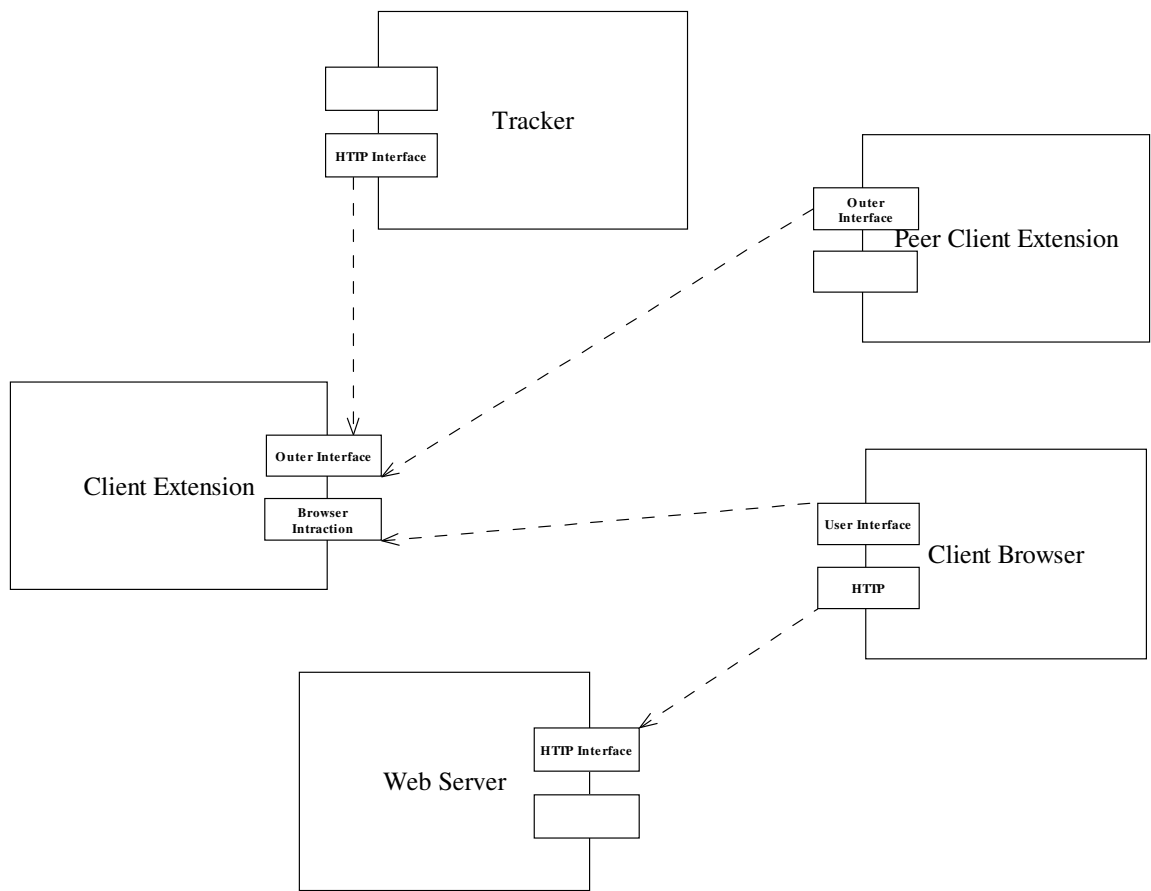
2.2 Structure and relationships

The structure of the modules is explained below.

Control information is transferred between modules to achieve coherent working.



Relationship between modules is explained below. The diagram below shows the relationship between the various modules



2.3 User interface issues

It is not applicable, as the user enters the URL into the browser's address bar from where it is taken by the BITAACJ extension. Hence there is not direct interaction with the user.

3. Detailed description of components

3.1 Client Module description

Identification	Client Module
Type	This is the program running on the client machine. It will contain many modules which have been briefly described above.
Purpose	The purpose of this module is to send the information from the client, to the tracker and the peers. Then to receive the corresponding data and display it accordingly.
Function	The functions of this module can be broadly divided in to the following main parts: <ul style="list-style-type: none">• Parsing the HTML page to get the objects needed by a particular webpage• Connecting to the tracker, and to receive the list of active peers from it.• Dividing the objects among the active peers.• Putting together the objects received from the web server and the different peers, to form the complete webpage.
Subordinates	The module has the following subordinates: <ul style="list-style-type: none">• Browser Interaction Module• Execution Module
Dependencies	The module depends on the URL address entered by the user in the browser and the list of active peers received from the tracker program. The client module initiates the connection to the tracker and waits till it gets the peer list. Based on the peer list it then downloads the different objects from the peers. This module takes care of the connections between the tracker and the respective peers.
Interfaces	The client module takes care of the connections between the tracker and the respective peers. All the connection between them, are done using TCP/IP.
Resources	The module uses all the resources used by individual subordinate modules.
Processing	The processing of the module is the sum of processing of each individual subordinate module.

Data	The data of this module is same as that of execution subordinate module, as explained below.
------	--

3.1.1 Browser Interaction Module

Identification	Browser Interaction Module
Type	This is a module which will be implemented as a set of components
Purpose	The main purpose of this module is take care of all kinds of interactions between our extension & the core Mozilla browser
Function	<p>The module performs the following functions:</p> <ul style="list-style-type: none"> • It gets the URL of all the requests that is send by the browser. • It also can block the GET request when required. • It parses the page frame obtained by the browser, from the Web Server. It obtains the URL of all the objects that are to be downloaded further. • When a particular object is downloaded from a peer instead of a web server, it will give that object to the browser to display. • Keeps record of all the objects which are getting cached in a log. • Searches for a particular object by its URL in the cache log. • When it finds the particular object it forwards that object to the requested component.
Subordinates	<p>This module consists of two subordinate modules which are</p> <ul style="list-style-type: none"> • Object Transfer module • Cache Retrieval module <p>The Object Transfer module performs the first four functions as explained above and the Cache Retrieval module performs the last three functions.</p>
Dependencies	Since this module is used for the interaction between the browser & our extension, its components are called at places where our extension has to get something from the browser or when it has to give something to the browser
Interfaces	There is no external interface to this module. It only has an internal interface thorough which the communication is done, which is according to the defined interface format of Mozilla.
Resources	Since this is a component of Mozilla, it uses shares the resources allocated for the browser.

Processing	It gets the URL for a request by capturing all the GET request send by the browser using the nsIChannel component. To parse a page frame it uses a Mozilla component nsIParser. It takes the help of nsICache component to search through the log which is created by this component in Mozilla.
Data	The Object transfer module does not stores any data. It only does the processing of data. In case of the Cache Retrieval module it contains a log of the cached objects which is updated every now & then, and which is referred for serving a request.

3.1.2 Execution Module

Identification	Execution Module
Type	Module
Purpose	Forms the core part of the extension
Function	The following are the core functions of this module <ul style="list-style-type: none"> • To parse the HTML file and find the links • Send a custom request to Tracker • To receive the response from Tracker • Send requests to Peers through a port • Listen on a port for requests • To communicate with Browser Interface module
Subordinates	Request Sending module, Response module and Allocation module
Dependencies	This component communicates with every other component in the system. It communicates with the Browser Interfacing module, Tracker module and other similar components found in Peers.
Interfaces	Internal interfaces - used to interact with the Browser Interface module. External Interfaces - All external communications are done through TCP Sockets.
Resources	Requires an Internet connection for communicating with the Tracker and Peers. It should be possible to open separate ports for communicating with external server and peers.
Processing	<ul style="list-style-type: none"> • HTML parsing – Once the browser gets the response from the Web Server, the extension gets the links from the parsed document. • Sending a custom request – The required details are collected and a custom request is formed according to a predefined format. This request is sent to the tracker. • Receiving a Tracker response – This component listens on a port for receiving tracker response. Once a response is detected, the required data (peers list) is

	<p>extracted.</p> <ul style="list-style-type: none"> • Sending requests to Peers - Request for an object to a peer is send in a predefined format. The request contains the URL of the actual object. This request is send through a predefined port. • Listening on a port for requests – The module listens on a port asynchronously for any external requests. As soon as it gets a request for an object, the component searches its cache to check the availability of the object. If the object exists in cache, it is sent back to the client. • Communication with Browser Interface Module – This is done through predefined interfaces.
Data	<p>For sending request to Tracker – MD5 of URL Response from Tracker – A comma separated list of Peers</p>

3.2 Tracker Module

Identification	Tracker_process
Type	This module is a part of the server program . This is a subprogram running on the server machine.
Purpose	<p>The module performs the following functions:</p> <ul style="list-style-type: none"> • Listens at a port for connections • Accepts a structure from the peer containing <ul style="list-style-type: none"> ○ MD5 hash of each object ○ IP ○ Size ○ Last accessed processed from the incoming request timestamp • Save structure as a Mysql database record
Function	This function implements the TCP listen and connect method to receive and process the request for the peers. This main object would be to index the peers and the domains accessed. In order to index the peers it will be using the Mysql api.
Subordinates	
Dependencies	This component does to interface or relate to any other function. This function is responsible for creating the records and ensuring accuracy in the central database
Interfaces	The interfaces for this function is the connection from the peers at port 27015. This function will be responsible for the response to the peers .

	Administration will this server will be from the command line(shell)
Resources	This module requires an installation of Perl and Mysql. No external interfaces are required as long as an openssh connection can be established. Since backend database Mysql is used and the module is threaded RAM not less than 512 Mb is required. Internet connection is needed.
Processing	
Data	Initially a blank database exists. Structure MD5 128 bit IP 8 byte Size 3 byte Last accessed 10 bytes

4. Reuse and relationships to other products

Since our project is to create an extension for the Mozilla firefox browser we will be using some of the components and modules which are already present in the Mozilla architecture than build new ones.

We will be also reusing the prefetch mechanism of another extension called Fasterfox, to parse the HTML page layout to get the information of the different objects present in the desired HTML page in advance.

As said before since this is an extension to the Mozilla browser it will be closely related to it, and will be used to enhance its existing features.

5. Design decisions and tradeoffs

The prototype extension for Firefox is to be coded mainly in JavaScript and parts of the extension which are processor intensive will be coded in C++ to get the performance enhancement. The frontend for Tracker will be coded in Perl and will use MySQL as the database. The database will store only the MD5 of the Object URL's so as to reduce the payload involved in communication. The current model will not work with systems behind a NAT or firewall.

6. Pseudocode for components

Acquiring URL from Browser:

```
Var Bitacj = {
```

```
  acquireURL : function() {
```

```
    Var href = document.location.href;
```

```
    Alert(href);
```

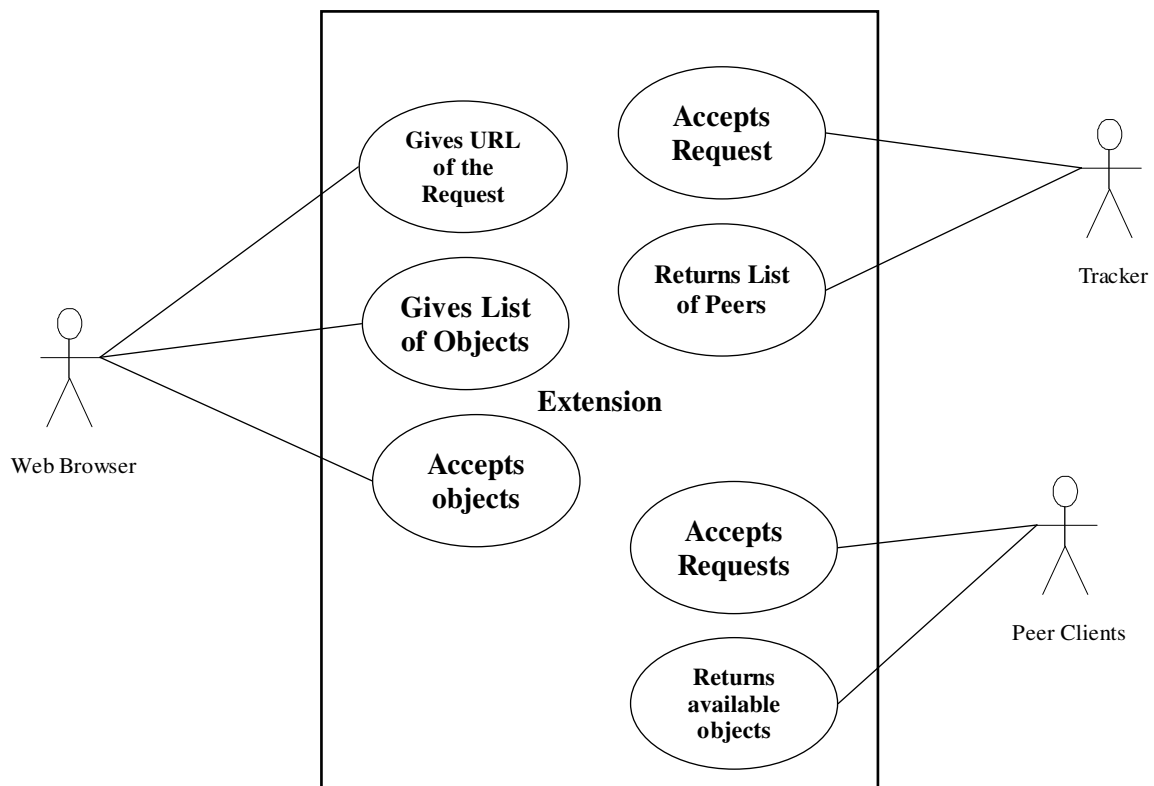
```
  },
```

```
}
```

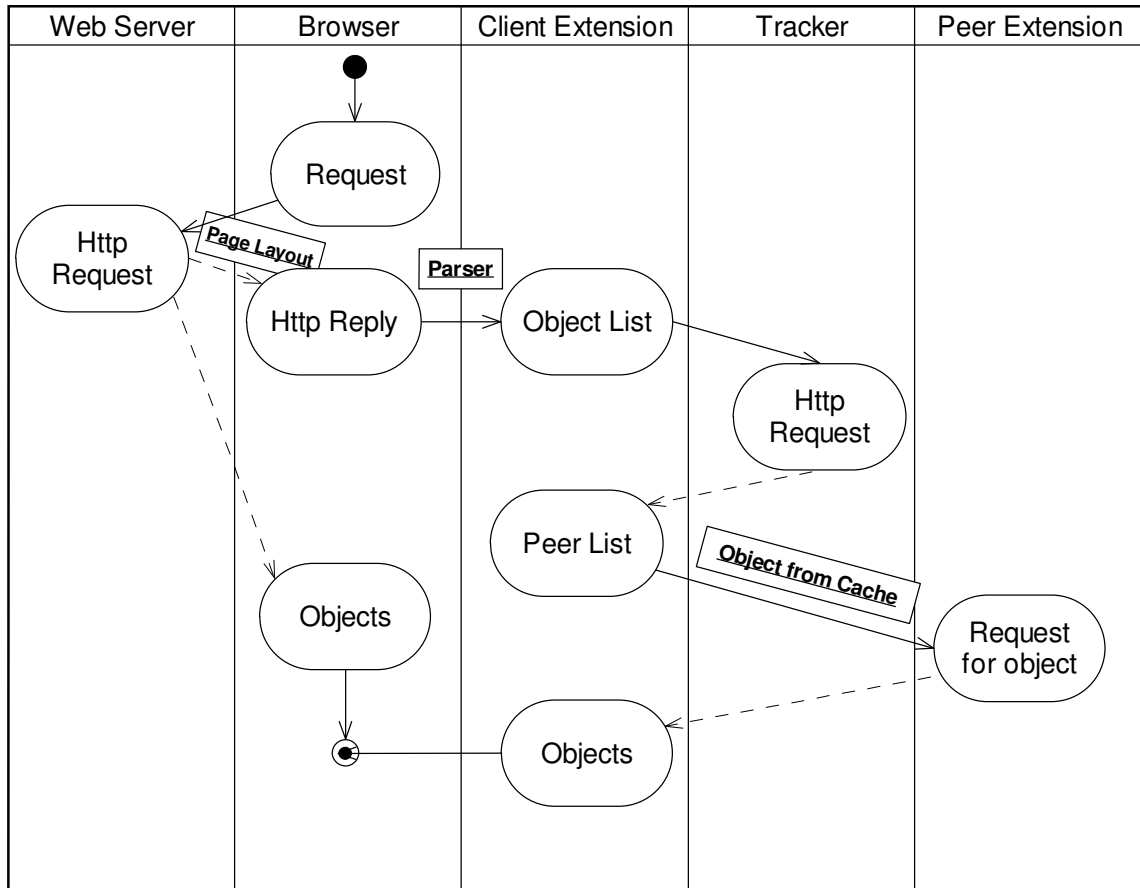
```
getBrowser().addEventListener("load", Bitacj.acquireURL, false);
```

7. Appendices

7.1 Use Case Diagrams

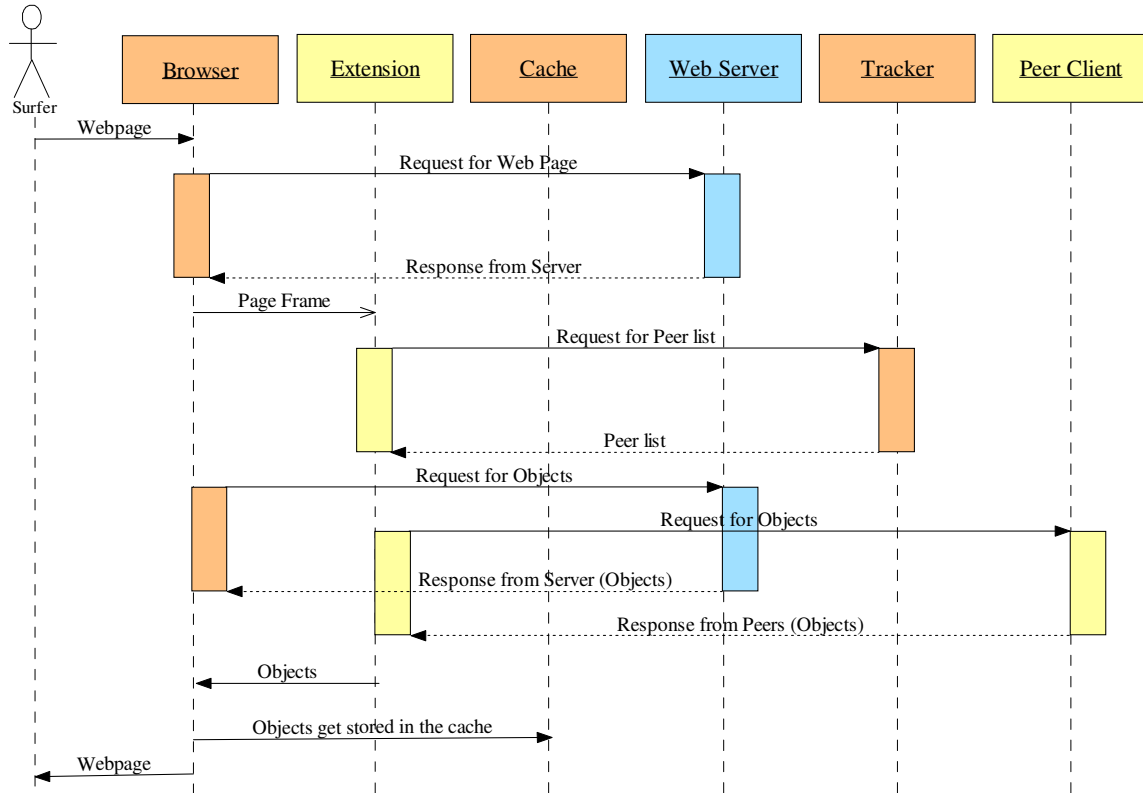


7.2 Activity Diagram



7.3 Sequence Diagram

7.3.1 Clients Sequence Diagram



7.3.2 Peers Sequence Diagram

