Bilkent University

Department of Computer Engineering

Senior Design Project

Low Level Design Report

Project Name: PeerNews

Group Members: Şamil İraz
Enes Emre Erdem
Muhammed İkbal Kazar

Supervisor: Prof. Özcan Öztürk

Jury Members: Prof. H. Altay Guvenir
Assoc. Prof. Selim Aksoy

Innovation Expert: Dr. Ali C. Beğen

Low Level Design Report

Oct 1st, 2019
This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Table of Contents

1. Introduction

Decentralized architectures are becoming more popular due to recent improvements in peer to peer networking technologies. Crypto currencies such as Bitcoin, Ethereum and the file sharing systems such as Bittorrent are popular examples of renowned decentralized systems. PeerNews will provide a decentralized approach to media distribution by creating a peer to peer network among users. This decentralized architecture will not have a single point of failure or central authority. Thus everyone's freedom of speech rights will be protected by design.

PeerNews targets everyone who wants to participate in decentralized social media based on freedom of speech and privacy. The main purpose of the system is to let users share and consume messages of various mediums such as image, video or plain text in a secure, efficient and decentralized way. When consuming content, users able to specify their preferences to interact with the messages they are interested in.

1.1 Object Design and Trade -offs

1.1.1 Security vs Cost

PeerNews does not hold the user data in a central server. Instead, due to its peer to peer structure, it stores the data on peers. Other peers can reach the data by downloading it from their peers via torrent. Once the data is uploaded into torrent, the

unique hash code of the data will be distributed among peers to keep it safe and secure. However, this structure may bring a highload of network usage due to its constant uploading and downloading procedure. Our responsibility is to balance out this network usage [1].

1.1.2 Functionality vs Usability

Usability of application is one of the main purpose that PeerNews aims. We want users to be able to easily interact with application. Users should be able to use our application without any worrying. However, since main idea behind our application requires some functionalities such as being connected to network, availability of contents etc; functionality is important as well as usability. Thus, we need to manage to keep the balance in between functionality and usability.

1.1.3 Space vs Durability

PeerNews clients garbage collect messages that are older than one day to avoid exhausting user device's disk storage. Therefore, system is designed with this trade-off in mind by limiting the space used by the application through compromising durability. In addition, this compromise on durability allows users to share more freely since they know that the messages they share won't be archived for longer periods of time.

1.1.4 Compatibility vs Extensibility

Although we are developing a desktop application, it is quite possible that in the future, PeerNews can be extended into different mediums. A mobile version or web version of the application can be developed. However, such extensions should be compatible with

PeerNews' current network structure. For example, they should be compatible with the peer to peer structure or the torrent.

1.2 Interface Documentation Guidelines

This report follows the standard convention for documentation guidelines. All the class names are named in the standard 'ClassName' form. Methods and variables follow the same convention as they are also named as 'methodName()' and 'variableName'. In a complete description of a class, class name comes first, followed by the class attributes, and finally the methods. A detailed layout template is given below:

Class Name

- A short description of the class

Attributes

- Attribute name and type

Methods

- Method Name
- Parameters
- Return Value

1.3 Engineering Standards

In this report, UML is used as the modeling language since UML is standardized and commonly used in the area of software engineering. Furthermore, IEEE standards are used as the citation method. It is also commonly used and widely accepted for citations in software engineering area [4].

1.4 Definitions, Acronyms, and Abbreviations

Peer: Basically each end-system connected to the network is a peer. Communication is taking place between peers.

P2P: Peer to peer

Torrent: Torrent is the peer to peer web service that lets us upload and download files.

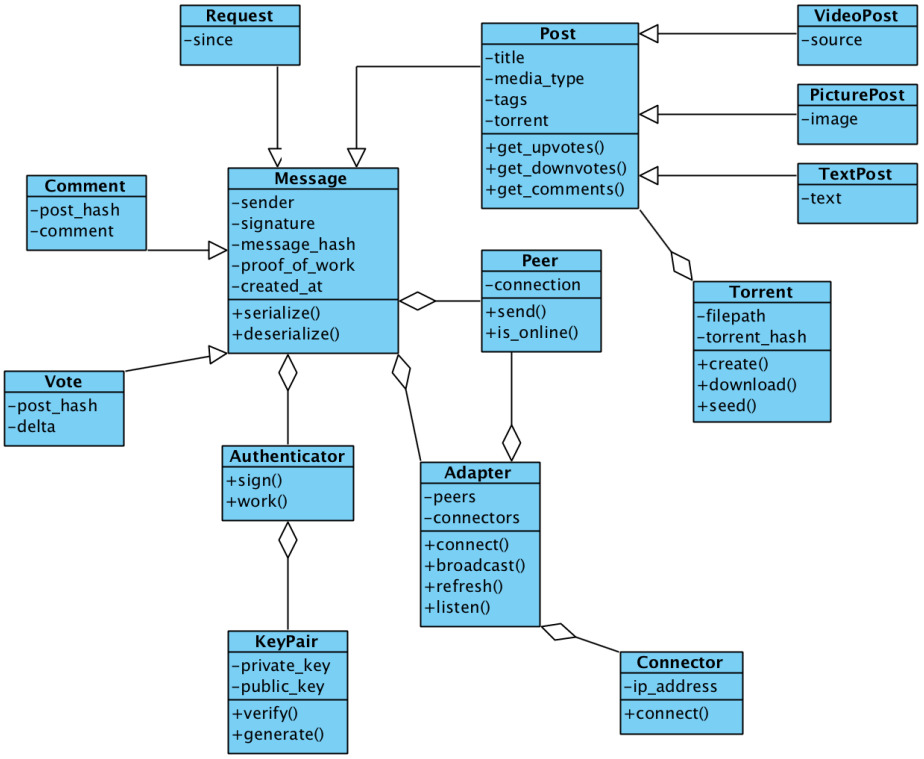
Connector: Connectors are special peers that have the ability to handle newcomers to the network. They receive the connection request from users and connect them by sharing their ip address. All peers can be connectors if they want to.

Server: Refers to connectors. Since PeerNews does not have a central server, connector peers act as decentralized servers.

Metadata: Metadata is simply data about data. It contains the description and context of the original data. In PeerNews metadata also contains torrent links.

2. Packages

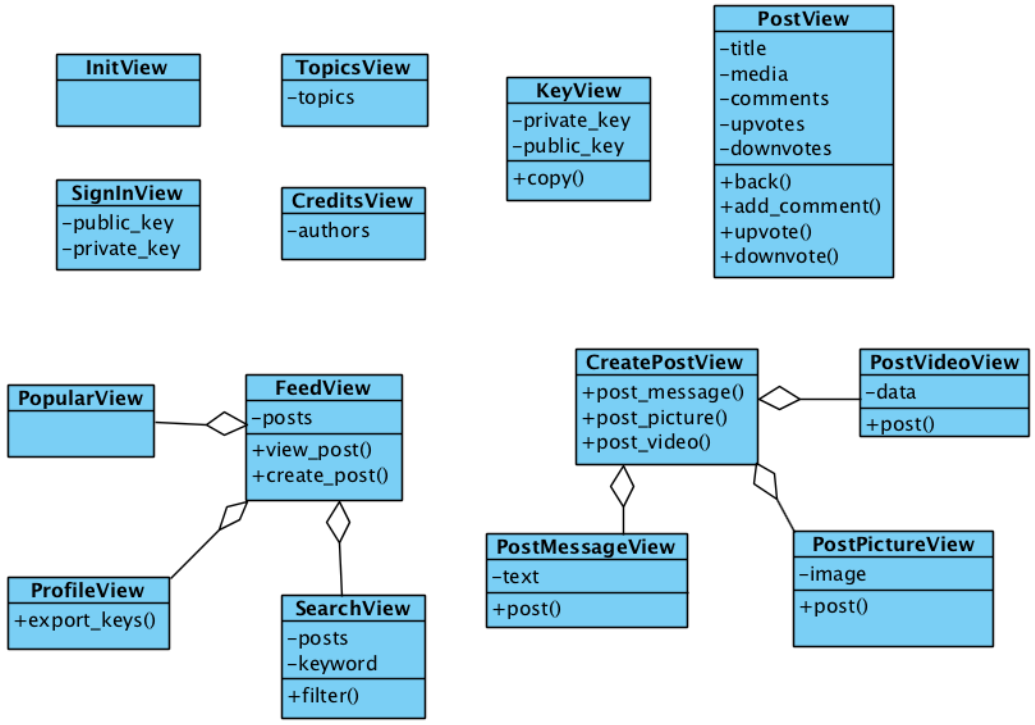
2.1 Peer Communication Protocol



- **Message** class is the base a class that represents a basic message object. Each message contains various meta-data such as signature, hash, proof of work.
- **Request**, **Comment**, **Vote**, **Post** are subclasses of the **Message** class. Each represents a particular type of message with additional fields.
- **Authenticator** class is responsible for creating a digital signature and proof of work for a given message [2].

- **KeyPair** class contains private and public keys of the user. It is also responsible for verification and initial generation of keys.
- **Adapter** class maintains a list of connectors and peers that are directly connected to the current user.
- **Torrent** class is responsible for the connection with the torrent network.

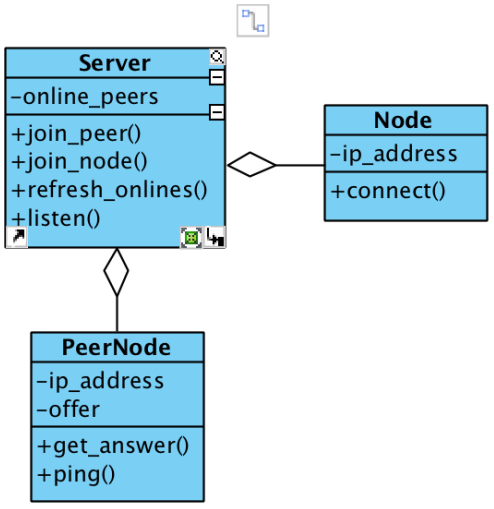
2.2 Client User Interface



- **InitView** class is responsible for verifying existing user credentials and redirecting to SignInView or FeedView based on that.
- **SignInView** class is responsible for creating private/public key pair for someone to join the network as a new user.

- **KeyView** class allows an existing user to see their private/public key pair and take a note of the private key in case they would like to migrate their user to a new device.
- **TopicsView** class renders a list of topics available and allow user to see a feed of messages tagged with the corresponding topic.
- **CreditsView** class is responsible for a simple page listing credits to developers and open-source library that's been used.
- **FeedView** class displays a list of posts (messages). It will also allow user to focus on a particular message to view more detailed info such as comments etc.
- **SearchView** class allows the user to view a list of messages containing the searched keyword.
- **ProfileView** class displays user's own posts and provides a button to view key pair by redirecting to **KeyView's** page.
- **PopularView** is a specialized feed view where list of messages consists of trending posts with most comments or upvotes.
- **CreatePostView** renders a page that allows the user to create their own messages to be shared with the network. **PostVideoView, PostPictureView , PostMessageView** classes are responsible for video, picture and plain-text post creation respectively.
- **PostView** class display's a post's metadata, content and associated comments, upvotes in a user-friendly way.

2.3 Connector Server



- **Node** class represents another connector server with it's ip address that this connector can connect to.
- **PeerNode** class represents a peer, a client node, that connector is aware of.
- **Server** class will be run as part of the main process of the connector server. It will be responsible for maintaining a list of peers that are potentially available to connect. Each peer that initially connects to a connector server will be provided with a sample of other peers that can be attempted to connect with directly.

3. Class Interfaces

3.1 Peer Communication Protocol

class Message
This is the base a class that represents a basic message object. Each message contains various meta-data such as signature, hash, proof of work [3].
Attributes
private String sender private String signature private String message_hash private String proof_of_work private Datetime created_at
Methods
public String serialize(Message message) Public Message deserialize(string serialized)

class Request
Subclass of the base Message class. Allows a peer to request missed messages since a specified time.
Attributes
private Datetime since
Methods
Overrides base message methods serialize and deserialize

class Post
Subclass of the base Message class that represents a post with related metadata.
Attributes
private String title private String media_type private String tags private Torrent torrent
Methods
public int get_upvotes() public int get_downvotes() public List<Comment> get_comments()

class Comment
Subclass of the base Message class. Allows a user to comment on a post.
Attributes
private String post_hash private String comment
Methods
Overrides base message methods serialize and deserialize

class Vote
Subclass of the base Message class. Allows a user to vote on a post.
Attributes
private string post_hash private int delta
Methods
Overrides base message methods serialize and deserialize

class Authenticator
Class with static methods that implements proof of work and signature related cryptology.
Attributes
N/A
Methods
public static Message sign(Message message) public static boolean proof_of_work(Message message)

class KeyPair
Class that represents a collection of public and private keys.
Attributes
private String public_key private String private_key
Methods
public bool verify() public static KeyPair generate()

class Torrent
This class is responsible for the connection with the torrent network.
Attributes
private String file_path private String torrent_hash
Methods
public static Torrent create() public void download() public void seed()

class Adapter
This class maintains a list of connectors and peers that are directly connected to the current user.
Attributes
private List<Peer> peers private List<Connector> connectors
Methods
public void connect() public void broadcast(Message message) public void refresh() public void listen()

class Connector
Represents a connector server from peer’s perspective.
Attributes
private string ip_address
Methods
public void connect()

3.2 Client User Interface

class InitView
Class that responsible for verifying existing user credentials and redirecting to SignInView or FeedView based on that
Attributes
None
Methods
None

class SignInView
Class that responsible for creating private/public key pair for someone to join the network as a new user
Attributes
private string public_key private string private_key
Methods
None

class KeyView
Class that allows an existing user to see their private/public key pair and take a note of the private key in case they would like to migrate their user to a new device
Attributes
private string public_key private string private_key
Methods
public void copy()

class TopicsView
Class that renders a list of topics available and allow user to see a feed of messages tagged with the corresponding topic
Attributes
private List<String> topics
Methods
None

class CreditsView
Class that is responsible for a simple page listing credits to developers and open-source library that’s been used
Attributes
private List<String> authors
Methods
None

class FeedView
Class that displays a list of posts (messages). It will also allow user to focus on a particular message to view more detailed info such as comments etc
Attributes
private String post
Methods
public void view_post(String post) public void create_post()

class SearchView
Class that allows the user to view a list of messages containing the searched keyword
Attributes
private String post private String keyword
Methods
public void filter(String attributes)

class ProfileView
Class that displays user's own posts and provides a button to view key pair by redirecting to KeyView's page
Attributes
None
Methods
public void export_keys()

class PopularView
Class that is a specialized feed view where list of messages consists of trending posts with most comments or upvotes
Attributes
None
Methods
None

class CreatePostView
Class that enders a page that allows the user to create their own messages to be shared with the network
Attributes
None
Methods
public void post_message(Media content) public void post_picture(Media content) public void post_video(Media content)

class PostView
Class that display’s a post’s metadata, content and associated comments, upvotes in a user-friendly way
Attributes
private String title private Media media private String comments private String upvotes private String downvotes

Methods
public void back() public void add_comment(String comment) public void upVote() public void downVote()

3.3 Connector Server

class Server
Server class will be run as part of the main process of the connector server. It will be responsible for maintaining a list of peers that are potentially available to connect. Each peer that initially connects to a connector server will be provided with a sample of other peers that can be attempted to connect with directly.
Attributes
private PeerNode online_peers
Methods
public void join_peer(PeerNode peer) public void join_node(Node peer) public void refresh_onlines() public void listen()

class Node
Node class represents another connector server with it's ip address that this connector can connect to.
Attributes
Public String ip_adress
Methods
public void connect()

class PeerNode
PeerNode class represents a peer, a client node, that connector is aware of.
Attributes
private String ip_adress private String offer
Methods
public String get_answer() public int ping()

4. References

[1] “Managing Trust in Peer-to-Peer Systems.” *10 Best Practices for Secure Software Development | Security, Data and Privacy | Subject Areas | Publishing and Editorial | BCS - The Chartered Institute for IT*, ITNOWextra, Jan. 2006, www.bcs.org/content/conWebDoc/3059.

[2] Kamvar, Sepandar D., et al. “The Eigentrust Algorithm for Reputation Management in P2P Networks.” *Contents: Using the Digital Library*, ACM, 20 May 2003, dl.acm.org/citation.cfm?id=775242.

[3] Adam Back, "Hashcash - A Denial of Service Counter-Measure", technical report, August 2002

[4] Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0.