

Tribler

ModerationCAST Design document

Efficient epidemic protocol for the propagation of rich metadata, with
pollution prevention methods

Vincent
9/18/2007

Introduction

This document describes the design of the ModerationCast module for Tribler. It describes the message sequence and structure, the classes and their relations with each other and the database-structure.

Messages

The following messages are used in the ModerationCAST protocol:

- MODERATION_HAVE
- MODERATION_REQUEST
- MODERATION_REPLY

The MODERATION_HAVE message is sent by both peers after a Buddycast-exchange.

Once a peer receives a MODERATION_HAVE message, it checks if it Buddycasted with this peer in the last minutes. Iff this is true then it will select zero or more of the infohashes for which it will request moderations. To select what infohashes to download moderations for, the peer can use the information in the MODERATION_HAVE message. This enables the peer to select moderations of a certain age, belonging to a certain torrent and up to a certain size (to limit download bandwidth usage).

Once a peer receives a MODERATION_REQUEST message, it chooses a (possibly empty) subset of the infohashes for which it has received a request and sends the corresponding moderations in a MODERATION_REPLY message. Reasons for limiting the number of moderations could be to limit upload bandwidth usage.

Message structure

MODERATION_HAVE

A Bencoded list with a maximum of 100 tuppels, which are requestable by the receiver:

| Index | Description | Type | Size (without Bencoding) |
|-------|---|------|--------------------------|
| 0 | Infohash of the torrent to which the moderation belongs | str | 20 bytes |
| 1 | Timestamp of the creation of the moderation in UTC (system time of the moderator) | int | 4 bytes |
| 2 | Size of the moderation in bytes | int | 4 bytes |

MODERATION_REQUEST

A Bencoded list with a maximum of 100 infohashes being requested by the sender.

MODERATION_REPLY

A Bencoded list of dictionaries with the following key-value-pairs:

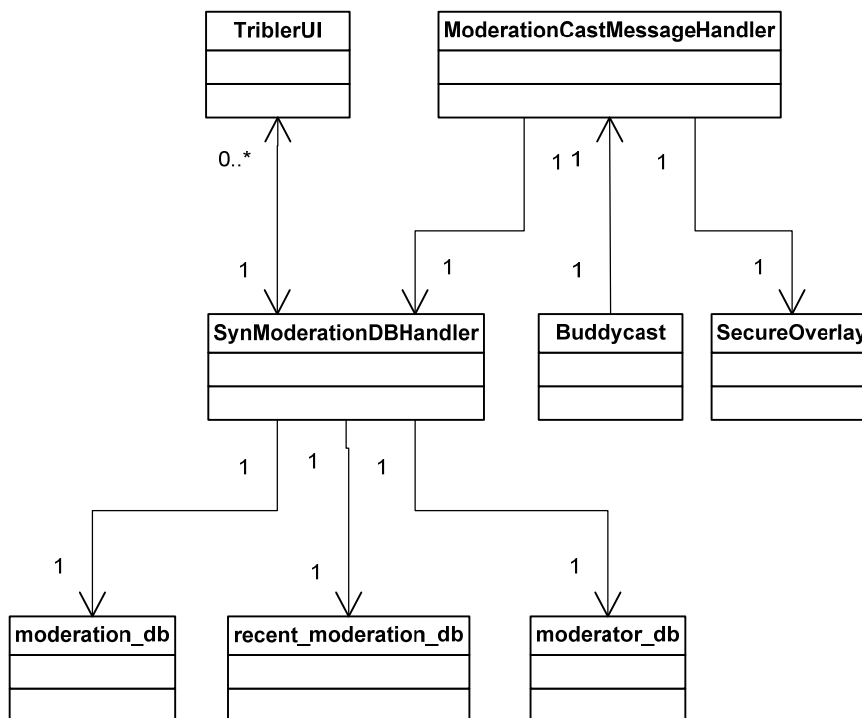
| Key | Description | Type | Maximum size (checked!) |
|------------------------|---|--------------|-------------------------------------|
| moderator | PermID of the moderator | str | ??? |
| timestamp | Timestamp of the creation of the moderation in UTC (system time of the moderator) | int | 4 bytes |
| <i>spoken_language</i> | ISO 639-3 coding of the spoken language | str | 3 letters |
| <i>subtitles</i> | Dictionary: ISO 639-3 str -> binary srt-data | {str:binary} | 150KB per subtitle, max 8 languages |
| <i>description</i> | Description of the torrent in text | str-unicode | 10.000 characters |
| <i>thumbnail</i> | Binary thumbnail for this torrent | binary | 100 KB |
| <i>tags</i> | List of strings that are relevant to this torrent | [str] | ??? |
| signature | binary ECDSA-signature over the moderation excluding the signature itself | binary | ??? |

Italic denotes optional, **bold** denotes required.

It might be a good idea to compress some of these messages (the srt-files are good candidates for compression).

Classes

UML class diagram for the ModerationCast module, including external modules that are directly required:



Method specification per class:

SynModerationDBHandler-class

This class must be used to access the moderation-databases. It will make sure that the databases remain consistent and access easy.

Methods

| Method | Description | Parameters | Return type |
|-------------------|--|---|-----------------------------------|
| get_moderation | Returns the moderation for a given infohash | infohash | bDecoded MODERATION_REPLY-message |
| add_moderation | Adds moderation to the moderation-db's, overwriting the old moderation | dictionary with a subset of the following keys (and their values): infohash <i>moderator</i> timestamp spoken_language subtitles : {lang:path} description thumbnail : path tags <i>signature</i> Specify moderator and signature if from other moderator or leave open for own moderation | bool |
| remove_moderation | Removes the moderation for a given infohash | infohash | None |
| block_moderator | Removes all moderations from this moderator and marks this moderator as blocked (moderator_db) | bool | None |
| forward_moderator | Marks this moderator as a moderator to forward for (moderation_db, forward) | bool | None |

ModerationCastMessageHandler-class

This class must be used to enable the ModerationCAST module. It registers itself with the buddycast module (which it is dependent upon).

Methods

| Method | Description | Parameters | Return type |
|-------------------------|---|-----------------------------|-------------|
| connection_handler | Callback function for a Buddycast connection event | None | None |
| have_message_handler | Callback function for the receipt of a MODERATION_HAVE-message Sends a MODERATION_REQUEST message for a subset of the infohashes specified in the HAVE_MESSAGE, to the sender of this message | BDecoded HAVE_MESSAGE | None |
| request_message_handler | Callback function for the receipt of a MODERATION_REQUEST-message Sends a MODERATION_REPLY message for a subset of the infohashes specified in the REQUEST_MESSAGE, to the sender of this message | BDecoded REQUEST_MESSAGE | None |
| reply_message_handler | Callback function for the receipt of a MODERATION_REPLY-message Stores the moderations that this REPLY_MESSAGE contains in the moderation databases | BDecoded REPLY_MESSAGE | None |
| select_have | Function that selects the infohashes for the HAVE_MESSAGE from all known moderations. | None | [infohash] |
| select_request | Function that selects the infohashes for the REQUEST_MESSAGE from the infohashes in the HAVE_MESSAGE, this is based on the following criteria: <ol style="list-style-type: none">1. Select only moderations for which we have the torrent2. Select newest infohashes up to a certain size-limit | [Infohash] | [infohash] |

| | | | |
|--------------|--|------------|------------|
| select_reply | Function that selects the infohashes for the REPLY_MESSAGE from the infohashes in the REQUEST_MESSAGE, this is based on the following criteria: 1. Select the infohashes in order of the REQUEST message up to a certain size-limit | [Infohash] | [Infohash] |
|--------------|--|------------|------------|

Database structure

The moderations will be stored in databases. These databases will be used by both the user-interface and the ModerationCAST-module. The structure of the databases will be as follows (please note that these databases will be accessible via the ModerationDBHandler-class to keep the data consistent):

ModerationDB

A dictionary {Infohash of a torrent : A dictionary with the following key-value-pairs:}

| Key | Description of value | Value-type |
|------------------------|---|-------------|
| moderator | PermID of the moderator | str |
| timestamp | Timestamp of the creation of the moderation in UTC (system time of the moderator) | int |
| <i>spoken_language</i> | ISO 639-3 coding of the spoken language | str |
| <i>subtitles</i> | Dictionary: ISO 639-3 str -> filename of subtitle file | {str:str} |
| <i>description</i> | Description of the torrent in text | str-unicode |
| <i>thumbnail</i> | filename of thumbnail file | str |
| <i>tags</i> | List of unicode-strings that are relevant to this torrent | [str] |
| signature | binary ECDSA-signature over the moderation excluding the signature itself | binary |

Italic denotes optional, **bold** denotes required.

The only items not being stored in the database are the subtitles and the thumbnails. For these items only references will be stored in the database. The items themselves will be stored on disk. The subtitles will be stored in TRIBLER_HOME_DIR/.subtitles/INFOHASH_LANGUAGE.EXT and the thumbnails will be stored in TRIBLER_HOME_DIR/.thumbnails/INFOHASH.EXT.

ModeratorDB

A dictionary { PermID of the moderator : A dictionary with the following key-value-pairs:}

| Key | Description of value | Value-type |
|--------------------|---|------------|
| moderations | List of infohashes for which the moderator has created the last moderation (local knowledge), keep up-to-date with moderation_db! | str |
| forward | Forward this users moderations to others | bool |
| blocked | Do not use moderations from this moderator | bool |

Italic denotes optional, **bold** denotes required.

RecentModerationDB

A list of 100 infohashes, which have the most recent moderations, in terms of receipt (not timestamp).