\* Важное замечание: имена чанков/файлов всегда совпадают с их InfoHash.

**Streamer (class StreamerSession)**

**OBS-Studio** после 'правильной' настройки генерирует плейлист и чанки

- obs-stream.m3u8

- obs-stream0.ts

- obs-stream1.ts

- ...

**Стример** цепляется на либторрентовкий таймер (1 сек)

и каждый раз **сканирует плейлист**.

Если появились **новые чанки**, то стример создает торрент файл для нового чанка и **загружает** его **в сессию**.

Плюс добавляет в m\_chunkInfoMap

struct ChunkInfo

{

std::array<uint8\_t,32> m\_streamId;

uint32\_t m\_chunkIndex;

std::array<uint8\_t,32> m\_chunkInfoHash;

uint32\_t m\_durationMks; // microseconds

uint64\_t m\_sizeBytes;

std::array<uint8\_t, 64> m\_sign;

};

И посылает репликаторам **DHT message "chunk-info"** (struct ChunkInfo).

Для того чтобы **стрим заканчился,** стример вызывает функцию finishStream(),

При этом:

- формируется вектор **finishStreamInfo** (struct FinishStreamChunkInfo).

- вычисляется для finishStreamInfo **торрент файл** и загружается в сессию.

- прекращается сканирование OBS-Studio **плейлиста**

Получив хеш **finishStreamInfo**, стример в блокчейн транзакцию **STREAM\_FINISHED**.

**Блокчейн** в свою очередь вызывает **asyncFinishStreamTxPublished**()

**Replicator (class StreamTask)**

Вначале репликатор получает от блокчейна уведомление о начале стриминга:

asyncStartStream( DRIVE\_PUB\_KEY, streamRequest )

где

struct StreamRequest

{

Hash256 m\_streamId; // transaction hash

Key m\_streamerKey; // streamer public key

std::string m\_folder; // where it will be saved in FsTree

uint64\_t m\_maxSizeBytes; // could be increased

ReplicatorList m\_replicatorList;

};

После этого репликатор ждет от стримера **DHT messages "chunk-info"**

Получив **"chunk-info”**, начинает **загрузку чанка**.

В это же время к репликатору приходят от вьюверов **DHT запросы "get-chunks-info"**.

Если **лайв-стрим закончился**, то репликатор отвечает, что **стрим закончился**.

Eсли же на стороне репликатора в текущий момент **нет активной StreamTask**,

то посылается ответ, что **стрим закончился** или **стрим не начался**.

Когда блокчейн вызывает у репликаторов **asyncFinishStreamTxPublished**(), то

- репликатор проверяет, все ли **чанки** указаны в **finishStreamInfo**

- если хотя бы что-то сохраняется на диск, то формируется **плейлист стрима** и добавляется в сессию

(А если ничего не сохраняется?)

- в **сендбоксе** формируется новый **FsTree**

- запускается механизм MODIFY\_APPROVE\_TX

После того, как от блокчейна приходит подтверждение MODIFY\_APPROVE\_TX,

и репликатор завершает StreamTask.

**Viewer (class ViewerSession)**

Для того, чтобы начать просматривать стрим, пользователь должен иметь (или создать) **download channel** для данного диска.

Затем нужно вызвать функцию **ViewerSession::startWatchingLiveStream()**.

Вьювер периодически (1 сек) посылает **DHT запросы "get-chunks-info"** к репликаторам.

Если появился чанк с запрашиваемым индексом, то возвращается его **InfoHash** (а так же по возможности нескольких последующих).

Вьювер скачивает чанк и после этого **обновляет плейлист**.

Если стрим закончился, то вьювер посылает **DHT запросы "get-playlist-hash"** к репликаторам.

После того, как приходит ответ, вьювер качает результирующий плейлист + недостающие чанки.