Livepeer Documentation

Release 0.0.1

Doug Petkanics

Apr 08, 2019

Contents

Livepeer is a decentralized video broadcasting platform powered by a crypto token on the Ethereum blockchain. Livepeer is for:

- Developers who want to build applications that include live video.
- Users who want to stream video, gaming, coding, entertainment, educational courses, and other types of content..
- Broadcasters who currently have large audiences and high streaming bills or infrastructure costs can use the Livepeer network to potentially reduce costs or infrastructure overhead.

Use this documentation to learn how to broadcast video through Livepeer, participate in the Livepeer protocol as a transcoder or delegator, and build apps or DApps with video based features using Livepeer.

We suggest you start with *Getting Started*.

CHAPTER 1

Contributing

The code for this documentation is open source and is available on Github. Updates and pull requests are much appreciated.

CHAPTER 2

Index

2.1 Getting Started

This guide will get you started with broadcasting your first livestream using the Livepeer tools in 5 minutes. We will avoid using the blockchain for simplicity. Livepeer is currently under active development, and it is accessed through the command line on Mac(darwin) or Linux.

The first step in getting started with Livepeer is to try to run the Livepeer executables and to broadcast a simple livestream.

2.1.1 Download Livepeer

Visit https://github.com/livepeer/go-livepeer/releases/tag/0.4.0. Depending on your operating system, choose the _darwin version for OS X and the _linux versions for Linux, and then untar them. There will be 2 files: `livepeer` and `livepeer_cli`:

```
$ tar -zxvf livepeer_darwin.tar.gz
$ mv ./livepeer_darwin/livepeer ./livepeer
$ mv ./livepeer_darwin/livepeer_cli ./livepeer_cli
$ ./livepeer -offchain -currentManifest
```

This will start a Livepeer node running in `offchain` mode. It will ask you to set a password and use this same password to unlock your ETH account.

2.1.2 Broadcast and Play Video

The easiest way to start broadcasting is by using OBS. For instructions, go to https://livepeer.readthedocs.io/en/latest/broadcasting.html.

2.1.3 What's Next?

You just demonstrated sending video to a Livepeer node. Time to learn how to use more convenient tools to broadcast and consume the streams. The next sections will teach you how to run a node on the blockchain, use Livepeer to broadcast to a large audience, how to build an app with video functionality using Livepeer, and how to participate in the Livepeer protocol as a delegator or transcoder.

2.2 The Livepeer Node

The Livepeer node is a command line executable called `livepeer` that connects to other nodes on the Livepeer network and speaks the Livepeer protocol. It comes with an accompanying command line interface (CLI) called `livepeer_cli` which makes it easy to take a number of actions on the network.

The below instructions are comprehensive for a number of scenarios, but generally running a single Livepeer node and joining the test network consists of simply running the command:

```
$ livepeer --rinkeby
```

2.2.1 Installation

You can download precompiled binaries, or you can build the latest version from source.

Download Executables

Follow the instructions on *Getting Started* to download the binaries for your platform and set their permissions.

Building from Source

The latest instructions for building the go-livepeer project can be found on Github.

2.2.2 Running a node

Once you have installed the executable, you can invoke it by running:

```
$ livepeer
```

Note: by default Livepeer listens to the local interface. This means if you are running Livepeer on a cloud-hosted instance, you need to set the `--rtmpAddr 0.0.0.0:1935` flag. However, there is no security built into the RTMP listener, so use with caution.

There are two other options that control the use of Livepeer services. The first is the API for the CLI interface. The CLI is meant to be a control interface towards the node: it can bond and transfer LPT, deposit and withdraw ETH, initialize rounds, manage broadcast and transcoding configurations, and so forth. Hence, it is strongly recommended to keep the CLI internal-only: the default setting is `` `--cliAddr 127.0.0.1:7935 ``. Only change the listening IP if you need to remotely configure your node, and you are absolutely certain that the listening interface is secure from the outside world.

The second option is the RPC/HTTP port. Broadcasters and transcoders use RPC messaging to interact and users can view streams via HTTP. The RPC and HTTP functions share the same port, and are configured with the same option. For the broadcaster, the default is `-httpAddr 127.0.0.1:8935` . For transcoders, the default is `-httpAddr 0.0.0.0:8935`.

In offchain mode

Using offchain mode does not require syncing with the Ethereum blockchain. Start a node in offchain mode with the command:

```
$ livepeer --offchain
```

You are now running a node, and can use it to develop and test Livepeer locally, or even use it as the basis to begin forming a *private network*.

Running a Livepeer node on the Ethereum Rinkeby Testnet

The Livepeer testnet is a set of nodes that are running on Ethereum's Rinkeby testnet blockchain.

Run Livepeer

Make sure that you have gone through the installation steps for both Livepeer, and its dependencies ffmpeg and. Now you can start Livepeer:

```
$ livepeer --rinkeby
```

In a separate terminal window, run livepeer_cli:

```
$ livepeer_cli
```

Livepeer CLI will print out your account address, ETH balance, Livepeer token balance, and more info. It should present an array of options for interacting with Livepeer:

```
What would you like to do? (default = stats)
1. Get node status
2. View protocol parameters
3. List registered transcoders
4. Print latest jobs
5. Invoke "initialize round"
6. Invoke "bond"
7. Invoke "unbond"
8. Invoke "rebond"
9. Invoke "withdraw stake" (LPT)
10. Invoke "withdraw fees" (ETH)
11. Invoke "claim" (for rewards and fees)
12. Invoke "transfer" (LPT)
13. Invoke "deposit" (ETH)
14. Invoke "withdraw deposit" (ETH)
15. Set broadcast config
16. Set Eth gas price
17. Get test LPT
18. Get test ETH
```

The testnet contains faucets for providing you with test ETH and test Livepeer Token (LPT), which you will need to take other actions in Livepeer. The options for the faucets are present only when running with the `--rinkeby` flag enabled.

- Get some test eth from the eth faucet from https://faucet.rinkeby.io/. Make sure to use the Eth account address printed out above in `livepeer_cli`. Remember to add 0x as a prefix to address, if not present.

  - You can check that the request is successful by going to `livepeer_cli` and selecting Get node status. You should see a positive Eth balance.

- Now get some test Livepeer tokens. Pick Get test Livepeer Token.
  - You can check that the request is successful by going to `livepeer_cli` and selecting Get node status. You should see your Token balance go up.

Now that you have Livepeer token and ETH you can use them broadcast, bond and delegate, or even become a transcoding node:

- *Broadcasting To A Livepeer Node*
- *Bonding and Delegation*
- *Transcoding*

Install and start Geth

Geth is the Ethereum client, and you can run your own Geth instances instead of using the Livepeer testnet Geth instances. The instructions for installing geth are available on the Ethereum installation guide. Generally this is just downloading a binary file for your platform. (If you are using Parity, you can use the *–geth* flag to emulate Geth behavior)

The "connect yourself" tab on the Testnet Homepage provides instructions for how to initialize Geth and launch it. It can be summarized as:

- Create a geth data directory. For example:

```
$ mkdir ~/.lpGeth
```

  We recommend creating a new directory even if you already have one, so the Livepeer testing data will be stored separately.

- Download the genesis json rinkeby.json. It can be saved anywhere. It'll just be used once for the next step
- Initialize your local geth node with testnet genesis block. For example:

```
$ geth --datadir ~/.lpGeth init lptestnet.json
```

Note: Depending on your geth version, you may see a complaint about 'genesis.number' related to your .json file. To fix the issue, delete the "number" field in the json.

- Create a new geth account and provide a password:

```
$ geth --datadir ~/.lpGeth account new
```

- Copy this account address down somewhere and remember the password, as you'll need them when you start the Livepeer node.
- Start geth with the network id `858585` and the Livepeer testnet bootnode. For example:

```
$ geth --datadir ~/.lpGeth --networkid 858585 --bootnodes "enode://
↪2975123a0b613588a52a4cc80981a1d101ce4dc0176e62757b771237073bccbf4066b03b5c647d36fcbdd7422fda43
↪216.122.204:30303"
```

  Now the geth node should be running, and it should soon start downloading blocks.

Running a node on a private network

You can also create your own private network without connecting to the public test network. To do so you'll initialize a private ethereum chain using Geth.

Instructions for creating a private ethereum chain are on the geth README.

Start Livepeer:

```
$ livepeer --v 4 --devenv --ethAcctAddr <ethereum address> --ethPassword <eth account␣
↪pw>
```

If you are on the same machine, specify new ports for `rtmpAddr`, `httpAddr` and `cliAddr`. In this example, we added 1 to each of the default ports which are in use by the first node Consider creating a second ethereum account address in the new data directory:

```
$ livepeer --v 4 --devenv --rtmpAddr 127.0.0.1:1936 --httpAddr 127.0.0.1:8936 --
↪cliAddr 127.0.0.1:7936 --datadir <new datadir eg. ~/.livepeer2> --ethAcctAddr
↪<ethereum address> --ethPassword <eth account pw>
```

The second node should start. You're now running a private network where the nodes can play different roles such as broadcaster and transcoder. Note that if you become an transcoder within a private network, the `--serviceAddr` option might need to be set in order to match the on-chain Service URI (which you will set when registering the transcoder).

2.3 Broadcasting To A Livepeer Node

Broadcasting to Livepeer using existing broadcasting tools is easy. After a Livepeer node is running, it exposes an RTMP interface on port 1935. You can broadcast into Livepeer using this port.

2.3.1 Install Livepeer and Have the Node Running

The following instructions assume that you have followed the installation instructions

and have the node running.

Note: make sure you have deposited ETH if you would like to broadcast.

2.3.2 Broadcasting To A Local Node Using OBS

Start by reading our step by step guide

It is far more convenient to broadcast using existing tools that have features for screen capture, composites, overlays, multiple video and audio sources, etc. One such tool is OBS. To use OBS you have to change two settings:

- Settings -> Stream -> URL. Set it as `rtmp://localhost:1935`
- Settings -> Output -> Output Mode. Set it to Advanced. Ensure the following settings are enabled:
  - Encoder: x264
  - Rate Control: CBR
  - Keyframe Interval: 4
- Start streaming as usual.

If the broadcast is successful, you should see a log in the terminal like:

```
Video Created With ManifestID: 710ed610
```

If you have used `-currentManifest` to start your Livepeer node, you can verify that the broadcast is successful by running `curl http://localhost:8935/stream/current.m3u8`. It should return a valid HLS playlist like:

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-STREAM-INF:PROGRAM-ID=0,BANDWIDTH=4000000,RESOLUTION=1120x700
710ed610/source.m3u8
```

2.3.3 Playing The Local Video Stream

Make sure you have used `-currentManifest` to start your Livepeer node. You can watch your stream via:

- Playing the stream through the Livepeer media player. Use `http://localhost:8935/stream/current.m3u8` as the URL for the video stream.
- Playing the stream using `ffplay`

```
$ ffplay http://localhost:8935/stream/current.m3u8
```

2.3.4 Broadcasting To A Local Node Using FFMPEG

To broadcast using ffmpeg you can try the following command:

For Mac:

```
ffmpeg -f avfoundation -framerate 30 -pixel_format uyvy422 -i "0:0" -vcodec libx264 -
↪tune zerolatency -b 1000k -x264-params keyint=60:min-keyint=60 -acodec aac -ac 1 -
↪b:a 96k -f flv rtmp://localhost:1935/movie
```

For Linux:

```
ffmpeg -f dshow -framerate 30 -pixel_format uyvy422 -i "0:0" -vcodec libx264 -tune␣
↪zerolatency -b 1000k -x264-params keyint=60:min-keyint=60 -acodec aac -ac 1 -b:a␣
↪96k -f flv rtmp://localhost:1935/movie
```

You can now verify if the broadcast is successful.

2.3.5 Broadcasting To A Remote Node From Mobile

There is not currently a natively Livepeer aware mobile app, but much like *using OBS*, as described above, you can use any existing mobile broadcasting tool such as ManyCam on iOS or RTMPCamera on Android to broadcast into Livepeer.

Instead of setting the rtmp output to `localhost:1935`, you'll want to set it to a remote Livepeer node that you are running on a server. Replace `localhost` with the IP address of the server.

Make sure the node is started with `-currentManifest` to make this process easier.

2.3.6 Reaching Many Viewers at Scale

If you would like to take your output video and make it available via a conventional CDN, then you have the option to do so.

- Run a Livepeer node on a server, and expose ports `8935 and 1935`.
- Boot up the livepeer node with the –rtmpAddr 0.0.0.0 and -httpAddr 0.0.0.0 flags
- Configure your CDN to cache video content running at `http://hostname:8935/stream/{streamID}.m3u8`

Now any requests that come into your site or DApp for video streaming through Livepeer will pull the video from the network, but will be served off of a CDN. In the future, we would like to replace this option with the p2p network that Livepeer forms around a stream.

2.3.7 FAQ

Check out our Broadcasting Forum for [frequently asked questions] (https://forum.livepeer.org/c/using-livepeer-for-broadcasting)

If you have any questions, reach out to Chris Hobcroft on our [community chat] (https://discord.gg/RR4kFAh)

2.4 Bonding and Delegation

Bonding is how most users participate in the Livepeer protocol and add value to the network long term by vetting and electing the best nodes to provide transcoding and video services to the network. See the Delegator Wikipage which describes what bonding and delegation is, how to do it, as well as tutorials on how to weigh various Transcoder statistics

The protocol mints new token every round and rewards participation in the network as a Delegator or Transcoder

2.4.1 Delegating using Explorer

Explorer is a tool we built to interface with livepeer cli in a less technical way

- How to Delegate

2.4.2 Assessing Transcoders

Assess transcoders based on performance, statistics and social campaigns

- Social Campaigns can be found in the Forum
- Stats can be viewed on Explorer
- Definitions and examples are on the Delegator Wiki

2.4.3 Delegating using the Terminal

In order to bond your Livepeer Token (LPT) you use `livepeer_cli`.

```
$ livepeer_cli
```

Make sure you have ETH and LPT and are running a Livepeer node as described in Getting Started.

The CLI presents options to

- `Bond`
- `Unbond`
- `Withdraw Bond`

When you choose to bond, it will present you with a table of transcoders to choose from in order to bond towards. You should select a transcoder based upon many factors including the fees that they're charging and sharing back to you, their statistics on past performance, and the social data that they've shared through forum posts or other Livepeer related resources. In the end, you're making a decision about whom you think will add the most value to the Livepeer network.

Keep in mind that if you delegate towards a high performing, honest transcoder you will earn a portion of the fees that they receive. If you delegate towards a transcoder who cheats or doesn't reliably do work in the network, you will lose out on the economic opportunity of fees and inflationary token issuance. Select wisely!

- Choose the option to `Bond` when you'd like to bond.

2.4.4 Unbonding

A ***Guide to Unbonding and Claiming Fees*** can be found on our Delegator Wiki

- Choose the option to `Unbond` when you'd like to withdraw your bond from a particular transcoder.

You will not yet be able to access your token while it's unbonding for the length of the `UnbondingPeriod`. You can rebond during this period to the same or a different transcoder.

- At the end of the `UnbondingPeriod` you can choose the option to `Withdraw` which will now give you access to your unbonded token.

2.5 Transcoding

Transcoding is the process of taking an input video in one format and bitrate, and converting it into many formats and bitrates to make it playable on the majority of devices on the planet at any connection speed.

In The Livepeer network, nodes who play the role of transcoder, perform this very important function, and as a result it's important that they have high bandwidth connections, sufficient hardware, and reliable devOps practices. These nodes are delegated towards and elected to perform this role, and they are rewarded with the ability to earn fees from the network.

Quicklinks:

Transcoder Megathread on Forum

Transcoder Election Dashboard (currently Rinkeby testnet)

Transcoder campaign thread

Livepeer Whitepaper

Transcoder chat

2.5.1 Becoming a Transcoder

We'll walk through the steps of becoming a transcoder on the test network. Start livepeer with the `--transcoder` flag:

```
$ livepeer --rinkeby --transcoder
```

Run `livepeer_cli`, and make sure you have test ETH and test LPT as described in Getting Started.

```
$ livepeer_cli
```

You should see the Transcoder Status as "Not Registered".

Pick "Become a transcoder" in the wizard. Make sure to choose "bond to yourself".

At this point the interface will ask you to set 3 values if you have not set them already:

- `PricePerSegment` - How many base unit Livepeer Token (LPT) will you charge to transcode a 4 second segment of video? Keep in mind that 1 LPT == 10^18 base unit LPT. Example 1000.
- `FeeShare` - You will collect fees from broadcasters based upon the above price that you charge and how many segments you transcode. What % of fees would you to keep? The remaining fees will be passed to your delegators. Example 98%.
- `BlockRewardCut` - All delegators are entitled to their share of newly minted inflationary Livepeer Token. Set the cut as a percentage that you will take from delegators who delegate towards you in exchange for doing the work of performing this valuable service of transcoding reliably. Example: 3%.
- `Public IP:Port` - Transcoders must be publicly accessible at the IP:port in order to receive streams from broadcasters.

If Successful, you should see the Transcoder Status change to "Registered"

Wait for the next round to start, and your transcoder will become active. At this point, the Livepeer node should handle everything for you. The important thing is that you keep the node running.

2.5.2 FAQ

After running the transcoder for a while, I get an error that says "too many open files".

- This means you have to increase the default file limit. This is a requirement for running an IPFS node. Since Livepeer transcoders run an internal IPFS node, we also have that requirement. The default file limit is 1024, increasing it to something like 4096 should be good. See this forum post for more details.

I get a lot of error messages saying things like "Error with x:EOF". And a lot of times, the transcoder doesn't do anything when it's suppose to take some action (like call `reward`, do transcoding jobs, etc).

- This is most likely because the connection between the Livepeer node and the Ethereum network is flaky. It is recommended to run a local `Geth` or `Parity` node when running a Livepeer transcoder. If you have a local `Geth` or `Parity` running, you can use the `--ethIpcPath` flag to specify the local IPC file location, which is a much more stable way to connect to the Ethereum network.

I get an error that looks something like "failed to estimate gas needed: gas required exceeds allowance or always failing transaction".

- This is because the gas estimator is giving incorrect estimates. To fix it, you can manually pass in a gas limit using `-gasLimit`. For example, `$ livepeer -transcoder -gasLimit 400000`.

What does being 'publicly accessible' mean? Can I run a transcoder from home?

- The transcoder should be reachable by broadcasters via the public IP and port that is set during transcoder configuration. Transcoders will not be able to serve the Livepeer network if they are behind a NAT (eg, a home router). If this is the case, special accommodations must be made for the transcoder, such as port forwarding or putting the the transcoder in the DMZ. The only port that is required to be public is the one that was set during the transcoder registration step (default 8935). Be aware that there are many risks to running a public server. Only set up a transcoder if you are comfortable with managing these risks.

What is the Service URI? Does this need to be an IP?

The Service Registry acts as a discovery mechanism to allow broadcasters to look up the addresses of transcoders on the network. Transcoders register their Service URI at configuration time; this is submitted to the blockchain as a standalone transaction. While the configuration tool only asks for your IP:port, the URI stored on the blockchain in the form of `https://IP:port`. Transcoders are expected to provide a consistent and reliable service, so IPs here *should* remain static. However, a host (DNS) name is also allowed for the service URI to give transcoders some flexibility.

What does this error mean? "Service address https://127.0.0.1:4433 did not match discovered address https://127.1.5.10:8935; set the correct address in livepeer_cli or use -serviceAddr"

- When starting up, the transcoder checks if the current public IP matches the IP that is stored on the blockchain. If there is a mismatch, there is a possibility that your node is not publicly accessible. Override the locally inferred IP address by setting `-serviceAddr IP:port` to what is on the blockchain. Ensure your node is actually accessible at that address.

TODO: These documents could be expanded with far more information about the transactions that a Livepeer Transcoder has to submit on a regular basis to avoid being penalized and to earn their rewards and fees.

2.6 Developing on Livepeer

2.6.1 Building Video Dapps

- Video-based Dapps (for example, livepeer.tv)
- Infrastructure tools and services for broadcasters or live streamers (for example, SAAS services on top of Livepeer)
- Livepeer Player - A react component for playing live video - https://github.com/livepeer/livepeerjs/tree/master/packages/chroma

2.6.2 Building Livepeer Protocol Dapps

- Dapps for the Livepeer ecosystem. (for example, livepeer protocol explorer or Supermax)

2.6.3 Building Tools for Livepeer

- SDKs for Livepeer (for example, livepeerjs-sdk or livepeerjs-graphql)
- Client implementation for Livepeer (for example, go-livepeer)

2.6.4 Open Projects

Livepeer also posts open problems for discussion, ideas, and collaboration on Github. Check out:

- Open Project Proposals

- Open Research Areas

2.6.5 Contributing to Livepeer

For developers who are looking for interesting to problems to work on related to decentralized tech, blockchain, cryptocurrency, video engineering, and peer-to-peer networking, Livepeer may provide some interesting challenges. The three technical areas that Livepeer focuses on today are:

- Protocol implementation (Smart Contract)
- Livepeer Node (Distributed Systems / Networking)
- Livepeer Media Server (Video Engineering)

For the protocol , you can follow the protocol repo. It requires some background in Solidity and the Livepeer Whitepaper.

For the livepeer node, check out the go-livepeer repo. It requires some understanding of Golang and Geth. Setting up a development enviroment can be done by following **'these instructions'_**.

For the livepeer media server implementation, take a look at the LPMS repo. It requires some video engineering knowledge. The demuxed conf videos and the Apple Live streaming doc are good resources to start learning.

If you're interested in the any of the above challenges, or are building video features into an application, jump into our development chat room on Discord and join the conversation.

2.7 License

2.7.1 Contact

Questions? Email contact@livepeer.org

2.8 Need Help

The Livepeer team and community are available to help with any additional questions. You can find them on:

- Discord chat room: https://discord.gg/7wRSUGX
- Forum: http://forum.livepeer.org
- Twitter: http://twitter.com/LivepeerOrg
- Reddit: http://reddit.com/r/livepeer