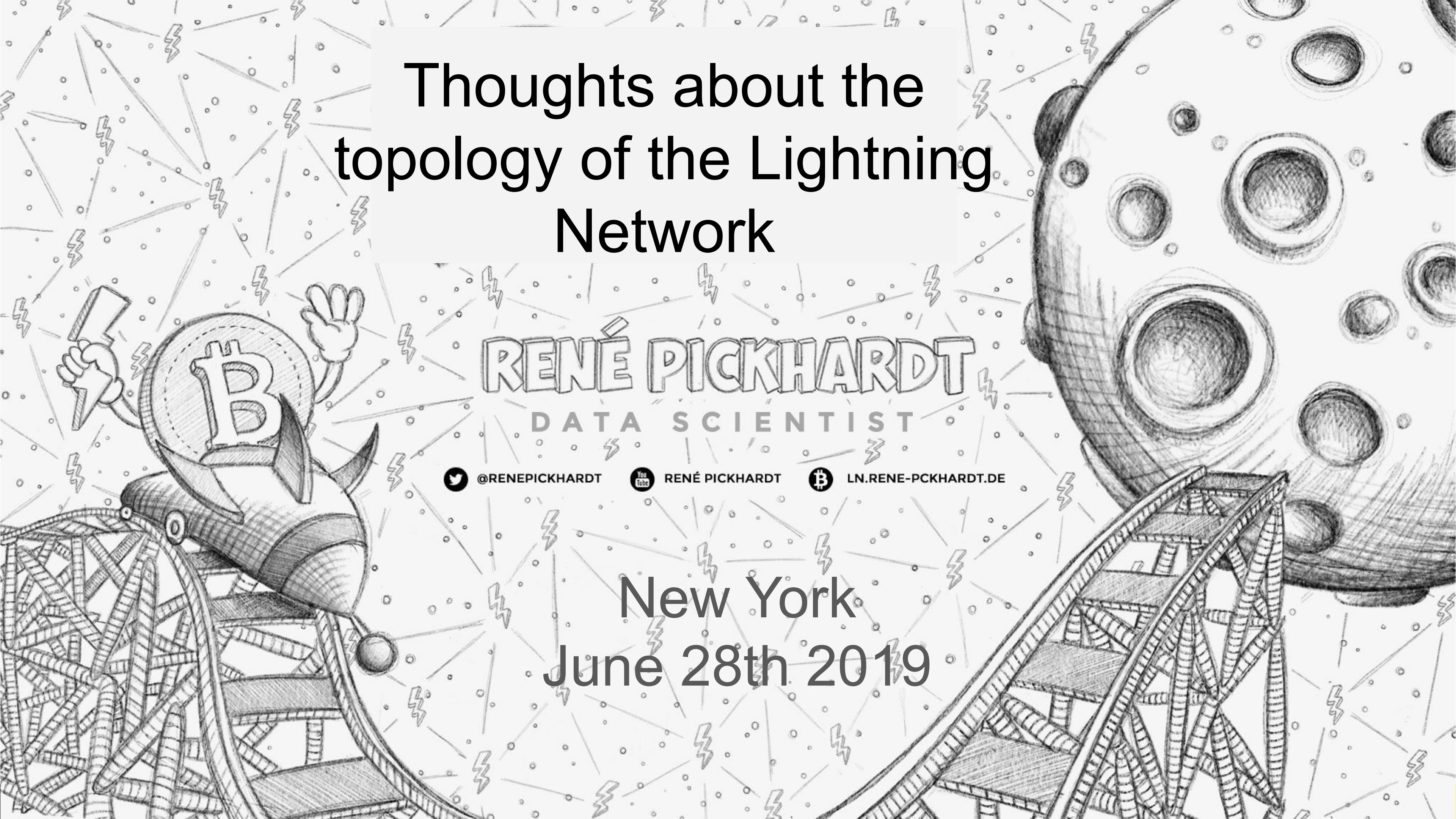
Thoughts about the topology of the Lightning Network



New York
June 28th 2019

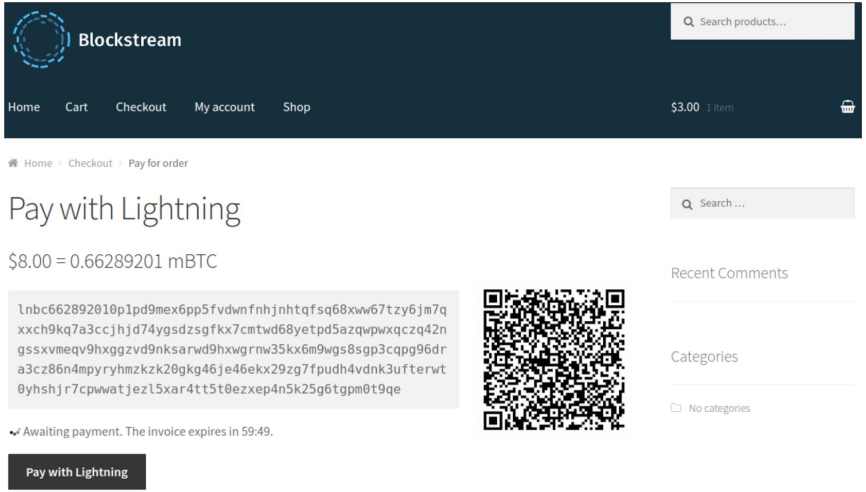
Disclaimer

- Please note the copyright notice at the end of the slide deck
- The pdf version - which you probably read has some weird formatting problems.
  - https://docs.google.com/presentation/d/1-eyceLISmcLpbPJLzj6_CnVYQdo1AUP3y5XD716U-Lg has the source files without formatting problems and with animations
- This slidedeck is crowd funded (see last slide) if you wish to learn more or contribute
- I do not take any guarantee that the information in this slide deck are 100% correct.
  - Sometimes I made simplifications: which I point out
  - Also I could just have misunderstood something or just made a mistake

Lightning Charge Powers Developers & Blockstream Store

Jan 16, 2018 by Christian Decker, Ph.D. Rusty Russell

With the release of the 1.0 spec for Lightning and the verification of interoperability among the three Lightning teams, the Lightning Network is racing forward. Bitcoin users are now able to take advantage of instant bitcoin transactions, improved scalability, and lower fees—enabling many new use cases, such as micropayments. In addition, like Bitcoin itself, the Lightning Network is



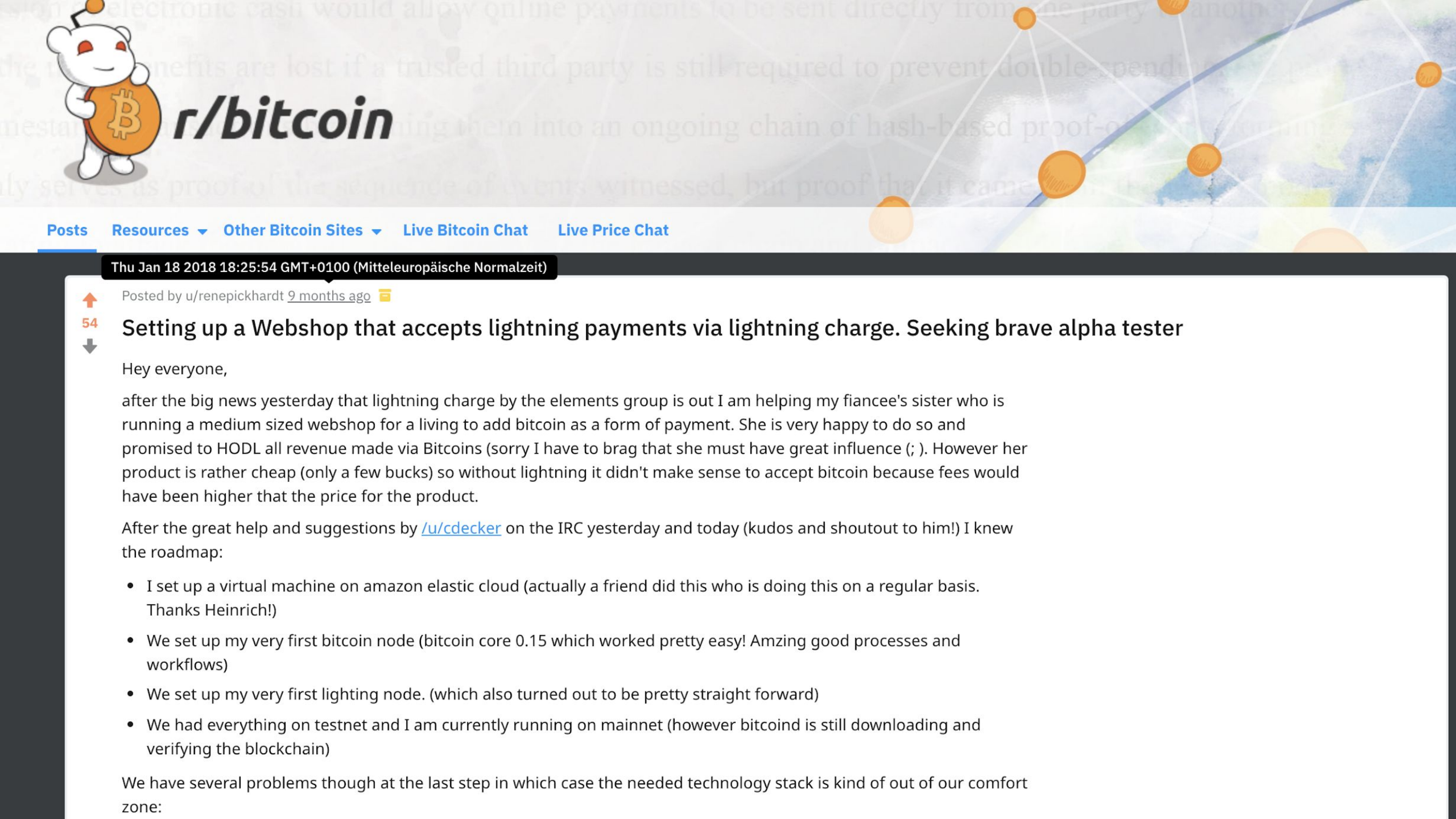
Pay with Lightning

$8.00 = 0.66289201 mBTC

```
lnbc662892010p1pd9mex6pp5fvdwnfnhjnhtqfsq68xww67tzy6jm7q
xxch9kq7a3ccjhjd74ygsdzsgfkx7cmtwd68yetpd5azqwpwxqczq42n
gssxvmeqv9hxggzvd9nksarwd9hxwgrnw35kx6m9wgs8sgp3cqpg96dr
a3cz86n4mpyryhmzkzk20gkg46je46ekx29zg7fpudh4vdnk3ufterwt
0yhshjr7cpwwatjezl5xar4tt5t0ezxep4n5k25g6tgpm0t9qe
```

Awaiting payment. The invoice expires in 59:49.

Lightning Charge and the WooCommerce Lightning Gateway are both available as part of the Elements Project: download them to begin work on your own c-lightning projects. We invite you to participate by reporting issues, asking questions, and making contributions through the Lightning



r/bitcoin

Posts · Resources · Other Bitcoin Sites · Live Bitcoin Chat · Live Price Chat

Thu Jan 18 2018 18:25:54 GMT+0100 (Mitteleuropäische Normalzeit)

54

Posted by u/renepickhardt 9 months ago

Setting up a Webshop that accepts lightning payments via lightning charge. Seeking brave alpha tester

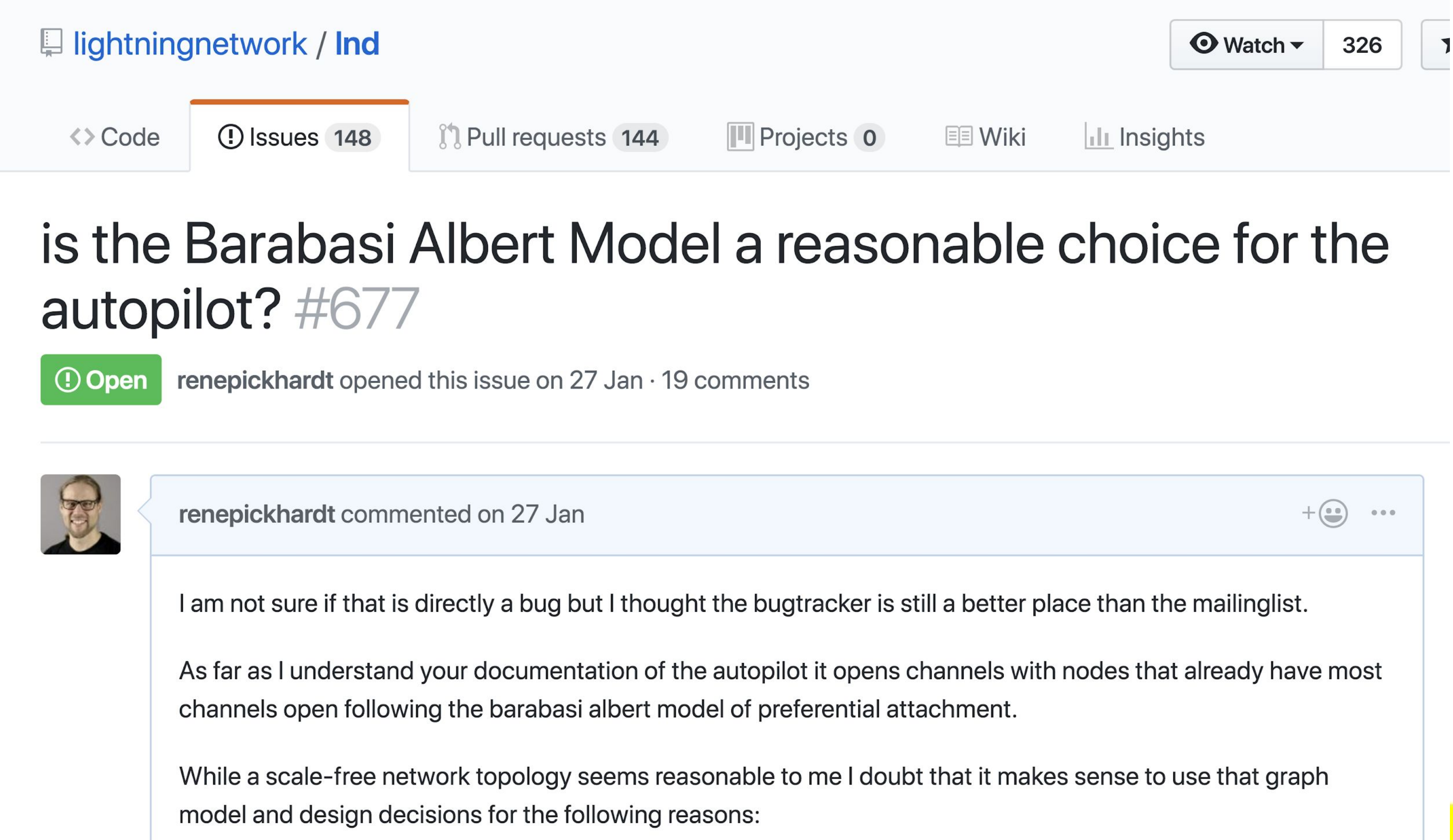
Hey everyone,

after the big news yesterday that lightning charge by the elements group is out I am helping my fiancee's sister who is running a medium sized webshop for a living to add bitcoin as a form of payment. She is very happy to do so and promised to HODL all revenue made via Bitcoins (sorry I have to brag that she must have great influence (;). However her product is rather cheap (only a few bucks) so without lightning it didn't make sense to accept bitcoin because fees would have been higher that the price for the product.

After the great help and suggestions by /u/cdecker on the IRC yesterday and today (kudos and shoutout to him!) I knew the roadmap:

- I set up a virtual machine on amazon elastic cloud (actually a friend did this who is doing this on a regular basis. Thanks Heinrich!)
- We set up my very first bitcoin node (bitcoin core 0.15 which worked pretty easy! Amzing good processes and workflows)
- We set up my very first lighting node. (which also turned out to be pretty straight forward)
- We had everything on testnet and I am currently running on mainnet (however bitcoind is still downloading and verifying the blockchain)

We have several problems though at the last step in which case the needed technology stack is kind of out of our comfort zone:



is the Barabasi Albert Model a reasonable choice for the autopilot? #677

renepickhardt opened this issue on 27 Jan · 19 comments

renepickhardt commented on 27 Jan

I am not sure if that is directly a bug but I thought the bugtracker is still a better place than the mailinglist.

As far as I understand your documentation of the autopilot it opens channels with nodes that already have most channels open following the barabasi albert model of preferential attachment.

While a scale-free network topology seems reasonable to me I doubt that it makes sense to use that graph model and design decisions for the following reasons:

(Current?) Autopilot in a nutshell

- A recommender system for a node
- Suggests channels to which the node should open channels
- Supposed to help users to automatically provide access to the lightning network

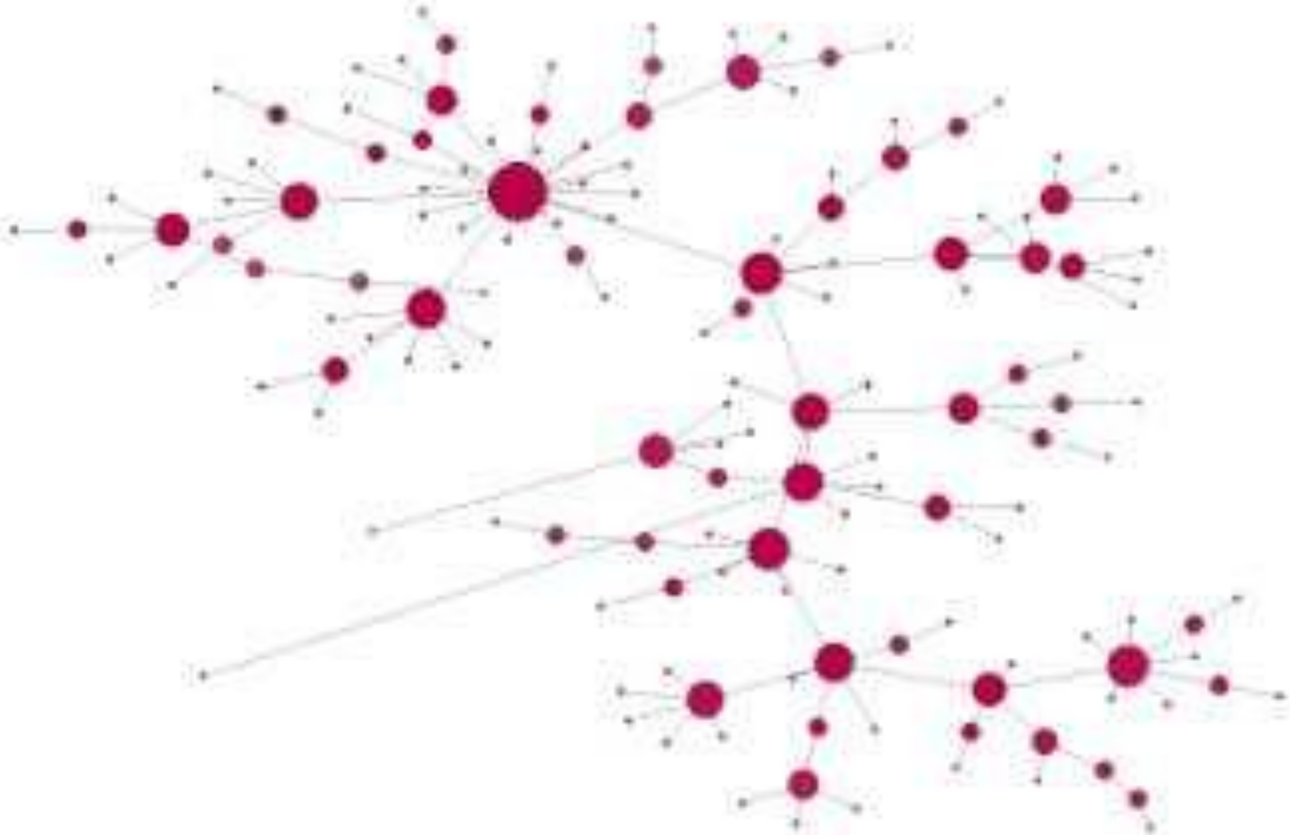
Autopilots could do so much more

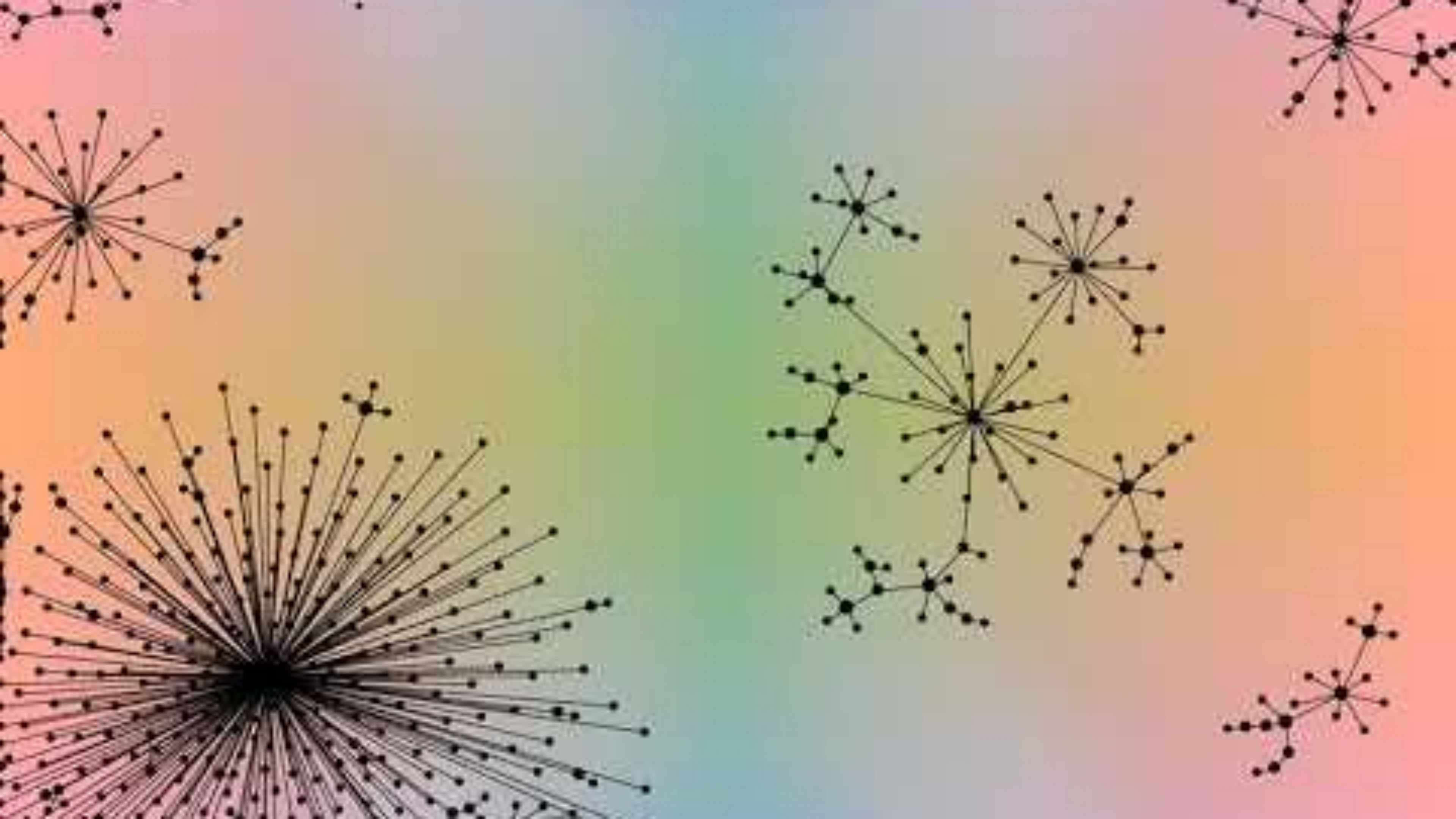
- Balance selection / maintenance
- Closing channels
- Blacklisting nodes
- Share information for pathfinding
- Do the pathfinding
- Do probing to check the health of a node on the network
  - How easily can I reach other nodes?

But there is criticism too: Do we really need them?

How does (did?) the LND Autopilot work?

- Barabasi Albert Model
  - Aka preferential attachment
  - Aka the rich get richer
- Random process
  - Introducing randomization is often a surprisingly strong idea
  - Don't use a uniform distribution
  - Use a weighted distribution with popularity (measured by node degree) as a weight
- Result will be a graph with following properties
  - Scale-free
    - Similar as social network graphs (seems natural)
  - Small world network (in particular small diameter)
    - Likely high reachability
  - Power law distribution of node degrees
- Interface for scores exist but it is weird
  - https://github.com/lightningnetwork/lnd/issues/2665 (arithmetic mean vs harmonic mean)



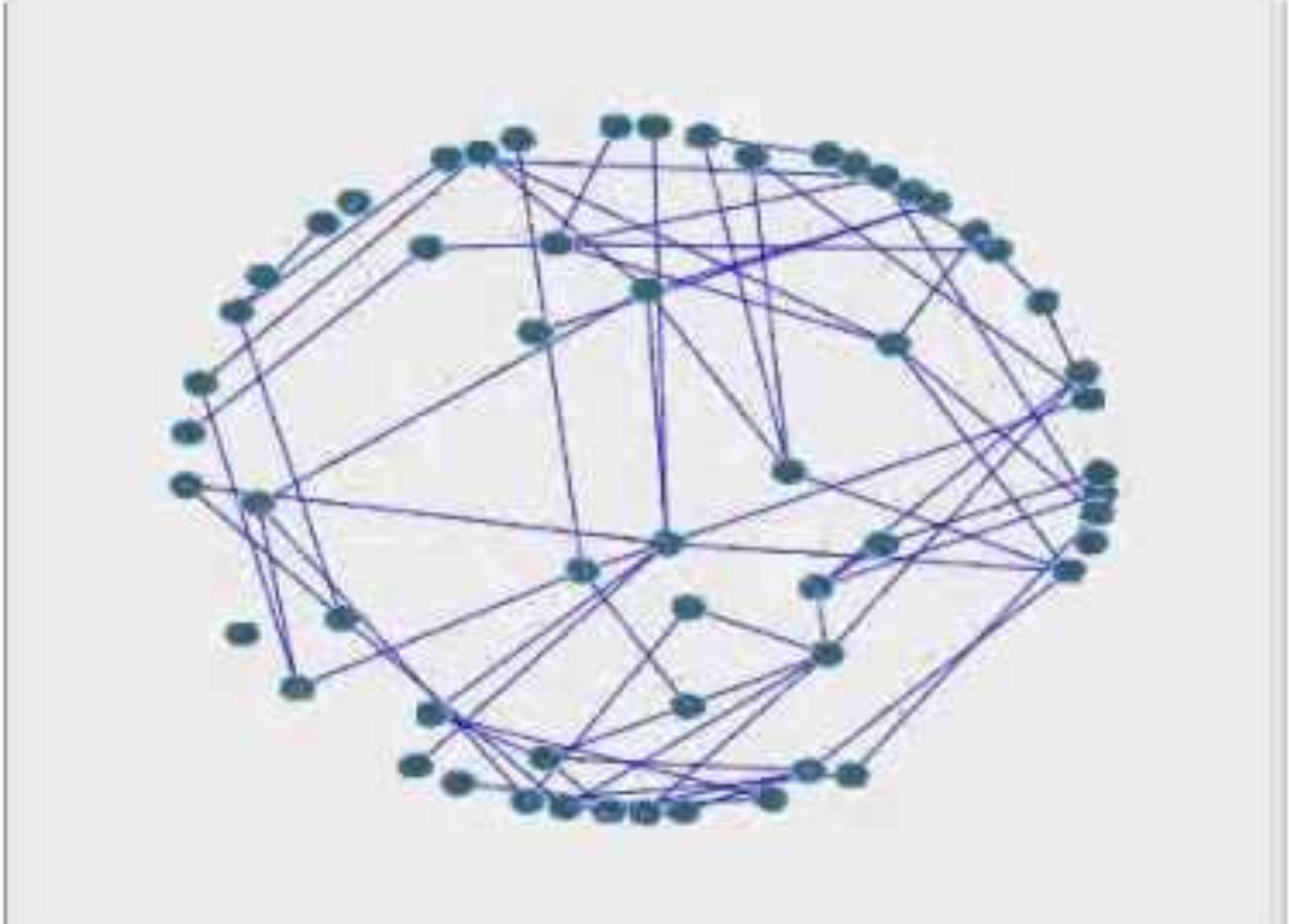


Pitfalls with the lnd autopilot

- It creates highly popular hubs
  - By the algorithm the hubs will have mainly inbound channels
  - Could technically overwhelm nodes

- Number of triangles in the graph are relatively low
  - Single edges (aka payment channels) become very important to ensure connectivity

- All created channels have the same balance
  - It would make way more sense to choose a reasonable capacity for each candidate



Why is it important to think about the autopilot early

- Even with the lightning network blockchain transactions will be precious
  - We can change about 7 channels per second
  - That is 630'000 per day
- If we grow the lightning network for a couple of years we
  - Might detect the topology of the lightning network was not so great
  - Can't change that topology quickly
- Routing success depends on network topology
  - max flow min cut (in particular with upcoming AMP Routing)
- Less Routing errors → better user experience → more utility

Solution: Grow the network in a smart way to avoid future problems

(Not perfect) analogy to routing and routed protocols

- The IP forwarding algorithm is a routed protocol.
  - Similarly: Onion routing of HTLCs is a routed protocol

- A routing protocol helps routers to build and maintain routing tables
  - Similarly: the autopilot helps nodes to build and maintain channels
  - Channels in turn determine how routing tables will be created.
  - Also gossip is obviously a routing protocol as it spreads the topology

- Historically on the Internet we had routed protocols before routing protocols
  - People have been setting routing tables manually
  - Same holds true in the Lightning Network (channels are created manually)

My approach / first prototype

- 4 different strategies for node selection
  - 2 Strategies which are good for the user
    - Barabasi Albert (look for popular loads)
    - Respect centrality
  - 2 Strategies which are healthy for the network
    - Decrease diameter
    - Erdoes Renyi (draw from uniform distribution just to increase entropy)
- Following an estimation / suggestion of balances for channels
- Let's look at the 4 strategies!
  - Barabasi Albert is clear as we already discussed it
  - Erdoes Renyi is also clear

Betweenness centrality

- For each pair of nodes calculate the shortest path between them
- For each node count on how many shortest path it lies
- The relative frequency is the betweenness centrality
- Nodes that have a high betweenness centrality can reach many other nodes quickly
- Being connected to such a node is certainly desirable
    - So connect to nodes with probability proportional to their betweenness centrality
- Connecting to such a node will generally increase their centrality

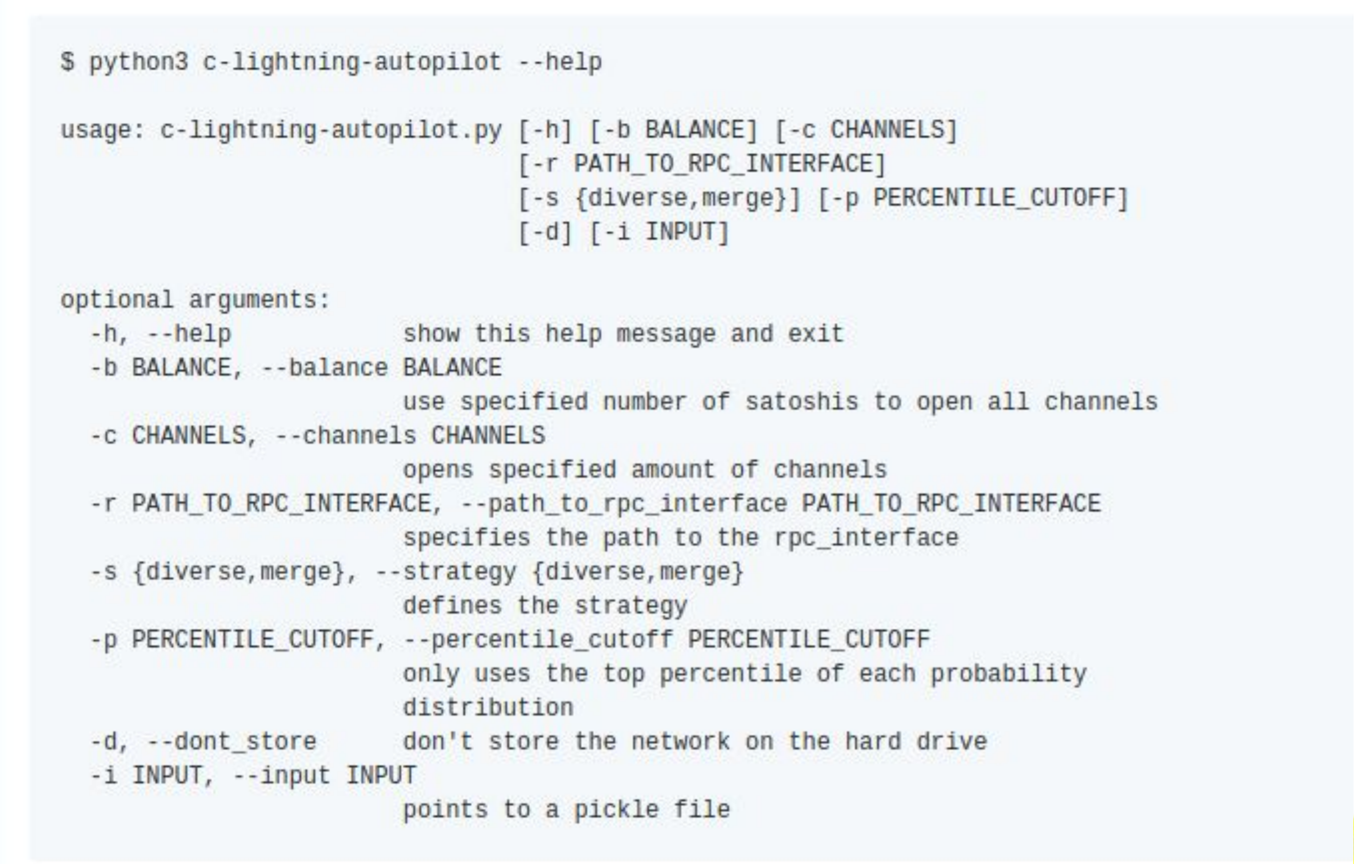
Decreasing the diameter of the network

- For each pair of nodes calculate the shortest path between them
- Keep track of the length of this path
- Find the pair for which the path is longest
- With higher probability open a channel to nodes which are increasing the diameter
- Makes the own node a little bit more central

Some technical details

- All strategies implemented to produce probability distributions
  - There is always some randomization coming from sampling
  - Can be smoothed with uniform distribution
  - Can be cutoff to only consider top candidates according to a strategy
- There are two different strategies
  - Diverse
    - Average the distributions and sample from the average
  - Merge
    - Sample from each and merge the top results
- Cutoffs
  - It is possible to only get a predefined top percentile of the distribution

API overview

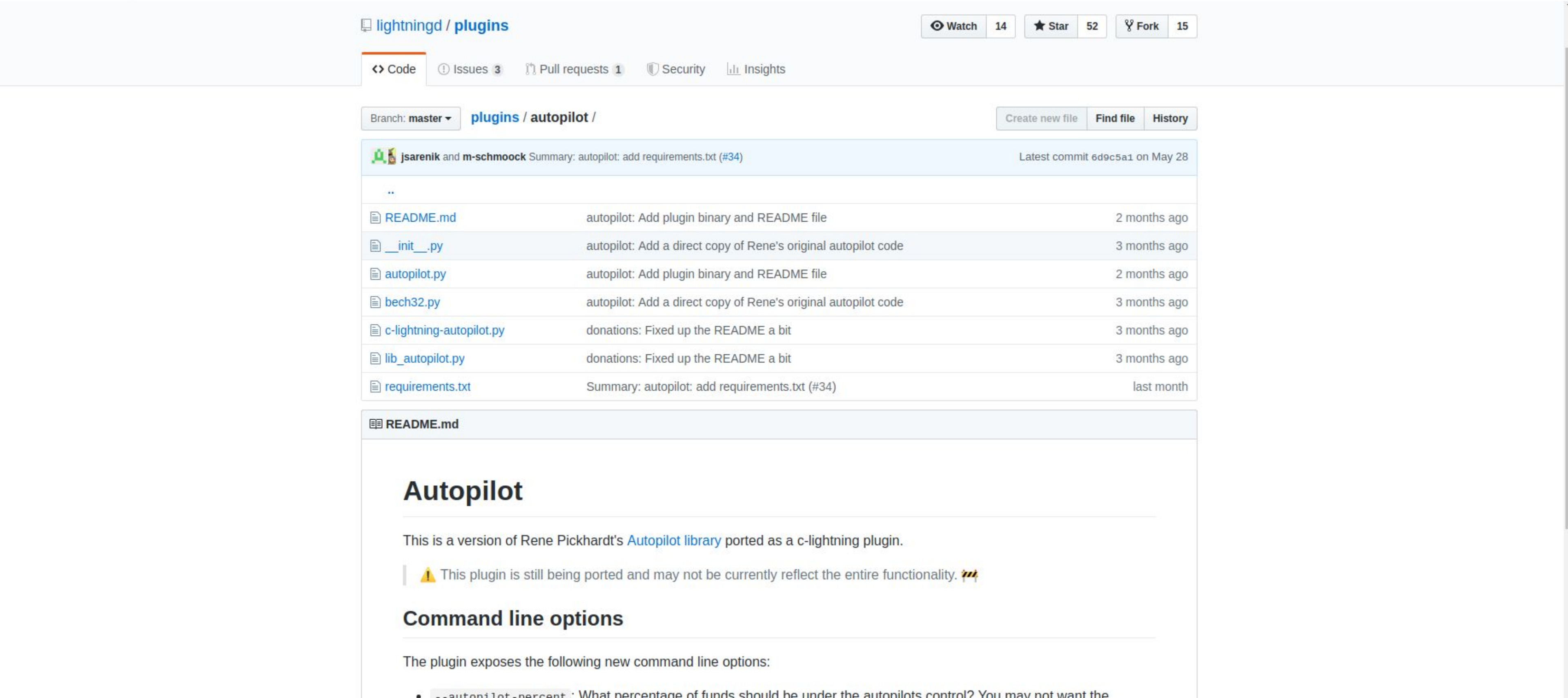


```
$ python3 c-lightning-autopilot --help

usage: c-lightning-autopilot.py [-h] [-b BALANCE] [-c CHANNELS]
                                [-r PATH_TO_RPC_INTERFACE]
                                [-s {diverse,merge}] [-p PERCENTILE_CUTOFF]
                                [-d] [-i INPUT]

optional arguments:
  -h, --help            show this help message and exit
  -b BALANCE, --balance BALANCE
                        use specified number of satoshis to open all channels
  -c CHANNELS, --channels CHANNELS
                        opens specified amount of channels
  -r PATH_TO_RPC_INTERFACE, --path_to_rpc_interface PATH_TO_RPC_INTERFACE
                        specifies the path to the rpc_interface
  -s {diverse,merge}, --strategy {diverse,merge}
                        defines the strategy
  -p PERCENTILE_CUTOFF, --percentile_cutoff PERCENTILE_CUTOFF
                        only uses the top percentile of each probability
                        distribution
  -d, --dont_store      don't store the network on the hard drive
  -i INPUT, --input INPUT
                        points to a pickle file
```

C-lightning plugin (lightning-cli autopilot-run-once)



plugins / autopilot /

jsarenik and m-schmoock Summary: autopilot: add requirements.txt (#34) Latest commit 6d9c5a1 on May 28

..

README.md	autopilot: Add plugin binary and README file	2 months ago
__init__.py	autopilot: Add a direct copy of Rene's original autopilot code	3 months ago
autopilot.py	autopilot: Add plugin binary and README file	2 months ago
bech32.py	autopilot: Add a direct copy of Rene's original autopilot code	3 months ago
c-lightning-autopilot.py	donations: Fixed up the README a bit	3 months ago
lib_autopilot.py	donations: Fixed up the README a bit	3 months ago
requirements.txt	Summary: autopilot: add requirements.txt (#34)	last month

README.md

Autopilot

This is a version of Rene Pickhardt's Autopilot library ported as a c-lightning plugin.

> ⚠ This plugin is still being ported and may not be currently reflect the entire functionality. 🚧

Command line options

The plugin exposes the following new command line options:

- `--autopilot-percent`: What percentage of funds should be under the autopilots control? You may not want the

Balance selection for candidates

- Estimate how much of the UTXO should be allocated to each channel

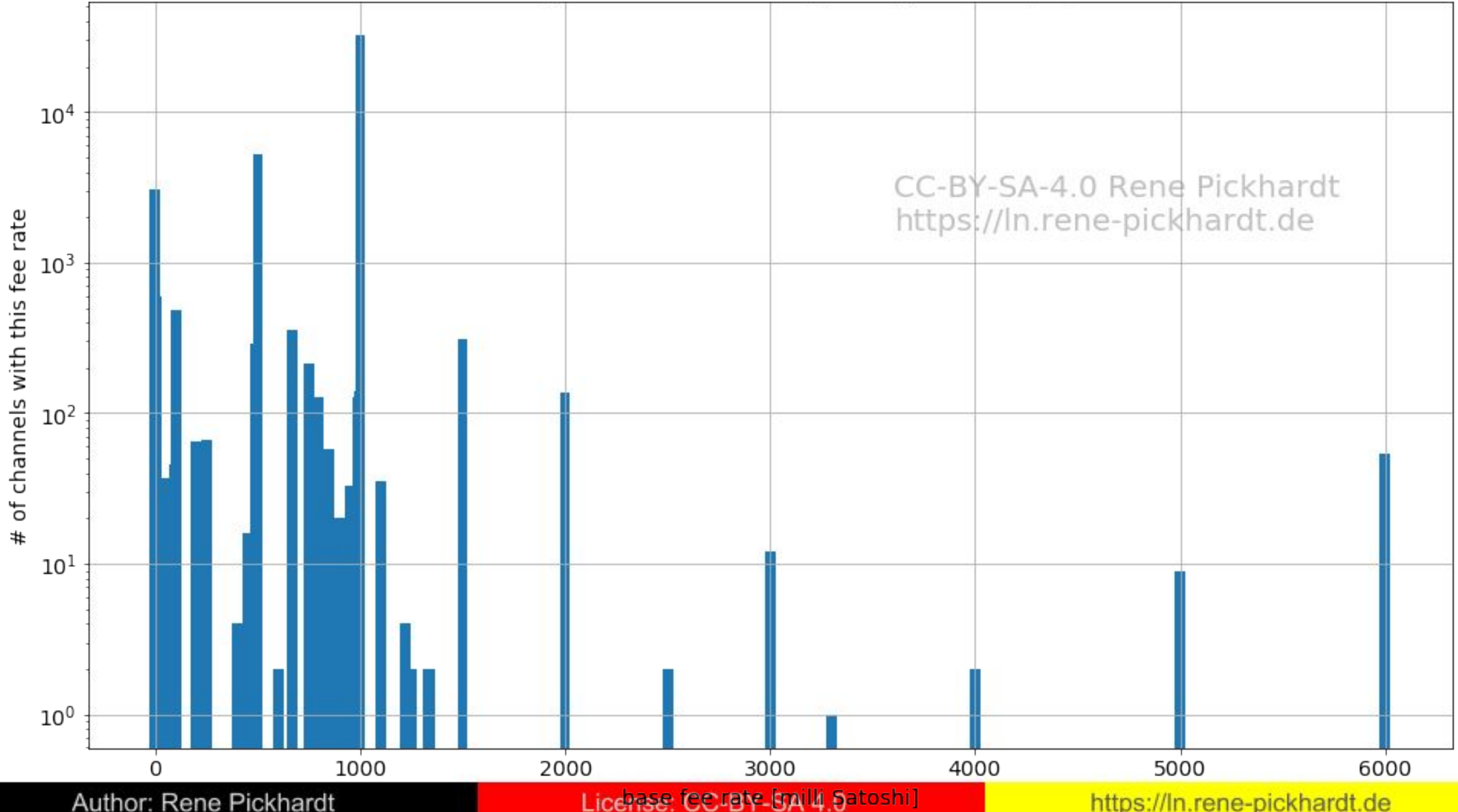
- Sum the the total balance of all channels for each node in the candidate set

- Create a probability distribution proportional to the results

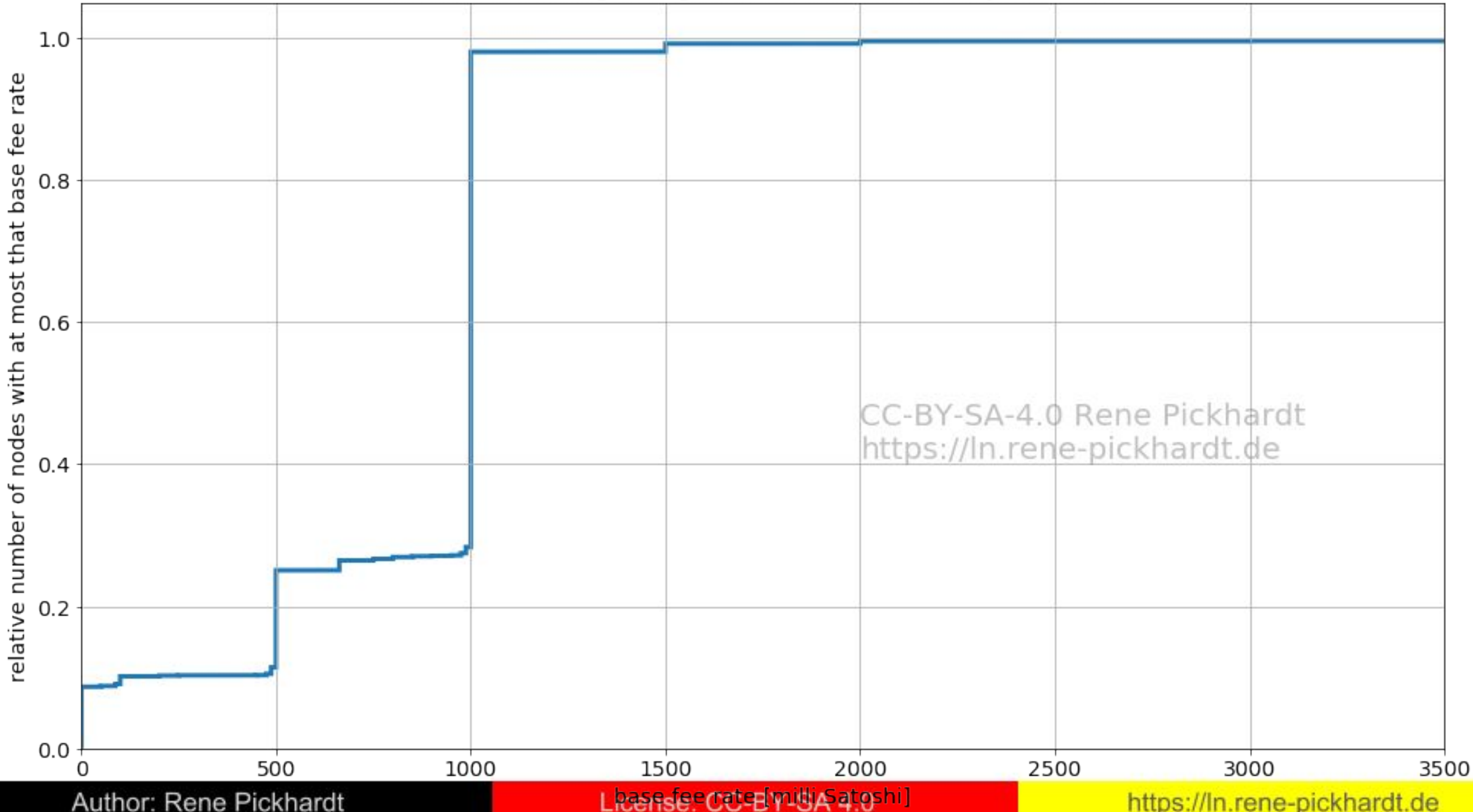
- Maybe smooth it to avoid dust channels

Some statistics about the lightning network with respect to routing fees

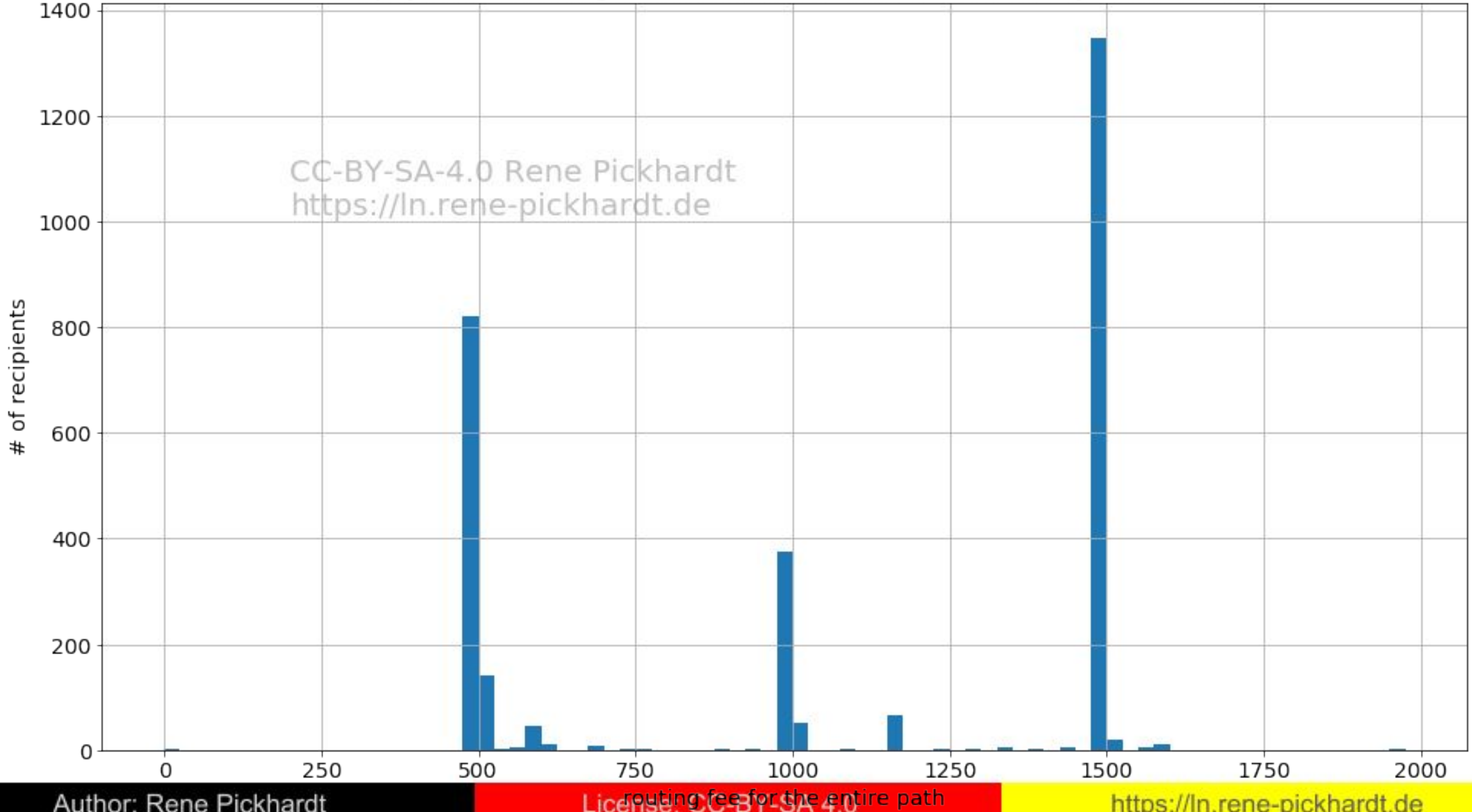
Distribution of routing base fee rates of Lightning Network payment channels

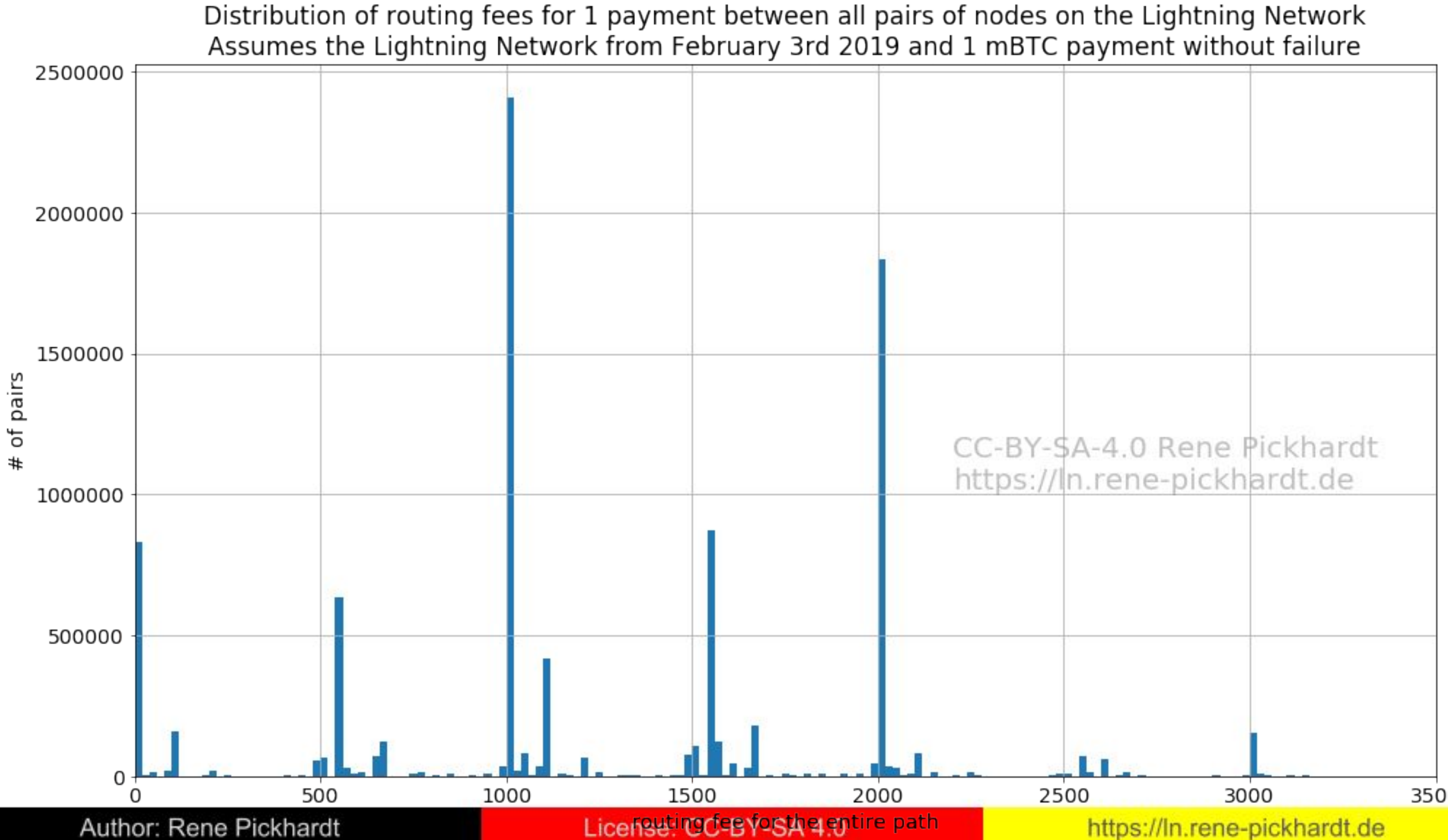


Cummulative Distribution Function of used base fee rate
Assumes the Lightning Network from February 3rd 2019

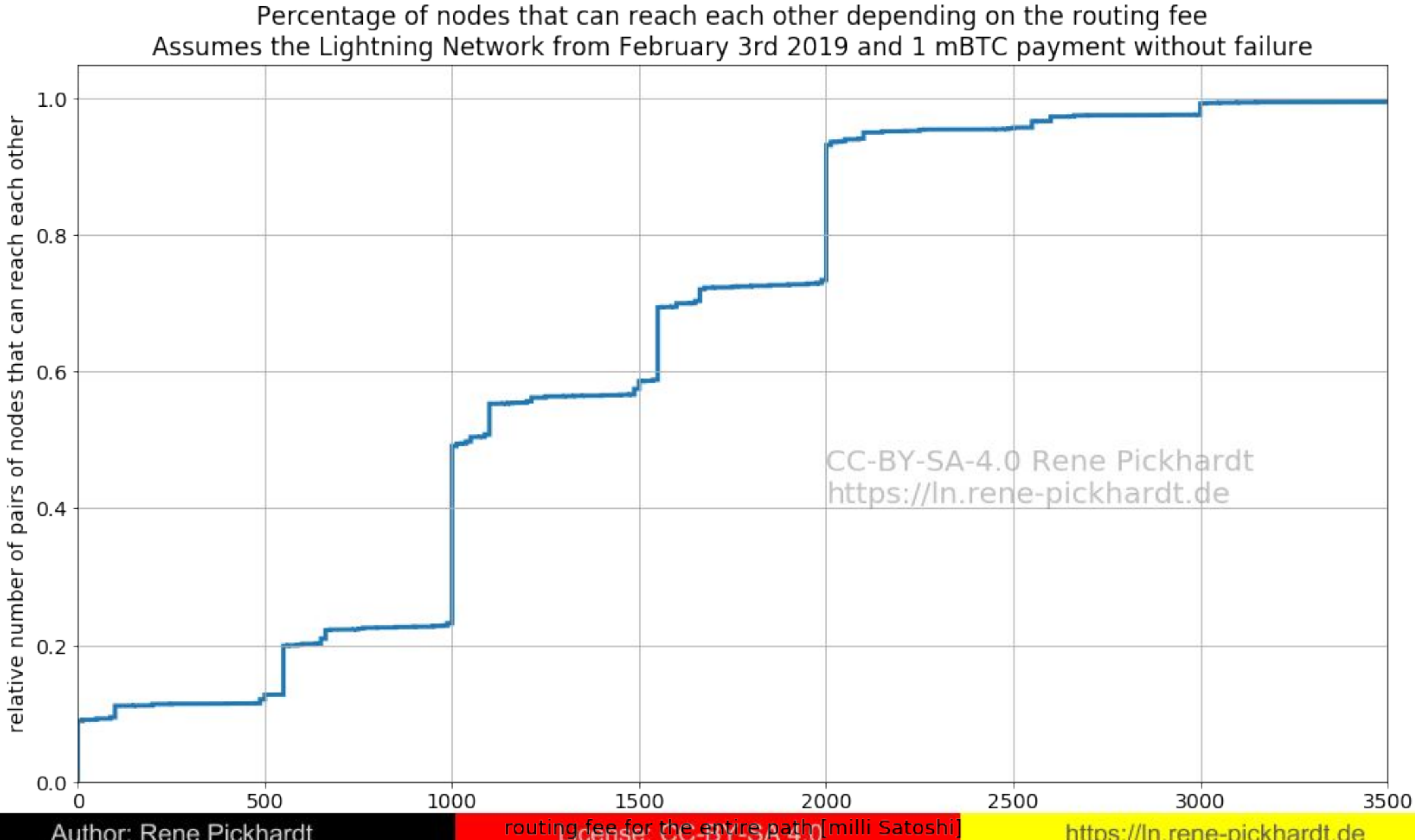


Distribution of routing base fee rates of Lightning Network payment channels
for payments starting from Rene Pickhardt's node to every other node





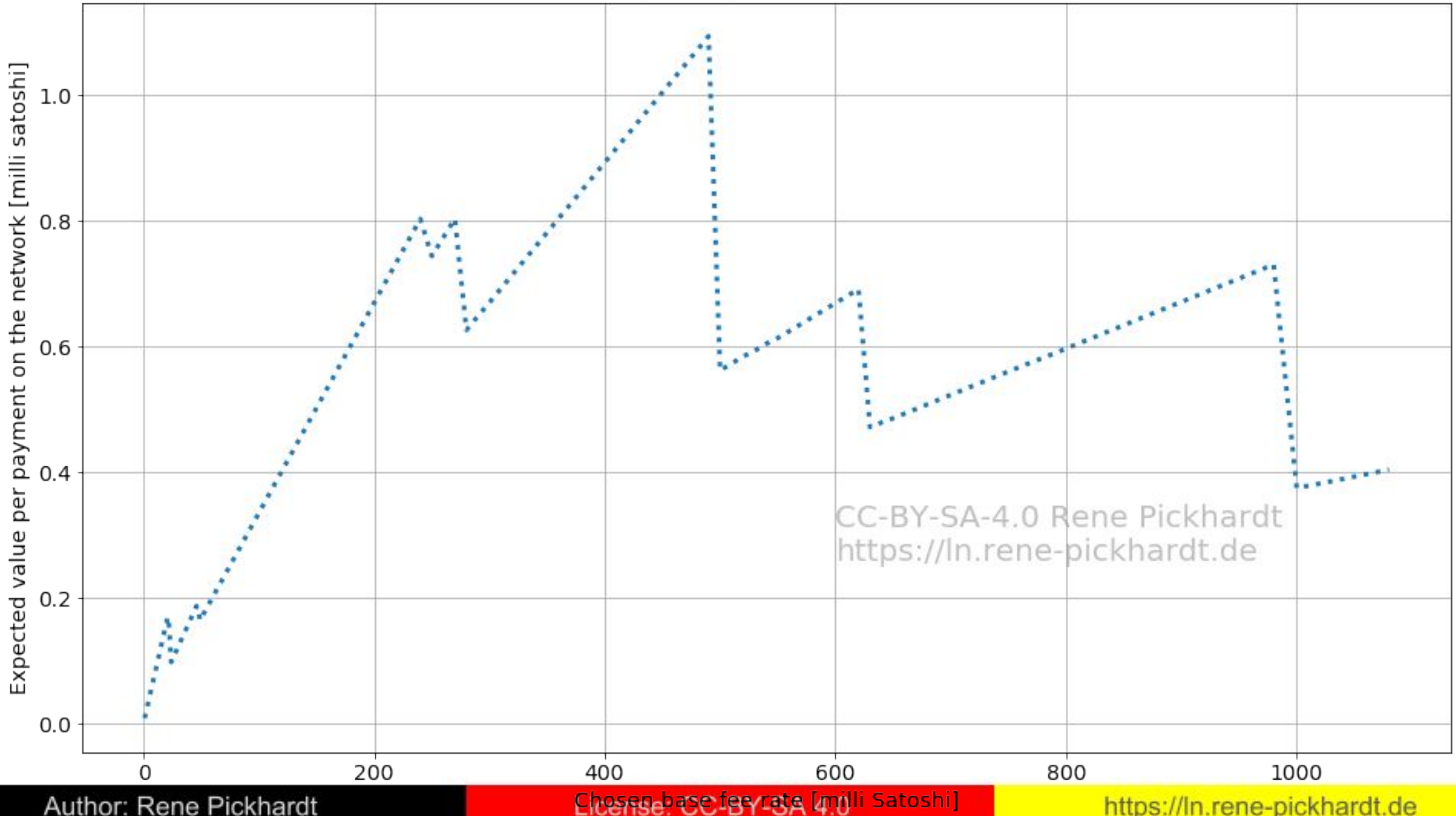
Distribution of routing fees for 1 payment between all pairs of nodes on the Lightning Network
Assumes the Lightning Network from February 3rd 2019 and 1 mBTC payment without failure

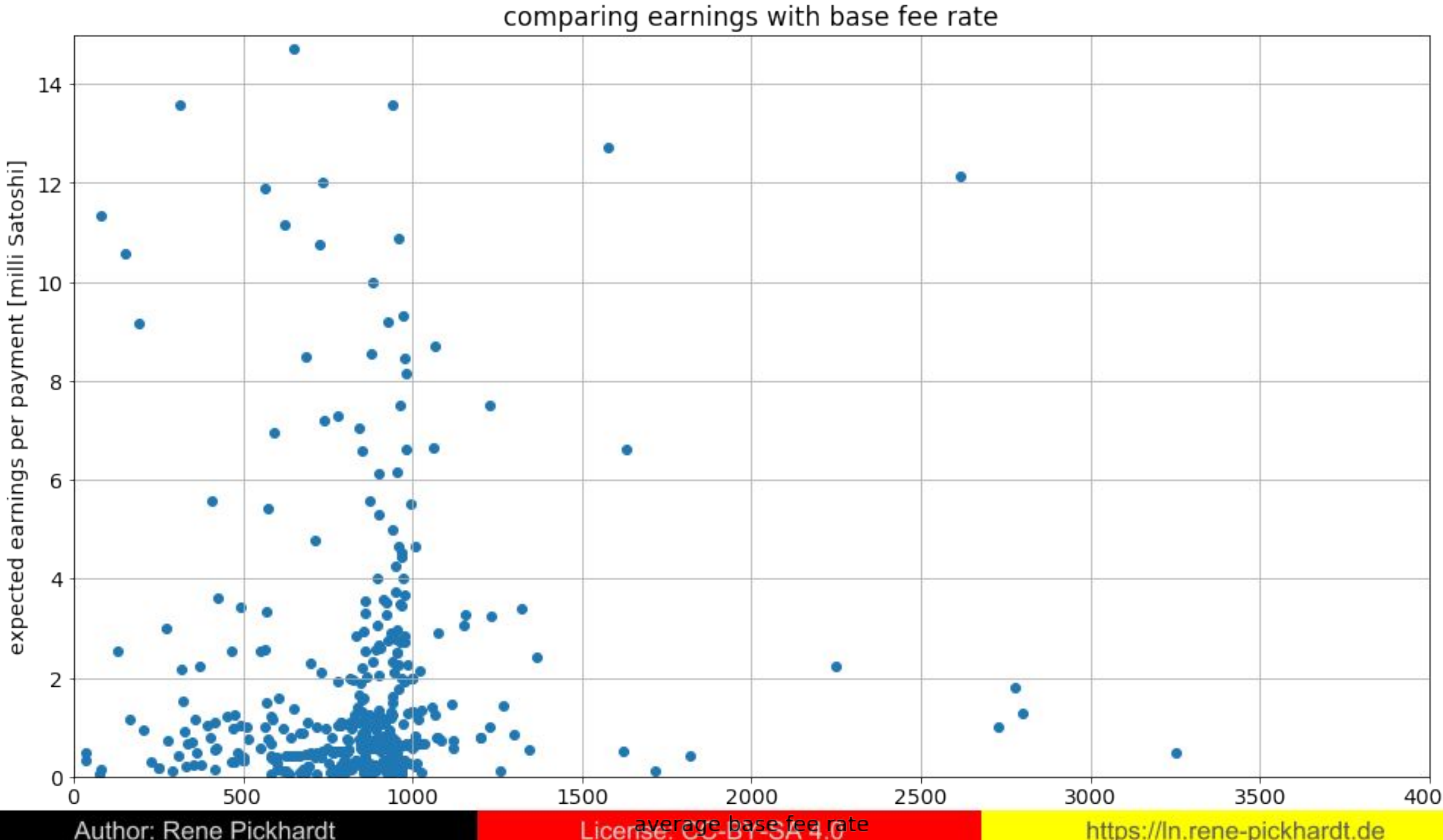

Percentage of nodes that can reach each other depending on the routing fee
Assumes the Lightning Network from February 3rd 2019 and 1 mBTC payment without failure
relative number of pairs of nodes that can reach each other
0.0
0.2
0.4
0.6
0.8
1.0
0
500
1000
1500
2000
2500
3000
3500
routing fee for the entire path [milli Satoshi]
CC-BY-SA-4.0 Rene Pickhardt
https://ln.rene-pickhardt.de

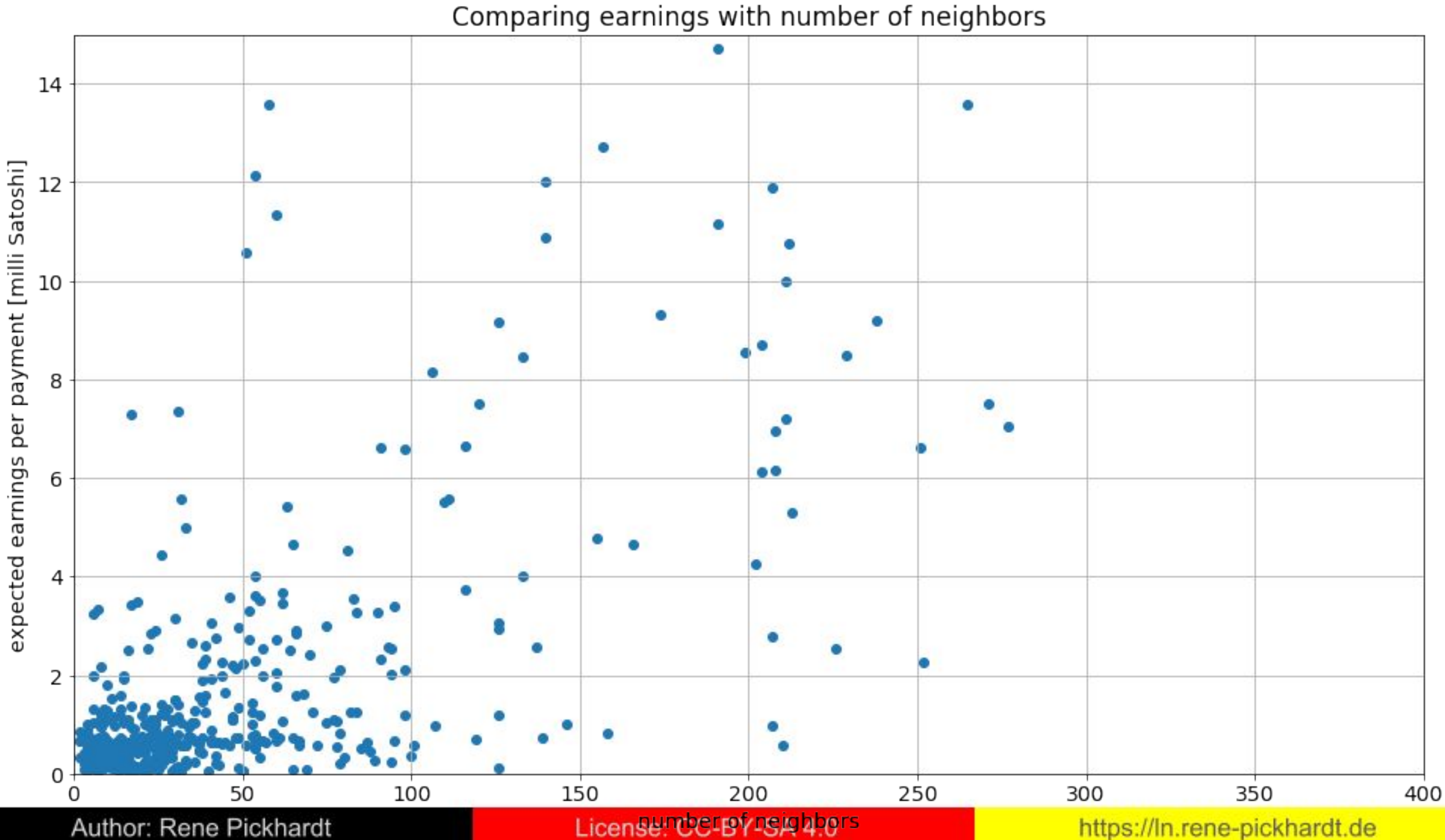
Expected routing fees

- Assume uniform distribution of sender and recipient pairs
  - Strong and most likely invalid assumption
- Probability my node is on the path between S and R
  - Will be higher if my fees are low
  - Proportional to the betweeness centrality on the fee graph
- E = betweeness centrality x fee
  - Optimization problem
  - Hard to solve since
    - Non linear
    - Betweeness centrality is hard to compute

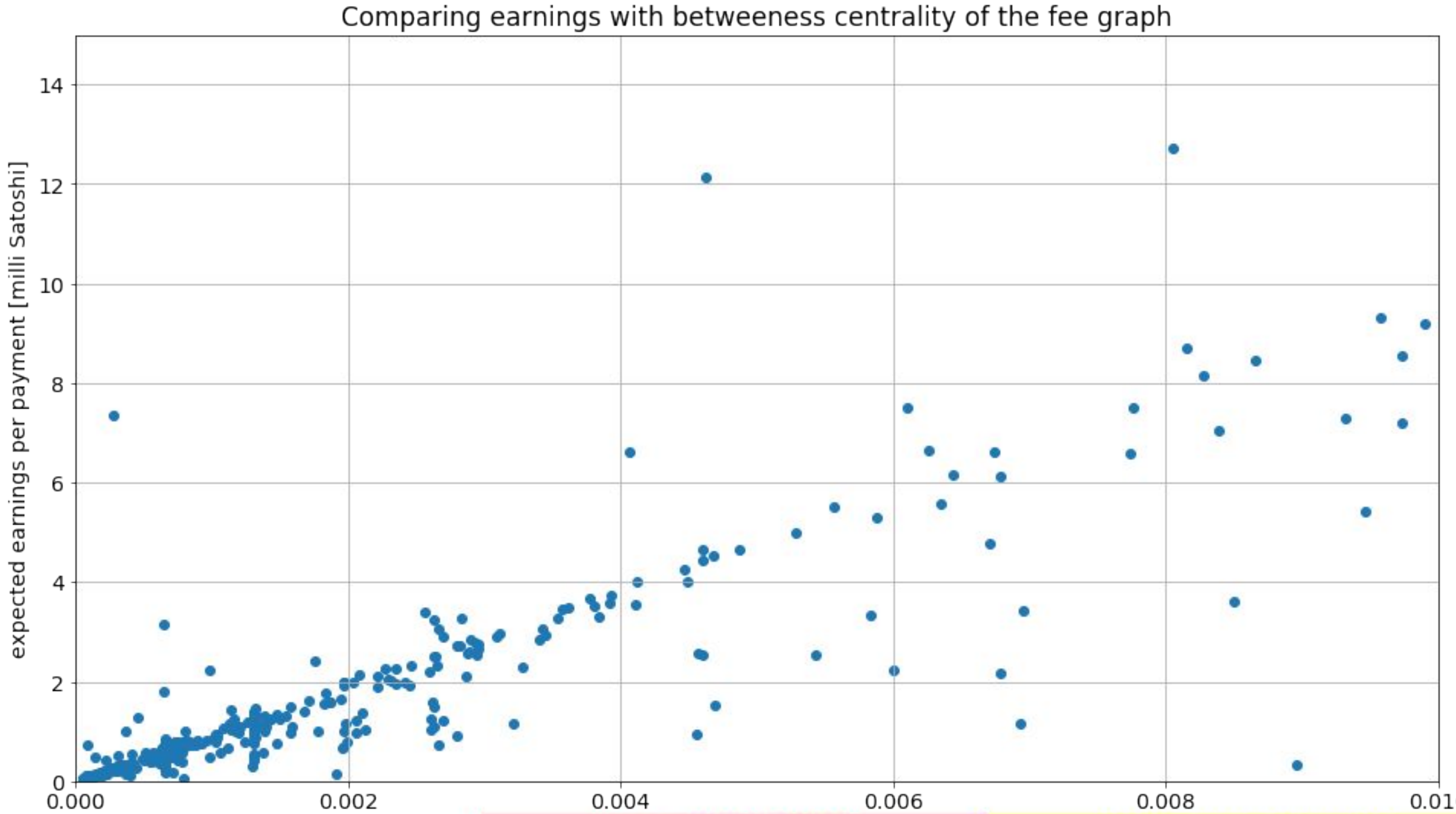
Expected earned routing fees in milli Satoshi per lightning network payment depending of base fee rate
(for Rene Pickhardt's lightning node with 31 channels on Feb 3th 2019)

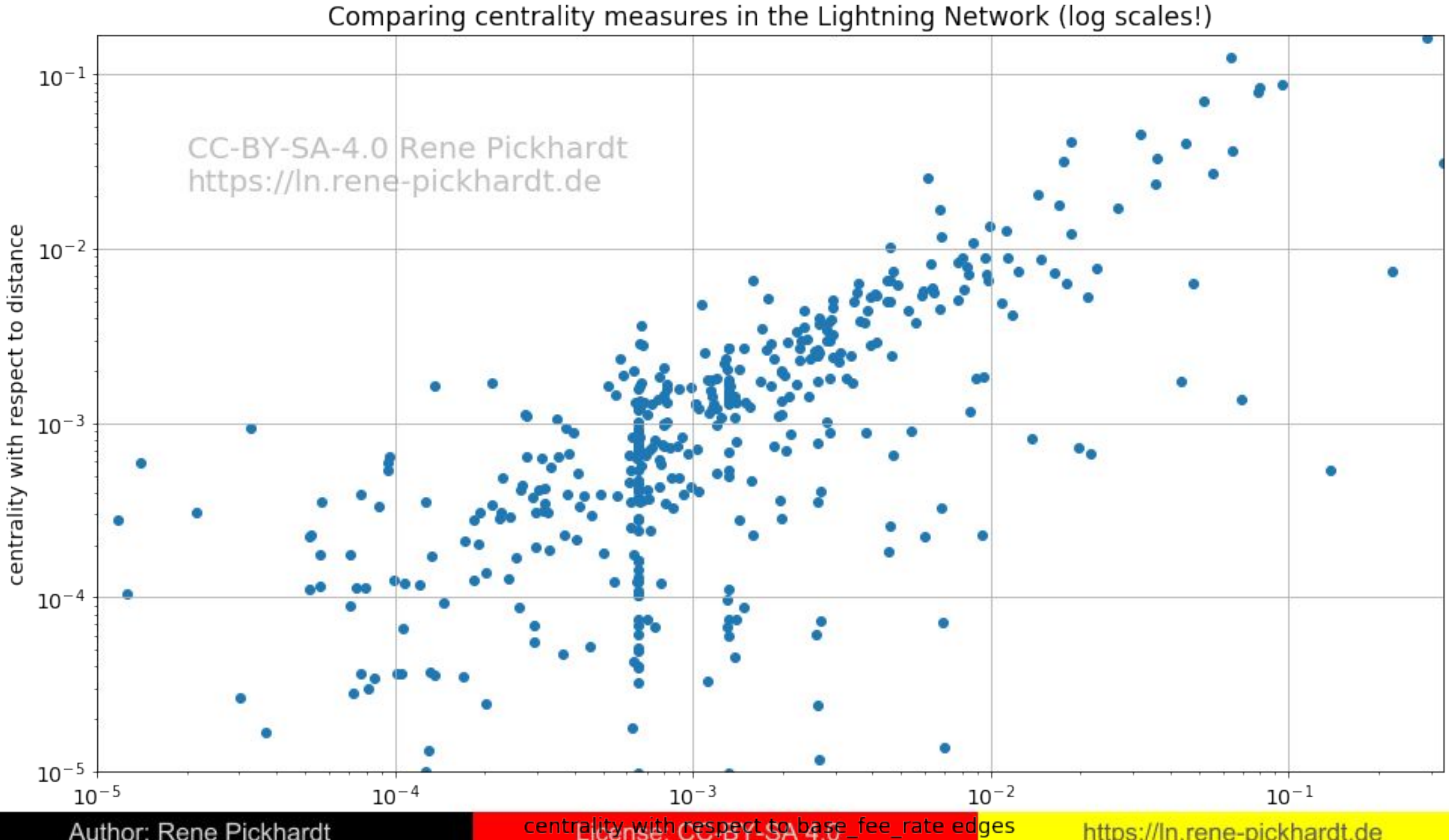



comparing earnings with base fee rate
expected earnings per payment [milli Satoshi]
0
2
4
6
8
10
12
14
0
500
1000
1500
2000
2500
3000
3500
4000
average base fee rate

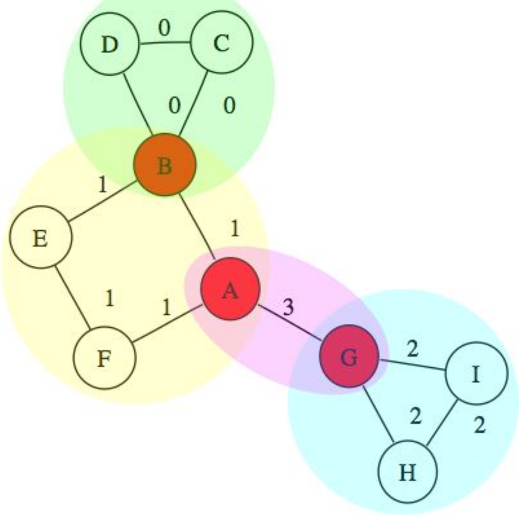


Comparing earnings with number of neighbors
expected earnings per payment [milli Satoshi]
0
2
4
6
8
10
12
14
0
50
100
150
200
250
300
350
400
number of neighbors


Comparing earnings with betweeness centrality of the fee graph
expected earnings per payment [milli Satoshi]
0
2
4
6
8
10
12
14
0.000
0.002
0.004
0.006
0.008
0.010
betweeness centrality on the fee graph


Comparing centrality measures in the Lightning Network (log scales!)
centrality with respect to distance
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
centrality with respect to base_fee_rate edges

Future ideas



- Articulation points by Kulpreet Singh
  - Look at points creating biconnected components
- Look at node age / channel age
- Make lightning nodes collect more statistics
  - Uptime of nodes
  - Routing fees
    - Could indicate if a node / channel is available for inbound capacity
  - Previous success / failure of routes
- Recommend to close channels
- Strategies to increase collection of routing fees
- Recommend routing fee for a certain channel
- Machine Learning

Roadmap

- Simulations
- Find out / describe desired properties
- Create / extend gossip protocol for autopilots to communicate on top of the lightning network (distributed autopilot)
  - Request Inbound channels
  - Offer outbound channels
  - Maybe even exchange some information about channel balance
- Get it merged as an "official" c-lightning plugin to the Elements/lightning repo

Conclusions / Thoughts about the autopilot

- Tradeoff between privacy and quality of recommendations
- Path finding and autopilot are related problems
- Recommending channel balances seems reasonable.
- Communication / information sharing might be the way to go
- Selecting routes yourself is perfectly fine
- We need simulation frameworks / research and some real world usage data
- We should take that topic serious within pathfinding
    - It's not only about channels but also about paths

Path finding has many open challenges!

(Join us and help to tinker about them!)

One slide in my own personal interest

- Thanks to chaincode for inviting everyone!
  - Like always I learnt a lot by teaching and from listening to the others
- I am in lightning since one year now
  - From a perspective of Community and Topic that was a great choice
  - Financially a disaster.
    - Since working I never earnt as little as during the last 12 months.
- I am currently literally unemployed
- Seeking funding for my youtube channel / open source activities
  - I want to do more regular / structured educational content on youtube about lightning
  - I would love to find a couple of companies / sponsors.
  - If you have ideas / contacts (to companies in the industry) please talk to me or help me
- I have a Tallycoin / Patreon Page for my lightning network book and youtube channel
  - Major announcement about progress on that coming end of August

References and helpful links

- https://github.com/lightningnetwork/lightning-rfc
- https://www.youtube.com/user/RenePickhardt
- https://bitcoin.stackexchange.com/questions/tagged/lightning-network
- https://lightning.network/lightning-network-paper.pdf
- C-lightning plugin: https://github.com/lightningd/plugins/tree/master/autopilot
- Lib-autopilot: https://github.com/renepickhardt/lightning-network-autopilot
- Earn routing fees: https://www.youtube.com/watch?v=L39IvFqTZk8

External youtube videos used:

- Preferential attachment: https://www.youtube.com/watch?v=IPgUi5y3_O8
- Growing Erdos Renyi: https://www.youtube.com/watch?v=8tMCFCkxbnw
- Barabasi Albert visualization: https://www.youtube.com/watch?v=4GDqJVtPEGg

Copyright notice

Thanks to Marietheres Viehler (aka journalspiration) for the design of the title slide.

About this slide deck

The purpose is to help spreading education about the Lightning Network Protocol so that the technology will be adopted more quickly by more people. This shall be my contribution to the Bitcoin / Lightning Network Community.

This slide deck was presented during Chaincodelabs Lightning Residency program in June 2019. It is part of https://commons.wikimedia.org/wiki/File:Introduction_to_the_Lightning_Network_Protocol_and_the_Basics_of_Lightning_Technology_(BOLT_aka_Lightning-rfc).pdf. To the best of my knowledge the original file is the most comprehensive work making an introduction to the BOLT standard.

The slides are part of my effort to create a book about the lightning network. You can follow that effort at: https://github.com/renepickhardt/The-Lightning-Network-Book or you can support the effort at my fundrasing pages at: https://tallyco.in/s/lnbook or at: https://www.patreon.com/renepickhardt or at 1GZx8tWgDd21Rd8b1QdMrzdZGHgyfVkzaD part of this effort also consists of creating video tutorials and teaching materials on my Youtube Channel over at: https://www.youtube.com/user/RenePickhardt

This work was funded (sorted by amount of contribution from top to bottom) by: Me personally, fulmo.org, everyone who contributed to the above mentioned fundraiser and George Danzer.

Thank you to the lightning developers and people in various telegram groups for helpful discussions