LNP/BP Standards Association

LNP/BP nodes initiative

Rationale and WIP on Bitcoin and Lightning nodes supported by LNP/BP Standards Association

Dr Maxim Orlovsky, CEO **Pandora Core**, Secretary of LNP/BP Standards Association

github.com/dr-orlovsky
Twitter: @dr_orlovsky

“Wait, but why we do need new nodes”?

–Anonymous

New exiting apps on layer 2 & 3 are coming!

- Channel factories
- Discreet log contracts on LN
- RGB & Spectrum: assets, identity, smart contracts & DEX on LN
- Lightspeed: high-frequency micropayments
- Storm: storage and messaging; with multi-peer channels
- Prometheus: high-load computing; with multi-peer channels

Current Bitcoin node

- No proper indexes, require third-party software (like Electrum server)
- … which are also very limited in functionality (wallet-focused)
- Slow, outdated, insecure JSON-RPC
- Monolithic architecture, non-scalable

LN nodes

- Hard to extend with custom messages (except c-lightning)
- One can’t modify the structure of commitment and other channel transactions
- “Hardcoded” to existing specs, no modularization

And also LN should upgrade for …

- Schnorr signatures
- Taproot
- Payment points
- eltoo
- … who knows?

All these upgrades are very complex with existing node architecture

LN software has to be ready for:

- Support for multi-peer channels
- Abstraction of commitment- and funding transaction structure
- Modularisation of penalty/escrow mechanics (HTLC->PTLC)
- Better separation of networking layers

But are existing LN nodes ready to adopt that?

- No, at least without a deep refactoring of their architecture and lots of rewrites.

Why?

- Hardcoded uni-directed channel parameters
- No channel / connection concept separation
- Monolithic architecture (except c-Lightning)
- No plugin support (except c-Lightning)

LNP/BP Standards Association

- github.com/LNP-BP

- oversees Layer 2 & 3-related standards and software dev

- Proposed bitcoin tx indexing/query service in March 2020:
  fast & scalable arbitrary complex queries against
  compact bitcoin blockchain index
  https://github.com/LNP-BP/txserv

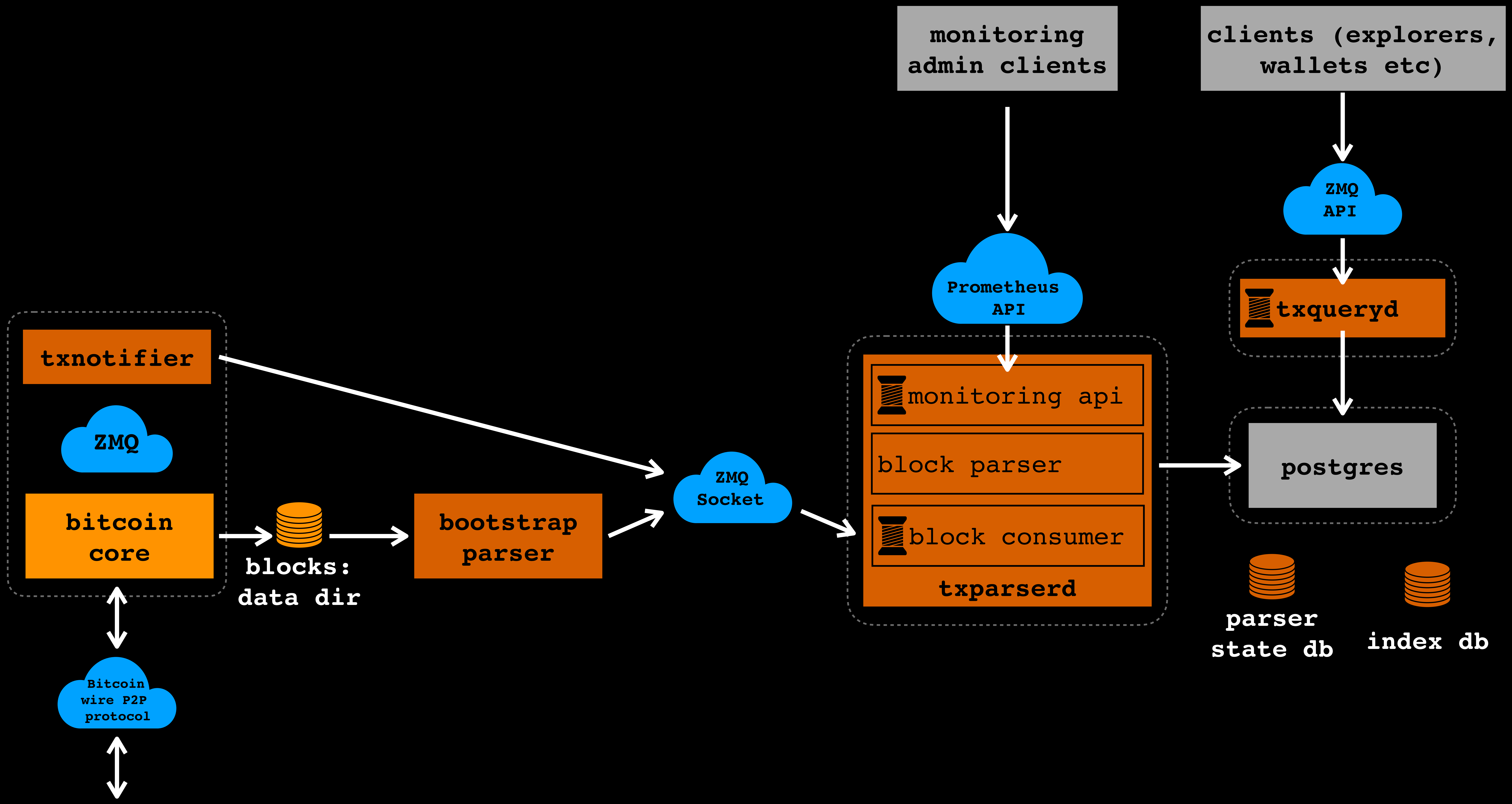
- Run Lightning Hacksprint hackaton in early April 2020
  with the challenge to create prototype of new Rust-based
  LN node "LNPd"
  https://github.com/LNP-BP/lnpd

Architecture requirements

- Microservice-based: scalability up to multi-docker enterprise environments
- High-load processing: usage of ZeroMQ APIs instead of JSON RPC and unreliable IPC
- Subscription/push-based notification model for clients, non-custodial wallets etc
- Separation of Peers and Channels
- Extensible with new modular functionality

First stage: Bitcoin Transaction Services

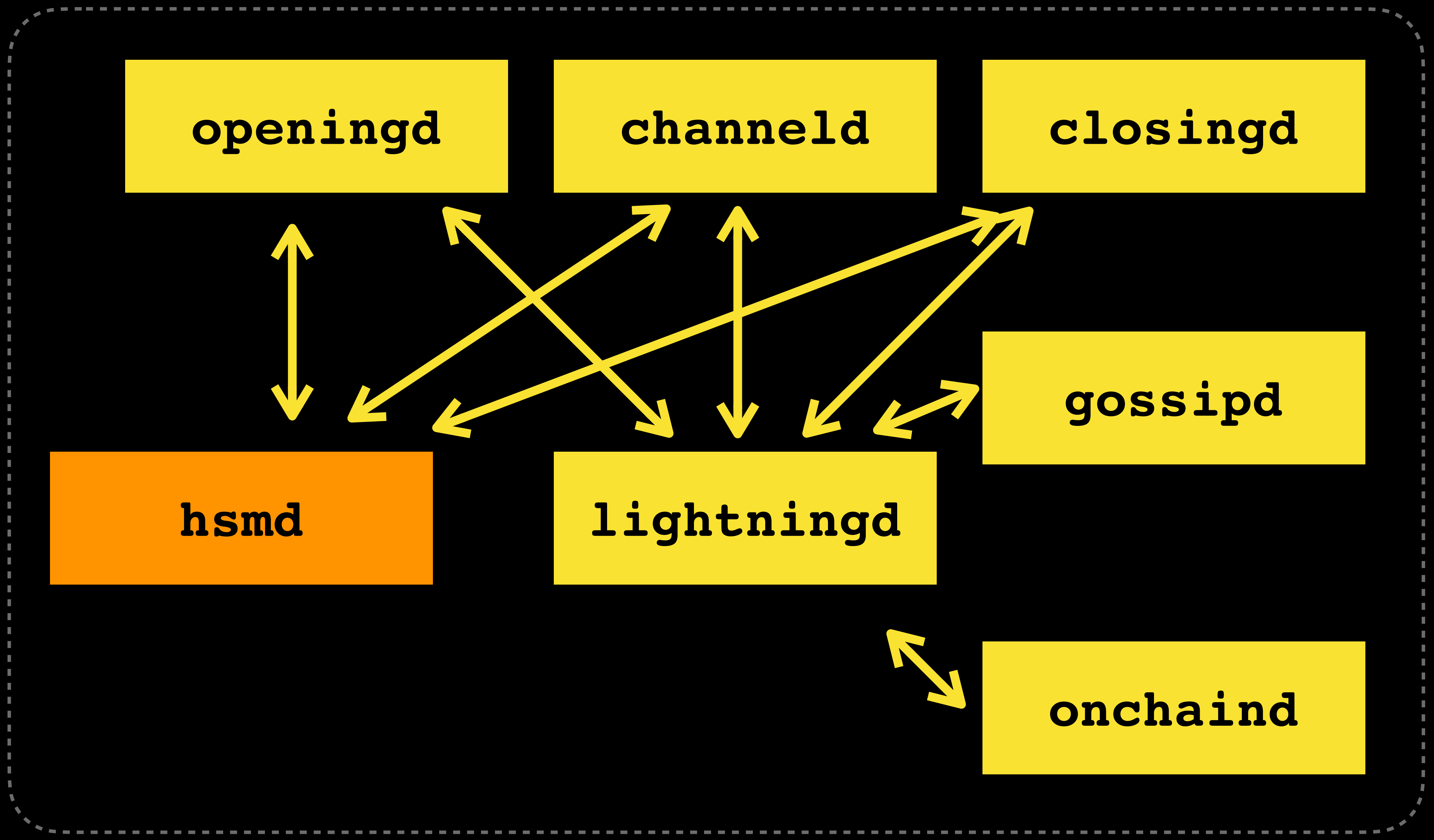
- Parse bitcoin blockchain into a compact database
- Based on **LNPBP-5** standard – joint work with C. Decker: https://github.com/LNP-BP/lnpbps/blob/master/lnpbp-0005.md
- Based on rust-bitcoin library by Andrew Poelstra @Blockstream
- Provide a query API for data that can't be provided by either Bitcoin Core and Electrum Server
  - getting transaction spending particular output
  - querying transactions by their script/**miniscript** code


monitoring admin clients
clients (explorers, wallets etc)
ZMQ API
Prometheus API
txqueryd
txnotifier
ZMQ
monitoring api
block parser
block consumer
txparserd
postgres
ZMQ Socket
bitcoin core
blocks: data dir
bootstrap parser
parser state db
index db
Bitcoin wire P2P protocol

Second stage: LNP node (Lightning node)

- Based on rust-lightning library by Matt Corallo @Square Crypto & @Chaincode Labs
- Utilizing the same multi-thread non-blocking microservice code as Bitcoin transaction service
- Following best practices from c-Lightning architecture & extensibility
- Suited for generalized Lightning Network, ready for:
    - ◉ multi-peer channels / channel factories
    - ◉ multiple channels per peer
    - ◉ payment points
    - ◉ RGB, Spectrum
    - ◉ Protocols, that require modification of channel transaction structure (discreet log contracts, Storm, Prometheus)

Our example to start with:
c-Lightning multiprocess architecture



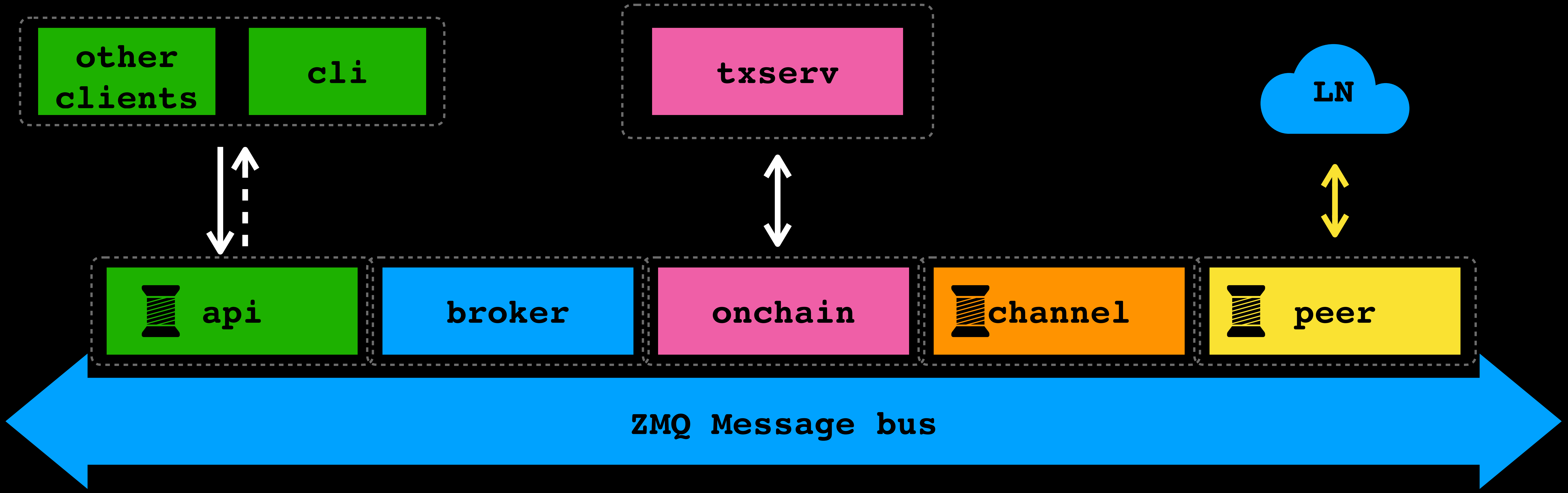
docker container

← c-lightning RPC

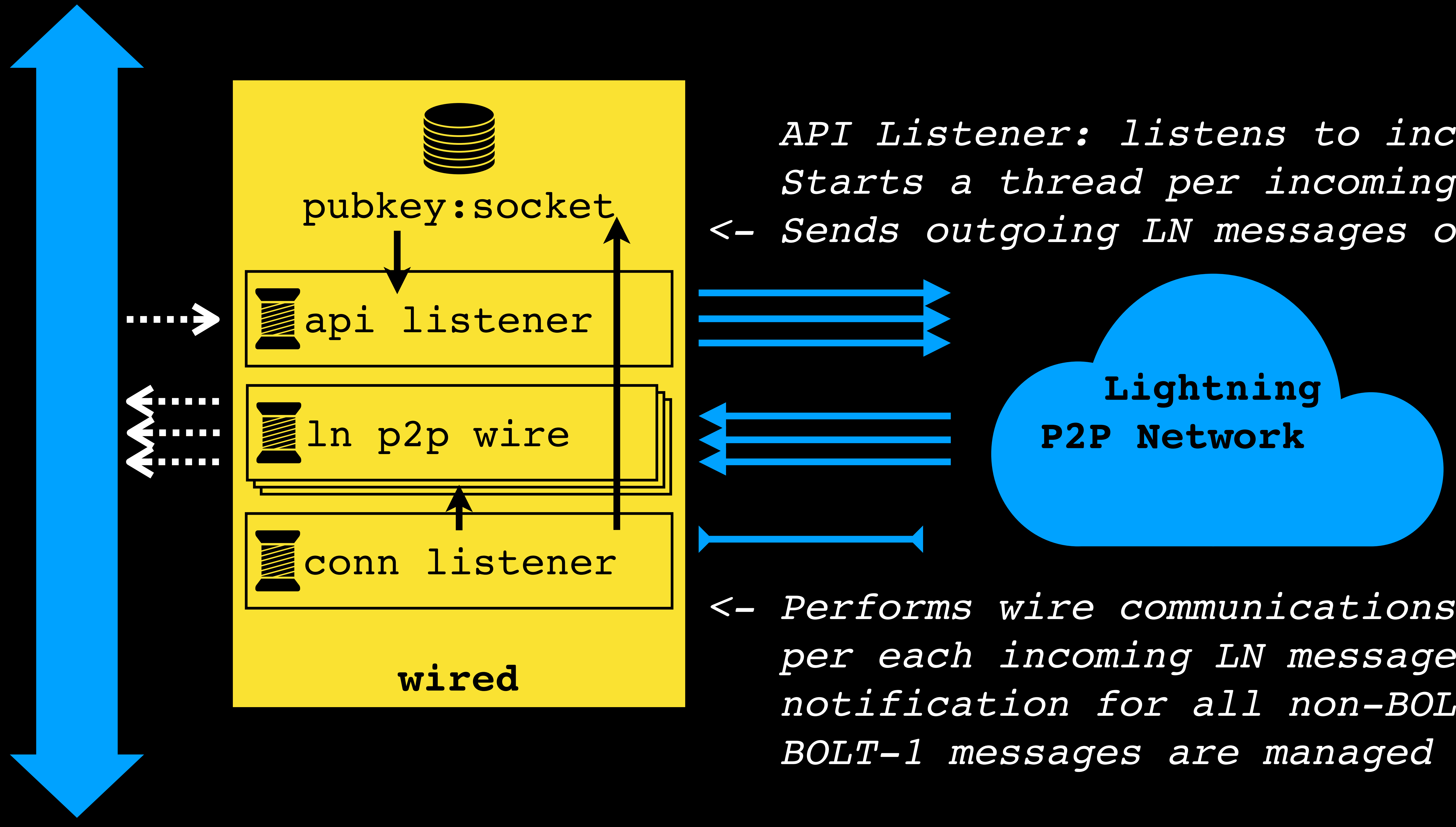
LNP: new LN node architecture, round 1

docker container or volume
ZMQ REQ/REP
Double ZMQ REQ/REP
ZMQ PUB/SUB
connected to all channels and peers
Extensions/mods
txserv
keys daemon
PNS
push-notifications used by mobile non-custodials for signing txs
multipeer channel
onchain
cli
other clients
api
broker
routing
channel
gossip
channel
DLC
channel
RGB
peer
peer
peer
DLC
RGB
dlcd
rgbd
routing sqlite
per-channel state sqlite
gossip sqlite

New LN node architecture, result of hackathon



Anatomy of a peer managing process (wired)



API Listener: *listens to incoming commands from other daemons. Starts a thread per incoming request.*

<- Sends outgoing LN messages originating from channels etc.

<- Performs wire communications. Starts a new "ln p2p wire" thread per each incoming LN message, that ends posting a ZMQ PUB notification for all non-BOLT1 messages. BOLT-1 messages are managed internally.

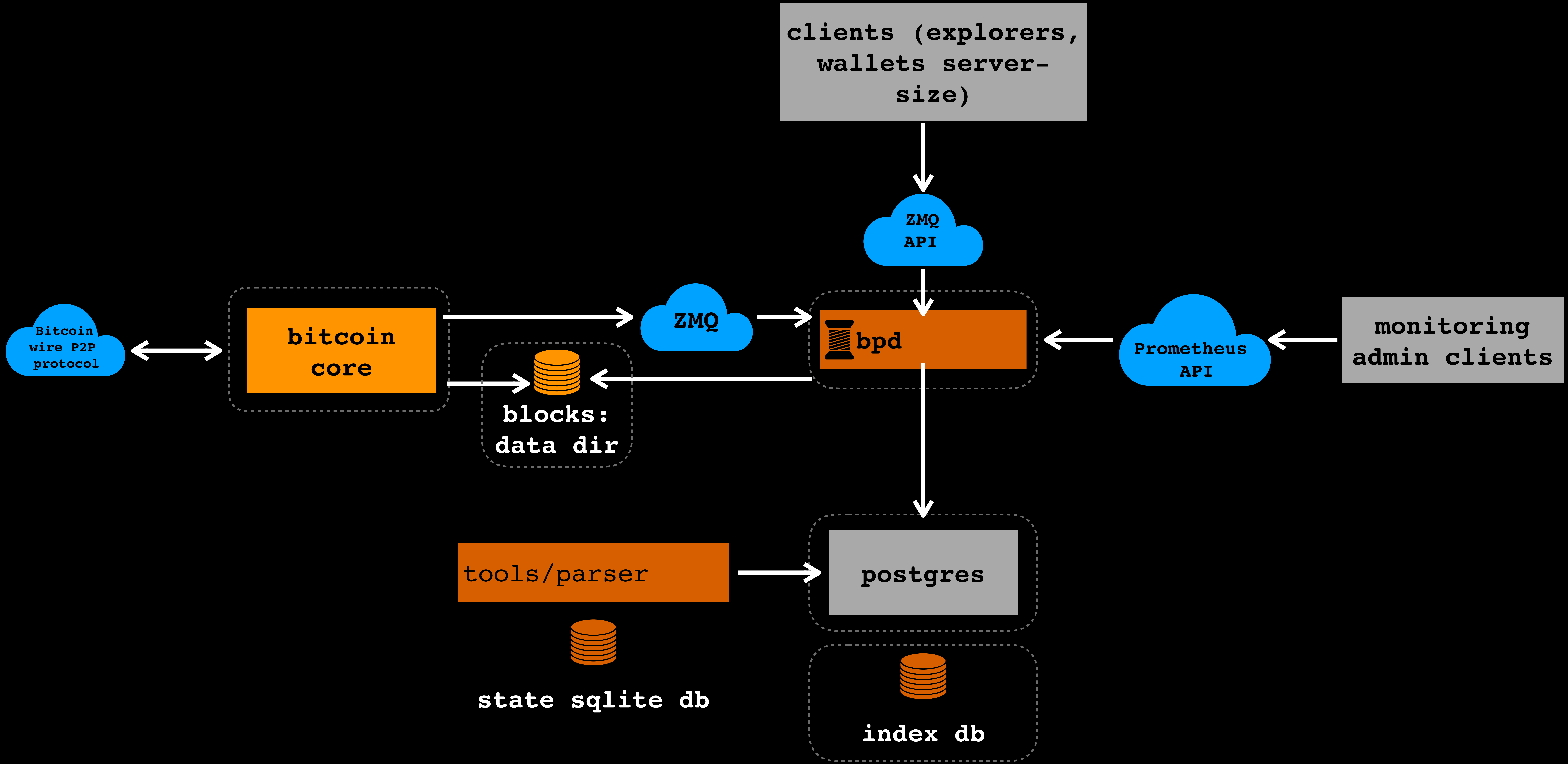
ZMQ Message Bus

Results of the hackathon

- Created the core of multi-process & multi-thread architecture for new lightning node
  github.com/LNP-BP/lnpd
- Abstracted LN layers for transport (BOLT-8) & peer messaging (BOLT-1)
  github.com/LNP-BP/rust-lnpbp
- Created a peer daemon able to run P2P remote node communications: extension point for channel negotiation
  github.com/LNP-BP/lnpd/src/bin/peerd.rs

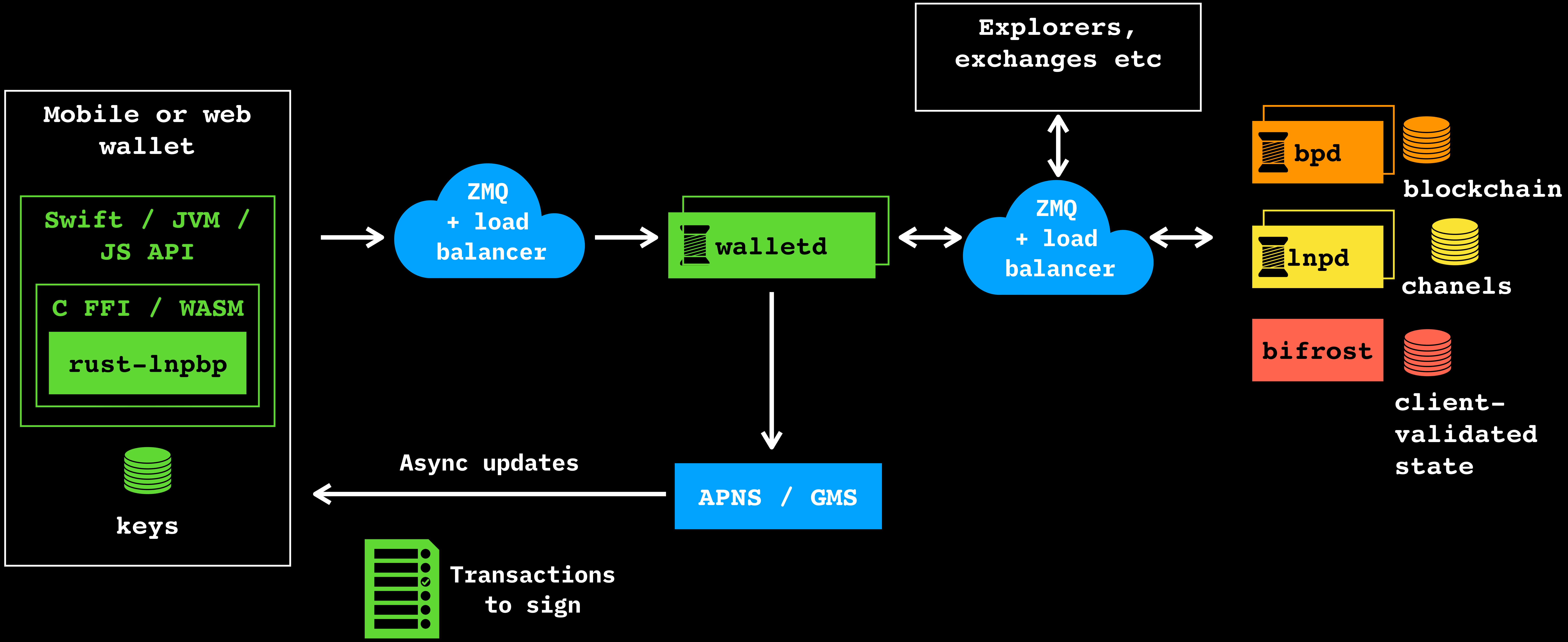
Third stage:
BP node (bitcoin protocol node)

- Based first on Bitcoin Core, later on libbitcoinconsensus
- Universal solution for Layer 2/Layer 3 protocols, including LN nodes to query bitcoin blockchain w/o the need to parse it
- Real-time notifications about new incoming transactions (with ZMQ subscriptions) satisfying complex **miniscript**-based queries


clients (explorers, wallets server-size)
ZMQ API
Bitcoin wire P2P protocol
bitcoin core
ZMQ
bpd
Prometheus API
monitoring admin clients
blocks: data dir
tools/parser
postgres
state sqlite db
index db

Putting node architecture into perspective

Non-custodial clients/wallets



Contribute to github.com/LNP-BP!

- LNP/BP Standards development & discussions:
  github.com/LNP-BP/lnpbps

- Standard Rust library:
  github.com/LNP-BP/rust-lnpbp

- Bitcoin node:
  github.com/LNP-BP/bp-node

- Lightning node:
  github.com/LNP-BP/lnp-node

- Discussions @ Freenode IRC: #lnp-bp