LNP/BP Standards Association

Bifrost

generalized lightning network

Dr Maxim Orlovsky,
CEO (chief engineering officer) at
Pandora Core AG

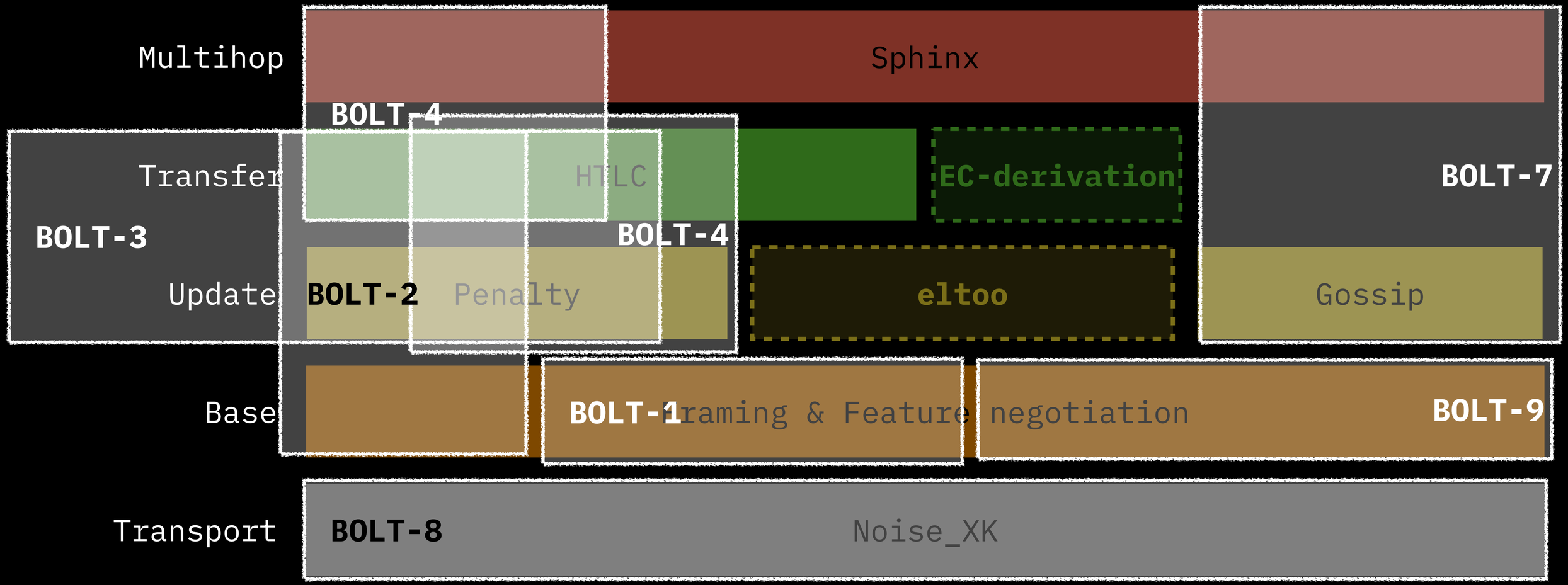
Background

During work on RGB it became apparent that

- Some of important RGB-on-LN features break existing BOLT standards
- LN specs (BOLTs) are hard to update
- Even without LN RGB requires data propagation P2P network; it is logical to leverage LN, but see previous two points

It was decided to create a generalization of LN defined as separate LNPBP standards – codenamed “Bifrost”

Real BOLT Specifications



Problems with existing LN

- BOLTs
  - Poorly structured
  - Hard to implement feature-wise
  - Takes forever to get PRs reviewed
- Transaction & message structuring: custom-tailored for two-peer payments and HTLCs
  - No key tweaking possible without standards breaking
  - Extremely complex to change structure of transactions
- Encoding: not compatible with client-side-validation standard (“strict encoding”, LNPBP-7)

LN should upgrade for …

- Dually-funded channels
- Channel factories, multiplier channels
- Channel splits
- Schnorr signatures
- Taproot
- Payment points (PTLCs)
- Eventually, eltoo

- RGB
- Discreet log contracts on LN
- Lightspeed: high-frequency micropayments
- Storm: storage and messaging; with multi-peer channels
- Prometheus: high-load computing; with multi-peer channels

Bifrost ideology

Lightning Network will evolve

…and Layer 3 apps will be built on it

Let’s draw a reckless way forward:
build a foundation for fast experimentation & extension

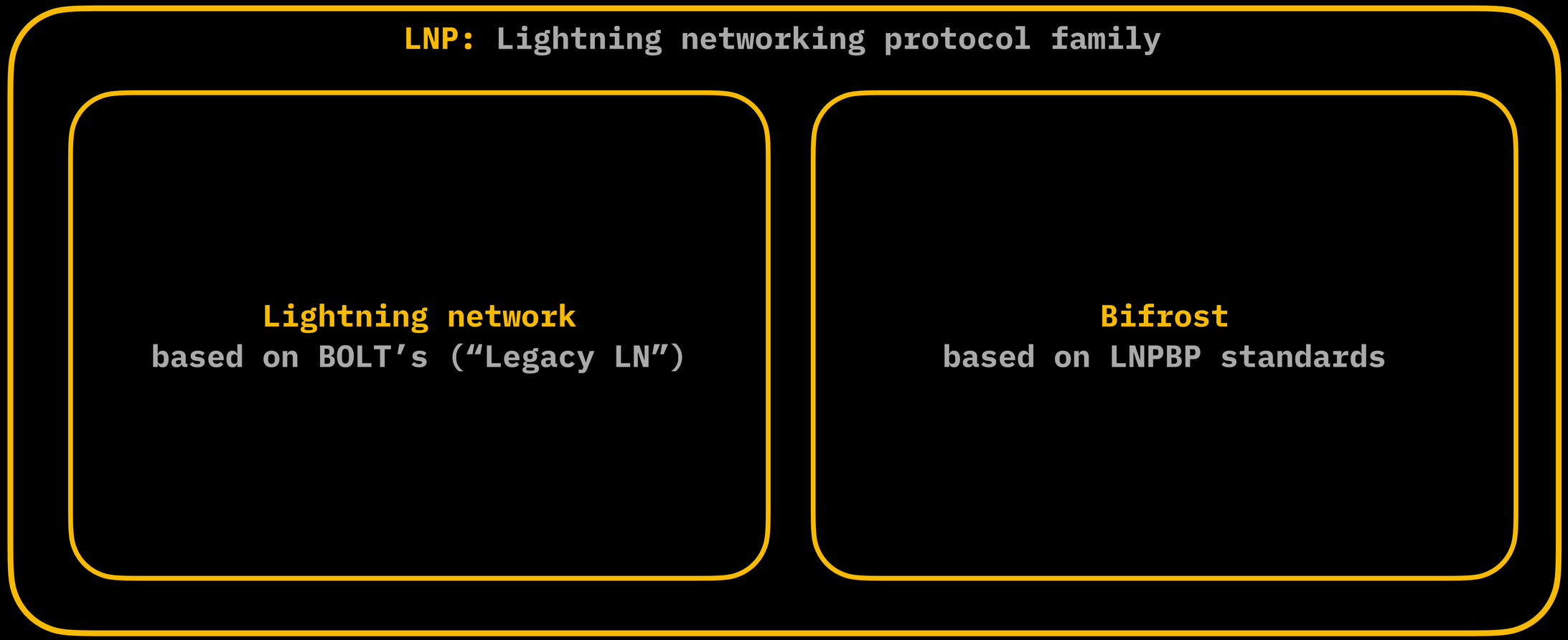
Doing Bifrost: Reckless^2

- We want to reduce the number of required changes to BOLTs to ZERO
- We need to separate risks of new experimental protocol from more stable LN
- We need clear layerization / abstractions

Part I

Defining Bifrost

LNP & Bifrost


LNP: Lightning networking protocol family
Lightning network
based on BOLT's ("Legacy LN")
Bifrost
based on LNPBP standards

LNP & Bifrost

LNP: Lightning networking protocol family

Lightning network
based on BOLT's ("Legacy LN")

- Single/dually-funded payment channels
- Gossip-based route exploration
- HTLC-based payment propagation
- Payment-specific onion routing with Sphinx protocol

Bifrost
based on LNPBP standards

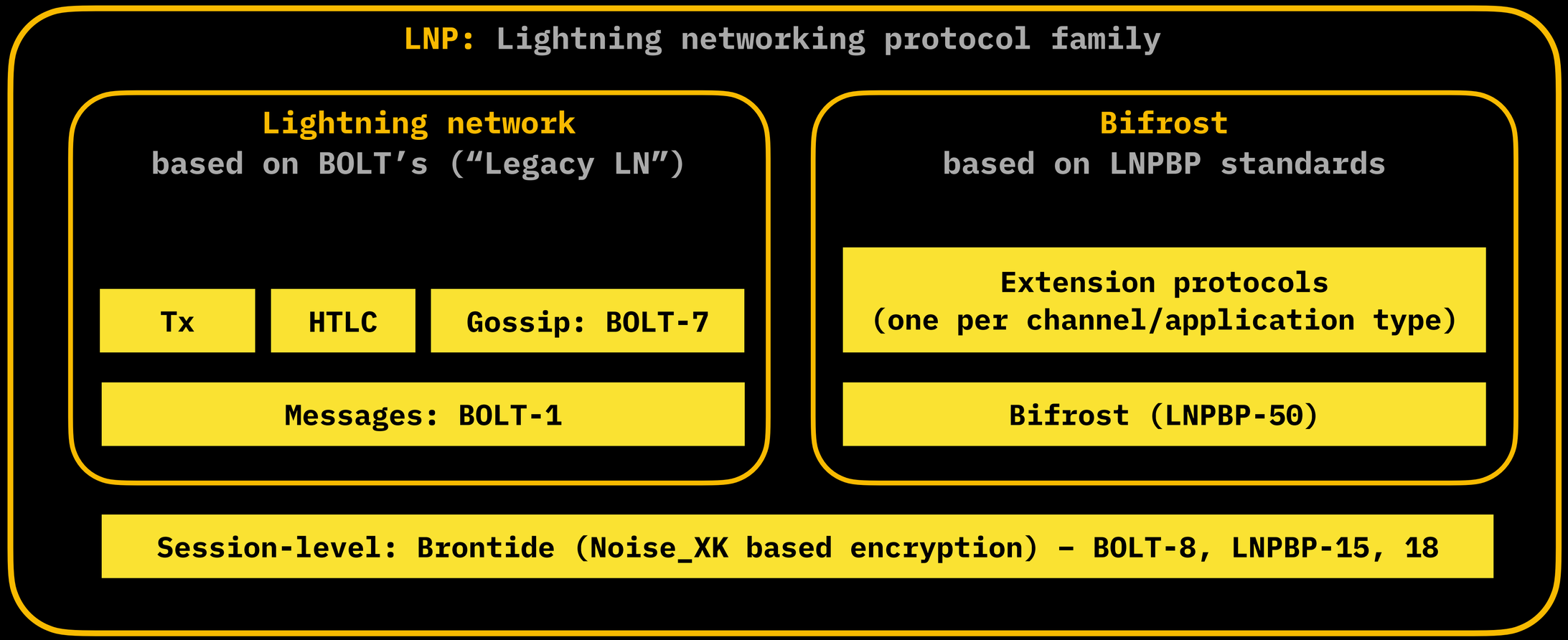
- Pure networking protocol, unaware of specific channel types
- General data propagation P2P network
- Application-defined custom channels
- Channel composability

Doing Bifrost: Reckless^2

Bifrost will become a lightning “subnetwork”:

- Separate port number (9999)
- Based on Brontide session management (BOLT-8, redefined as LNPBP-15 and LNPBP-18 standards)
- A single custom message over “legacy” LN connection to upgrade to the new port/protocol (like from HTTP to WebSocket)

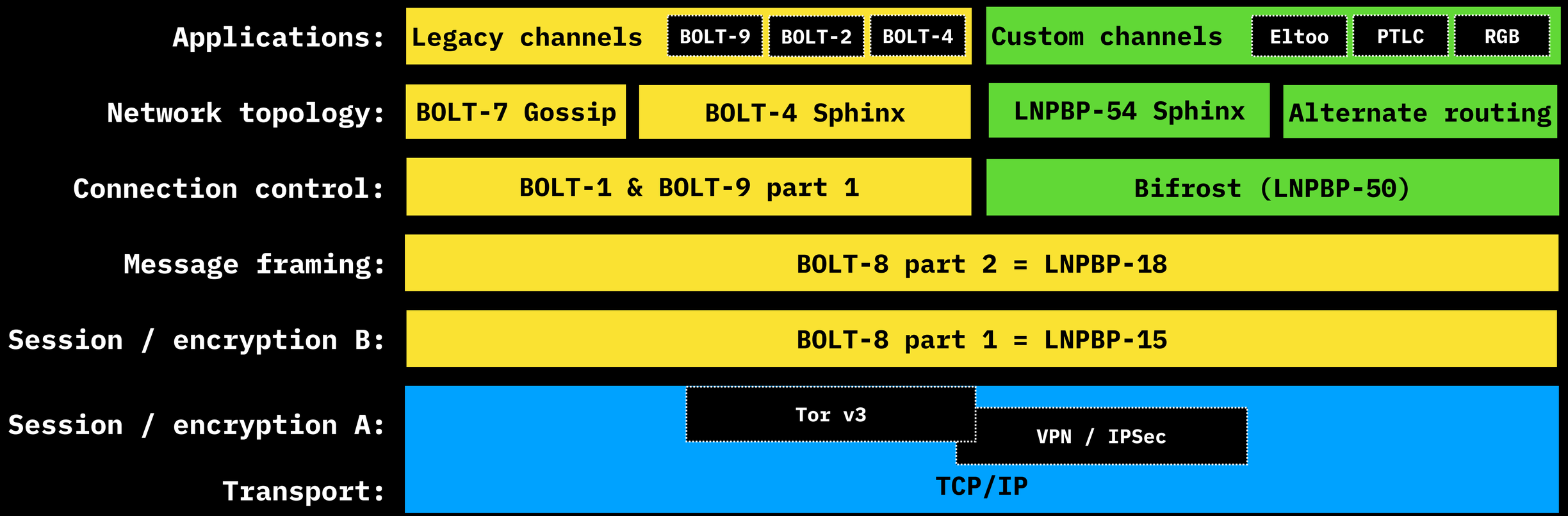
LNP & Bifrost


LNP: Lightning networking protocol family
Lightning network
based on BOLT's ("Legacy LN")
Tx
HTLC
Gossip: BOLT-7
Messages: BOLT-1
Bifrost
based on LNPBP standards
Extension protocols
(one per channel/application type)
Bifrost (LNPBP-50)
Session-level: Brontide (Noise_XK based encryption) – BOLT-8, LNPBP-15, 18

Bifrost is pure network-level protocol, abstracting away channel-specific functionality into application levels above

This solves one of the main issues with the design of the modern Lightning network

LNP protocols



Bifrost as generic P2P data network

- Messaging
  - Leveraging privacy of Sphinx protocol and anonymity of addresses with LN node pub keys and underlying Tor onion addresses
- Information storage & retrieval network
  - Pre-paid information storage
    - Channel state
    - RGB consignments
    - Public RGB contracts (media resources for NFTs, asset information)
    - Generic files & info (Torrent-like)
  - Watchtowers
  - Storm (long-term trustless information storage contracts)

Platform for the future social networks: censorship-resistant, private, incentivized

Bifrost as BiFi platform

- Bitcoin finance (BiFi) - or Bifrost Finance?
  - RGB
  - DEX
  - DLCs
  - Liquidity pools
- This unlocks complex smart contracts at L2/L3: private, scalable, uncensorable
  - algorithmic stable coins
  - futures/options etc

Part II

Designing Bifrost

Design goals

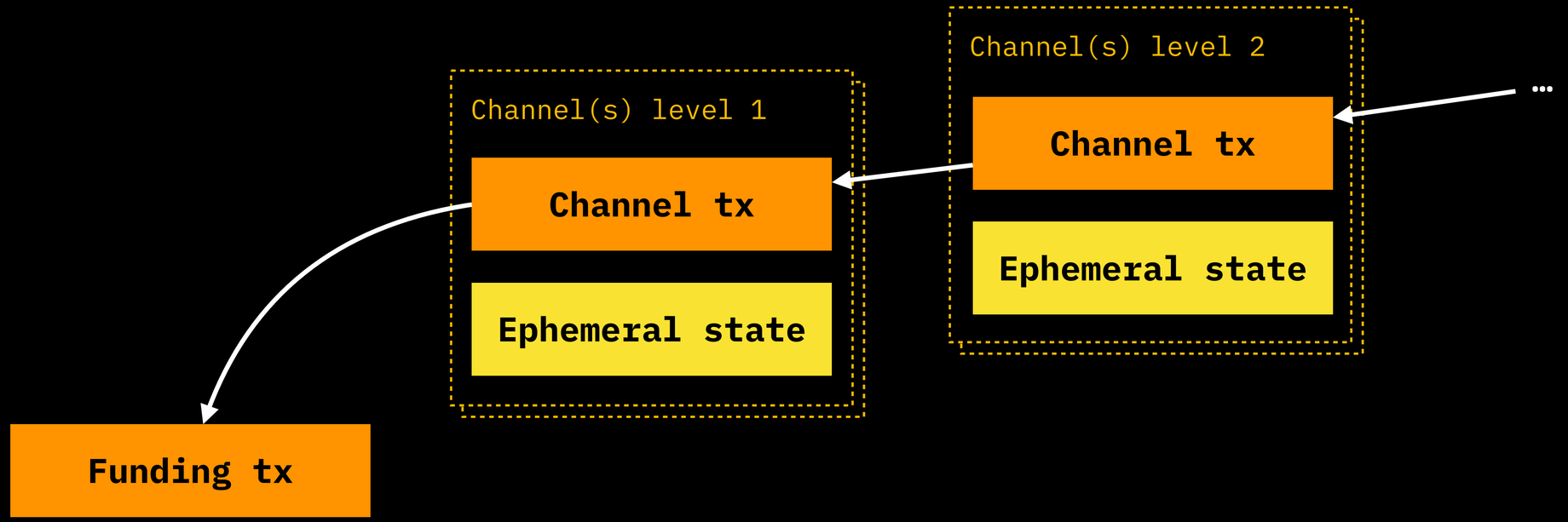
1. Extensibility, including
   - non-payment / custom lightning channels
   - non-payment network messaging & communications
   - arbitrary extension of channel transaction structure (combining different types of channels)
   - custom / new route discovery mechanisms
2. Maximal use of existing LNPBP standards, in particular
   - LNPBP-7 strict encoding
   - LNPBP-9 client-side-validation
   - LNPBP-15 Noise_XK handshake & network encryption (BOLT-8 extract)
   - LNPBP-18 Native message framing (BOLT-8 extract)
   - LNPBP-16 handshake over WebSockets
3. Privacy: improved re-use of onion messaging from Lightning network

1. Abstracting channels

What types of channels there can be?

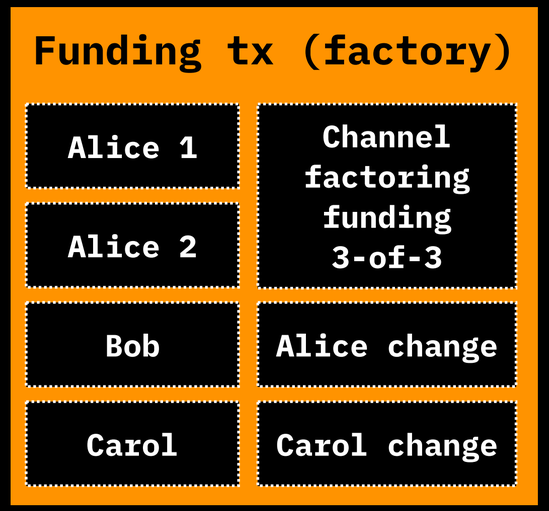
- Normal lightning payment channels (BOLT-2 & 3) with all
  - existing additional features (BOLT-9), and
  - future non-BOLT-9 extensions:
    RGB, splits/joins, taproot & miniscript, eltoo
- Channel factories / multi-peer channels
- DLCs (descrete log contracts)
- Storm (storage)
- Lightspeed (ultra-fast multi-asset nano-payments)
- … many other will appear soon

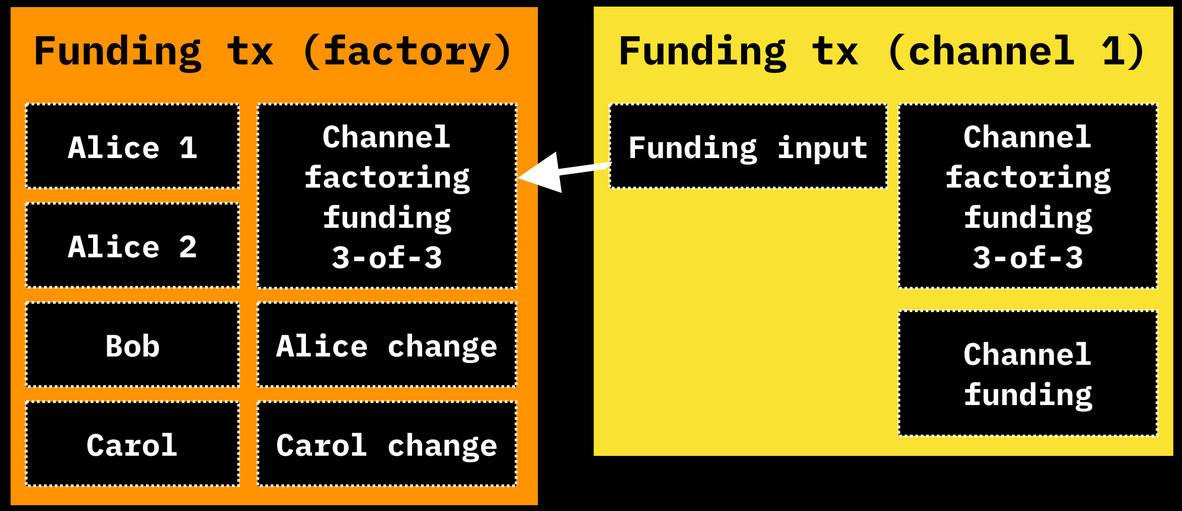
Channel hierarchy

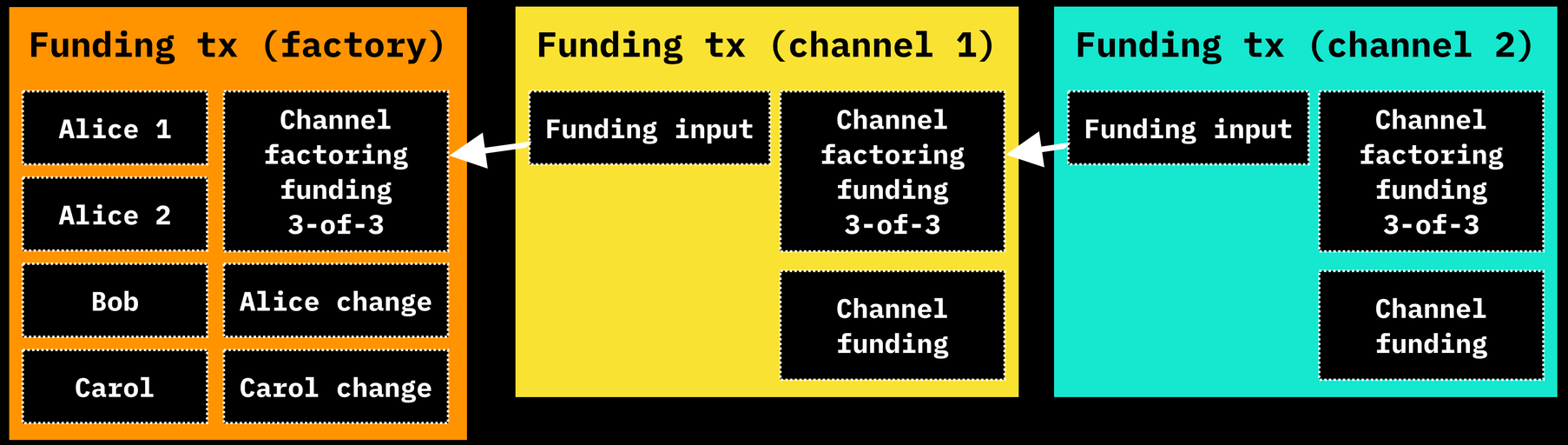


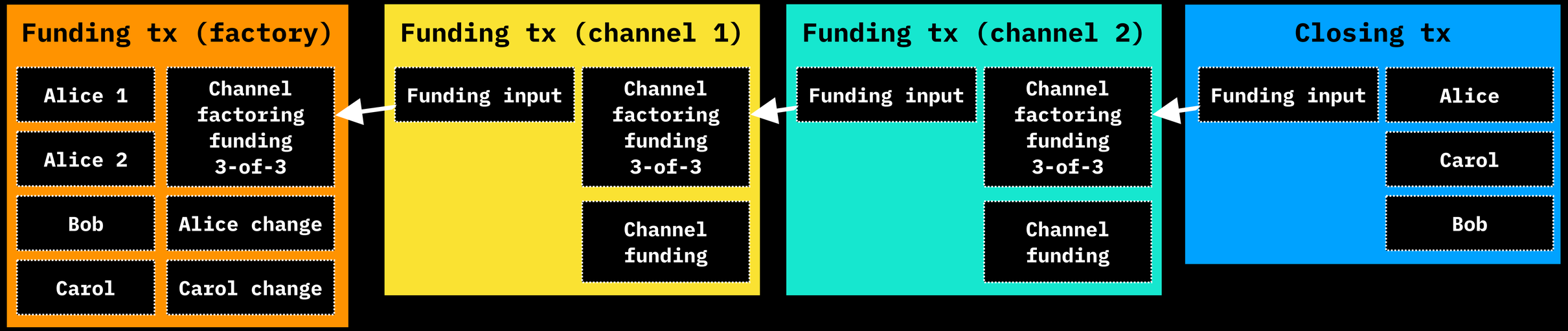
Each channel

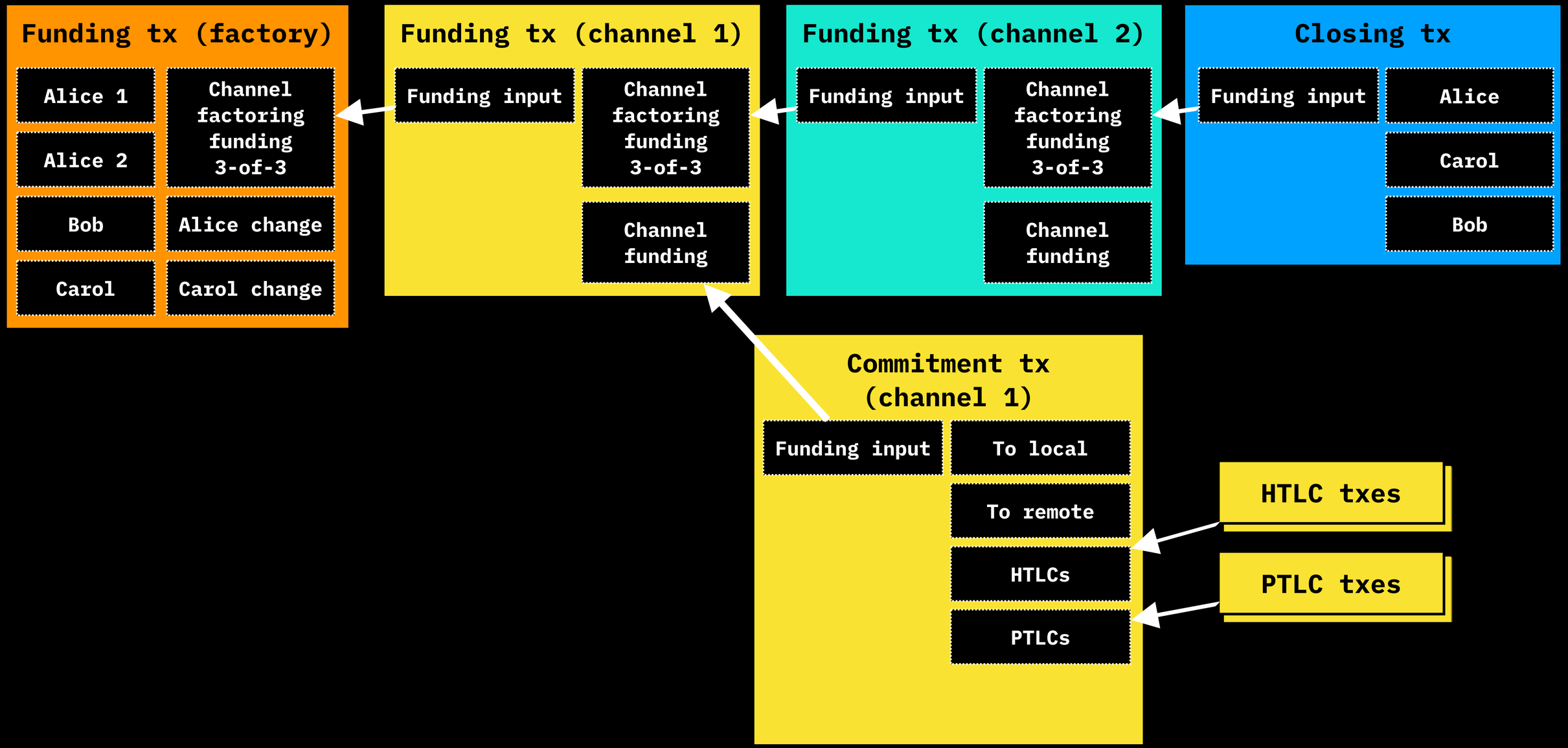
- Starts with funding transaction, which can’t be modified by channel updates. Funding transaction is
  - Mined to bitcoin blockchain for level 1 channels
  - Is a part of other channel transaction graph for channels >1
- Represented by transaction graphs
  - Each individual channel has a separate transaction graph
  - Transactions from one channel may be funding transactions for other channels
- If the channel is closed, closing transaction(s) should be added to the level above (blockchain for level 1 and upper channels for layer above 1)

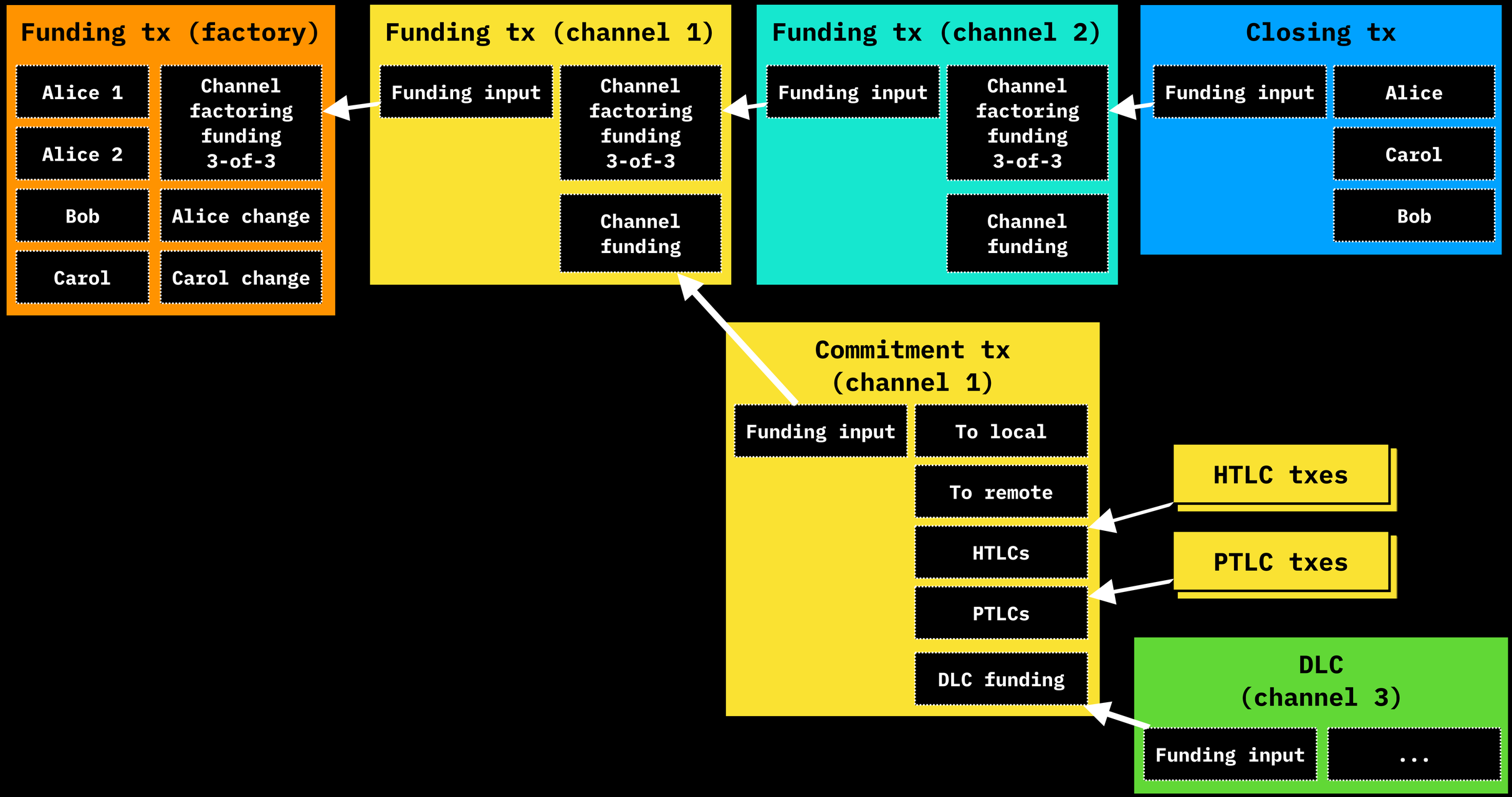

Funding tx (factory)
Alice 1
Alice 2
Channel factoring funding 3-of-3
Bob
Alice change
Carol
Carol change


Funding tx (factory)
Alice 1
Alice 2
Bob
Carol
Channel factoring funding 3-of-3
Alice change
Carol change
Funding tx (channel 1)
Funding input
Channel factoring funding 3-of-3
Channel funding


Funding tx (factory)
Alice 1
Alice 2
Bob
Carol
Channel factoring funding 3-of-3
Alice change
Carol change
Funding tx (channel 1)
Funding input
Channel factoring funding 3-of-3
Channel funding
Funding tx (channel 2)
Funding input
Channel factoring funding 3-of-3
Channel funding


Funding tx (factory)
Alice 1
Alice 2
Bob
Carol
Channel factoring funding 3-of-3
Alice change
Carol change
Funding tx (channel 1)
Funding input
Channel factoring funding 3-of-3
Channel funding
Funding tx (channel 2)
Funding input
Channel factoring funding 3-of-3
Channel funding
Closing tx
Funding input
Alice
Carol
Bob


Funding tx (factory)
Alice 1
Channel factoring funding 3-of-3
Alice 2
Bob
Alice change
Carol
Carol change
Funding tx (channel 1)
Funding input
Channel factoring funding 3-of-3
Channel funding
Funding tx (channel 2)
Funding input
Channel factoring funding 3-of-3
Channel funding
Closing tx
Funding input
Alice
Carol
Bob
Commitment tx (channel 1)
Funding input
To local
To remote
HTLCs
PTLCs
HTLC txes
PTLC txes


Funding tx (factory)
Alice 1
Channel factoring funding 3-of-3
Alice 2
Bob
Alice change
Carol
Carol change
Funding tx (channel 1)
Funding input
Channel factoring funding 3-of-3
Channel funding
Funding tx (channel 2)
Funding input
Channel factoring funding 3-of-3
Channel funding
Closing tx
Funding input
Alice
Carol
Bob
Commitment tx (channel 1)
Funding input
To local
To remote
HTLCs
PTLCs
DLC funding
HTLC txes
PTLC txes
DLC (channel 3)
Funding input
...

Generalization of channels require nodes to keep not just tx signatures, but full transaction structure, which may be channel-specific / highly custom

Negotiating channel structure

Done with **channel-based smart contract proposals (CSCP):**

- PSBT-like data structure
- “Templates” for transactions and channel transaction graph
- Can be contained within each other (allow channel composability)
- Signed by node key proposing them, helping in multiplier negotiation
- Limited to 64 kb in size (to fit both LN message and AluVM register)

2. Message types and encoding

Encoding changes

- Getting rid of BigInt type
- Switching to strict encoding (LNPBP-7),
  also used in client-side-validation
- Data types are fully AluVM compatible
- TLVs remain the same

We can do better than BOLT-9

- BOLT-9 de facto represents censorship for the new network upgrades, since each new feature must be approved by BOLT maintainers and must become part of the standard

  - blocks LN extensibility

- No simple hacks: use of high feature numbers explodes the length of feature vector

- Instead of BOLT-9 features Bifrost uses string numbers for standards describing features: LNPBP, BIP, BOLTs, ISO, RFC, … any. One can even create a custom standard name.

Part IV

Bifrost interoperability

Upgrading existing connection to Bifrost

- Done via hack into onion payment: most internal message contains custom message type and payload, specifying Bifrost connection point
- Allows NAT bypassing similarly to UDP hole punching technology
- Channels can be operated only in legacy or Bitforst, not both
  - Legacy LN -> Bifrost channel upgrade is always possible
  - Bifrost -> Legacy LN downgrade is possible if a channel is not being used (or removed use of) any of Bifrost-specific functionality

Upgrading channels to Bifrost

- Channels can be created in Bifrost directly
- Channels can be operated only in legacy or Bitforst, not both
- Network used to operate channel may be switched with the following rules:
  - Legacy LN -> Bifrost channel upgrade is always possible
  - Bifrost -> Legacy LN downgrade is possible if a channel is not being used (or removed use of) any of Bifrost-specific functionality

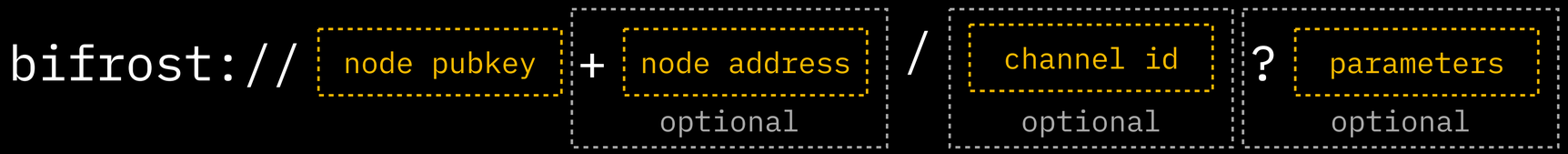
Invoicing

- All bifrost-operated channels MUST use LNPBP-38 invoices (Universal bitcoin invoices)
- BOLT-11 invoices MUST NOT be created for any channel which was moved into the Bifrost network

URLs

Bifrost URLs are designed to be compliant with RFC 3986

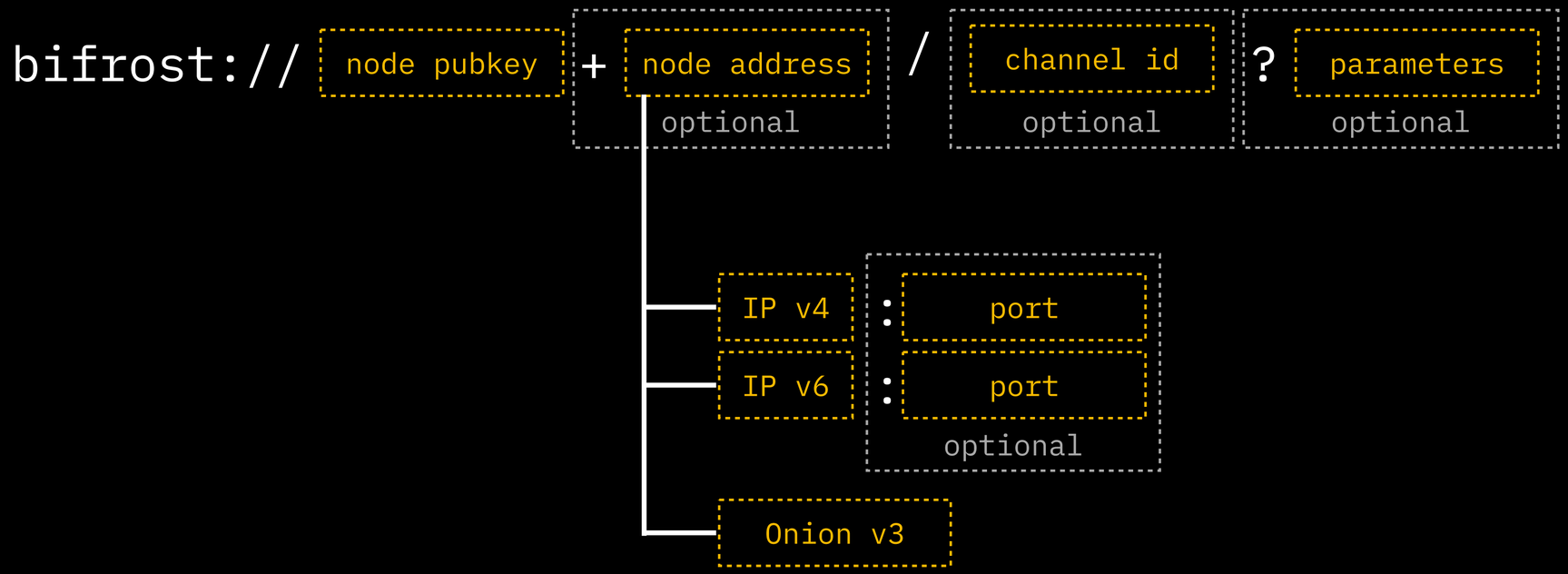
(legacy LN URLs are not standard-compatible)



URLs

Bifrost URLs are designed to be compliant with RFC 3986

(legacy LN URLs are not standard-compatible)



URLs

Bifrost URLs are designed to be compliant with RFC 3986

(legacy LN URLs are not standard-compatible)

Bifrost URLs for resource retrieval

Bifrost can operate as a distributed resource-storing network – as a better alternative to IPFS/Swarm/Bittorrent with incentivization

Part III

Implementing Bifrost

Resources

- Bifrost standard proposal: LNPBP-50
  github.com/LNP-BP/LNPBPs/pull/97
- LNP Core library implementing Bifrost data types & business logic
  github.com/LNP-BP/lnp-core/pull/6
- LNP Node: full lightning node with support for Bifrost
  github.com/LNP-BP/lnp-node
- Bifrost project on GitHub:
  github.com/orgs/LNP-BP/projects/5

Bifrost extensibility

- Today, a Bifrost-based custom application will require
  - writing code in Rust using LNP Core Lib
  - updating / forking LNP Node and adding app as a module to it
- In future it can be done as simple as
  - writing new custom channel or messaging business logic in any AluVM-compilable language
  - hot plug it into working LNP Node (without forking or compilation)
  - or even re-use the same AluVM module with other nodes, as soon as they will integrate AluVM (**quite sure this will not happen soon*)