

Networking with LNP

universal networking protocol for bitcoin world
– and much beyond!

LNP/BP Standards Association

Prepared & supervised by **Dr Maxim Orlovsky, Pandora Core AG**

Created with support from **Bitfinex** and **Fulgur Ventures**

The need for P2P & RPC solutions for Bitcoin ecosystem

Protocols

- DLCs: networking part
- LN extensions: channel factories etc
- RGB: P2P, RPC, Bifrost...
- Watchtowers (BOLT-13)
- Storm, prometheus & much more will follow

Products

- Better RPC for Bitcoin & LN nodes (JSON-RPC is really outdated)
- Microservice architectures (c-lightning a first example, but more will follow)

What networking is made of?

- Encoding protocol
- Transport protocols (framing, encryption & session management)
- Procedure invocation standard (P2P, RPC ...)

API Types

- **P2P**: peer-to-peer
 - peers, i.e. equal roles
 - sends message, no response
- **RPC**: remote procedure call or “client-server”
 - asymmetric roles (client & server)
 - client sends request to server and waits for reply
- **REST** and other RPC alternatives (**GraphAPI**): the same pattern as RPC; used in Web apps only, can be easily built with the same tools as RPC
- **SUB**: publication-subscription or PUB/SUB
 - asymmetric roles (publisher & subscriber)
 - publisher provides async event notifications to (potentially) multiple subscribers
 - subscriber does not send data to publisher

More information: <https://github.com/LNP-BP/LNPBPs/issues/21>

Encoding standards

	Transport	Languages	Code generation	Speed	Security	Interoperability	Community support
JSON / XML	Any	All	unvalidated key-values	low	low	perfect	perfect
Strict / consensus	Most	Rust	?	high	high	bitcoin	extra low
BOLT-1+9 (Message)	Most	Rust, C, Go, Scala *	schema-validated key-values	intermediate	moderate	lightning	niche
Avaro	Only Avaro transport	Most	schema-validated key-values	high	moderate	hadoop	big data community
Thrift binary	Only Thrift transport	Most	generated native code	high	moderate	poor	moderate
Thrift binary compact	Only Thrift transport	Most	generated native code	high	moderate	poor	moderate
Protobufs	Most	Most	generated native code	high	moderate	high	good

Transport framing protocols

	Connection	Security	Languages	Firewall performance
ZMQ Framing	POSIX Sockets, TCP, Inproc	None	Most	Bad
Apache Thrift Framed	HTTP, TCP, File, Inproc	None	Most	Moderate
BOLT-8	TCP, POSIX Sockets	Decentralized	Rust, C, Go, Scala *	Bad (good with Tor)
gRPC Cronet	TCP, POSIX Sockets	None	Most	Moderate
HTTP	TCP	TLS	Most	Good
WebSocket	TCP	TLS	Most	Moderate
Raw TCP	IP	TLS	Most	Moderate

Remote procedure calls & REST

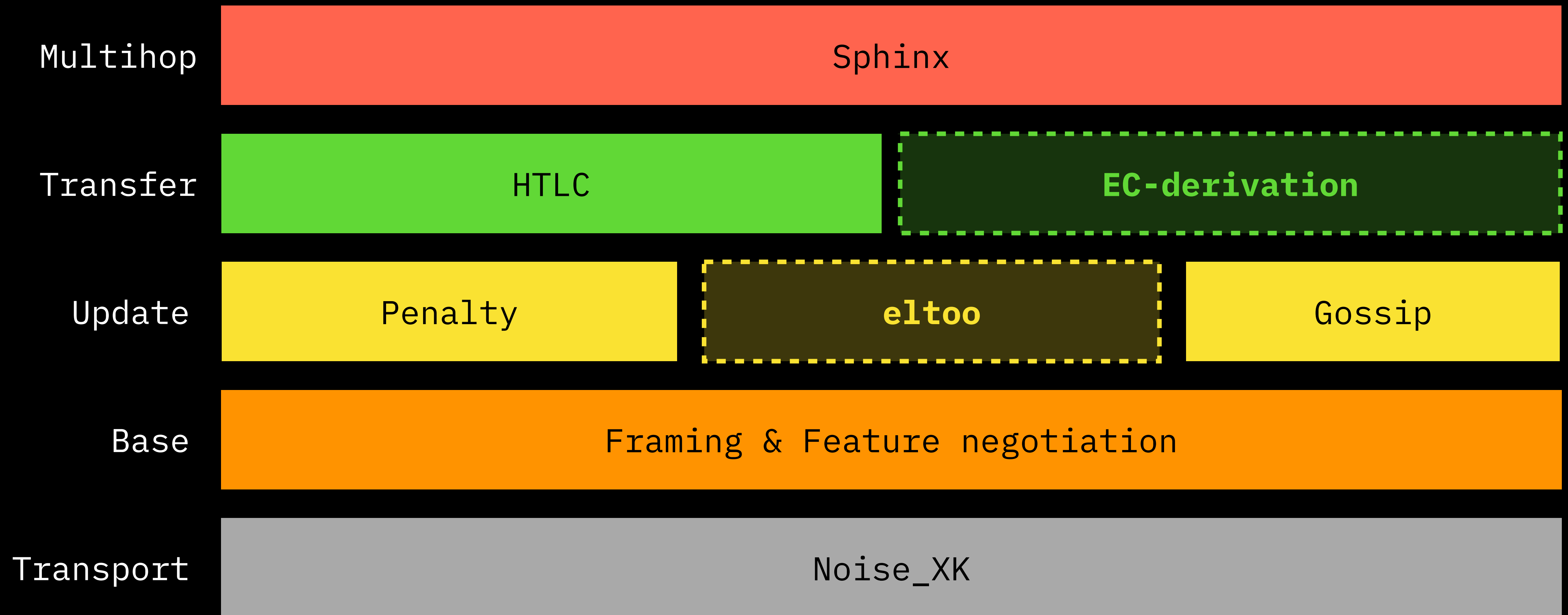
	Transport	Encoding	Category	Languages
Apache Thrift	HTTP, TCP, Inproc	JSON, Thrift	RPC	Most
Apache Avaro	HTTP, TCP, Inproc	Avaro	RPC	Most
gRPC	HTTP, TCP, Inproc	Protobuf	RPC	Most
JSON-RPC	HTTP, TCP	JSON	RPC	Most
OpenAPI	HTTP	JSON	REST	JS+
SOAP/WSDL	HTTP	XML	RPC	Most
WAMP	WebSockets	JSON	RPC	JS+
XML-RPC	HTTP	XML	RPC	Most
ZMQs	ZMQ Framing	Any	RPC	Most
BOLT-1 (RPC)	TCP, BOLT-8	BOLT-1	RPC	Rust, C, Go, Scala *

Requirements

- SSL + DNS -> Tor-like id's
- Routed
- End-to-end encrypted. Always.
- Native work with hashes, public keys etc
- Already have adoption
- Suited for both P2P & RPC
- Must work over Tor
- Works with ZMQ (optionally)
- Works over Websockets
- ~~OpenAPI~~
- ~~Thrift~~
- ~~Protobufs~~
- ~~Avaro~~
- ~~WAMP, crossbar.io~~
- **LN P2P!**

Lightning Network Architecture

after Christian Decker



Presenting LNP:

- We took LN P2P protocols (BOLT-8, BOLT-1, BOLT-9)
- ...dissected into layers
- ...added support for Websockets & ZMQ
- ...added support for RPC & Pub/Sub APIs
- ...added encoding enhancements

LNP: universal networking protocol for bitcoin world

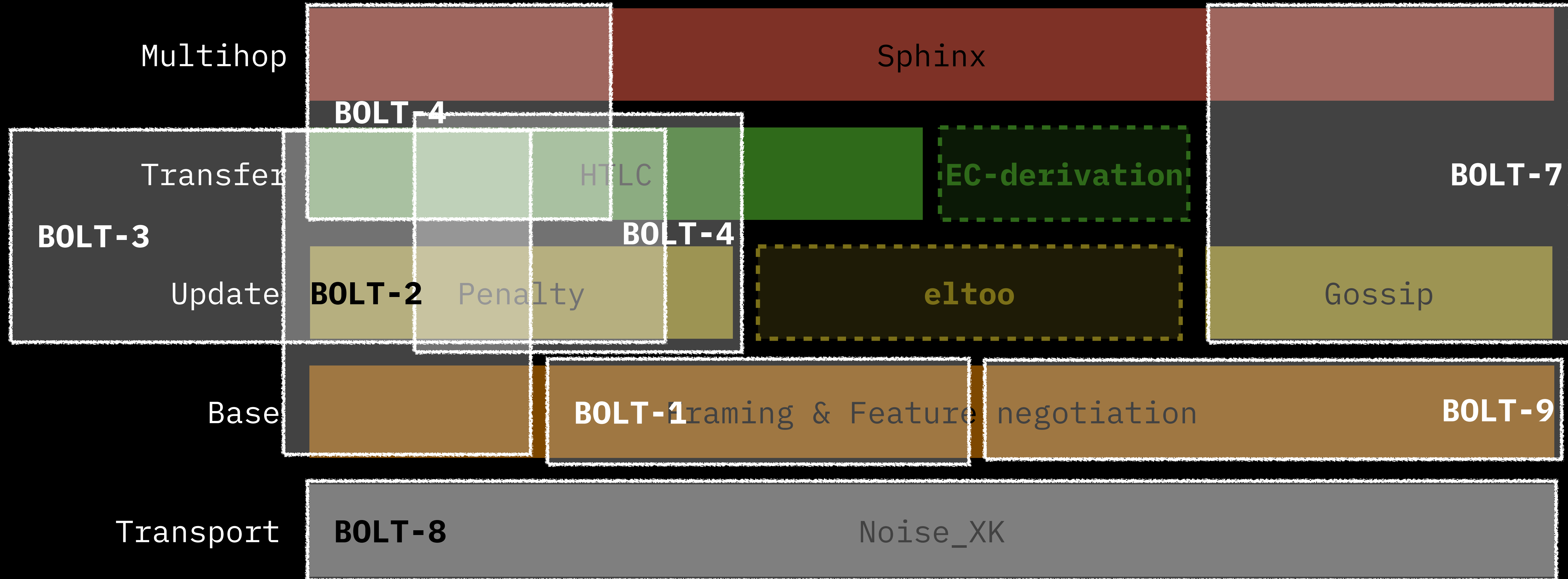
Decentralized & encrypted

- **No SSL; no PKI**, meaning
 - no centrally-issued certificates & authorities
 - no dependency on DNSes, that can be censored
 - no dependency on CAs, that can be censored
- **Tor-like node ids** and **onion-routing**
- Complete **end-to-end encryption** for all data
- Uses
 - native bitcoin consensus encoding (where defined)
 - LN encoding (data types from BOLT-1, 2, 4, 7)
 - LNP/BP strict encoding (LNPBP-7) used by RGB
- Can pass firewalls (with Tor and UDP hole punching)

Interoperable

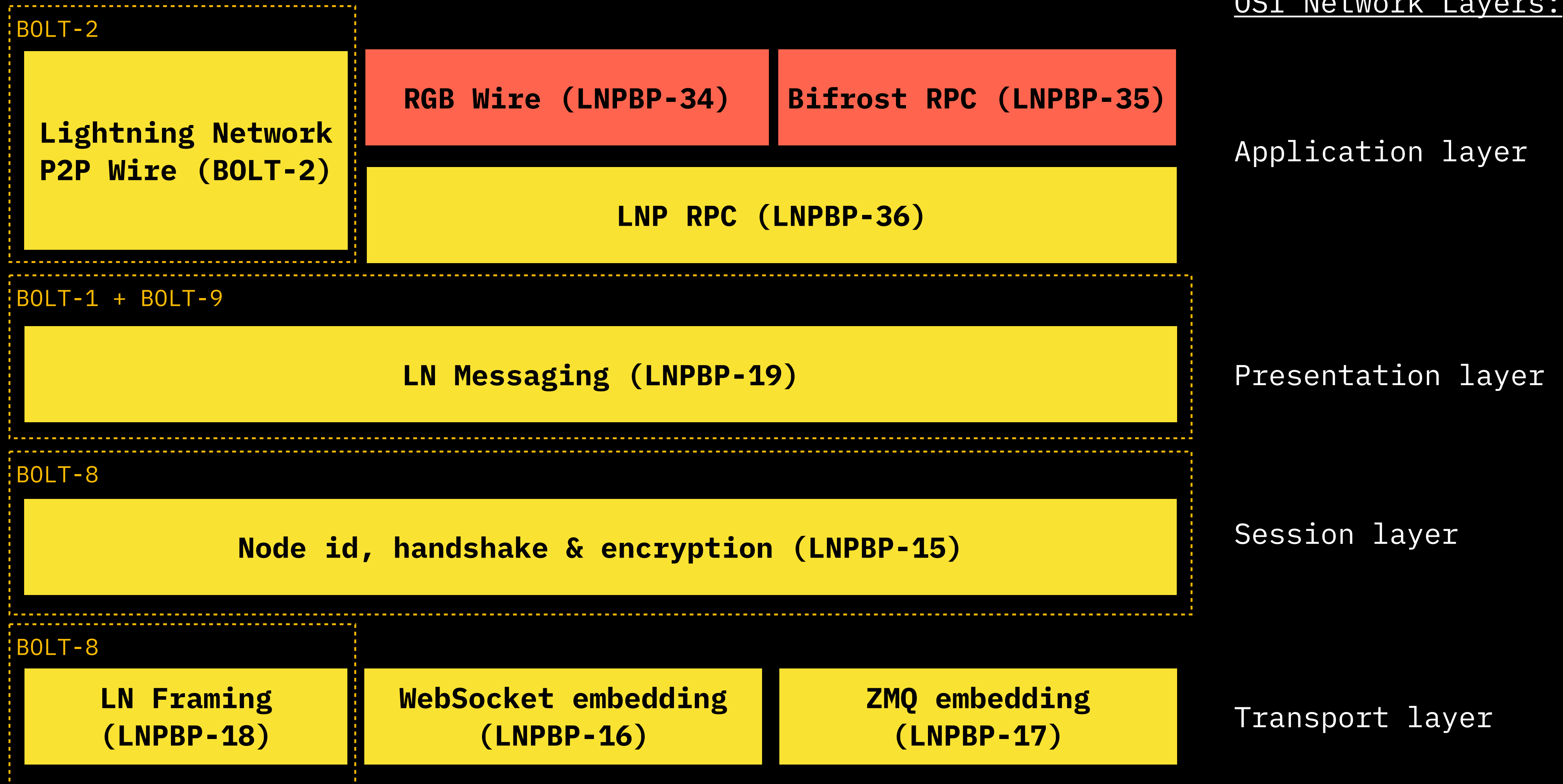
- Lightning native citizen: already used by LN
- Natively works with Tor and raw TCP sockets
- Now extended to work over
 - UDP & UDP hole punching
 - WebSockets
 - ZMQ for in-process, inter-process and network comms.
 - May work with MTCP, QUICK
- Single RPC protocol standard for all LNP/BP apps
 - Microservice architectures (used by LNP, BP and RGB nodes internally)
 - Peer wire protocols (LN wire protocol, RGB wire protocol)
 - Client-server protocols (like cli tools): replacement for JSON-RPC

Real BOLT Specifications



Lightning Network Protocols (LNP) suite

OSI Network Layers:



Lightning Network Protocols:

LN wire protocol layers dissected

- **Session layer:** identification & encryption
 - Defined in **LNPBP-15**
 - Noise_XK based (first half of **BOLT-8**)
 - manage decentralized node ID:
(Tor-like identity with Secp256k1 keys)
 - set up session-level encryption
 - do key rotation
- **Presentation layer:** identification & encryption
 - Message structure:
 - type (command)
 - payload parsing
 - TLVs
 - Defined in LNPBP-18
 - In fact, **BOLT-1** + **BOLT-9**
+ some additional recommendations
- **Transport layer:** framing protocols
 - LN native framing over TCP/IP or TCP/Tor (**BOLT-8** second half)
 - Support for ZMQ Sockets framing protocol in multiple variants:
 - P2P (PUSH/PULL ZMQ)
 - RPC (REQ/REP ZMQ)
 - Pub/Sub
 - ... over multiple connection layers:
 - Inproc & IPC (unencrypted)
 - TCP (encrypted & unencrypted)
 - UDP (potentially, important for Mesh & Satellite networks)
 - Support for WebSockets protocol
 - Support for SMTP protocol?
(Christian Decker proposal)

Rules for data serialisation

- Do not compress the data
- Use deterministically-defined value length
(Bitcoin/LN VarInt are bad practices)
- Define both lower and upper bounds for each type validity:
 - ranges for the number of occurrences
 - ranges for possible value (or length in case of strings)
- No pointers/offsets/shifts, no linked lists

LNP API (with C.Decker): LNPBP-36, 38, 39

Interface description

- Another IDL standard?! – No!
- Already works in c-lightning
- You can describe interface in:
 - YAML
 - TOML
 - JSON
 - CSV-based custom c-lightning format
 - Special language for LNP API (LIDL)
 - Binary form
(for network transfers & commitments)
- Can be provided in init message TLV extensions
- Will be defined in **LNPBP-19**

Toolset

github.com/LNP-BP/lnp-api-tools

- Cross-conversion of the standards
- API validation
- Generate language-specific wrappers –
but very small amount of code, audited by
developers
- Already used for LNP and c-lightning hybrids
- Used by all three nodes internally:
BP, LNP, RGB
- Used for RPC APIs to all three nodes
- Will be used by Bifrost

LNP API YAML interface description

- Follows strict encoding paradigm
- Language-specific customization
- Multi-file, allows extensions to existing protocols
- Can be used to write **formal deterministic** API specs (LNPBPs, BOLTS)

```
1  %YAML 1.2
2  %TAG !strict! tag:https://lnp-bp.org/lnp/strict.yaml
3  %TAG !wallet! tag:https://lnp-bp.org/lnp/wallet.yaml
4  ---
5  name: keyring
6  type: RPC
7  description: RPC API for Keyring service by Pandora Core
8  author: Dr Maxim Orlovsky <orlovsky@pandoracore.com>
9  ---
10
11 types:
12   second_auth_factor: !!u32
13     max: 999999
14
15   key:
16     - id: !wallet!xpubid
17     - xpubkey: !wallet!xpubkey
18     - path: !wallet!derivation_path
19     - fingerprint: !wallet!key_fingerprint
20
21   version:
22     - features: 0
23
24   messages:
25     &success 1:
26
27     &failure 0:
28       - code: !!u16
29       - info: !strict!utf8
30         max: 256
31
32     # Requests key listing
33     &keys 1000:
34
35     # Returned key list
36     &keylist 1001:
37       - keys: !strict!array
38         item: !key
39         max: 1024
40
```

```
41  # Generates a new seed & extended master private key
42  &seed 2000:
43    - auth_code: !second_auth_factor
44
45  &export 2100:
46    - key_id: !wallet!xpubid
47    - auth_code: !second_auth_factor
48
49  &xpriv 2101:
50    - xpriv: !wallet!xprivkey
51
52  &xpub 2102:
53    - xpub: !wallet!xpubkey
54
55  &derive 3000:
56    - from: !wallet!xpubid
57    - path: !wallet!derivation_path
58    - auth_code: !second_auth_factor
59
60  &sign 4000:
61    - psbt: !wallet!psbt
62
63  &psbt 4001:
64    - psbt: !wallet!psbt
65
66  extensions:
67    # No TLV extensions are defined
68
69  rpc:
70    # Responses returning either ok or error
71    - requests:
72      - &seed
73      - &derive
74    responses:
75      - &ok
76      - &error
77
78  - request: &keys
79    responses:
80      - &error
81      - &keylist
82
83  - request: &xpriv
84    responses:
85      - &error
86      - &xpriv
87
88  - request: &xpub
89    responses:
90      - &error
91      - &xpub
92
93  - request: &derive
94    responses:
95      - &error
96      - &derive
97
98  - request: &sign
99    responses:
100      - &error
101      - &sign
102
103  - request: &psbt
104    responses:
105      - &error
106      - &psbt
107
108  - request: &export
109    responses:
110      - &error
111      - &export
112
113  - request: &seed
114    responses:
115      - &error
116      - &seed
117
118  - request: &keys
119    responses:
120      - &error
121      - &keys
122
123  - request: &keylist
124    responses:
125      - &error
126      - &keylist
127
128  - request: &success
129    responses:
130      - &error
131      - &success
132
133  - request: &failure
134    responses:
135      - &error
136      - &failure
137
138  - request: &psbt
139    responses:
140      - &error
141      - &psbt
142
143  - request: &sign
144    responses:
145      - &error
146      - &sign
147
148  - request: &derive
149    responses:
150      - &error
151      - &derive
152
153  - request: &xpriv
154    responses:
155      - &error
156      - &xpriv
157
158  - request: &xpub
159    responses:
160      - &error
161      - &xpub
162
163  - request: &export
164    responses:
165      - &error
166      - &export
167
168  - request: &seed
169    responses:
170      - &error
171      - &seed
172
173  - request: &keys
174    responses:
175      - &error
176      - &keys
177
178  - request: &keylist
179    responses:
180      - &error
181      - &keylist
182
183  - request: &success
184    responses:
185      - &error
186      - &success
187
188  - request: &failure
189    responses:
190      - &error
191      - &failure
192
193  - request: &psbt
194    responses:
195      - &error
196      - &psbt
197
198  - request: &sign
199    responses:
200      - &error
201      - &sign
202
203  - request: &derive
204    responses:
205      - &error
206      - &derive
207
208  - request: &xpriv
209    responses:
210      - &error
211      - &xpriv
212
213  - request: &xpub
214    responses:
215      - &error
216      - &xpub
217
218  - request: &export
219    responses:
220      - &error
221      - &export
222
223  - request: &seed
224    responses:
225      - &error
226      - &seed
227
228  - request: &keys
229    responses:
230      - &error
231      - &keys
232
233  - request: &keylist
234    responses:
235      - &error
236      - &keylist
237
238  - request: &success
239    responses:
240      - &error
241      - &success
242
243  - request: &failure
244    responses:
245      - &error
246      - &failure
247
248  - request: &psbt
249    responses:
250      - &error
251      - &psbt
252
253  - request: &sign
254    responses:
255      - &error
256      - &sign
257
258  - request: &derive
259    responses:
260      - &error
261      - &derive
262
263  - request: &xpriv
264    responses:
265      - &error
266      - &xpriv
267
268  - request: &xpub
269    responses:
270      - &error
271      - &xpub
272
273  - request: &export
274    responses:
275      - &error
276      - &export
277
278  - request: &seed
279    responses:
280      - &error
281      - &seed
282
283  - request: &keys
284    responses:
285      - &error
286      - &keys
287
288  - request: &keylist
289    responses:
290      - &error
291      - &keylist
292
293  - request: &success
294    responses:
295      - &error
296      - &success
297
298  - request: &failure
299    responses:
300      - &error
301      - &failure
302
303  - request: &psbt
304    responses:
305      - &error
306      - &psbt
307
308  - request: &sign
309    responses:
310      - &error
311      - &sign
312
313  - request: &derive
314    responses:
315      - &error
316      - &derive
317
318  - request: &xpriv
319    responses:
320      - &error
321      - &xpriv
322
323  - request: &xpub
324    responses:
325      - &error
326      - &xpub
327
328  - request: &export
329    responses:
330      - &error
331      - &export
332
333  - request: &seed
334    responses:
335      - &error
336      - &seed
337
338  - request: &keys
339    responses:
340      - &error
341      - &keys
342
343  - request: &keylist
344    responses:
345      - &error
346      - &keylist
347
348  - request: &success
349    responses:
350      - &error
351      - &success
352
353  - request: &failure
354    responses:
355      - &error
356      - &failure
357
358  - request: &psbt
359    responses:
360      - &error
361      - &psbt
362
363  - request: &sign
364    responses:
365      - &error
366      - &sign
367
368  - request: &derive
369    responses:
370      - &error
371      - &derive
372
373  - request: &xpriv
374    responses:
375      - &error
376      - &xpriv
377
378  - request: &xpub
379    responses:
380      - &error
381      - &xpub
382
383  - request: &export
384    responses:
385      - &error
386      - &export
387
388  - request: &seed
389    responses:
390      - &error
391      - &seed
392
393  - request: &keys
394    responses:
395      - &error
396      - &keys
397
398  - request: &keylist
399    responses:
400      - &error
401      - &keylist
402
403  - request: &success
404    responses:
405      - &error
406      - &success
407
408  - request: &failure
409    responses:
410      - &error
411      - &failure
412
413  - request: &psbt
414    responses:
415      - &error
416      - &psbt
417
418  - request: &sign
419    responses:
420      - &error
421      - &sign
422
423  - request: &derive
424    responses:
425      - &error
426      - &derive
427
428  - request: &xpriv
429    responses:
430      - &error
431      - &xpriv
432
433  - request: &xpub
434    responses:
435      - &error
436      - &xpub
437
438  - request: &export
439    responses:
440      - &error
441      - &export
442
443  - request: &seed
444    responses:
445      - &error
446      - &seed
447
448  - request: &keys
449    responses:
450      - &error
451      - &keys
452
453  - request: &keylist
454    responses:
455      - &error
456      - &keylist
457
458  - request: &success
459    responses:
460      - &error
461      - &success
462
463  - request: &failure
464    responses:
465      - &error
466      - &failure
467
468  - request: &psbt
469    responses:
470      - &error
471      - &psbt
472
473  - request: &sign
474    responses:
475      - &error
476      - &sign
477
478  - request: &derive
479    responses:
480      - &error
481      - &derive
482
483  - request: &xpriv
484    responses:
485      - &error
486      - &xpriv
487
488  - request: &xpub
489    responses:
490      - &error
491      - &xpub
492
493  - request: &export
494    responses:
495      - &error
496      - &export
497
498  - request: &seed
499    responses:
500      - &error
501      - &seed
502
503  - request: &keys
504    responses:
505      - &error
506      - &keys
507
508  - request: &keylist
509    responses:
510      - &error
511      - &keylist
512
513  - request: &success
514    responses:
515      - &error
516      - &success
517
518  - request: &failure
519    responses:
520      - &error
521      - &failure
522
523  - request: &psbt
524    responses:
525      - &error
526      - &psbt
527
528  - request: &sign
529    responses:
530      - &error
531      - &sign
532
533  - request: &derive
534    responses:
535      - &error
536      - &derive
537
538  - request: &xpriv
539    responses:
540      - &error
541      - &xpriv
542
543  - request: &xpub
544    responses:
545      - &error
546      - &xpub
547
548  - request: &export
549    responses:
550      - &error
551      - &export
552
553  - request: &seed
554    responses:
555      - &error
556      - &seed
557
558  - request: &keys
559    responses:
560      - &error
561      - &keys
562
563  - request: &keylist
564    responses:
565      - &error
566      - &keylist
567
568  - request: &success
569    responses:
570      - &error
571      - &success
572
573  - request: &failure
574    responses:
575      - &error
576      - &failure
577
578  - request: &psbt
579    responses:
580      - &error
581      - &psbt
582
583  - request: &sign
584    responses:
585      - &error
586      - &sign
587
588  - request: &derive
589    responses:
590      - &error
591      - &derive
592
593  - request: &xpriv
594    responses:
595      - &error
596      - &xpriv
597
598  - request: &xpub
599    responses:
600      - &error
601      - &xpub
602
603  - request: &export
604    responses:
605      - &error
606      - &export
607
608  - request: &seed
609    responses:
610      - &error
611      - &seed
612
613  - request: &keys
614    responses:
615      - &error
616      - &keys
617
618  - request: &keylist
619    responses:
620      - &error
621      - &keylist
622
623  - request: &success
624    responses:
625      - &error
626      - &success
627
628  - request: &failure
629    responses:
630      - &error
631      - &failure
632
633  - request: &psbt
634    responses:
635      - &error
636      - &psbt
637
638  - request: &sign
639    responses:
640      - &error
641      - &sign
642
643  - request: &derive
644    responses:
645      - &error
646      - &derive
647
648  - request: &xpriv
649    responses:
650      - &error
651      - &xpriv
652
653  - request: &xpub
654    responses:
655      - &error
656      - &xpub
657
658  - request: &export
659    responses:
660      - &error
661      - &export
662
663  - request: &seed
664    responses:
665      - &error
666      - &seed
667
668  - request: &keys
669    responses:
670      - &error
671      - &keys
672
673  - request: &keylist
674    responses:
675      - &error
676      - &keylist
677
678  - request: &success
679    responses:
680      - &error
681      - &success
682
683  - request: &failure
684    responses:
685      - &error
686      - &failure
687
688  - request: &psbt
689    responses:
690      - &error
691      - &psbt
692
693  - request: &sign
694    responses:
695      - &error
696      - &sign
697
698  - request: &derive
699    responses:
700      - &error
701      - &derive
702
703  - request: &xpriv
704    responses:
705      - &error
706      - &xpriv
707
708  - request: &xpub
709    responses:
710      - &error
711      - &xpub
712
713  - request: &export
714    responses:
715      - &error
716      - &export
717
718  - request: &seed
719    responses:
720      - &error
721      - &seed
722
723  - request: &keys
724    responses:
725      - &error
726      - &keys
727
728  - request: &keylist
729    responses:
730      - &error
731      - &keylist
732
733  - request: &success
734    responses:
735      - &error
736      - &success
737
738  - request: &failure
739    responses:
740      - &error
741      - &failure
742
743  - request: &psbt
744    responses:
745      - &error
746      - &psbt
747
748  - request: &sign
749    responses:
750      - &error
751      - &sign
752
753  - request: &derive
754    responses:
755      - &error
756      - &derive
757
758  - request: &xpriv
759    responses:
760      - &error
761      - &xpriv
762
763  - request: &xpub
764    responses:
765      - &error
766      - &xpub
767
768  - request: &export
769    responses:
770      - &error
771      - &export
772
773  - request: &seed
774    responses:
775      - &error
776      - &seed
777
778  - request: &keys
779    responses:
780      - &error
781      - &keys
782
783  - request: &keylist
784    responses:
785      - &error
786      - &keylist
787
788  - request: &success
789    responses:
790      - &error
791      - &success
792
793  - request: &failure
794    responses:
795      - &error
796      - &failure
797
798  - request: &psbt
799    responses:
800      - &error
801      - &psbt
802
803  - request: &sign
804    responses:
805      - &error
806      - &sign
807
808  - request: &derive
809    responses:
810      - &error
811      - &derive
812
813  - request: &xpriv
814    responses:
815      - &error
816      - &xpriv
817
818  - request: &xpub
819    responses:
820      - &error
821      - &xpub
822
823  - request: &export
824    responses:
825      - &error
826      - &export
827
828  - request: &seed
829    responses:
830      - &error
831      - &seed
832
833  - request: &keys
834    responses:
835      - &error
836      - &keys
837
838  - request: &keylist
839    responses:
840      - &error
841      - &keylist
842
843  - request: &success
844    responses:
845      - &error
846      - &success
847
848  - request: &failure
849    responses:
850      - &error
851      - &failure
852
853  - request: &psbt
854    responses:
855      - &error
856      - &psbt
857
858  - request: &sign
859    responses:
860      - &error
861      - &sign
862
863  - request: &derive
864    responses:
865      - &error
866      - &derive
867
868  - request: &xpriv
869    responses:
870      - &error
871      - &xpriv
872
873  - request: &xpub
874    responses:
875      - &error
876      - &xpub
877
878  - request: &export
879    responses:
880      - &error
881      - &export
882
883  - request: &seed
884    responses:
885      - &error
886      - &seed
887
888  - request: &keys
889    responses:
890      - &error
891      - &keys
892
893  - request: &keylist
894    responses:
895      - &error
896      - &keylist
897
898  - request: &success
899    responses:
900      - &error
901      - &success
902
903  - request: &failure
904    responses:
905      - &error
906      - &failure
907
908  - request: &psbt
909    responses:
910      - &error
911      - &psbt
912
913  - request: &sign
914    responses:
915      - &error
916      - &sign
917
918  - request: &derive
919    responses:
920      - &error
921      - &derive
922
923  - request: &xpriv
924    responses:
925      - &error
926      - &xpriv
927
928  - request: &xpub
929    responses:
930      - &error
931      - &xpub
932
933  - request: &export
934    responses:
935      - &error
936      - &export
937
938  - request: &seed
939    responses:
940      - &error
941      - &seed
942
943  - request: &keys
944    responses:
945      - &error
946      - &keys
947
948  - request: &keylist
949    responses:
950      - &error
951      - &keylist
952
953  - request: &success
954    responses:
955      - &error
956      - &success
957
958  - request: &failure
959    responses:
960      - &error
961      - &failure
962
963  - request: &psbt
964    responses:
965      - &error
966      - &psbt
967
968  - request: &sign
969    responses:
970      - &error
971      - &sign
972
973  - request: &derive
974    responses:
975      - &error
976      - &derive
977
978  - request: &xpriv
979    responses:
980      - &error
981      - &xpriv
982
983  - request: &xpub
984    responses:
985      - &error
986      - &xpub
987
988  - request: &export
989    responses:
990      - &error
991      - &export
992
993  - request: &seed
994    responses:
995      - &error
996      - &seed
997
998  - request: &keys
999    responses:
1000      - &error
1001      - &keys
1002
1003  - request: &keylist
1004    responses:
1005      - &error
1006      - &keylist
1007
1008  - request: &success
1009    responses:
1010      - &error
1011      - &success
1012
1013  - request: &failure
1014    responses:
1015      - &error
1016      - &failure
1017
1018  - request: &psbt
1019    responses:
1020      - &error
1021      - &psbt
1022
1023  - request: &sign
1024    responses:
1025      - &error
1026      - &sign
1027
1028  - request: &derive
1029    responses:
1030      - &error
1031      - &derive
1032
1033  - request: &xpriv
1034    responses:
1035      - &error
1036      - &xpriv
1037
1038  - request: &xpub
1039    responses:
1040      - &error
1041      - &xpub
1042
1043  - request: &export
1044    responses:
1045      - &error
1046      - &export
1047
1048  - request: &seed
1049    responses:
1050      - &error
1051      - &seed
1052
1053  - request: &keys
1054    responses:
1055      - &error
1056      - &keys
1057
1058  - request: &keylist
1059    responses:
1060      - &error
1061      - &keylist
1062
1063  - request: &success
1064    responses:
1065      - &error
1066      - &success
1067
1068  - request: &failure
1069    responses:
1070      - &error
1071      - &failure
1072
1073  - request: &psbt
1074    responses:
1075      - &error
1076      - &psbt
1077
1078  - request: &sign
1079    responses:
1080      - &error
1081      - &sign
1082
1083  - request: &derive
1084    responses:
1085      - &error
1086      - &derive
1087
1088  - request: &xpriv
1089    responses:
1090      - &error
1091      - &xpriv
1092
1093  - request: &xpub
1094    responses:
1095      - &error
1096      - &xpub
1097
1098  - request: &export
1099    responses:
1100      - &error
1101      - &export
1102
1103  - request: &seed
1104    responses:
1105      - &error
1106      - &seed
1107
1108  - request: &keys
1109    responses:
1110      - &error
1111      - &keys
1112
1113  - request: &keylist
1114    responses:
1115      - &error
1116      - &keylist
1117
1118  - request: &success
1119    responses:
1120      - &error
1121      - &success
1122
1123  - request: &failure
1124    responses:
1125      - &error
1126      - &failure
1127
1128  - request: &psbt
1129    responses:
1130      - &error
1131      - &psbt
1132
1133  - request: &sign
1134    responses:
1135      - &error
1136      - &sign
1137
1138  - request: &derive
1139    responses:
1140      - &error
1141      - &derive
1142
1143  - request: &xpriv
1144    responses:
1145      - &error
1146      - &xpriv
1147
1148  - request: &xpub
1149    responses:
1150      - &error
1151      - &xpub
1152
1153  - request: &export
1154    responses:
1155      - &error
1156      - &export
1157
1158  - request: &seed
1159    responses:
1160      - &error
1161      - &seed
1162
1163  - request: &keys
1164    responses:
1165      - &error
1166      - &keys
1167
1168  - request: &keylist
1169    responses:
1170      - &error
1171      - &keylist
1172
1173  - request: &success
1174    responses:
1175      - &error
1176      - &success
1177
1178  - request: &failure
1179    responses:
1180      - &error
1181      - &failure
1182
1183  - request: &psbt
1184    responses:
1185      - &error
1186      - &psbt
1187
1188  - request: &sign
1189    responses:
1190      - &error
1191      - &sign
1192
1193  - request: &derive
1194    responses:
1195      - &error
1196      - &derive
1197
1198  - request: &xpriv
1199    responses:
1200      - &error
1201      - &xpriv
1202
1203  - request: &xpub
1204    responses:
1205      - &error
1206      - &xpub
1207
1208  - request: &export
1209    responses:
1210      - &error
1211      - &export
1212
1213  - request: &seed
1214    responses:
1215      - &error
1216      - &seed
1217
1218  - request: &keys
1219    responses:
1220      - &error
1221      - &keys
1222
1223  - request: &keylist
1224    responses:
1225      - &error
1226      - &keylist
1227
1228  - request: &success
1229    responses:
1230      - &error
1231      - &success
1232
1233  - request: &failure
1234    responses:
1235      - &error
1236      - &failure
1237
1238  - request: &psbt
1239    responses:
1240      - &error
1241      - &psbt
1242
1243  - request: &sign
1244    responses:
1245      - &error
1246      - &sign
1247
1248  - request: &derive
1249    responses:
1250      - &error
1251      - &derive
1252
1253  - request: &xpriv
1254    responses:
1255      - &error
1256      - &xpriv
1257
1258  - request: &xpub
1259    responses:
1260      - &error
1261      - &xpub
1262
1263  - request: &export
1264    responses:
1265      - &error
1266      - &export
1267
1268  - request: &seed
1269    responses:
1270      - &error
1271      - &seed
1272
1273  - request: &keys
1274    responses:
1275      - &error
1276      - &keys
1277
1278  - request: &keylist
1279    responses:
1280      - &error
1281      - &keylist
1282
1283  - request: &success
1284    responses:
1285      - &error
1286      - &success
1287
1288  - request: &failure
1289    responses:
1290      - &error
1291      - &failure
1292
1293  - request: &psbt
1294    responses:
1295      - &error
1296      - &psbt
1297
1298  - request: &sign
1299    responses:
1300      - &error
1301      - &sign
1302
1303  - request: &derive
1304    responses:
1305      - &error
1306      - &derive
1307
1308  - request: &xpriv
1309    responses:
1310      - &error
1311      - &xpriv
1312
1313  - request: &xpub
1314    responses:
1315      - &error
1316      - &xpub
1317
1318  - request: &export
1319    responses:
1320      - &error
1321      - &export
1322
1323  - request: &seed
1324    responses:
1325      - &error
1326      - &seed
1327
1328  - request: &keys
1329    responses:
1330      - &error
1331      - &keys
1332
1333  - request: &keylist
1334    responses:
1335      - &error
1336      - &keylist
1337
1338  - request: &success
1339    responses:
1340      - &error
1341      - &success
1342
1343  - request: &failure
1344    responses:
1345      - &error
1346      - &failure
1347
1348  - request: &psbt
1349    responses:
1350      - &error
1351      - &psbt
1352
1353  - request: &sign
1354    responses:
1355      - &error
1356      - &sign
1357
1358  - request: &amp
```

Code autogeneration

```
13
14     #[derive(Clone, Debug, Display, LnpApi)]
15     #[lnp_api(encoding = "strict")]
16     #[display_from(Debug)]
17     #[non_exhaustive]
18     pub enum Request {
19         #[lnp_api(type = 0x0201)]
20         List,
21
22         #[lnp_api(type = 0x0203)]
23         Seed(crate::api::message::Seed),
24
25         #[lnp_api(type = 0x0301)]
26         Export(crate::api::message::Export),
27
28         #[lnp_api(type = 0x0401)]
29         Derive(crate::api::message::Derive),
30     }
31
```

```
18     #[derive(Clone, Debug, Display, StrictEncode, StrictDecode)]
19     #[display_from(Debug)]
20     #[non_exhaustive]
21     pub struct Seed {
22         pub auth_code: AuthCode,
23         pub name: String,
24         pub description: Option<String>,
25     }
26
27     #[derive(Clone, Debug, Display, StrictEncode, StrictDecode)]
28     #[display_from(Debug)]
29     #[non_exhaustive]
30     pub struct Export {
31         pub key_id: XpubIdentifier,
32         pub auth_code: AuthCode,
33     }
34
35     #[derive(Clone, Debug, Display, StrictEncode, StrictDecode)]
36     #[display_from(Debug)]
37     #[non_exhaustive]
38     pub struct Derive {
39         pub from: XpubIdentifier,
40         pub path: DerivationPath,
41         pub auth_code: AuthCode,
42     }
43
44     #[derive(Clone, Debug, Display, StrictEncode, StrictDecode)]
45     #[display_from(Debug)]
46     #[non_exhaustive]
47     pub struct Failure {
48         pub code: u16,
49         pub info: String,
50     }
51
```

Simplicity of implementation

```
96  async fn rpc_process(&mut self, raw: Vec<u8>) -> Result<Reply, Reply> {
97      trace!("Got {} bytes over ZMQ RPC: {:?}", raw.len(), raw);
98      let message :&? = (&*self.unmarshaller.unmarshall(&raw)?).clone();
99      debug!("Received ZMQ RPC request: {:?}", message);
100     match message {
101         Request::Seed(seed :&Seed) => self.rpc_seed_create(seed).await,
102         Request::List => self.rpc_list().await,
103         _ => unimplemented!(),
104     }
105 }
106
107 async fn rpc_seed_create(&mut self, seed: message::Seed) -> Result<Reply, Reply> {
108     trace!("Awaiting for the vault lock");
109     self.vault
110         .lock()
111         .await
112         .seed(seed.name, seed.description, encryption_key: &self.config.node_id())?;
113     trace!("Vault lock released");
114     Ok(Reply::Success)
115 }
116
117 async fn rpc_list(&mut self) -> Result<Reply, Reply> {
118     trace!("Awaiting for the vault lock");
119     let accounts :Vec<AccountInfo> = self.vault.lock().await.list()?;
120     trace!("Vault lock released");
121     Ok(Reply::Keylist(accounts))
122 }
```


LNP API outside of RGB & LN

- Can be used to build **messenger** outside of LN network
 - optional (not required) bitcoin payments:
 - lightning invoices
 - LSAT
 - Lightspeed
 - always end-to-end encrypted, even if central server is present
 - can work over Tor and Mesh networks from day 0
- A proposal by A. Riard to move **Bitcoin Core RPC** on (de facto) this protocol

<https://twitter.com/Snyke/status/1262024134088970243?s=19>

Not a new standard!

- any protocol designed in the same way as LN P2P will be automatically compliant
- **LNP API** is just a “soft-fork” extension of **LN P2P** protocols enabling them for different types of networks & transport layers (Mesh, Satellite, interprocess/IPC, Websockets etc)
- “Compatible without being (previously) aware”:
 - BOLT-13 (watchtowers)
 - Bitcoin Core RPC proposal (A. Riard)

LNP API Summary

Framing protocol	Standard	Encryption (BOLT-9 / LNPBP-15)	Possible API types	When to use
TCP/IP & TCP/Tor	BOLT-9,1 / LNPBP-18	always	P2P, RPC	Default in network
ZMQ	LNPBP-17	none	P2P, RPC, SUB	DMZ networking, ESB, IPC, inproc
Websockets	LNPBP-16	always	P2P, RPC	Web apps
UDP	WIP	always	P2P	Low connectivity, Mesh, Satellite
SMTP	WIP	always	P2P	Mesh, Satellite, "Offline" (ultra-low connectivity)

LNP API transport layer selection

- Inter-process and in-process (inter-thread) APIs:
 - use unencrypted Inproc & IPC ZMQ
 - PULL/PUSH, REQ/REP and PUB/SUB sockets
- Client-server RPCs &
- P2P networks
 - use either TCP/IP, TCP/Tor or Websockets (for web-related systems); always encrypted
 - for DMZ, use ZMQ-based variant (may be unencrypted)
- Mesh and satellite networks
 - use UDP or SMTP; always encrypted

```
26 // Universal Node Locator (from LNPBP-19)
27 // NB: DNS addressing is not used since it is considered insecure in terms of
28 // censorship resistance.
29 #[derive(Clone)]
30 pub enum NodeLocator {
31     // Native Lightning network connection: uses end-to-end encryption and
32     // runs on top of either TCP or Tor socket
33     // # URL Schema
34     // lnp://<node-id>@<ip>|<onion>:<port>
35     Native(secp256k1::PublicKey, InetAddr, Option<u16>),
36
37     // UDP-based connection that uses UDP packets instead of TCP. Can't work
38     // with Tor, but may use UDP hole punching in a secure way, since the
39     // connection is still required to be encrypted.
40     // # URL Schema
41     // lnp-udp://<node-id>@<ip>:<port>
42     Udp(secp256k1::PublicKey, IpAddr, Option<u16>),
43
44     // Local (for inter-process communication based on POSIX sockets)
45     // connection without encryption. Relies on ZMQ IPC sockets internally;
46     // specific socket pair for ZMQ is provided via query parameter
47     // # URL Schema
48     // lnp:<file-path>?api=<p2p|rpc|sub>
49     #[cfg(feature = "zmq")]
50     Ipc(PathBuf, ZmqType),
51
52     // In-process communications (between threads of the same process using
53     // Mutex'es and other sync managing routines) without encryption.
54     // Relies on ZMQ IPC sockets internally; specific socket pair for ZMQ is
55     // provided via query parameter
56     // # URL Schema
57     // lnp:?api=<p2p|rpc|sub>#<id>
58     #[cfg(feature = "zmq")]
59     Inproc(String, zmq::Context, ZmqType),
60
61     // SHOULD be used only for DMZ area connections; otherwise Native or
62     // Websocket-based connection MUST be used
63     // # URL Schema
64     // lnp-zmq://<node-id>@<ip>|<onion>:<port>/?api=<p2p|rpc|sub>
65     #[cfg(feature = "zmq")]
66     ZmqEncrypted(secp256k1::PublicKey, ZmqType, IpAddr, Option<u16>),
67
68     // SHOULD be used only for DMZ area connections; otherwise Native or
69     // Websocket-based connection MUST be used
70     // # URL Schema
71     // lnp-zmq://<ip>|<onion>:<port>/?api=<p2p|rpc|sub>
72     #[cfg(feature = "zmq")]
73     ZmqUnencrypted(ZmqType, IpAddr, Option<u16>),
74
75     // # URL Schema
76     // lnp-ws://<node-id>@<ip>|<onion>:<port>
77     #[cfg(feature = "websocket")]
78     Websocket(secp256k1::PublicKey, IpAddr, Option<u16>),
79 }
```

LNP API URL schemes (LNPBP-39)

- Native (over **TCP/IP** and **TCP/Tor**)

lnp:// <node-id> @ <ip>|<onion> : <port>

- LNP over **Websockets**

lnp-ws:// <node-id> @ <ip>|<onion> : <port>

- LNP over **UDP** (UDP hole punching or low throughput/mesh networks)

lnp-udp:// <node-id> @ <ip> : <port>

- Inter-process and in-process communications (with **ZMQ**)

lnp-zmq: [<file-path>] ? api=<p2p|rpc|sub>

- LNP over **ZMQ** over TCP/IP or TCP/Tor

lnp-zmq:// <ip>|<onion> : <port>/ ? api=<p2p|rpc|sub>

Big picture

- LNP networking is a first step towards generalized Lightning network
- LNP API stack can fix problems of modern TCP/IP combined with DNS & SSL:
 - decentralized network ids (public keys instead of certificates)
 - self-issues names (again, public keys)
 - end-to-end encryption, always
- Combined with TCP/IP/Tor, LNP API can help in building Internet2 (and not Web3:) confidential & censorship-resistant
- We design LNP API code to make future way into POSIX (Linux/UNIX) kernels
- May be, one day, Bitcoin/LN/RGB nodes will be part of OS kernel/distribution as well

Let's work together!

- Rust implementation:

github.com/LNP-BP/rust-lnpbp/tree/master/src/lnp

- Sample usage:

github.com/LNP-BP/rgb-node/blob/master/src/contracts/fungible/runtime.rs

(LNP node and BP node will follow soon)

- API tools:

github.com/LNP-BP/lnp-api-tools

- Standards:

github.com/LNP-BP/LNPBPs

RGB integration

Universal architecture and components for
personal nodes, wallets, exchanges & payment providers

LNP/BP Standards Association

Prepared & supervised by **Dr Maxim Orlovsky, Pandora Core AG**

Created with support from **Bitfinex** and **Fulgur Ventures**

Current 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)

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

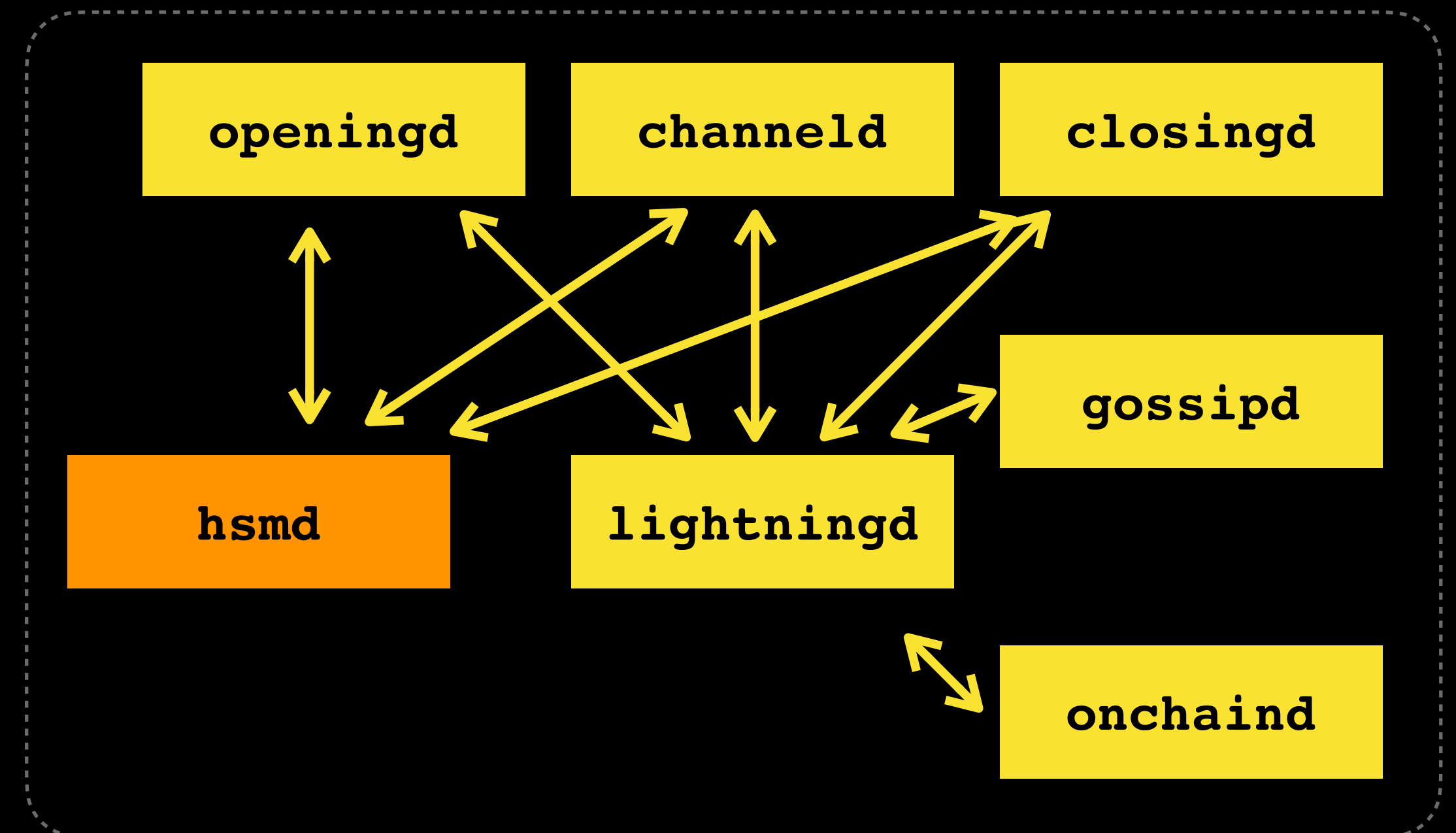
LNP node (Lightning node)

- Based on rust-lightning library by Matt Corallo @Square Crypto & @Chaincode Labs
- Utilising the same multi-thread non-blocking microservice code as Bitcoin transaction service
- Following best practices from c-Lightning architecture & extensibility
- Suited for generalised 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



```
docker container  
or volume
```

connected to
all channels and
peers

```
cli
other
clients
```

PNS

*push-notifications
used by mobile
non-custodials for
signing txs*

peer

peer

peer

DLC

RGB

d1cd

rgb

routing sqlite

per-channel state sqlite

gossip sqlite

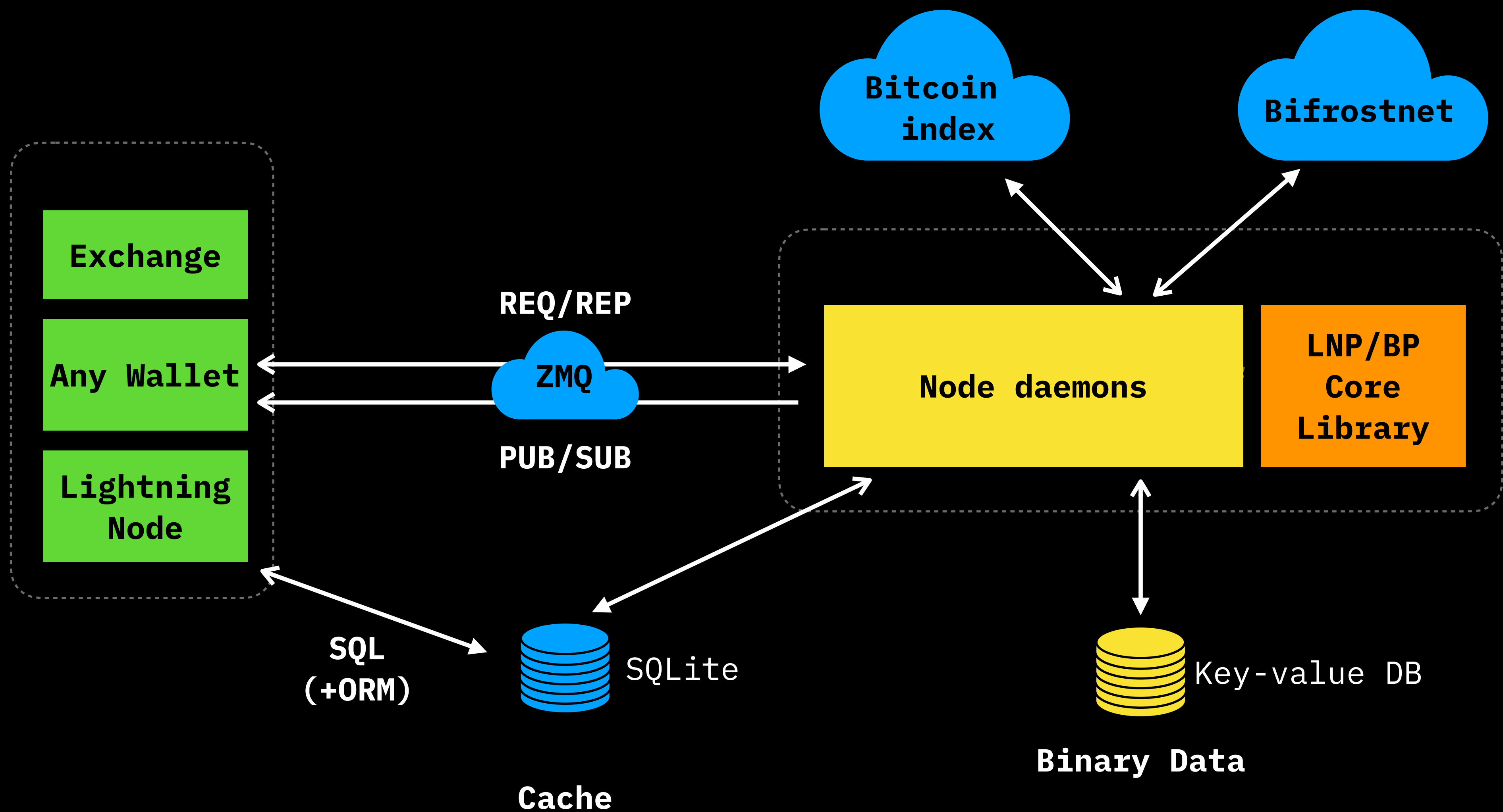
Variants for node integration

- **Daemon-based**: multi-process elastic configuration (c-lightning-like)
 - Can be dockerized and scaled independently (by module)
 - Can run as geo-distributed cluster
 - Can be used on enterprise server or personal server
- **Service-based**: multi-threaded runtime
 - Runs in the same process as client app
 - Best for mobile
- **Proxy-based**: web model
 - Service- or daemon-based backed
 - NodeJS proxy storing RGB data on server
 - JS client library (cache-less)
- **Direct**: may be implemented in the future; not recommended
 - WASM and C FFI bindings with language-specific wrappers
 - No ZMQ, no multithreading

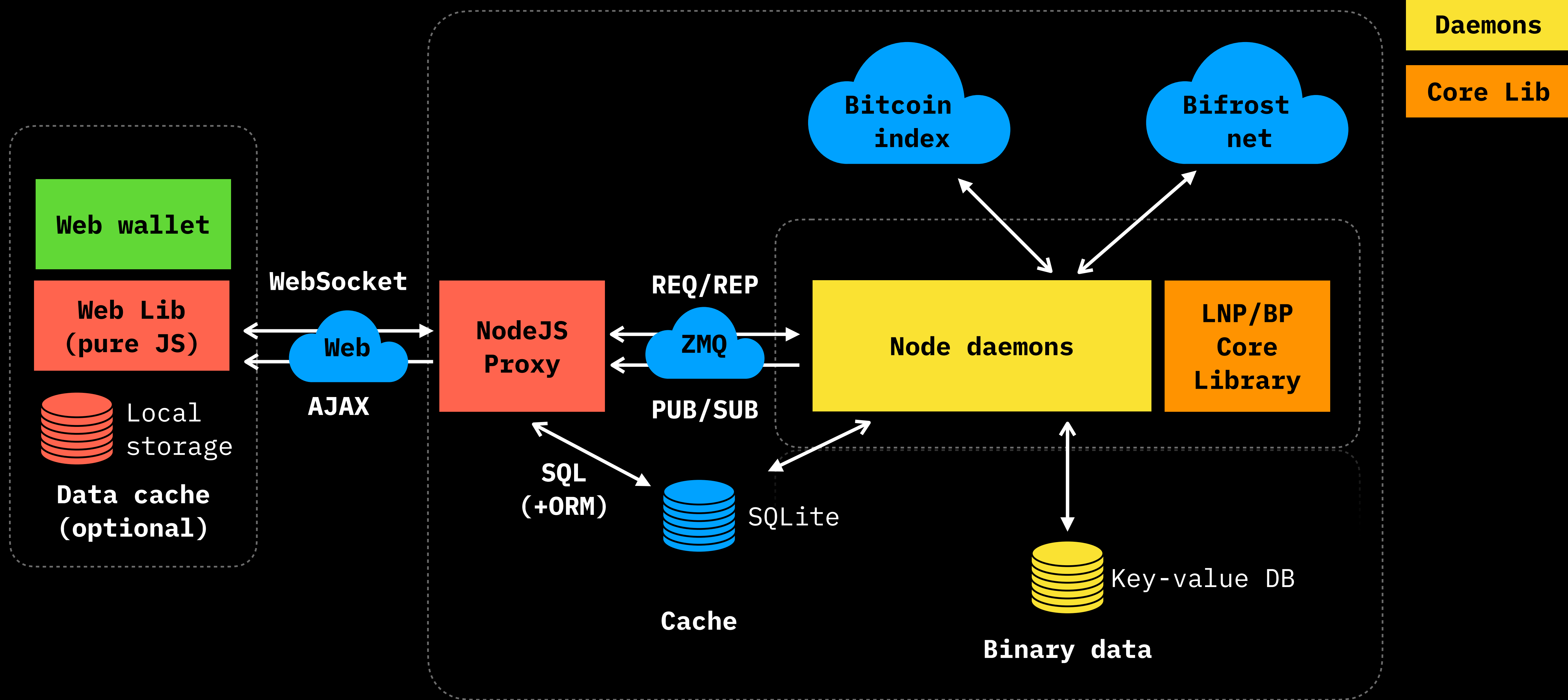
RGB Integration SDK

- **Binaries**
 - platform-specific runtime library (for service-based integration)
 - executables (for daemon-based integration)
 - in future: WASM & binary library for direct integration (not recommended)
- **Docker images** (can be used in daemon-based integration only)
- Language-specific integration for loading RGB runtime as service
 - Swift
 - Kotlin
- Class abstractions in JS, Swift & Kotlin for
 - ZMQ Client API
 - ORM data objects
- Web proxy service (NodeJS)
- Future: language-specific direct integration class libraries (not recommended)

Universal LNP/BP node architecture

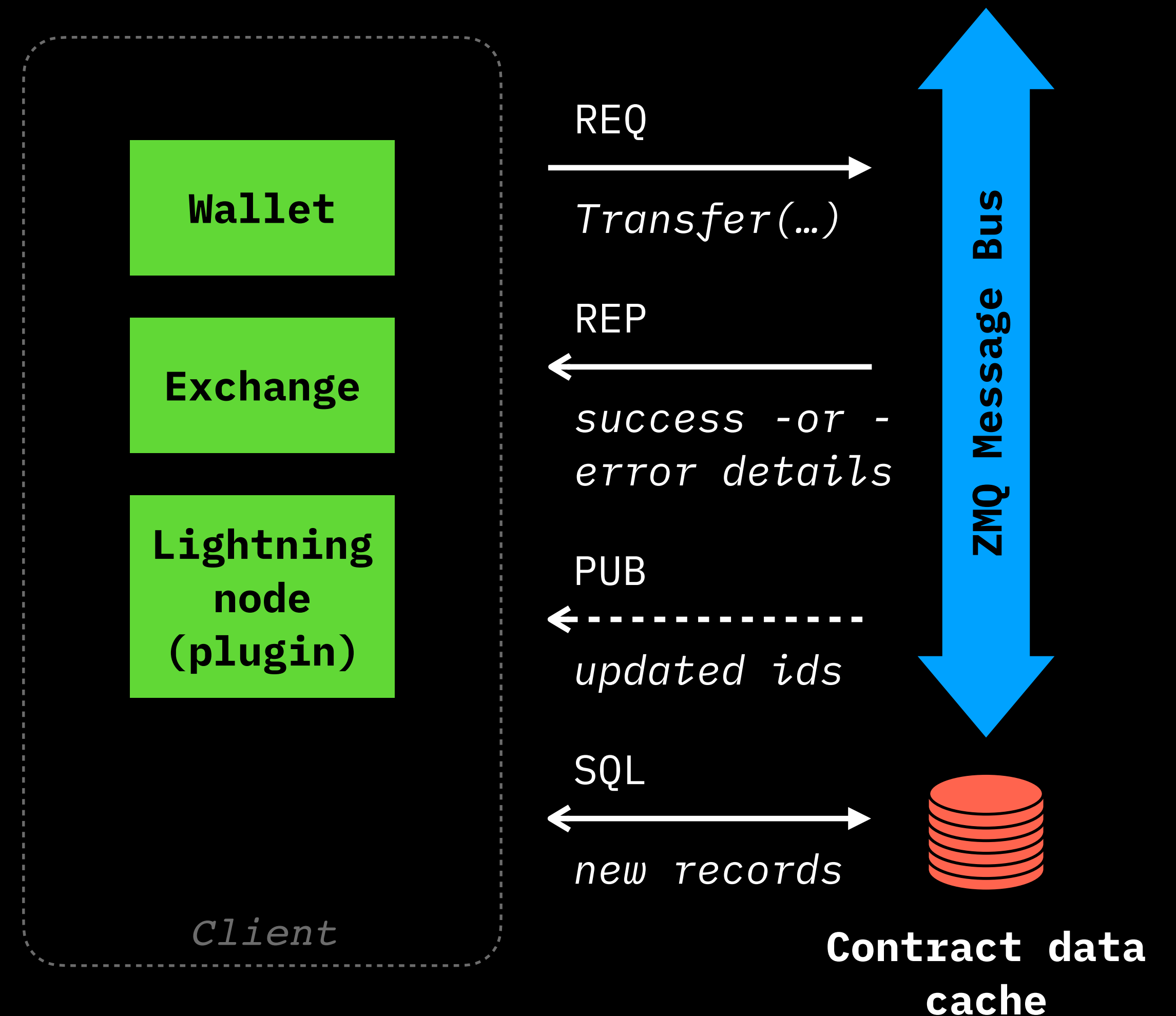


Universal LNP/BP node architecture: Web



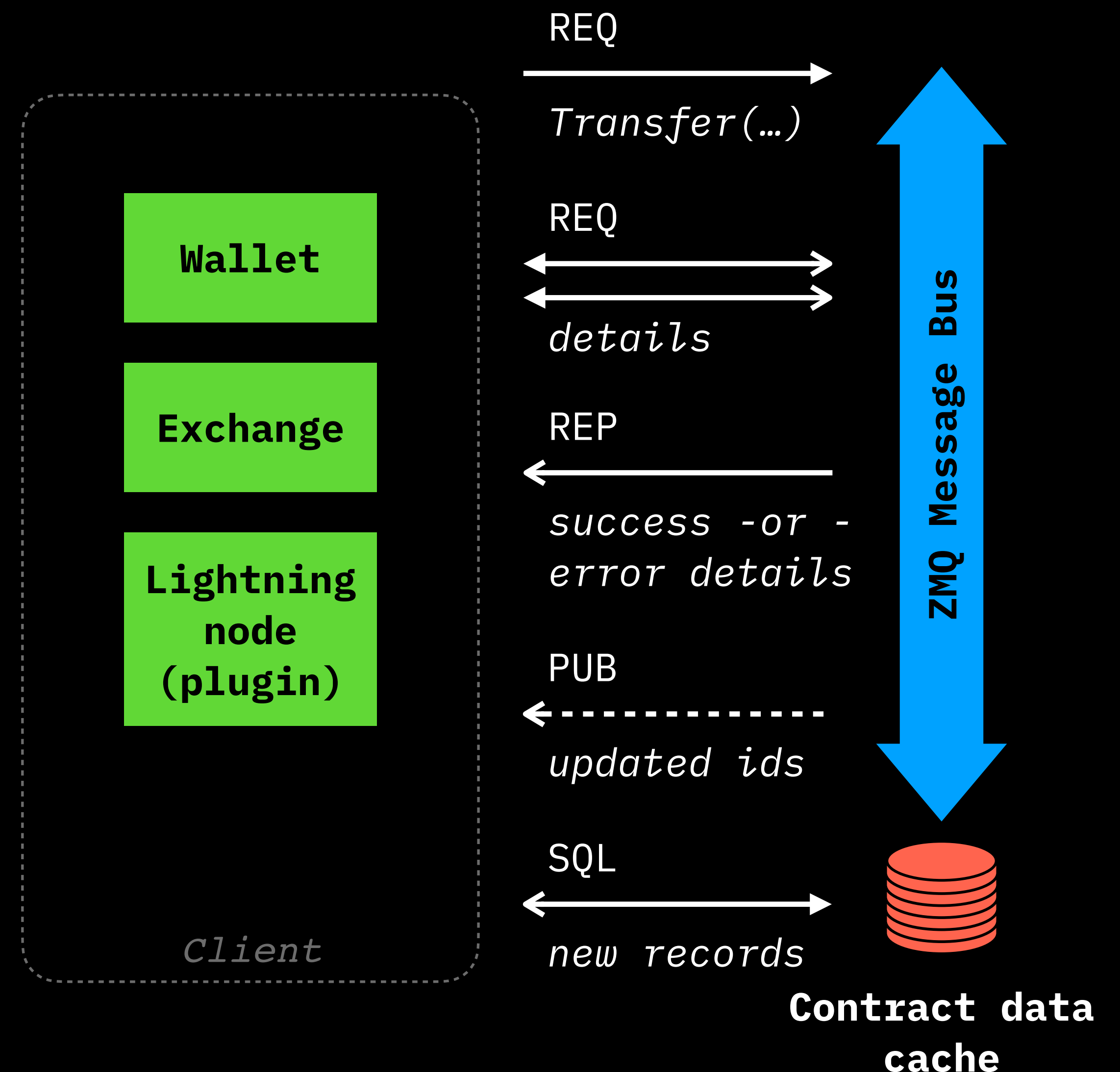
High-level RGB API integration

- Send simple request via ZMQ API to corresponding contract service (like fungible assets)
- Get response of success/failure code
- Wait for PUB information which entities were updated
- Read updated entities from SQL using your ORM



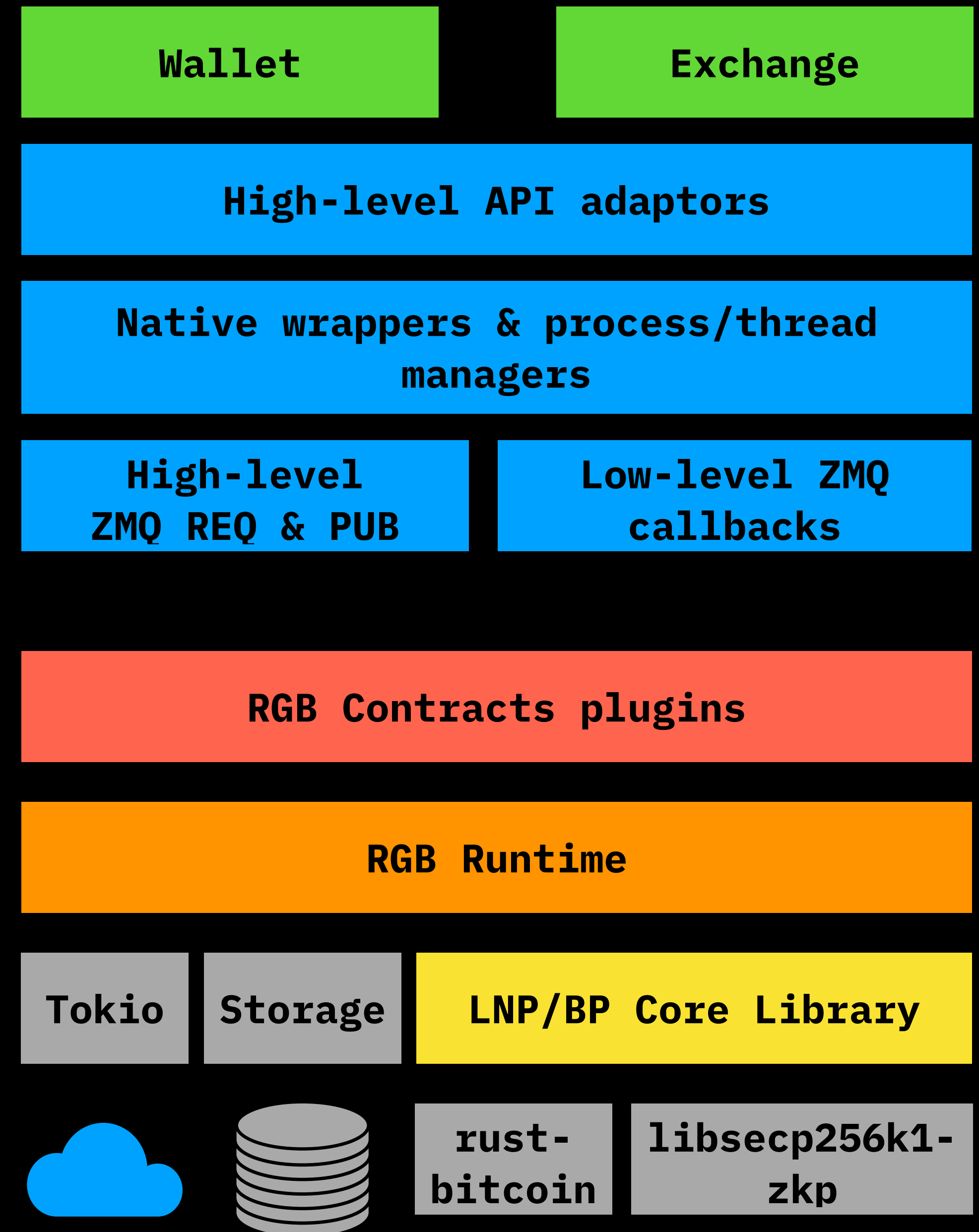
Low-level RGB API integration

- Open special per-request ZMQ REQ socket
- Send request via ZMQ API providing all required details via optional parameters; pass socket details
- Reply with necessary information on each incoming REQ ("callbacks") until you'll...
- ...get response of success/failure code
- Wait for PUB information which entities were updated
- Read updated entities from SQL using your ORM



Integration Stack

- Launch point:
`node(config).launch()`
- Wallet using high-level class wrapper
(sample for RGB node):
`RGB(context).pay(invoice)`
- Wallet using low-level class wrapper:
`rgb_pay_invoice(
 invoice,
 context,
 coordinator_callback,
 transaction_constructor_callback,
 coin_selection_callback)`



LNP/BP DEX

Integrating third-layer protocols to build DEX:
RGB, DLC, LNP

LNP/BP Standards Association

Prepared by **Dr Maxim Orlovsky, Pandora Core AG**

Created with support from **Bitfinex & Fulgur Ventures**

Leverage existing & don't reinvent the wheel

- Protocols:
 - **LNP**: networking protocol for end-to-end encrypted P2P and RPC APIs used by LN, RGB (potentially in future – DLC)
 - **RGB**: information exchange between LN nodes on their assets (based on LNP)
 - **DLC**: ongoing work on decentralized arbitrage network with oracles
 - **Dazaar**: real-time data exchange P2P protocol
- Software:
 - **LNP Node**: modular lightning node (under development) with support for RGB & LNP
 - **Bifrost**: data storage node (planned) used in RGB & Storm, which can be used for maintaining data on orderbooks

Bifrost node & infrastructure

- General decentralized storage server:
manages key-value indexes of encrypted, hashed or blinded blob data
- Used by RGB for:
 - accept payments confirmation
(RGB Consignments) when receiver's wallet is offline
 - hold backup for RGB Stash (in encrypted form)
- Other usage:
 - LN watchtowers
 - DEX orderbooks
 - Gossip message propagation (outside of native LN payments)
 - DLC oracle?
- Paid with Lightspeed & Storm (when its out)

DEX with RGB, DLCs & generalized LN

