Blockstream

Magical Minisketch
and the Lightning Gossip Network

Alex Myers - Blockstream

June 7, 2022 - Bitcoin++

Topics

Role of Gossip

What is Minisketch?

How minisketch can improve lightning gossip

Role of Gossip

Make a Lightning Payment

Open an App

Make a Lightning Payment

Open an App

Scan QR or copy BOLT 11 invoice

Make a Lightning Payment

Open an App

Scan QR or copy BOLT 11 invoice

Verify amount, press send

111698sats?

Make a Lightning Payment

Open an App

Scan QR or copy BOLT 11 invoice

Verify amount, press send

Success!

111698sats?

✓

Lightning Payment - Enhance

```
$ lightning-cli listpays <BOLT 11>
```

Lightning Payment - Enhance

```
$ lightning-cli listpays <BOLT 11>

      "bolt11": "LNBC1116…",
      "destination": "028…",
      "payment_hash": "af1721…",
      "status": "complete",
      "created_at": 1639436393,
      "preimage": "6a4b839…",
      "amount_msat": "111698000msat",
      "amount_sent_msat": "111941376msat",
      "number_of_parts": 12
```

Lightning Payment - Enhance!

```
$ lightning-cli listpaystatus <BOLT 11>
```

53 payment parts

MPP timeouts

failure replies (onion messages):0x1007…

```
"erring_index": 2,
"erring_node": "020…",
"erring_channel": "724…",
```

Lightning Payment - Enhance!

```
$ lightning-cli listpaystatus <BOLT 11>
```

53 payment parts

MPP timeouts

failure replies (onion messages):0x1007…

```
    "erring_index": 2,
    "erring_node": "020…",
    "erring_channel": "724…",
```

Lightning Payment - Enhance!!

BOLT4:

“The top byte of failure_code can be read as a set of flags:

- 0x8000 (BADONION): unparsable onion encrypted by sending peer
- 0x4000 (PERM): permanent failure (otherwise transient)
- 0x2000 (NODE): node failure (otherwise channel)
- 0x1000 (UPDATE): new channel update enclosed

BOLT4:

“type: UPDATE|7 (temporary_channel_failure)

1. data:
    - [u16:len]
    - [len*byte:channel_update]

The channel from the processing node was unable to handle this HTLC, but may be able to handle it, or others, later.

Lightning Payment - Enhance!!!

```
$ devtools/decodemsg
```

```
WIRE_CHANNEL_UPDATE:
signature=304…
chain_hash=000000000019d6689…
short_channel_id=724…
timestamp=1639436394
message_flags=1
channel_flags=1
cltv_expiry_delta=34
htlc_minimum_msat=1msat
fee_base_msat=0
fee_proportional_millionths=1000
(option_channel_htlc_max):htlc_maximum_msat=495000000msat
```

Lightning Payment - Enhance!!!

```
$ devtools/decodemsg
```

```
WIRE_CHANNEL_UPDATE:
signature=304…
chain_hash=000000000019d6689…
short_channel_id=724…
timestamp=1639436394
message_flags=1
channel_flags=1  ←———  This held us up!
cltv_expiry_delta=34
htlc_minimum_msat=1msat
fee_base_msat=0
fee_proportional_millionths=1000
(option_channel_htlc_max):htlc_maximum_msat=495000000msat
```

Lightning Payment - Takeaway

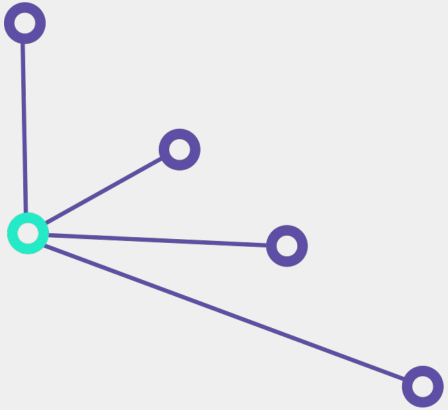
70+ lightning channels utilized

Outdated channel info

41 payment failures (due in part to outdated gossip/graph)

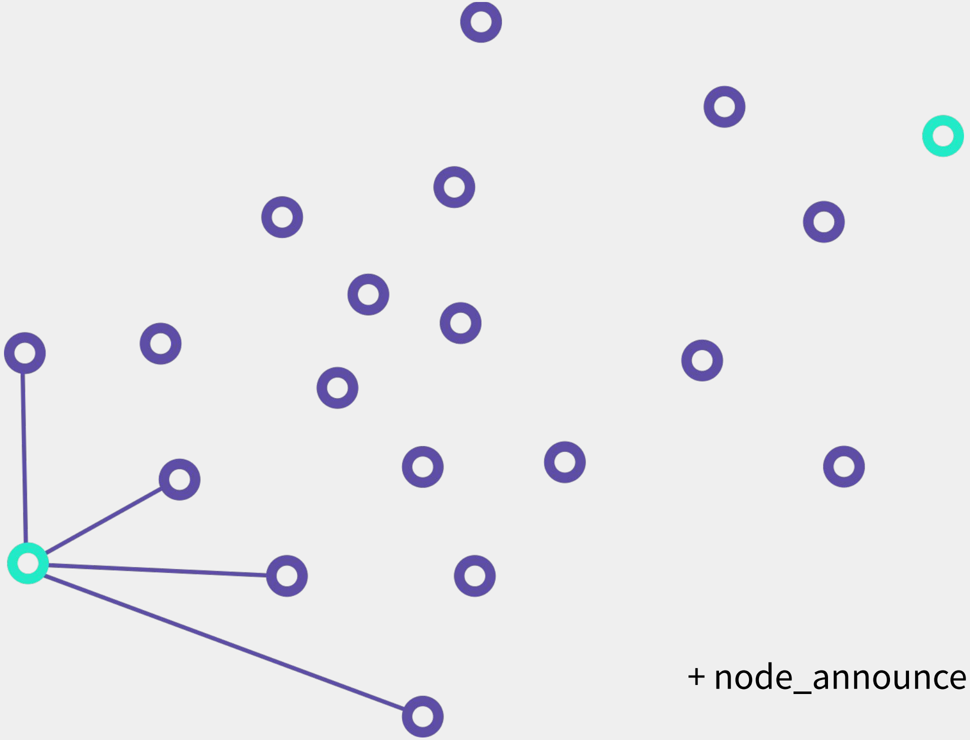
Node was able to construct payment routes to complete the payment

Role of Gossip



Role of Gossip

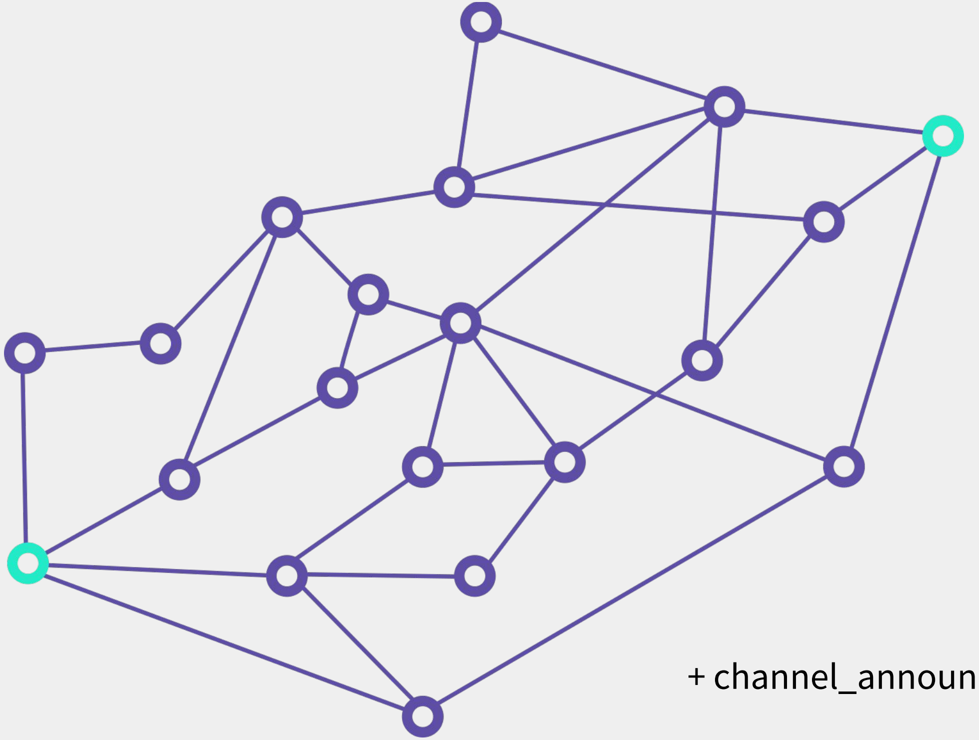
Construct the graph



+ node_announcement

Role of Gossip

Construct the graph

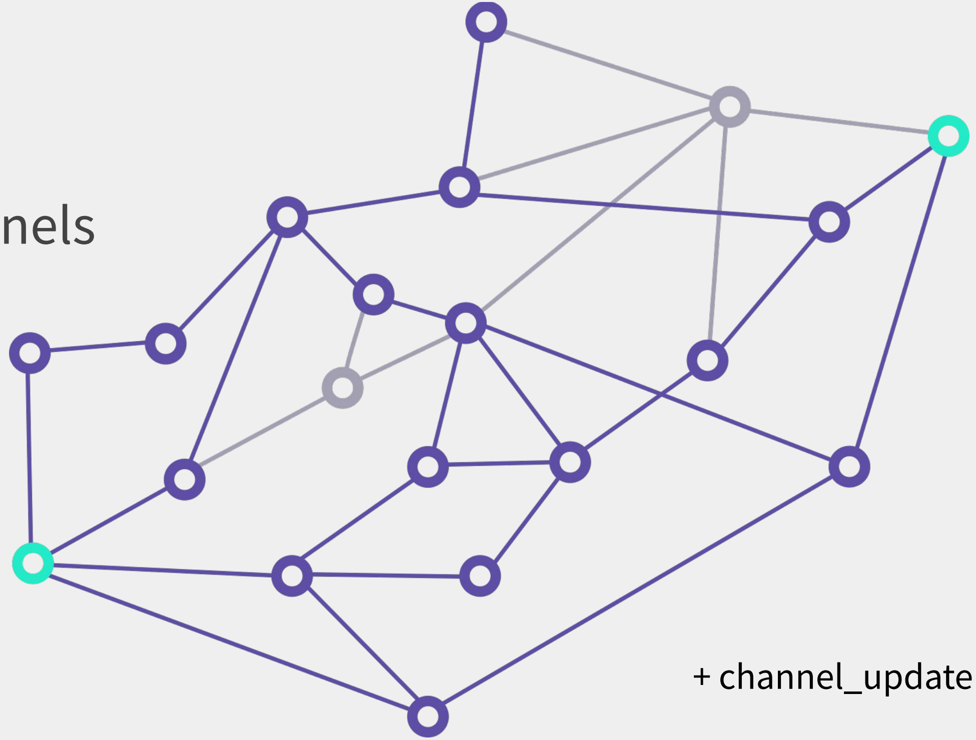


+ channel_announcement

Role of Gossip

Construct the graph

Update nodes / channels



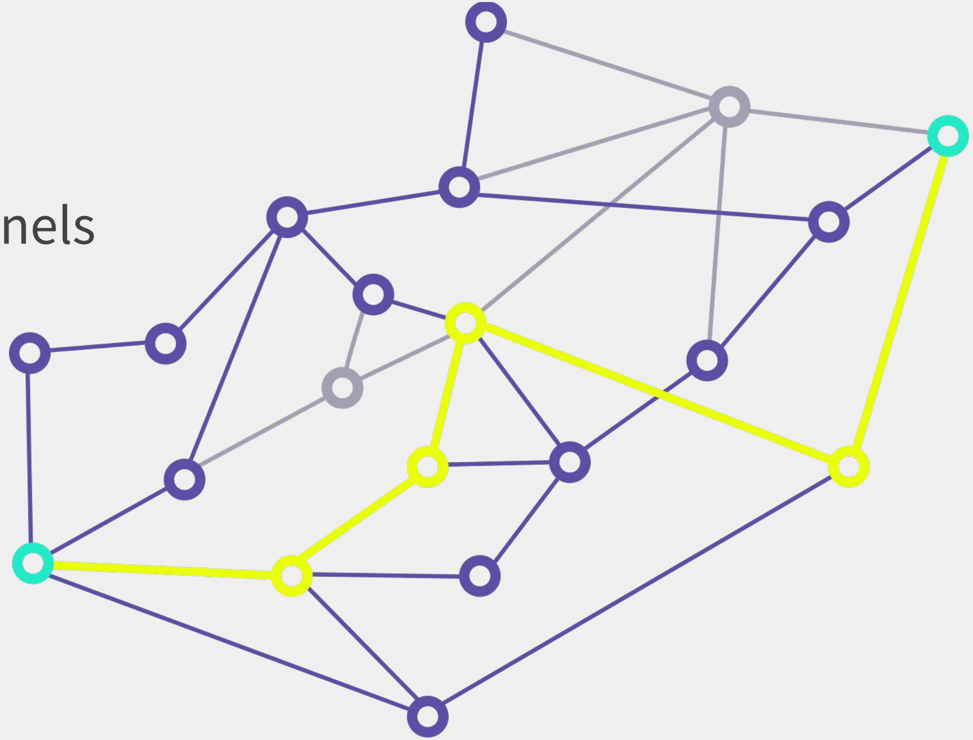
+ channel_update

Role of Gossip

Construct the graph

Update nodes / channels

Construct routes



Gossip Network

Connected node != gossiper

3-5 active gossip peers

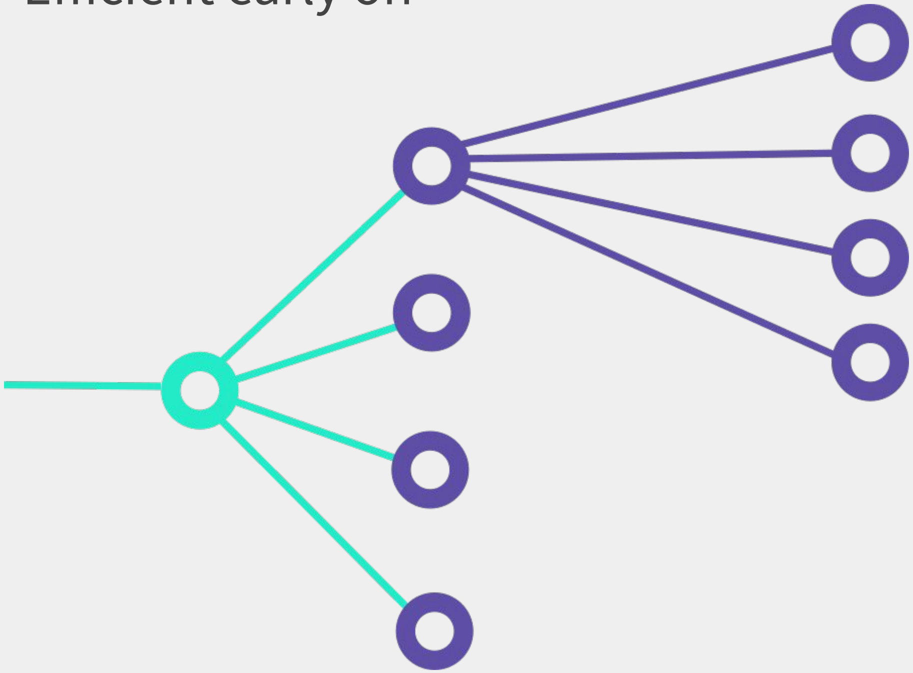
Gossip peer does not require a channel

Gossip peer may not even be a node

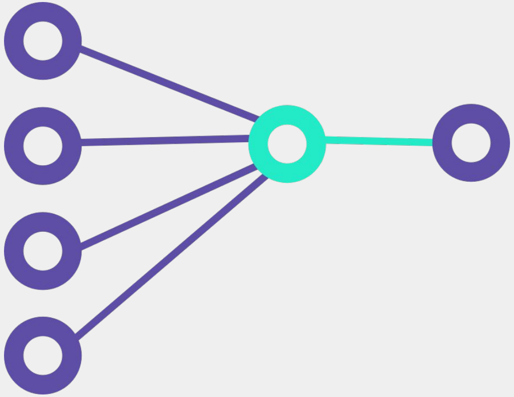
Relayed by flood propagation

Flood Propagation

Efficient early on



Redundant later



Gossip Statistics

Overall gossip bandwidth consumption: ~2.5x ideal

~85,000 channels

~17,500 nodes

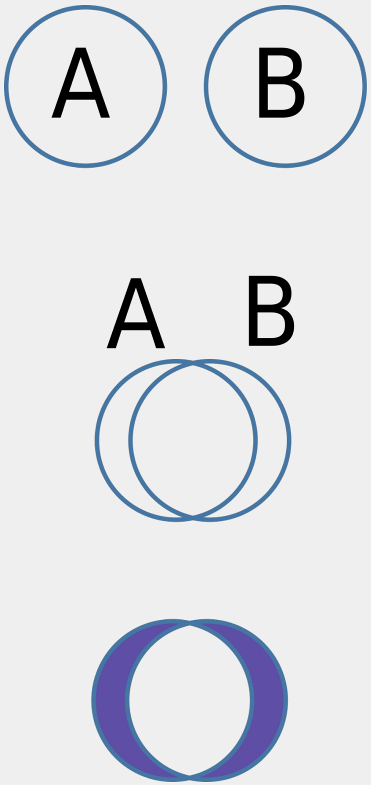
3 gossip peers: minimum 14 hops to fully traverse the network

Gossip is batched prior to broadcast: 60s-90s cycle

95% of peers are reached within 13 minutes

What is Minisketch?

Set Reconciliation



Background

Error correction codes: Hammond, Bell Labs 1950

BCH error correction: 1959/1960

Berlekamp-Massey Algorithm, 1969

PinSketch: Dodis, Ostrovsky, Reyzin, Smith 2004

BCH Example

Sets

[1, 2, 3]   and   [1, 2, 3, 4]

Sum the elements

[1+2+3]   [1+2+3+4]

[6]   [10]

Difference = 10 - 6 = 4

capacity = 1

BCH Example

2 differences? Sum sets, then sum the squares.

[1, 2, 3, 4] and [2, 3]

$$\begin{pmatrix} 1 + 2 + 3 + 4 \\ 1 + 4 + 9 + 16 \end{pmatrix} \qquad \begin{pmatrix} 2 + 3 \\ 4 + 9 \end{pmatrix}$$

$$\begin{pmatrix} 10 \\ 30 \end{pmatrix} \qquad \begin{pmatrix} 5 \\ 13 \end{pmatrix}$$

BCH Example

2 differences?

[1, 2, 3, 4] and [2, 3]

$$\begin{pmatrix} 10 \\ 30 \end{pmatrix} - \begin{pmatrix} 5 \\ 13 \end{pmatrix} = \begin{pmatrix} 5 \\ 17 \end{pmatrix} = \begin{pmatrix} a_1 \\ a_1{}^2 \end{pmatrix} + \begin{pmatrix} a_2 \\ a_2{}^2 \end{pmatrix}$$

Thanks to Gleb Naumenko for this explanation:
https://www.youtube.com/watch?v=ZUWs00Anpaw

Constructing a large sketch

$$\begin{pmatrix} a_1 + a_2 + \ldots + a_n \\ a_1^2 + a_2^2 + \ldots a_n^2 \\ \ldots \\ a_1^n + a_2^n + \ldots a_n^n \end{pmatrix}$$

Minisketch

C++ library developed by Pieter Wuille to implement PinSketch algorithm

Compiles on a range of hardware and architectures (Armv7l, x86_64)

Pure python implementation

https://github.com/sipa/minisketch

Using Minisketch

Alice

Initialize Sketch (bits, capacity)

Add data, calculate syndromes

Serialize and transmit

Bob

Build sketch

Merge with Alice’s

Calculate Polynomial

Extract roots - result is difference between the two data sets

Black Box Properties

2 - 64 bit wide data supported

Serialized size == sketch capacity * data width

Reconciliation time scales linearly with sketch capacity

Reconciliation time scales quadratically with set differences

Can merge sketches with different capacities

Black Box Properties

2 - 64 bit wide data supported

Serialized size == sketch capacity * data width

Reconciliation time scales linearly with sketch capacity

Reconciliation time scales quadratically with set differences

Can merge sketches with different capacities

Limitations

Sketch elements must not be 0

Helps to verify difference in set sizes < sketch capacity

How do we use this?

Erlay

Transaction relay protocol for bitcoin

Uses Minisketch set reconciliation

32 byte TXID -> 64 bit transaction fingerprint

- Hashed with shared secret on a per-peer basis

Reconcile inventory sets

LN Gossip vs bitcoin tx relay

Short Channel ID already creates unique fingerprint

- No collision grinding concern
- No timing analysis concern - this is all public information

Three different message types

Application to Gossip

Gossip Messages

- channel_update: ~140bytes
- node_announcement: 150+ bytes
- channel_announcement: ~430 bytes

Application to Gossip

Gossip Messages

- channel_update: ~140bytes
- node_announcement: 150+ bytes
- channel_announcement: ~430 bytes

Valid for 2 weeks

Challenge

Uniquely identify a gossip message in only 8 bytes

Channels can be identified by SCID (8 bytes)

Node_id is a 32 byte string

Encoding Scheme

Offset	Bits	Data
0	2	Message type (chan announce / chan update / node announce)
2	1	Direction
3	24	Block Height
27	15	Transaction Index
42	10	Output Index
52	12	Timestamp

Encoding Scheme

Offset	Bits	Data
0	2	Message type (chan announce / chan update / node announce)
2	1	Direction
3	24	Block Height
27	15	Transaction Index
42	10	Output Index
52	12	Timestamp

Short Channel ID (SCID)

Set Reconciliation benefits

- Bandwidth of gossip is decreased by 60%+
- Possible to gossip with more peers
- More gossip peers -> more reliable propagation
  - Node_announcements will benefit

What’s next?

- Global sketch vs. per-peer sketches
  - Tighter consensus vs. more robust
- Common rate-limit improves efficiency
  - Using block_height makes this easier
- Gossip may drop SCID in the future

Conclusion

Gossip lets us construct payment routes across the lightning network.

Minisketch is an incredibly efficient tool for data propagation.

We can improve LN gossip function by encoding and transmitting gossip sketches.

Questions?

Lightning Dev mailing list

@endothermicdev

alex@endothermic.dev

References

1. Lightning Spec: https://github.com/lightning/bolts
2. Naumenko, G., Maxwell, G., Wuille, P., Fedorova, A., Beschastnikh, I. (2019). “Erlay: Efficient Transaction Relay for Bitcoin” https://people.ece.ubc.ca/sasha/papers/ccs19.pdf
3. Wuille, Pieter (2018). *Minisketch: A Library for BCH-Based Set Reconciliation* https://github.com/sipa/minisketch
4. Russel, Rusty (2018). “Minisketch and lightning gossip” https://lists.linuxfoundation.org/pipermail/lightning-dev/2018-December/001741.html
5. Gögge, N., Rohrer, E., and Tschorsch, F. (2022). *On the Routing Convergence Delay in the Lightning Network* https://arxiv.org/abs/2205.12737
6. London Bitcoin Devs. (2019, Nov 23) *Gleb Naumenko - Current State of P2P Research in Bitcoin/Erlay* YouTube https://www.youtube.com/watch?v=ZUWs00Anpaw