Bootstrapping and Maintaining a Lightning Node

Elaine Ou
Chaincode Lightning Residency
October 22 2018

Bootstrapping and Maintaining a Lightning Node

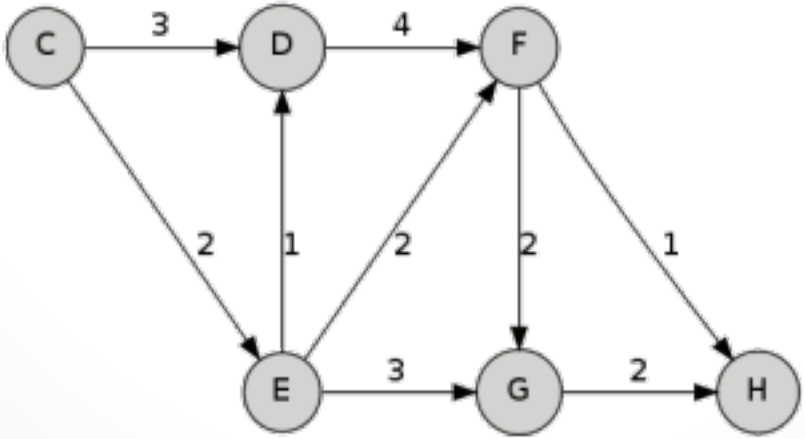
- How Routing Works
- Finding Peers
- Getting Incoming Capacity
- Maintaining that Capacity
- Future Work

Route Formation

(In Theory)

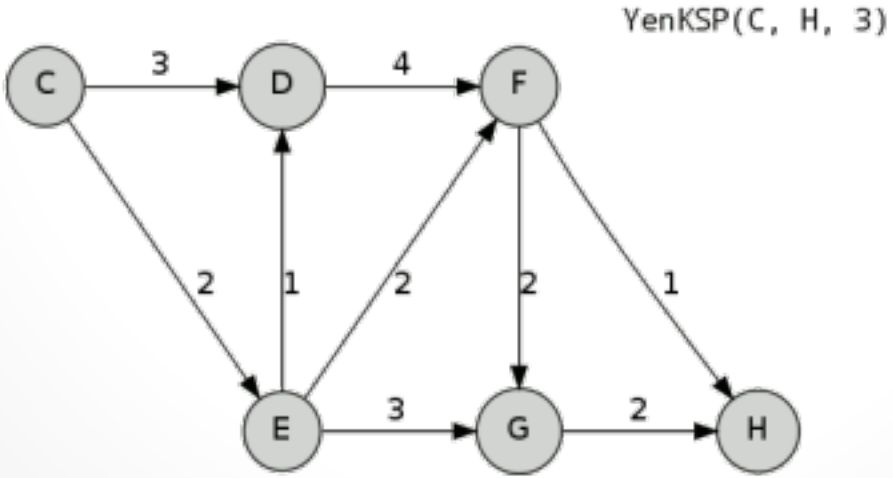
Calculating a Route

- Nodes have a local network view
- Source-based, varies by implementation
  - Find k-shortest paths that has sufficient channel capacity for payment, weighted by fees plus time lock penalty



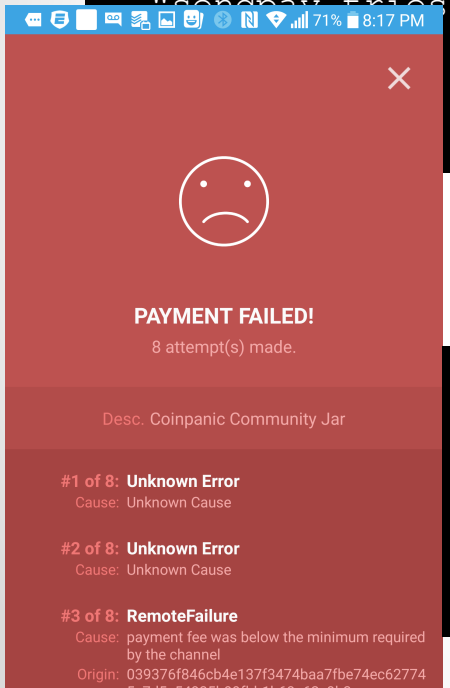
Calculating a Route

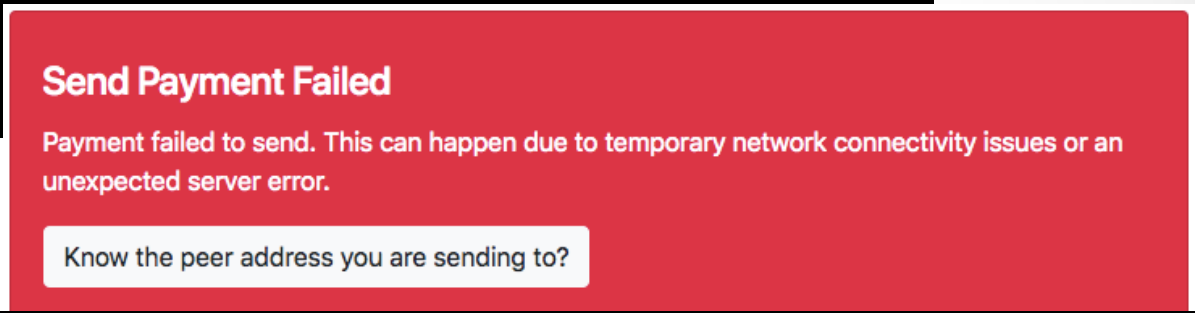
- Try paths until one works!
  - C->E->F->H
  - C->D->F->H
  - C->E->G->H



Routing (in practice)

```
{ "code" : 205, "message" : "Could not find a route", "data" :
{
  "getroute_tries": 1,
  "sendpay_tries": 0,
```






```
Confirm payment (yes/no): yes
{
     "payment_error": "unable to find a path to destination",
     "payment_preimage": "",
     "payment_route": null
}
```

Making Friends...

Node discovery and channel formation

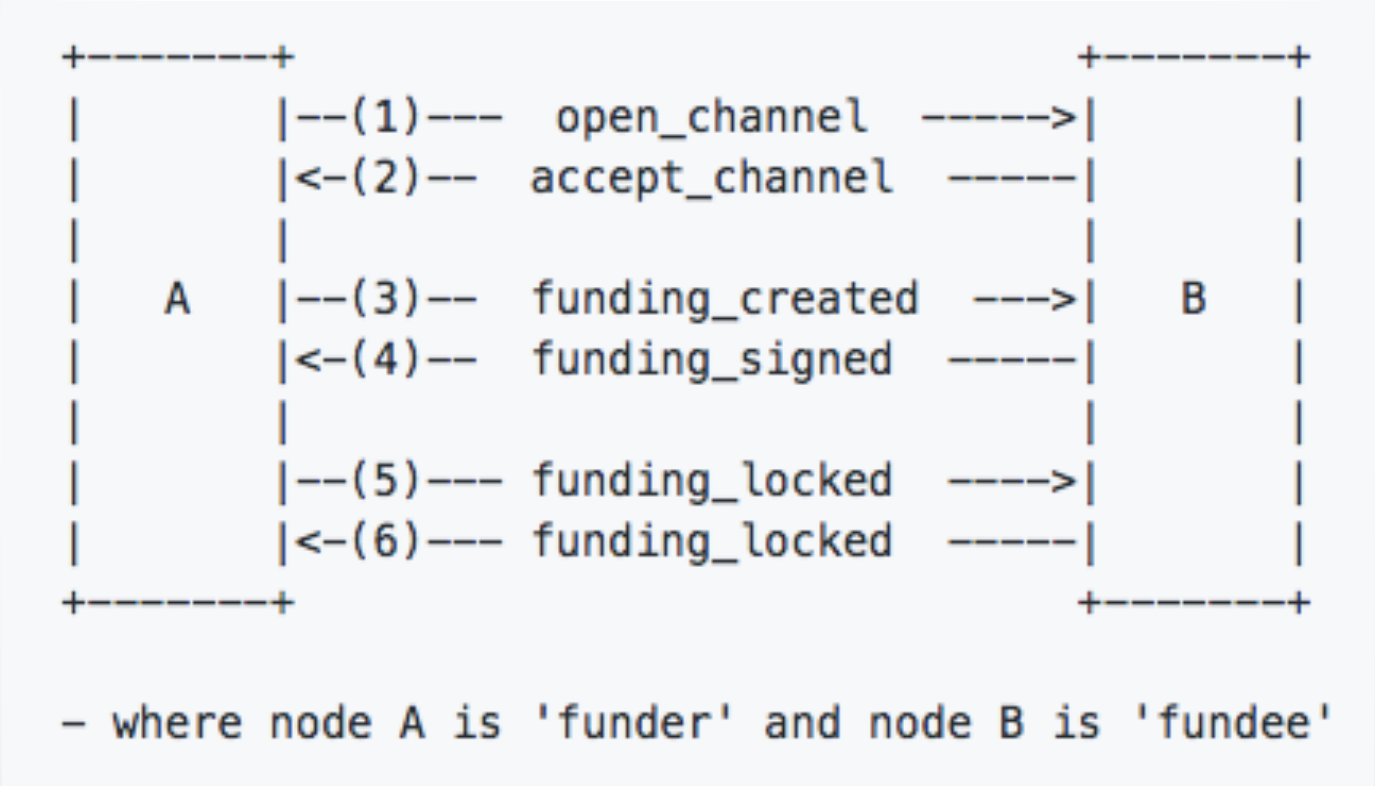
BOLT#10: DNS Bootstrap and Assisted Node Location

- It's hard to be discovered
  - Nodes need peers to get started
- DNS seed server indexes nodes and return a random set for bootstrapping
  - %s.lseed.bitcoinstats.com
  - https://github.com/cdecker/lseed
- Can search for nodes by node_id if address has changed

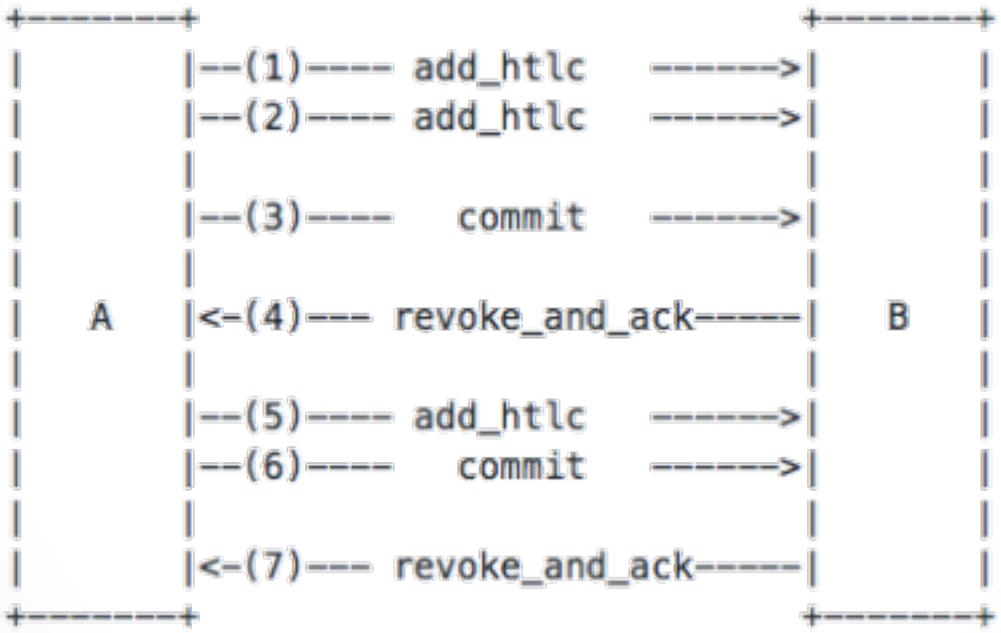
BOLT#2: Peer Protocol for Channel Management

- open_channel message
  - Funding amount, push amount, max_htlc_value_in_flight, htlc_minimum, max_accepted_htlcs, channel reserve, feerate, to_self_delay, dust_limit_satoshis, announce_channel
  - Revocation basepoint

BOLT#2: Peer Protocol for Channel Management



BOLT#2: Peer Protocol for Channel Management

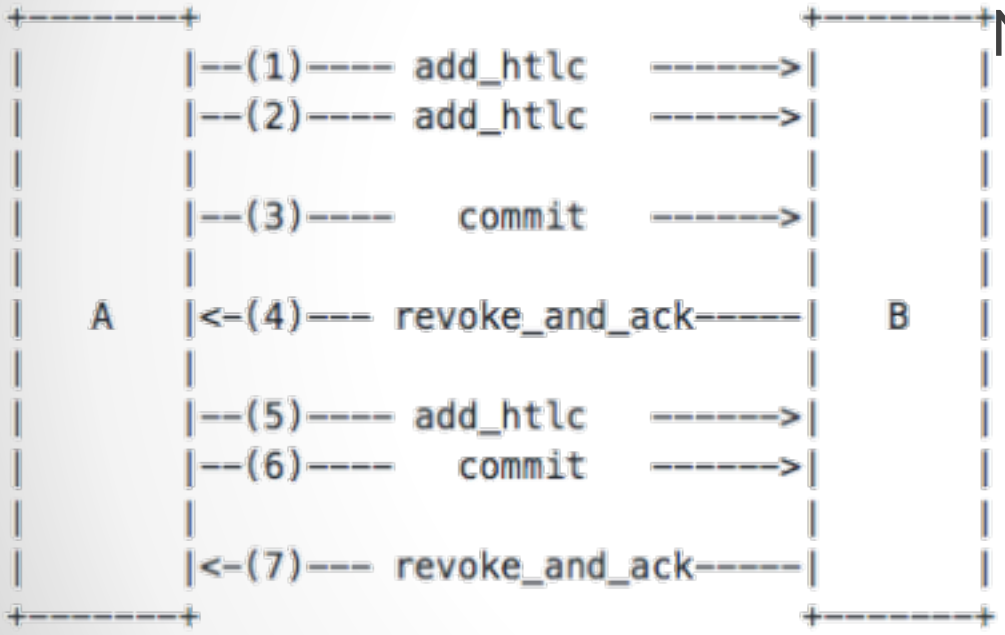


BOLT#2: Peer Protocol for Channel Management

- Commitment Transaction
  - Revocation_basepoint + commitment_point = revocation_pubkey

```
OP_IF
  <revocation_key>
OP_ELSE
  `to_self_delay`
  OP_CSV
  OP_DROP
  <local_delayedkey>
OP_ENDIF OP_CHECKSIG
```

BOLT#2: Peer Protocol for Channel Management

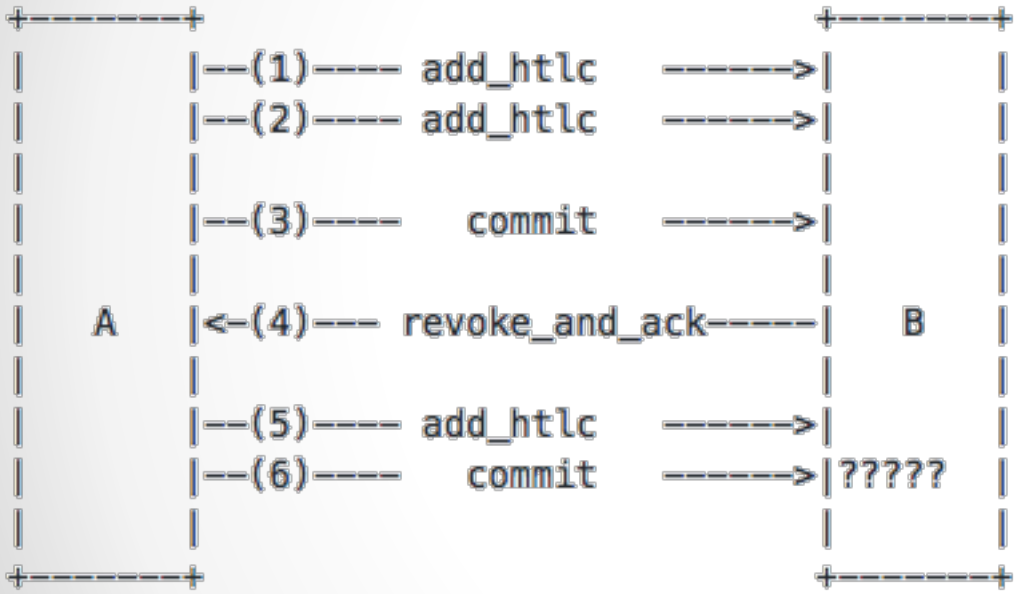


New Commitment

- Previous commitment_secret + revocation_basepoint_secret = revocation_privkey

```
OP_IF
  <revocation_key>
OP_ELSE
  `to_self_delay`
  OP_CSV
  OP_DROP
  <local_delayedkey>
OP_ENDIF OP_CHECKSIG
```

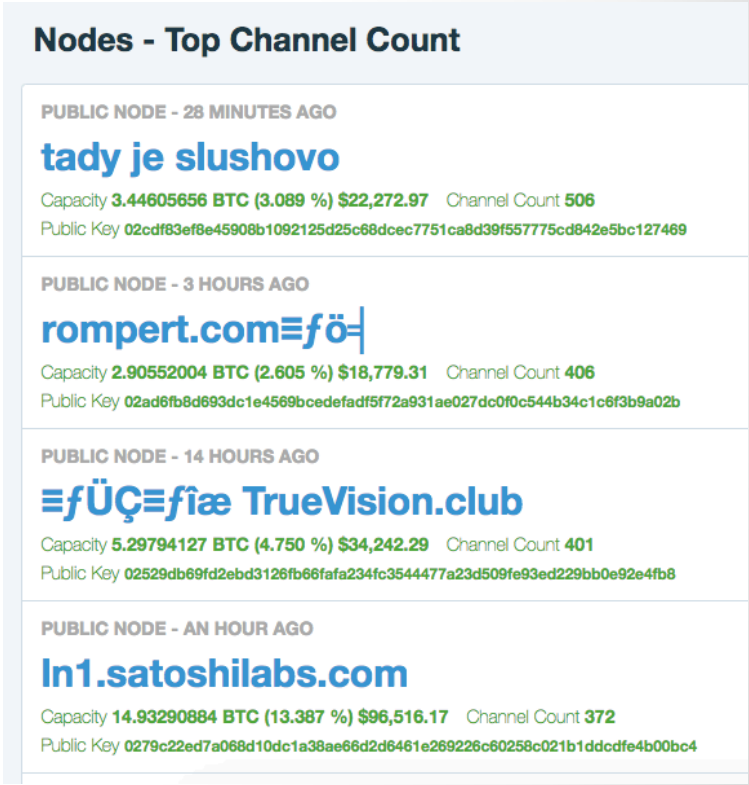
BOLT#2: Peer Protocol for Channel Management




```
OP_IF
  <revocation_key>
OP_ELSE
  `to_self_delay`
  OP_CSV
  OP_DROP
  <local_delayedkey>
OP_ENDIF  OP_CHECKSIG
```

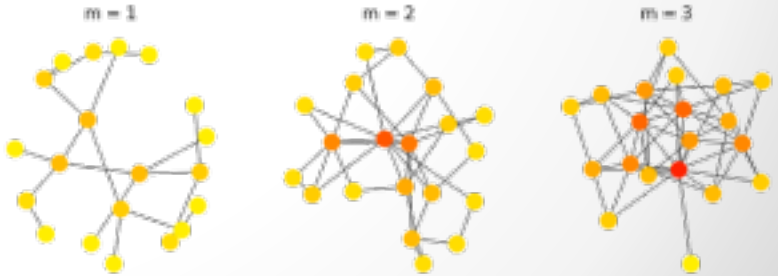
Finding Good Peers

- Reliability
- Connect to nodes that maximize connectivity to the network
  - High uptime, low fees, good capacity (balanced channels), strongly connected
  - https://1ml.com/
- Funds are tied up, so choose wisely



Autopilot (LND)

- Automatically open channels to routing nodes
  - Set max channels, fraction of funds to commit
- Connect to nodes randomly, with probability of connection proportional to number of existing channels
  - Power law distribution
- Goal: Global scale-free network
  - Many hubs



Additional Considerations

- Channel capacity vs. channel balance
- Reliability/Uptime
- Path diversity and redundancy (if one node disappears, will there be a connectivity problem?)
- Fewest hops to favorite destinations
- All these factors optimize for outgoing transactions

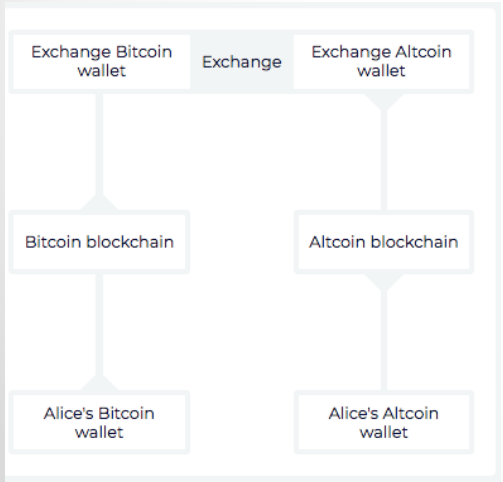
Incoming Capacity

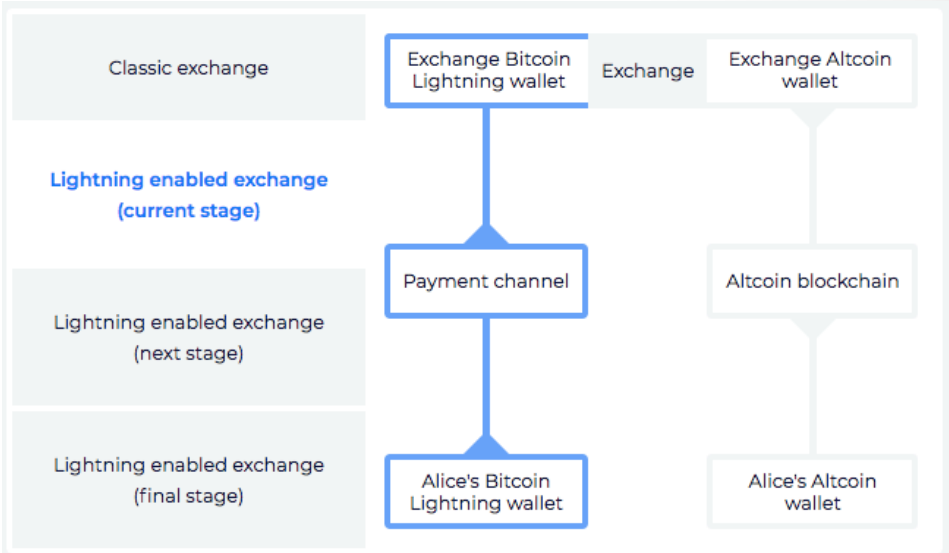
Why would someone open a channel to you?

- Responsibilities
  - A funding transaction forms a smart contract (commitments)
  - An unresponsive node causes routing failures, payment delays, lost transaction fees, or money to be unavailable until the timeout period
- Be a good routing node (chicken-and-egg problem)
- Build a store that people like
- Pay for the channel yourself
  - Spend money (create channel and push some of it)
  - Exchange on-chain capacity for lightning payments
  - Submarine Swaps

Exchange On-chain Bitcoin for Lightning

- Buy Bitcoin on exchange, exchange opens a channel to your node, pushes funds to you (custodial swap)
  - Now you're connected to a hub
  - Eg: Zigzag.io





Non-custodial swaps

- Preimage obtained from paid invoice to claim on-chain funds
  - Generate a hash of the preimage for swap
  - Create on-chain commitment transaction containing hash
- Submarineswaps.org
  - https://github.com/submarineswaps/swaps-service
- Sparkswap.com

Maintaining Channel Liquidity

How do we keep things running?

Why Payments Fail

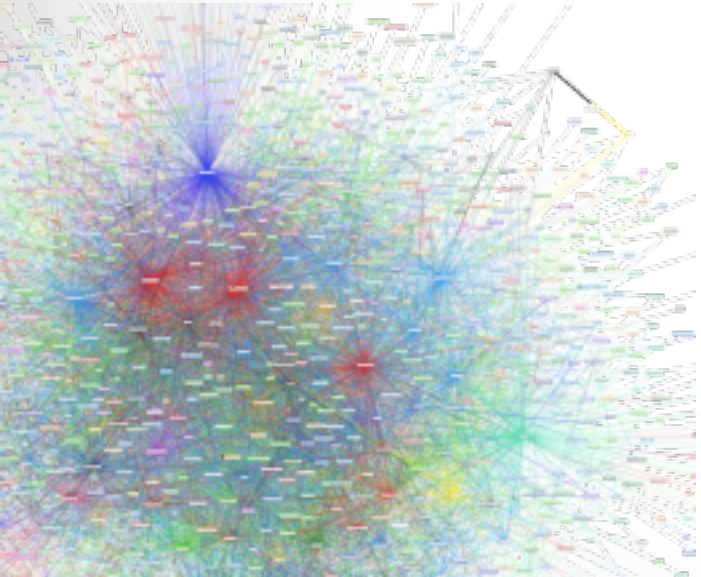
- Can’t find a route
- The payment is too large #reckless
  - Some transactions are better on-chain
  - Larger transactions shorten channel lifetime, hence higher fees
- Channels have sufficient capacity, but insufficient balance
  - Only the two channel endpoints know the balance
- Zombies
  - Nodes made public channels, then went offline
- Sender has limited information

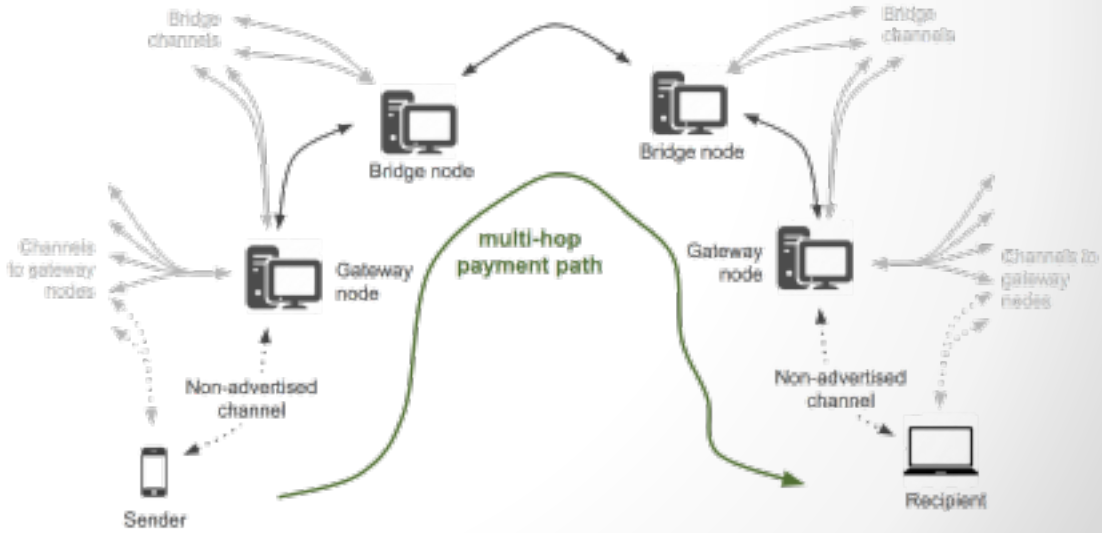
BOLT#7: P2P Node and Channel Discovery

- Nodes share gossip to disseminate information about nodes and channels
- Gossip messages
  - channel_announcement: new public channel!
    - Confirmed funding transaction, public keys of nodes
  - node_announcement: node data (addresses, pubkey, color, alias)
    - Node must be associated with known channel
  - channel_update: fees, minimum expiries for HTLCs
    - Each channel_announcement should have two channel_updates
- Gossip can be queried or rebroadcast on reconnection

Let Nodes Do Their Jobs

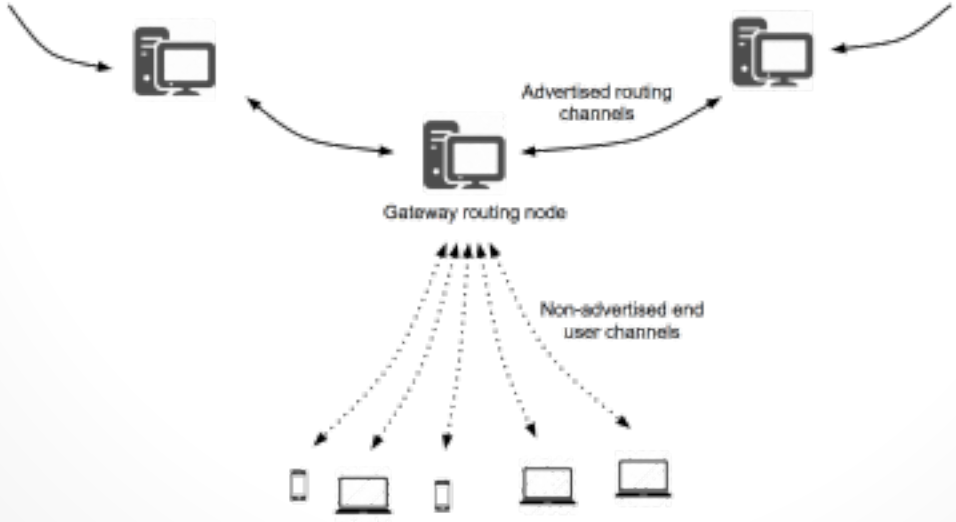
- You don’t need to have public channels
  - Many Lightning users will be on mobile devices, reliability should not be expected





Routing Hints

- Invoices support a tagged field with extra routing information for private routes
  - Payer not required to use hints, but add it to pathfinding process



Be a Good Node

- Balance Channels if public
  - Negative-fee channels can encourage balance
  - Keep an eye on node and channel activity
- Shutdown gracefully
- Punish bad nodes
  - Disable inactive channels
  - Close offline channels

WIP

...

Things to look forward to

Atomic Multi-Path Transactions

- High capacity channels are expensive and scarce
- When sending a payment over multiple channels, you'll be restricted by the size of your the smallest channel
  - Split a payment into multiple partial payments
  - The receiver is able to claim the total amount only when all partial payments have been received
- More path options, better privacy

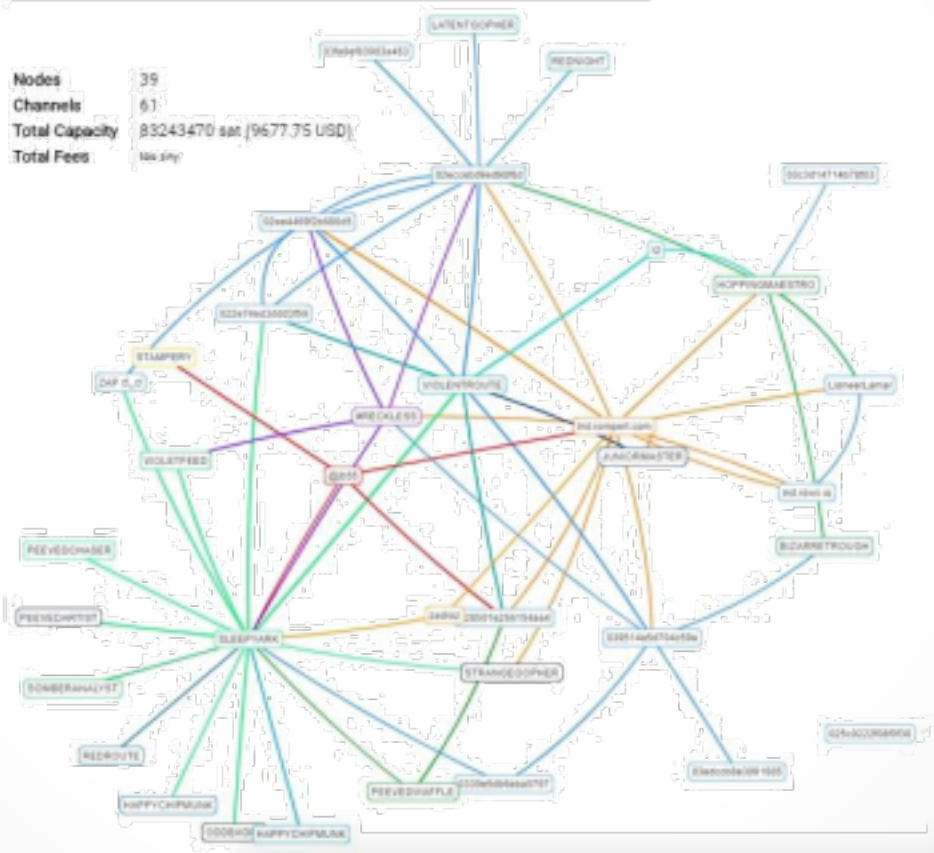
Lots More…

- Splicing
  - Resize open channels
- Channel Factories
  - Payment channels can be used to create more payment channels
  - Funds are locked into a shared wallet between a group of nodes instead of a specific channel.

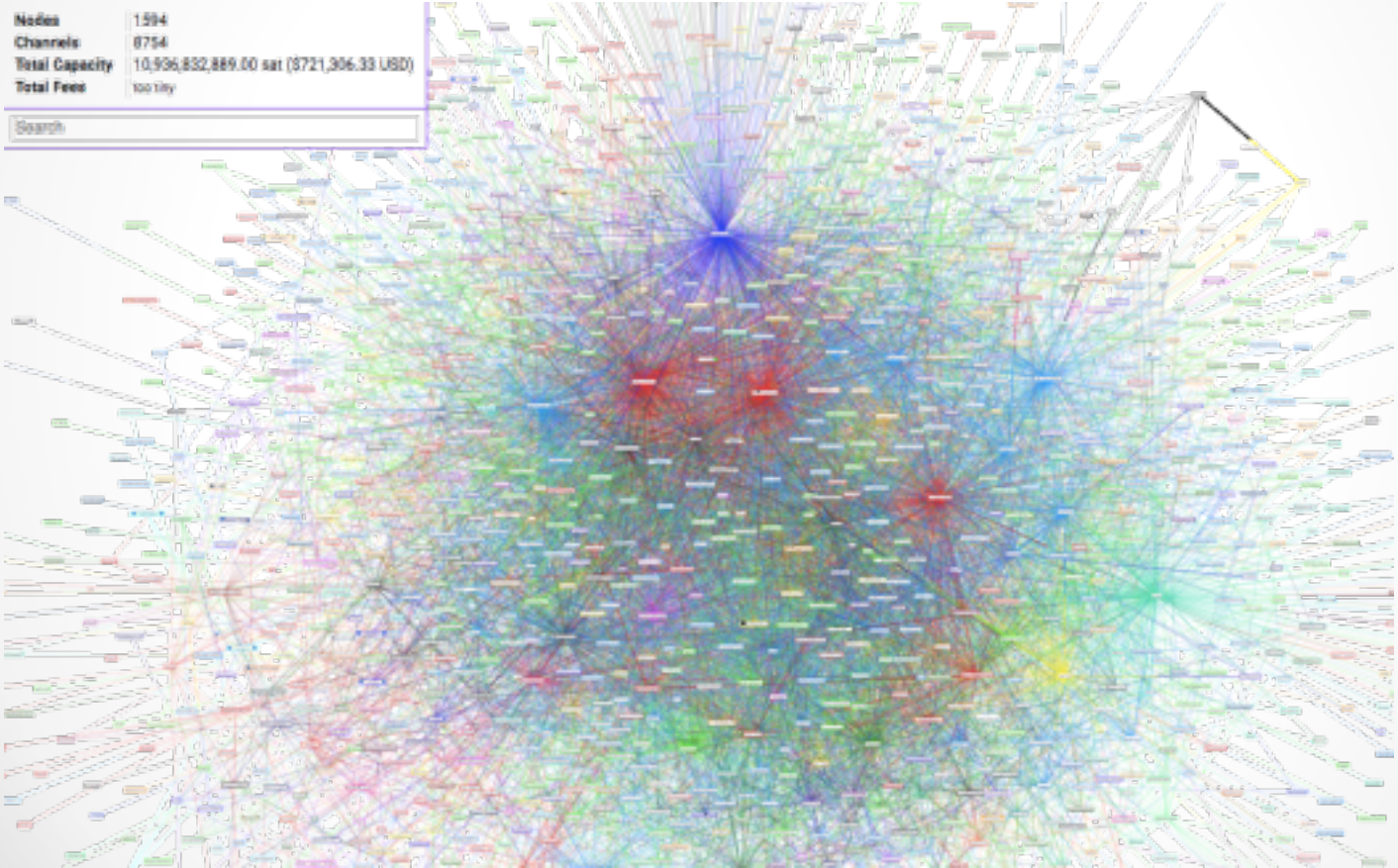
More Information, Better Decisions

- Nodes will track more metrics
  - Peer Node History
    - Past routing successes and failures
    - Forced close channels due to inactivity or route timeout/delays
    - Channels closed, opened, turnover frequency
- Active node management
- Different strategies for peer management, pathfinding

January 2018

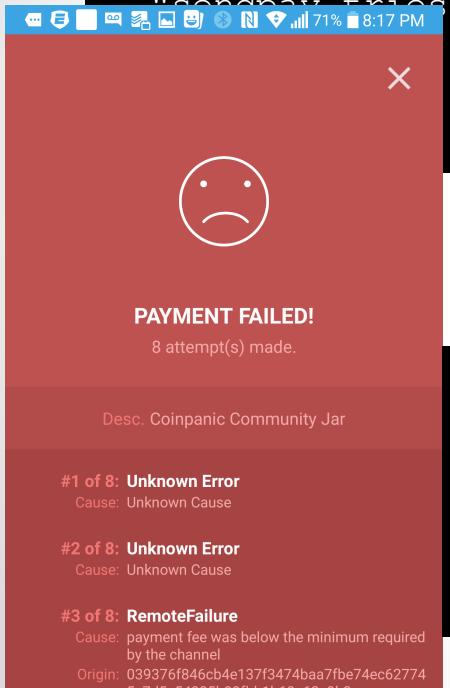


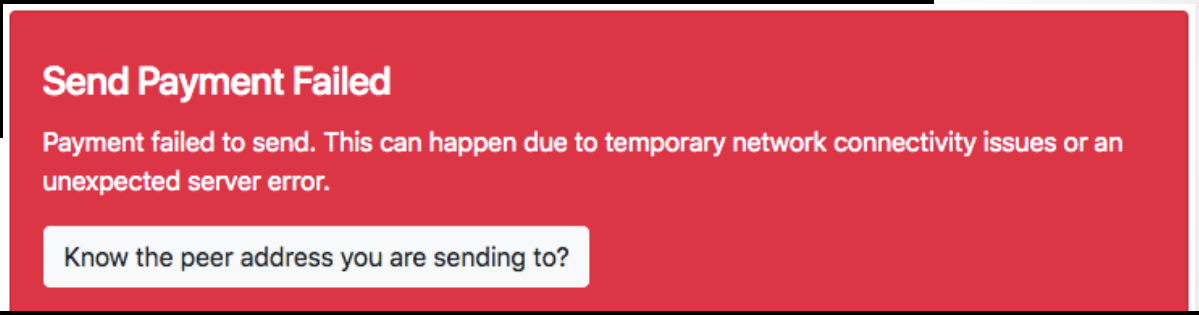
October 2018



Less of This

```
{ "code" : 205, "message" : "Could not find a route", "data" :
{
  "getroute_tries": 1,
  "sendpay_tries": 0,
```






```
Confirm payment (yes/no): yes
{
    "payment_error": "unable to find a path to destination",
    "payment_preimage": "",
    "payment_route": null
}
```

More of This