Bolt 7

P2P Node & Channel discovery

short_channel_id

What it is

Unique description of the funding transaction.

Purpose

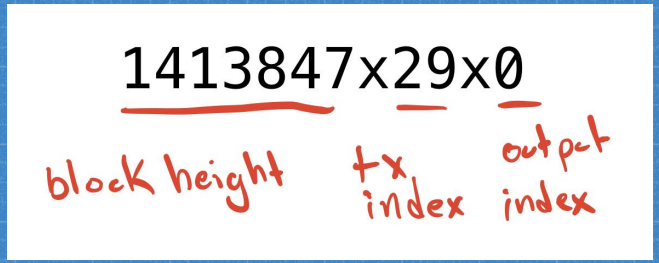
Implementation agnostic method of communicating unique channels

Breakdown

3 bytes - block height

3 bytes - transaction index

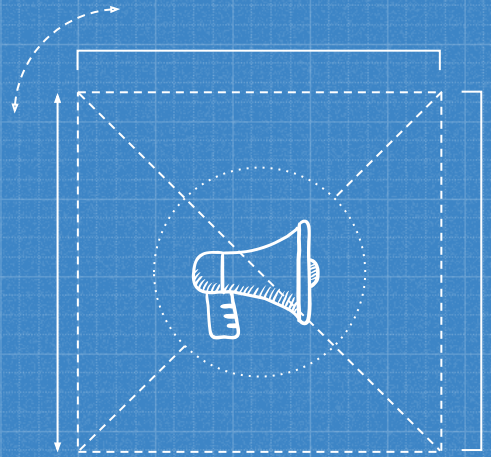
2 bytes - output index



Drawbacks

* Confirmations necessary before usage
* 2 methods to reference a channel is complex!

references
https://github.com/lightning/bolts/issues/301
https://www.derpturkey.com/lightning-network-routing-gossip-shortchannelid/



Messages

channel_announcement, node_announcement,
channel_update, announcement_signatures

1

announcement_signatures

gossip messages

announcement_signatures

Purpose

Opt in mechanism to allow announcement to rest of network

What it is

signatures necessary for peer to construct channel_announcement message

How it's used

Proves node ownership over funding transaction

1. type: 259 (`announcement_signatures`)
2. data:
   - [`channel_id` : `channel_id`]
   - [`short_channel_id` : `short_channel_id`]
   - [`signature` : `node_signature`]
   - [`signature` : `bitcoin_signature`]

When is it broadcasted

- if
  open_channel.announce_channel is set
  && shutdown message not sent
  && funding_locked is sent and recv
  && funding_tx has 6 confs
- on reconnection if above is met

references
http://site.ieee.org/icbc-2019/files/2019/05/ICBC-2019-Tutorial-3-Lightning-Network-Protocol.pdf
https://github.com/lightningnetwork/lnd/issues/1636
https://bitcoin.stackexchange.com/questions/80019/what-is-the-purpose-of-the-announcement-signatures-message-as-specified-by-the-l

announcement_signatures

How the peer processes it

- if the `short_channel_id` is NOT correct:
  - SHOULD fail the channel.
- if the `node_signature` OR the `bitcoin_signature` is NOT correct:
  - MAY fail the channel.
- if it has sent AND received a valid `announcement_signatures` message:
  - SHOULD queue the `channel_announcement` message for its peers.
- if it has not sent funding_locked:
  - MAY defer handling the announcement_signatures until after it has sent funding_locked
  - otherwise:
    - MUST ignore it.

1. type: 259 (`announcement_signatures`)
2. data:
   - [`channel_id` : `channel_id`]
   - [`short_channel_id` : `short_channel_id`]
   - [`signature` : `node_signature`]
   - [`signature` : `bitcoin_signature`]

Failure to broadcast means peer cannot create their channel_announcement edge (remember, most channels are 2 edges (bi-directional))

references
http://site.ieee.org/icbc-2019/files/2019/05/ICBC-2019-Tutorial-3-Lightning-Network-Protocol.pdf
https://github.com/lightningnetwork/lnd/issues/1636
https://bitcoin.stackexchange.com/questions/80019/what-is-the-purpose-of-the-announcement-signatures-message-as-specified-by-the-l

2

channel_announcement

gossip messages

channel_announcement

Purpose

communicates ownership info of a channel across the network

What it is

Proofs of the existence of a channel between node_1 and node_2

How it's used

Nodes pass this msg throughout the network, check the proofs to to connect the onchain bitcoin key to the lightning key.

Channel is not usable until fee and expiry is broadcast via channel_update

Proving the existence of a channel between `node_1` and `node_2` requires:

1. proving that the funding transaction pays to `bitcoin_key_1` and `bitcoin_key_2`
2. proving that `node_1` owns `bitcoin_key_1`
3. proving that `node_2` owns `bitcoin_key_2`

1. type: 256 (`channel_announcement`)
2. data:
   - [`signature` : `node_signature_1`]
   - [`signature` : `node_signature_2`]
   - [`signature` : `bitcoin_signature_1`]
   - [`signature` : `bitcoin_signature_2`]
   - [`u16` : `len`]
   - [`len*byte` : `features`]
   - [`chain_hash` : `chain_hash`]
   - [`short_channel_id` : `short_channel_id`]
   - [`point` : `node_id_1`]
   - [`point` : `node_id_2`]
   - [`point` : `bitcoin_key_1`]
   - [`point` : `bitcoin_key_2`]

When is it broadcasted

If open_channel.announce_channel is set

references
http://site.ieee.org/icbc-2019/files/2019/05/ICBC-2019-Tutorial-3-Lightning-Network-Protocol.pdf
https://github.com/lightningnetwork/lnd/issues/1636

channel_announcement

valid signature of h using each node secret

Proving the existence of a channel between `node_1` and `node_2` requires:

1. proving that the funding transaction pays to `bitcoin_key_1` and `bitcoin_key_2`
2. proving that `node_1` owns `bitcoin_key_1`
3. proving that `node_2` owns `bitcoin_key_2`

1. type: 256 (`channel_announcement`)
2. data:
   - [`signature` : `node_signature_1`]
   - [`signature` : `node_signature_2`]
   - [`signature` : `bitcoin_signature_1`]
   - [`signature` : `bitcoin_signature_2`]
   - [`u16` : `len`]
   - [`len*byte` : `features`]
   - [`chain_hash` : `chain_hash`]
   - [`short_channel_id` : `short_channel_id`]
   - [`point` : `node_id_1`]
   - [`point` : `node_id_2`]
   - [`point` : `bitcoin_key_1`]
   - [`point` : `bitcoin_key_2`]

h: double sha256 hash of the entire message, excluding the first 4 signatures

channel_announcement

valid signature of h using each node secret

must relate to respective node funding keys

Proving the existence of a channel between `node_1` and `node_2` requires:

1. proving that the funding transaction pays to `bitcoin_key_1` and `bitcoin_key_2`
2. proving that `node_1` owns `bitcoin_key_1`
3. proving that `node_2` owns `bitcoin_key_2`

1. type: 256 (`channel_announcement`)
2. data:
   - [`signature` : `node_signature_1`]
   - [`signature` : `node_signature_2`]
   - [`signature` : `bitcoin_signature_1`]
   - [`signature` : `bitcoin_signature_2`]
   - [`u16` : `len`]
   - [`len*byte` : `features`]
   - [`chain_hash` : `chain_hash`]
   - [`short_channel_id` : `short_channel_id`]
   - [`point` : `node_id_1`]
   - [`point` : `node_id_2`]
   - [`point` : `bitcoin_key_1`]
   - [`point` : `bitcoin_key_2`]

h: double sha256 hash of the entire message, excluding the first 4 signatures

channel_announcement

valid signature of h using each node secret

must relate to respective node funding keys

min length for features

bolt 9 features negotiated

Proving the existence of a channel between `node_1` and `node_2` requires:

1. proving that the funding transaction pays to `bitcoin_key_1` and `bitcoin_key_2`
2. proving that `node_1` owns `bitcoin_key_1`
3. proving that `node_2` owns `bitcoin_key_2`

1. type: 256 (`channel_announcement`)
2. data:
   - [`signature` : `node_signature_1`]
   - [`signature` : `node_signature_2`]
   - [`signature` : `bitcoin_signature_1`]
   - [`signature` : `bitcoin_signature_2`]
   - [`u16` : `len`]
   - [`len*byte` : `features`]
   - [`chain_hash` : `chain_hash`]
   - [`short_channel_id` : `short_channel_id`]
   - [`point` : `node_id_1`]
   - [`point` : `node_id_2`]
   - [`point` : `bitcoin_key_1`]
   - [`point` : `bitcoin_key_2`]

h: double sha256 hash of the entire message, excluding the first 4 signatures

channel_announcement

valid signature of h using each node secret

must relate to respective node funding keys

min length for features

bolt 9 features negotiated

Proving the existence of a channel between `node_1` and `node_2` requires:

1. proving that the funding transaction pays to `bitcoin_key_1` and `bitcoin_key_2`
2. proving that `node_1` owns `bitcoin_key_1`
3. proving that `node_2` owns `bitcoin_key_2`

1. type: 256 (`channel_announcement`)
2. data:
   - [`signature` : `node_signature_1`]
   - [`signature` : `node_signature_2`]
   - [`signature` : `bitcoin_signature_1`]
   - [`signature` : `bitcoin_signature_2`]
   - [`u16` : `len`]
   - [`len*byte` : `features`]
   - [`chain_hash` : `chain_hash`]
   - [`short_channel_id` : `short_channel_id`]
   - [`point` : `node_id_1`]
   - [`point` : `node_id_2`]
   - [`point` : `bitcoin_key_1`]
   - [`point` : `bitcoin_key_2`]

if recv node sees unknown EVEN bit - it can't route thru!

h: double sha256 hash of the entire message, excluding the first 4 signatures

channel_announcement

valid signature of h using each node secret

must relate to respective node funding keys

min length for features

bolt 9 features negotiated

genesis block hash to identify chain

Proving the existence of a channel between `node_1` and `node_2` requires:

1. proving that the funding transaction pays to `bitcoin_key_1` and `bitcoin_key_2`
2. proving that `node_1` owns `bitcoin_key_1`
3. proving that `node_2` owns `bitcoin_key_2`

1. type: 256 (`channel_announcement`)
2. data:
   - [`signature` : `node_signature_1`]
   - [`signature` : `node_signature_2`]
   - [`signature` : `bitcoin_signature_1`]
   - [`signature` : `bitcoin_signature_2`]
   - [`u16` : `len`]
   - [`len*byte` : `features`]
   - [`chain_hash` : `chain_hash`]
   - [`short_channel_id` : `short_channel_id`]
   - [`point` : `node_id_1`]
   - [`point` : `node_id_2`]
   - [`point` : `bitcoin_key_1`]
   - [`point` : `bitcoin_key_2`]

if recv node sees unknown EVEN bit - it can't route thru!

h: double sha256 hash of the entire message, excluding the first 4 signatures

channel_announcement

valid signature of h using each node secret

must relate to respective node funding keys

min length for features

bolt 9 features negotiated

genesis block hash to identify chain

reference to funding tx

Proving the existence of a channel between `node_1` and `node_2` requires:

1. proving that the funding transaction pays to `bitcoin_key_1` and `bitcoin_key_2`
2. proving that `node_1` owns `bitcoin_key_1`
3. proving that `node_2` owns `bitcoin_key_2`

1. type: 256 (`channel_announcement`)
2. data:
   - [`signature` : `node_signature_1`]
   - [`signature` : `node_signature_2`]
   - [`signature` : `bitcoin_signature_1`]
   - [`signature` : `bitcoin_signature_2`]
   - [`u16` : `len`]
   - [`len*byte` : `features`]
   - [`chain_hash` : `chain_hash`]
   - [`short_channel_id` : `short_channel_id`]
   - [`point` : `node_id_1`]
   - [`point` : `node_id_2`]
   - [`point` : `bitcoin_key_1`]
   - [`point` : `bitcoin_key_2`]

if recv node sees unknown EVEN bit - it can't route thru!

h: double sha256 hash of the entire message, excluding the first 4 signatures

channel_announcement

valid signature of h using each node secret

must relate to respective node funding keys

min length for features

bolt 9 features negotiated

genesis block hash to identify chain

reference to funding tx

pubkeys in lexicographically ascending order

Proving the existence of a channel between `node_1` and `node_2` requires:

1. proving that the funding transaction pays to `bitcoin_key_1` and `bitcoin_key_2`
2. proving that `node_1` owns `bitcoin_key_1`
3. proving that `node_2` owns `bitcoin_key_2`

1. type: 256 (`channel_announcement`)
2. data:
   - [`signature` : `node_signature_1`]
   - [`signature` : `node_signature_2`]
   - [`signature` : `bitcoin_signature_1`]
   - [`signature` : `bitcoin_signature_2`]
   - [`u16` : `len`]
   - [`len*byte` : `features`]
   - [`chain_hash` : `chain_hash`]
   - [`short_channel_id` : `short_channel_id`]
   - [`point` : `node_id_1`]
   - [`point` : `node_id_2`]
   - [`point` : `bitcoin_key_1`]
   - [`point` : `bitcoin_key_2`]

if recv node sees unknown EVEN bit - it can't route thru!

h: double sha256 hash of the entire message, excluding the first 4 signatures

channel_announcement

valid signature of h using each node secret

must relate to respective node funding keys

min length for features

bolt 9 features negotiated

genesis block hash to identify chain

reference to funding tx

pubkeys in lexicographically ascending order

Proving the existence of a channel between `node_1` and `node_2` requires:

1. proving that the funding transaction pays to `bitcoin_key_1` and `bitcoin_key_2`
2. proving that `node_1` owns `bitcoin_key_1`
3. proving that `node_2` owns `bitcoin_key_2`

1. type: 256 (`channel_announcement`)
2. data:
   - [`signature` : `node_signature_1`]
   - [`signature` : `node_signature_2`]
   - [`signature` : `bitcoin_signature_1`]
   - [`signature` : `bitcoin_signature_2`]
   - [`u16` : `len`]
   - [`len*byte` : `features`]
   - [`chain_hash` : `chain_hash`]
   - [`short_channel_id` : `short_channel_id`]
   - [`point` : `node_id_1`]
   - [`point` : `node_id_2`]
   - [`point` : `bitcoin_key_1`]
   - [`point` : `bitcoin_key_2`]

if recv node sees unknown EVEN bit - it can't route thru!

if short_channel_id is already seen with diff node_ids, receiving node will blacklist both nodes and forget all channels

h: double sha256 hash of the entire message, excluding the first 4 signatures

channel_announcement

valid signature of h using each node secret

must relate to respective node funding keys

min length for features

bolt 9 features negotiated

genesis block hash to identify chain

reference to funding tx

pubkeys in lexicographically ascending order

respective funding_pubkey

Proving the existence of a channel between `node_1` and `node_2` requires:

1. proving that the funding transaction pays to `bitcoin_key_1` and `bitcoin_key_2`
2. proving that `node_1` owns `bitcoin_key_1`
3. proving that `node_2` owns `bitcoin_key_2`

1. type: 256 (`channel_announcement`)
2. data:
   - [`signature` : `node_signature_1`]
   - [`signature` : `node_signature_2`]
   - [`signature` : `bitcoin_signature_1`]
   - [`signature` : `bitcoin_signature_2`]
   - [`u16` : `len`]
   - [`len*byte` : `features`]
   - [`chain_hash` : `chain_hash`]
   - [`short_channel_id` : `short_channel_id`]
   - [`point` : `node_id_1`]
   - [`point` : `node_id_2`]
   - [`point` : `bitcoin_key_1`]
   - [`point` : `bitcoin_key_2`]

if recv node sees unknown EVEN bit - it can't route thru!

if short_channel_id is already seen with diff node_ids, receiving node will blacklist both nodes and forget all channels

h: double sha256 hash of the entire message, excluding the first 4 signatures

3

node_announcment

gossip messages

node_announcement

Purpose

communicates node metadata across the network

What it is

connection info, arbitrary attributes for display on explorers, broadcasted features, etc

How it's used

Useful for nodes with a changing IP. Is ignored unless a channel is associated with it.

When is it broadcasted

After channel_announcement

1. type: 257 (`node_announcement`)
2. data:
   - [`signature` : `signature`]
   - [`u16` : `flen`]
   - [`flen*byte` : `features`]
   - [`u32` : `timestamp`]
   - [`point` : `node_id`]
   - [`3*byte` : `rgb_color`]
   - [`32*byte` : `alias`]
   - [`u16` : `addrlen`]
   - [`addrlen*byte` : `addresses`]

references
http://site.ieee.org/icbc-2019/files/2019/05/ICBC-2019-Tutorial-3-Lightning-Network-Protocol.pdf
https://bitcoin.stackexchange.com/questions/80546/how-does-the-lightning-network-handle-changing-ips

node_announcement

Purpose

communicates node metadata across the network

What it is

connection info, arbitrary attributes for display on explorers, broadcasted features, etc

How it's used

Useful for nodes with a changing IP. Is ignored unless a channel is associated with it.

When is it broadcasted

After channel_announcement

signature of double sha256 of rest of packet with sec key of node_id

1. type: 257 (`node_announcement`)
2. data:
   - [`signature` : `signature`]
   - [`u16` : `flen`]
   - [`flen*byte` : `features`]
   - [`u32` : `timestamp`]
   - [`point` : `node_id`]
   - [`3*byte` : `rgb_color`]
   - [`32*byte` : `alias`]
   - [`u16` : `addrlen`]
   - [`addrlen*byte` : `addresses`]

references
http://site.ieee.org/icbc-2019/files/2019/05/ICBC-2019-Tutorial-3-Lightning-Network-Protocol.pdf
https://bitcoin.stackexchange.com/questions/80546/how-does-the-lightning-network-handle-changing-ips

node_announcement

Purpose

communicates node metadata across the network

What it is

connection info, arbitrary attributes for display on explorers, broadcasted features, etc

How it's used

Useful for nodes with a changing IP. Is ignored unless a channel is associated with it.

When is it broadcasted

After channel_announcement

signature of double sha256 of rest of packet with sec key of node_id

address descriptors so others can reach directly

1. type: 257 (`node_announcement`)
2. data:
   - [`signature` : `signature`]
   - [`u16` : `flen`]
   - [`flen*byte` : `features`]
   - [`u32` : `timestamp`]
   - [`point` : `node_id`]
   - [`3*byte` : `rgb_color`]
   - [`32*byte` : `alias`]
   - [`u16` : `addrlen`]
   - [`addrlen*byte` : `addresses`]

defined types:

1 – ipv4

2 – ipv6

3 – TorV2 [Deprecated]

4 - TorV3

references
http://site.ieee.org/icbc-2019/files/2019/05/ICBC-2019-Tutorial-3-Lightning-Network-Protocol.pdf
https://bitcoin.stackexchange.com/questions/80546/how-does-the-lightning-network-handle-changing-ips

4

channel_update

gossip messages

channel_update

Purpose

Update properties and policies of an active edge

What it is

Policy info. Fees, cltv, supported features, active timestamp

How it's used

Used to broadcast if a channel is disabled. Channels can't be used without at least one of these messages reaching a user as it conveys ctlv and fee data

1. type: 258 (`channel_update`)
2. data:
   - [`signature` : `signature`]
   - [`chain_hash` : `chain_hash`]
   - [`short_channel_id` : `short_channel_id`]
   - [`u32` : `timestamp`]
   - [`byte` : `message_flags`]
   - [`byte` : `channel_flags`]
   - [`u16` : `cltv_expiry_delta`]
   - [`u64` : `htlc_minimum_msat`]
   - [`u32` : `fee_base_msat`]
   - [`u32` : `fee_proportional_millionths`]
   - [`u64` : `htlc_maximum_msat`] (option_channel_htlc_max)

When is it broadcasted

- if funding_locked has been received
- may be sent to peer to communicate channel parameters

MUST NOT be forwarded to other peers in this case

references
https://github.com/lnbook/lnbook/blob/ec806916edd6f4d1b2f9da2fef08684f80acb671/appendix_protocol_messages.asciidoc#L592
http://site.ieee.org/icbc-2019/files/2019/05/ICBC-2019-Tutorial-3-Lightning-Network-Protocol.pdf

channel_update

double-sha256 of rest of packet with nodeID

genesis block hash for chain

identify which channel this is in regards to

1. type: 258 (`channel_update`)
2. data:
   - [`signature` : `signature`]
   - [`chain_hash` : `chain_hash`]
   - [`short_channel_id` : `short_channel_id`]
   - [`u32` : `timestamp`]
   - [`byte` : `message_flags`]
   - [`byte` : `channel_flags`]
   - [`u16` : `cltv_expiry_delta`]
   - [`u64` : `htlc_minimum_msat`]
   - [`u32` : `fee_base_msat`]
   - [`u32` : `fee_proportional_millionths`]
   - [`u64` : `htlc_maximum_msat`] (option_channel_htlc_max)

references
https://github.com/lnbook/lnbook/blob/ec806916edd6f4d1b2f9da2fef08684f80acb671/appendix_protocol_messages.asciidoc#L592
http://site.ieee.org/icbc-2019/files/2019/05/ICBC-2019-Tutorial-3-Lightning-Network-Protocol.pdf

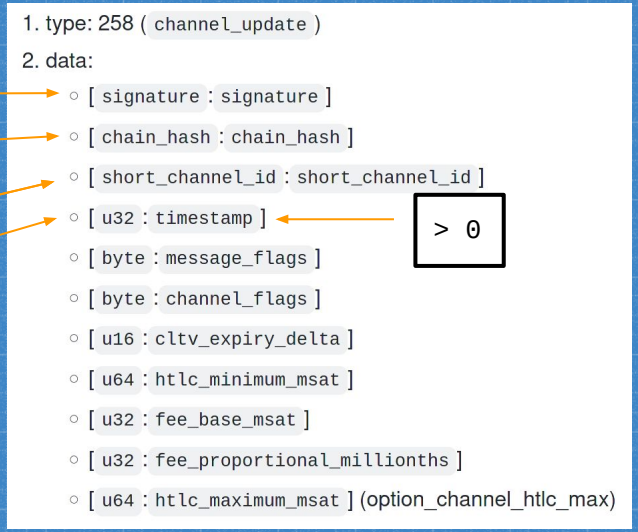
channel_update

double-sha256 of rest of packet with nodeID

genesis block hash for chain

identify which channel this is in regards to

greater than previous timestamp for edge



references
https://github.com/lnbook/lnbook/blob/ec806916edd6f4d1b2f9da2fef08684f80acb671/appendix_protocol_messages.asciidoc#L592
http://site.ieee.org/icbc-2019/files/2019/05/ICBC-2019-Tutorial-3-Lightning-Network-Protocol.pdf

channel_update

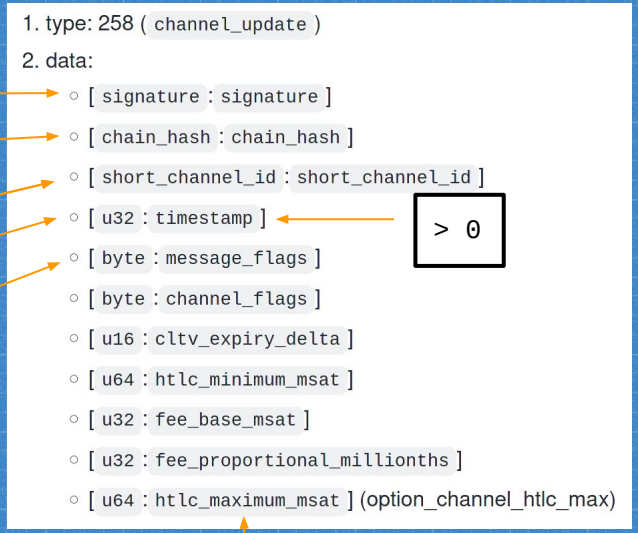
double-sha256 of rest of packet with nodeID

genesis block hash for chain

identify which channel this is in regards to

greater than previous timestamp for edge

indicate if optional messages exist at the end of the core fields



only BOLT specified optional message.
Indicates max amount sendable with one htlc

references
https://github.com/lnbook/lnbook/blob/ec806916edd6f4d1b2f9da2fef08684f80acb671/appendix_protocol_messages.asciidoc#L592
http://site.ieee.org/icbc-2019/files/2019/05/ICBC-2019-Tutorial-3-Lightning-Network-Protocol.pdf

channel_update

double-sha256 of rest of packet with nodeID

genesis block hash for chain

identify which channel this is in regards to

greater than previous timestamp for edge

indicate if optional messages exist at the end of the core fields

The `channel_flags` bitfield is used to indicate the direction of the channel: it identifies the node that this update originated from and signals various options concerning the channel. The following table specifies the meaning of its individual bits:

Bit Position	Name	Meaning
0	`direction`	Direction this update refers to.
1	`disable`	Disable the channel.

1. type: 258 (`channel_update`)
2. data:
   - [`signature` : `signature`]
   - [`chain_hash` : `chain_hash`]
   - [`short_channel_id` : `short_channel_id`]
   - [`u32` : `timestamp`] > 0
   - [`byte` : `message_flags`]
   - [`byte` : `channel_flags`]
   - [`u16` : `cltv_expiry_delta`]
   - [`u64` : `htlc_minimum_msat`]
   - [`u32` : `fee_base_msat`]
   - [`u32` : `fee_proportional_millionths`]
   - [`u64` : `htlc_maximum_msat`] (option_channel_htlc_max)

only BOLT specified optional message.
Indicates max amount sendable with one htlc

references
https://github.com/lnbook/lnbook/blob/ec806916edd6f4d1b2f9da2fef08684f80acb671/appendix_protocol_messages.asciidoc#L592
http://site.ieee.org/icbc-2019/files/2019/05/ICBC-2019-Tutorial-3-Lightning-Network-Protocol.pdf

channel_update

double-sha256 of rest of packet with nodeID

genesis block hash for chain

identify which channel this is in regards to

greater than previous timestamp for edge

indicate if optional messages exist at the end of the core fields

cltv policy

The `channel_flags` bitfield is used to indicate the direction of the channel: it identifies the node that this update originated from and signals various options concerning the channel. The following table specifies the meaning of its individual bits:

Bit Position	Name	Meaning
0	`direction`	Direction this update refers to.
1	`disable`	Disable the channel.

1. type: 258 (`channel_update`)
2. data:
   - [`signature` : `signature`]
   - [`chain_hash` : `chain_hash`]
   - [`short_channel_id` : `short_channel_id`]
   - [`u32` : `timestamp`] > 0
   - [`byte` : `message_flags`]
   - [`byte` : `channel_flags`]
   - [`u16` : `cltv_expiry_delta`]
   - [`u64` : `htlc_minimum_msat`]
   - [`u32` : `fee_base_msat`]
   - [`u32` : `fee_proportional_millionths`]
   - [`u64` : `htlc_maximum_msat`] (option_channel_htlc_max)

only BOLT specified optional message.
Indicates max amount sendable with one htlc

references
https://github.com/lnbook/lnbook/blob/ec806916edd6f4d1b2f9da2fef08684f80acb671/appendix_protocol_messages.asciidoc#L592
http://site.ieee.org/icbc-2019/files/2019/05/ICBC-2019-Tutorial-3-Lightning-Network-Protocol.pdf

channel_update

The `channel_flags` bitfield is used to indicate the direction of the channel: it identifies the node that this update originated from and signals various options concerning the channel. The following table specifies the meaning of its individual bits:

Bit Position	Name	Meaning
0	`direction`	Direction this update refers to.
1	`disable`	Disable the channel.

1. type: 258 (`channel_update`)
2. data:
   - [`signature` : `signature`]
   - [`chain_hash` : `chain_hash`]
   - [`short_channel_id` : `short_channel_id`]
   - [`u32` : `timestamp`] > 0
   - [`byte` : `message_flags`]
   - [`byte` : `channel_flags`]
   - [`u16` : `cltv_expiry_delta`]
   - [`u64` : `htlc_minimum_msat`]
   - [`u32` : `fee_base_msat`]
   - [`u32` : `fee_proportional_millionths`]
   - [`u64` : `htlc_maximum_msat`] (option_channel_htlc_max)

double-sha256 of rest of packet with nodeID

genesis block hash for chain

identify which channel this is in regards to

greater than previous timestamp for edge

indicate if optional messages exist at the end of the core fields

cltv policy

min amount sendable with one htlc

only BOLT specified optional message. Indicates max amount sendable with one htlc

references
https://github.com/lnbook/lnbook/blob/ec806916edd6f4d1b2f9da2fef08684f80acb671/appendix_protocol_messages.asciidoc#L592
http://site.ieee.org/icbc-2019/files/2019/05/ICBC-2019-Tutorial-3-Lightning-Network-Protocol.pdf

channel_update

The `channel_flags` bitfield is used to indicate the direction of the channel: it identifies the node that this update originated from and signals various options concerning the channel. The following table specifies the meaning of its individual bits:

Bit Position	Name	Meaning
0	`direction`	Direction this update refers to.
1	`disable`	Disable the channel.

1. type: 258 (`channel_update`)
2. data:
   - [`signature` : `signature`]
   - [`chain_hash` : `chain_hash`]
   - [`short_channel_id` : `short_channel_id`]
   - [`u32` : `timestamp`] > 0
   - [`byte` : `message_flags`]
   - [`byte` : `channel_flags`]
   - [`u16` : `cltv_expiry_delta`]
   - [`u64` : `htlc_minimum_msat`]
   - [`u32` : `fee_base_msat`]
   - [`u32` : `fee_proportional_millionths`]
   - [`u64` : `htlc_maximum_msat`] (option_channel_htlc_max)

double-sha256 of rest of packet with nodeID

genesis block hash for chain

identify which channel this is in regards to

greater than previous timestamp for edge

indicate if optional messages exist at the end of the core fields

cltv policy

min amount sendable with one htlc

static fee per payment

dynamic feerate per payment

only BOLT specified optional message.
Indicates max amount sendable with one htlc

references
https://github.com/lnbook/lnbook/blob/ec806916edd6f4d1b2f9da2fef08684f80acb671/appendix_protocol_messages.asciidoc#L592
http://site.ieee.org/icbc-2019/files/2019/05/ICBC-2019-Tutorial-3-Lightning-Network-Protocol.pdf

Sequence of events

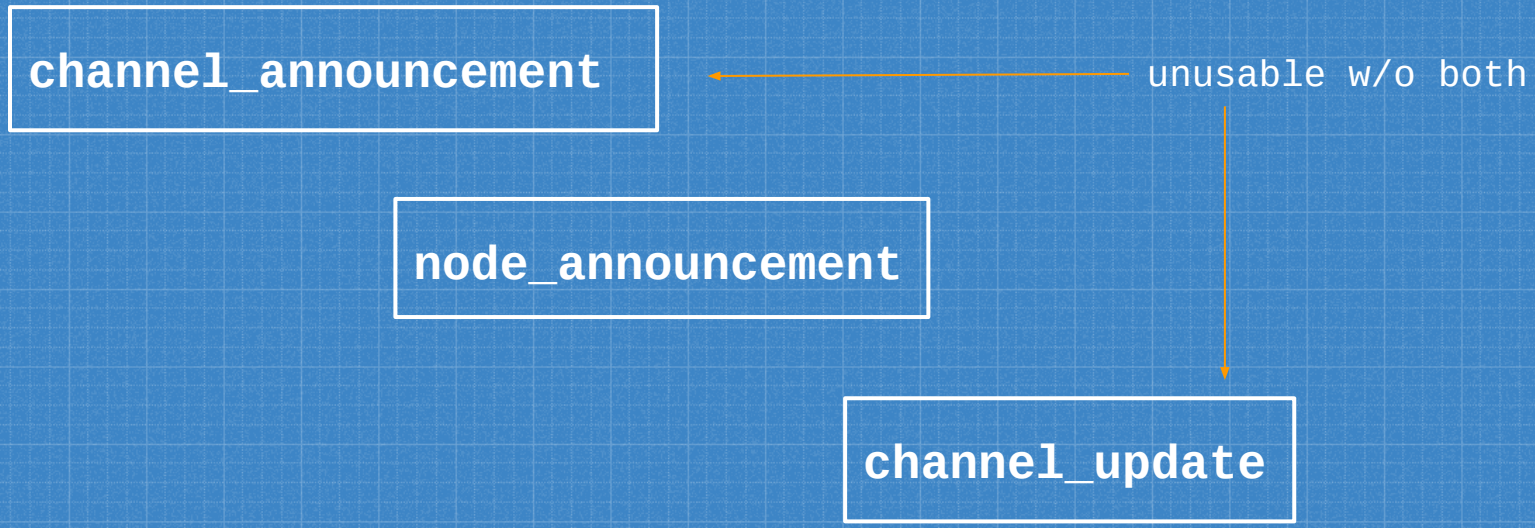
channel_announcement

node_announcement

channel_update

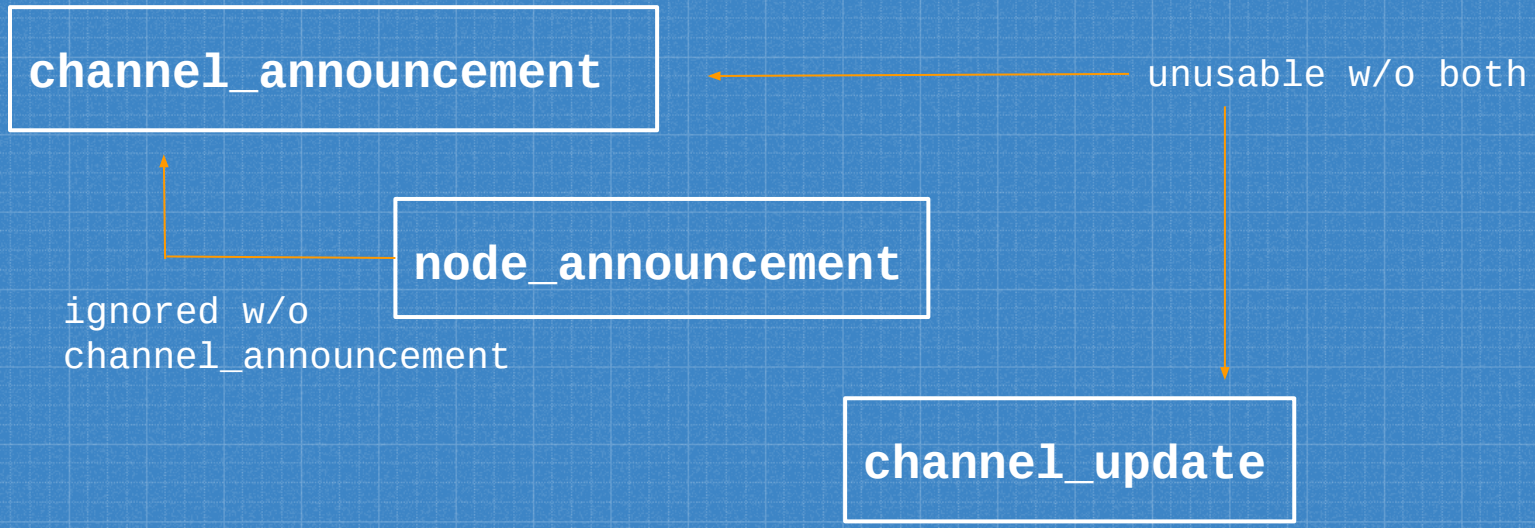
references
http://site.ieee.org/icbc-2019/files/2019/05/ICBC-2019-Tutorial-3-Lightning-Network-Protocol.pdf
https://github.com/lightningnetwork/lnd/issues/1636

Sequence of events



references
http://site.ieee.org/icbc-2019/files/2019/05/ICBC-2019-Tutorial-3-Lightning-Network-Protocol.pdf
https://github.com/lightningnetwork/lnd/issues/1636

Sequence of messages



references
http://site.ieee.org/icbc-2019/files/2019/05/ICBC-2019-Tutorial-3-Lightning-Network-Protocol.pdf
https://github.com/lightningnetwork/lnd/issues/1636

Querying tools

query_short_channel_ids

returns
channel_announcement and
channel_update messages
for specific channels

Used when node sees
channel_update but not
channel_announcement for
a channel

query_channel_range

returns
all channels in
specified block range

Used to discover new
channels

gossip_timestamp_filter

returns
channel_announcements and
channel_update messages by
date range

Used to receive real time
updates in channel graph

references
https://github.com/lnbook/lnbook/blob/ece69d5c2ac8116ef83c1826bd43bd4b33c74dca/appendix_protocol_messages.asciidoc#the-query_channel_range-message
https://github.com/lnbook/lnbook/blob/ece69d5c2ac8116ef83c1826bd43bd4b33c74dca/appendix_protocol_messages.asciidoc#the-gossip_timestamp_range-message
http://site.ieee.org/icbc-2019/files/2019/05/ICBC-2019-Tutorial-3-Lightning-Network-Protocol.pdf
https://docs.rs/lightning/0.0.101/lightning/ln/msgs/struct.ReplyShortChannelIdsEnd.html
https://bitcoinops.org/en/newsletters/2019/08/07/

Thank you!

ANY QUESTIONS?

You can find me at:

@_arshbot
harshagoli@protonmail.com