LIGHTNING DEV KIT

Or: How I Learned to Stop Worrying and Embrace the Rust

@jkczyz @arikaleph

What's LDK?

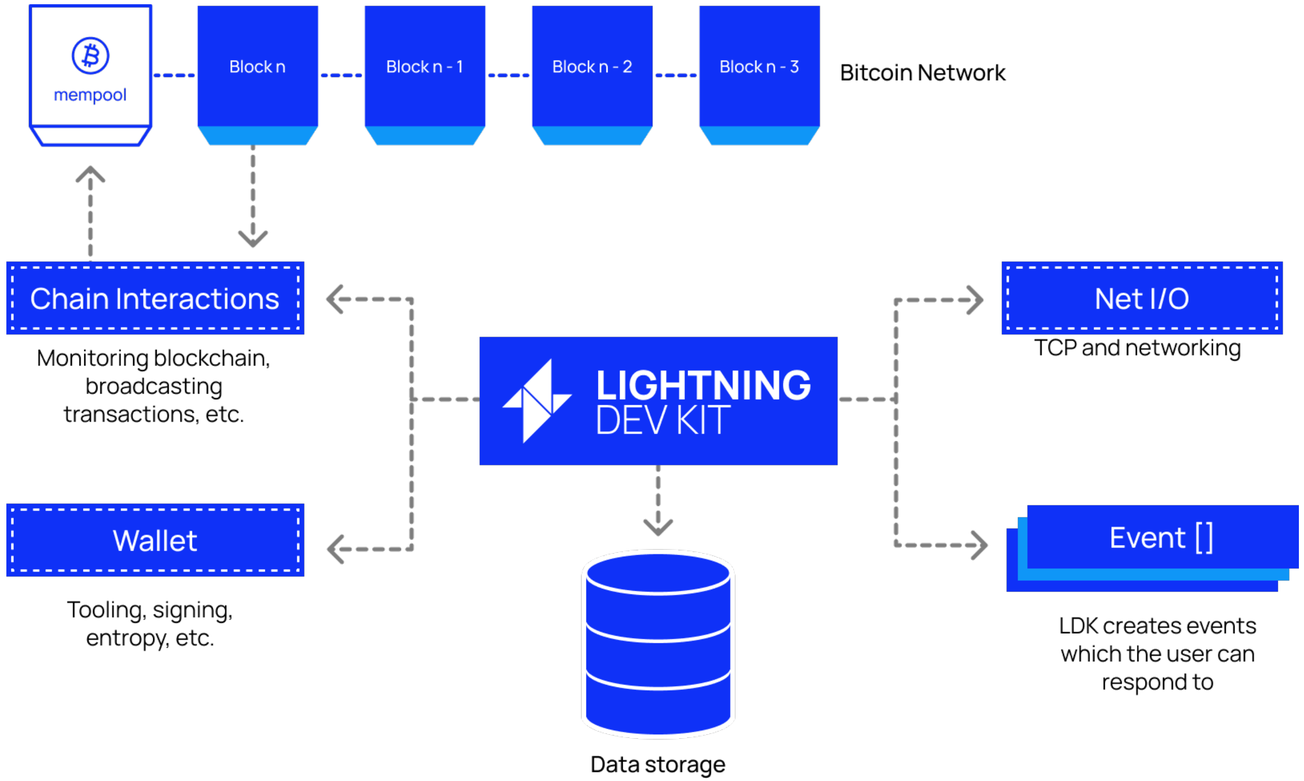
- Lightning Network implementation as SDK
- Handles channel state machine (i. e. the difficult stuff)
- lightningdevkit.org

Why LDK?

- Lightweight & performant
- Multilingual
- Customizable out of the box

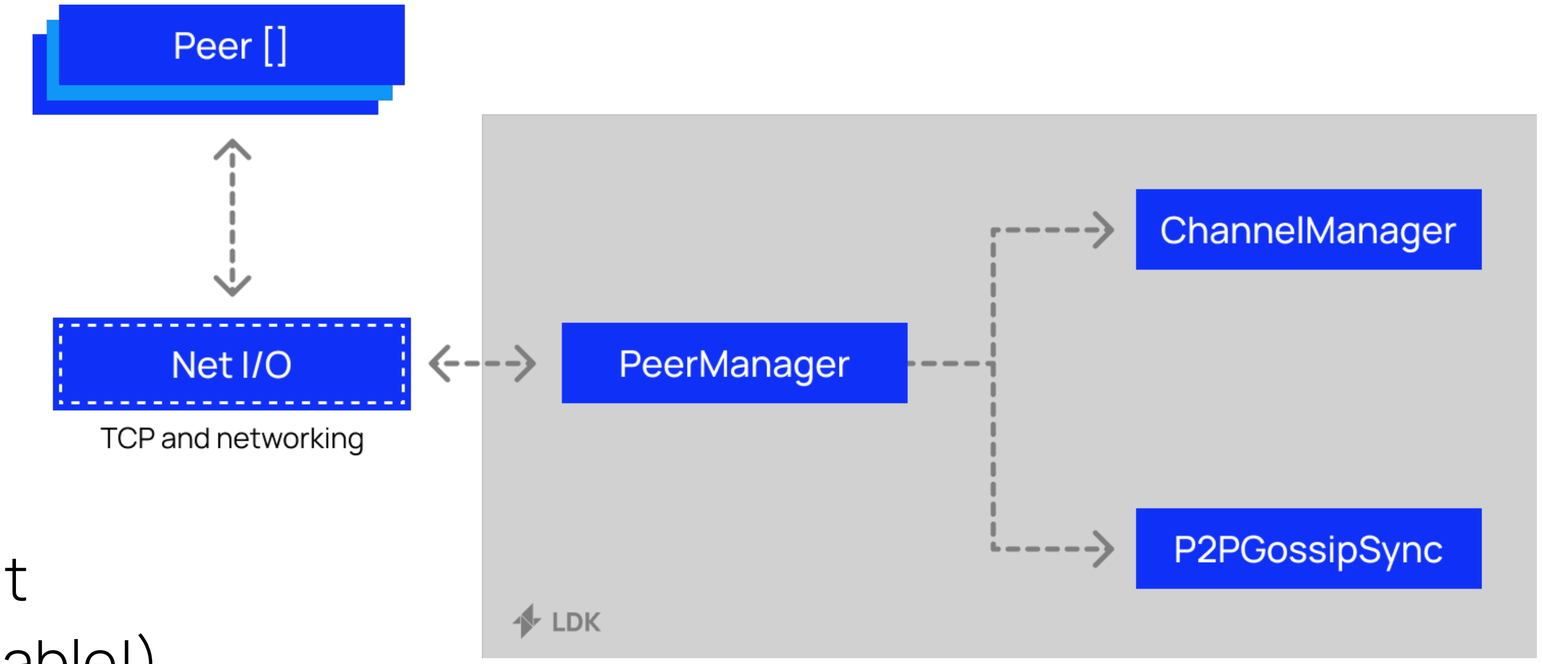
Architecture

- Major components:
  - Peer management
  - Channel management
  - Payments & routing
- Language Bindings



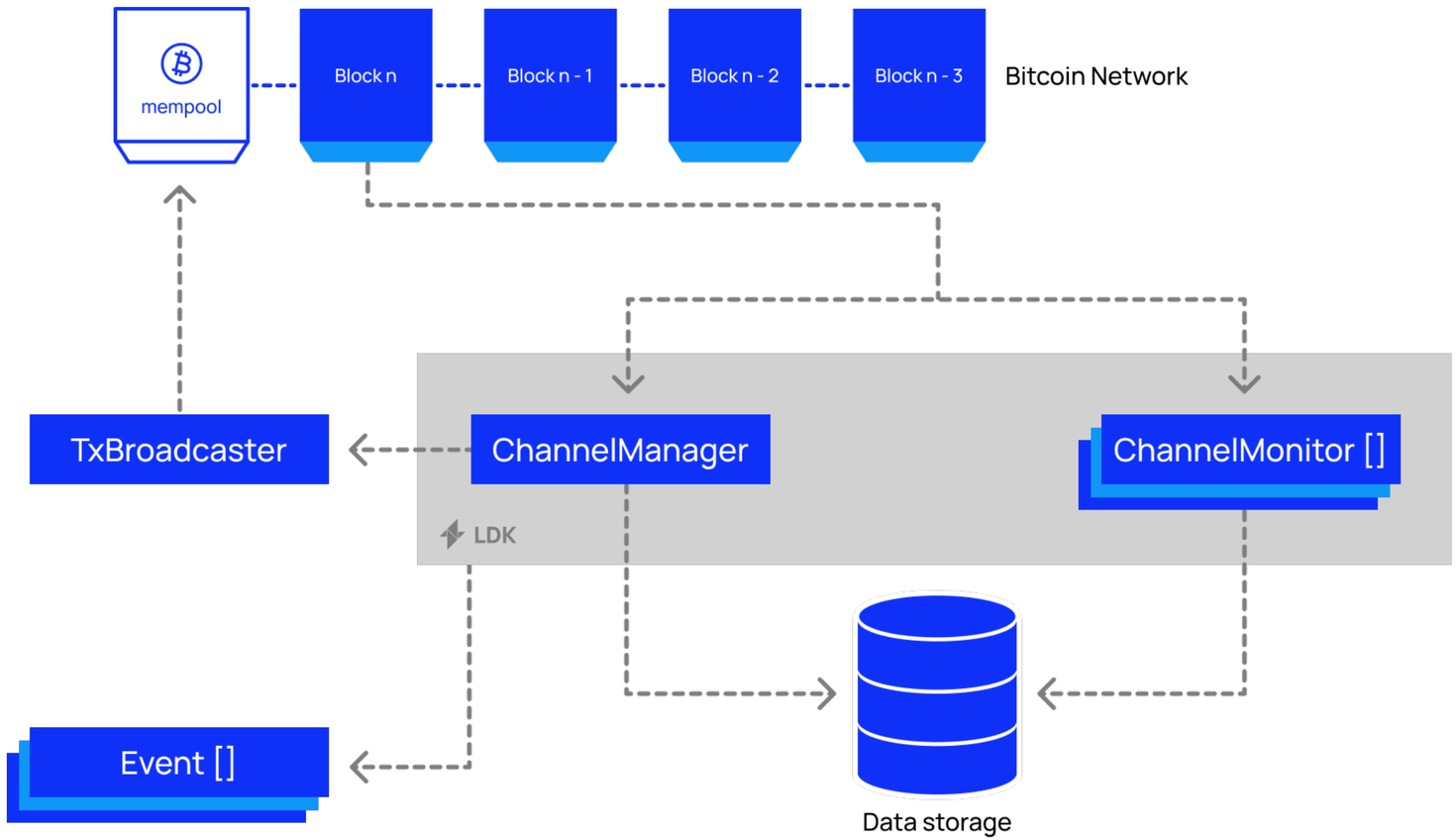
Peer Management

- BOLTs 1, 8, 9
- Requirements
  - Network I/O
  - Entropy
  - Periodic calls (default implementation available!)



Channel Management

- BOLT 2
- Requirements
  - Disk persistence
  - Tx broadcasting
  - Block data
  - Event handling
  - Fee estimation



Channel Management

Persistence

- Choose how/where to persist data
- Locally
- Encrypted
- Remotely
- Block channel state machine until advancement is safe

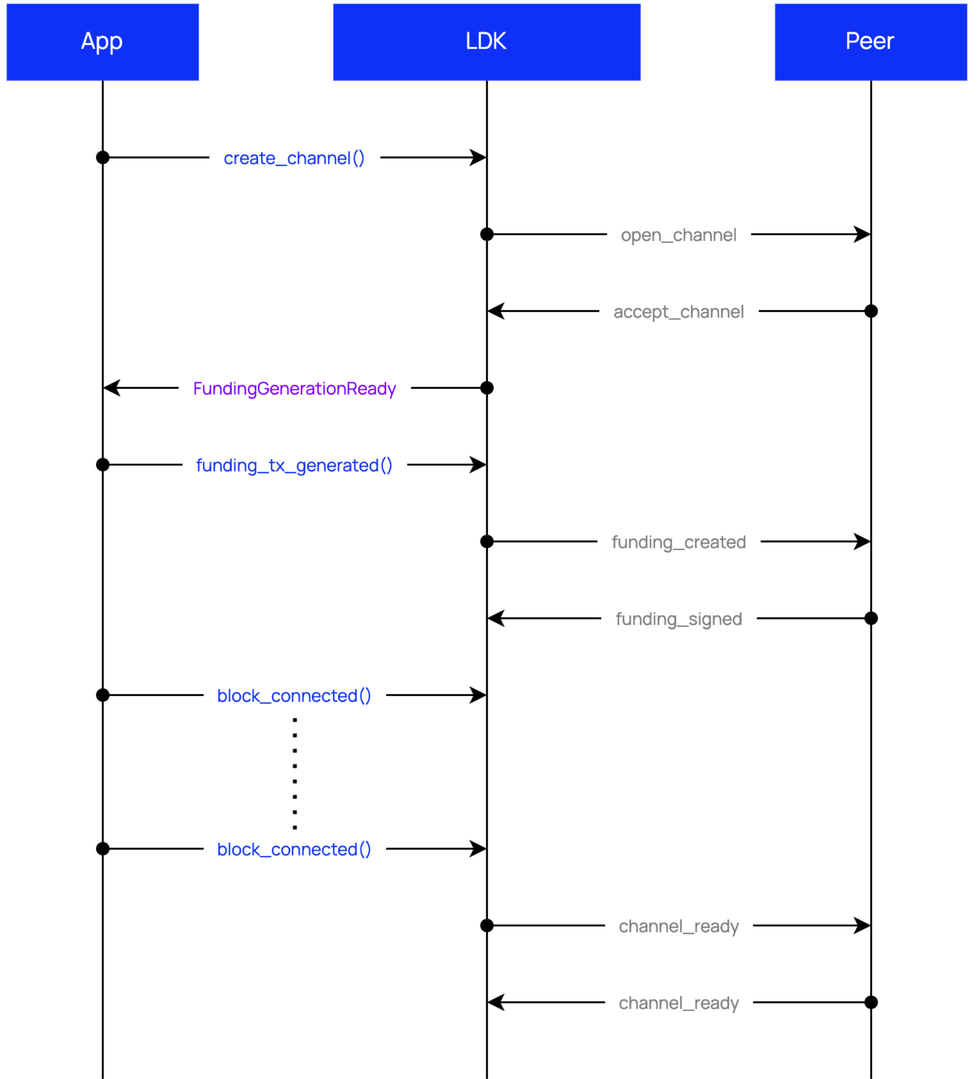
Block Source Data

- Choice of paradigm:
  - Full blocks and reorgs (e. g. full node)
  - Individual transactions and output scripts (e. g. Electrum, Neutrino, etc.)
- Notify listeners

Channel Management

Event Handling: Outbound

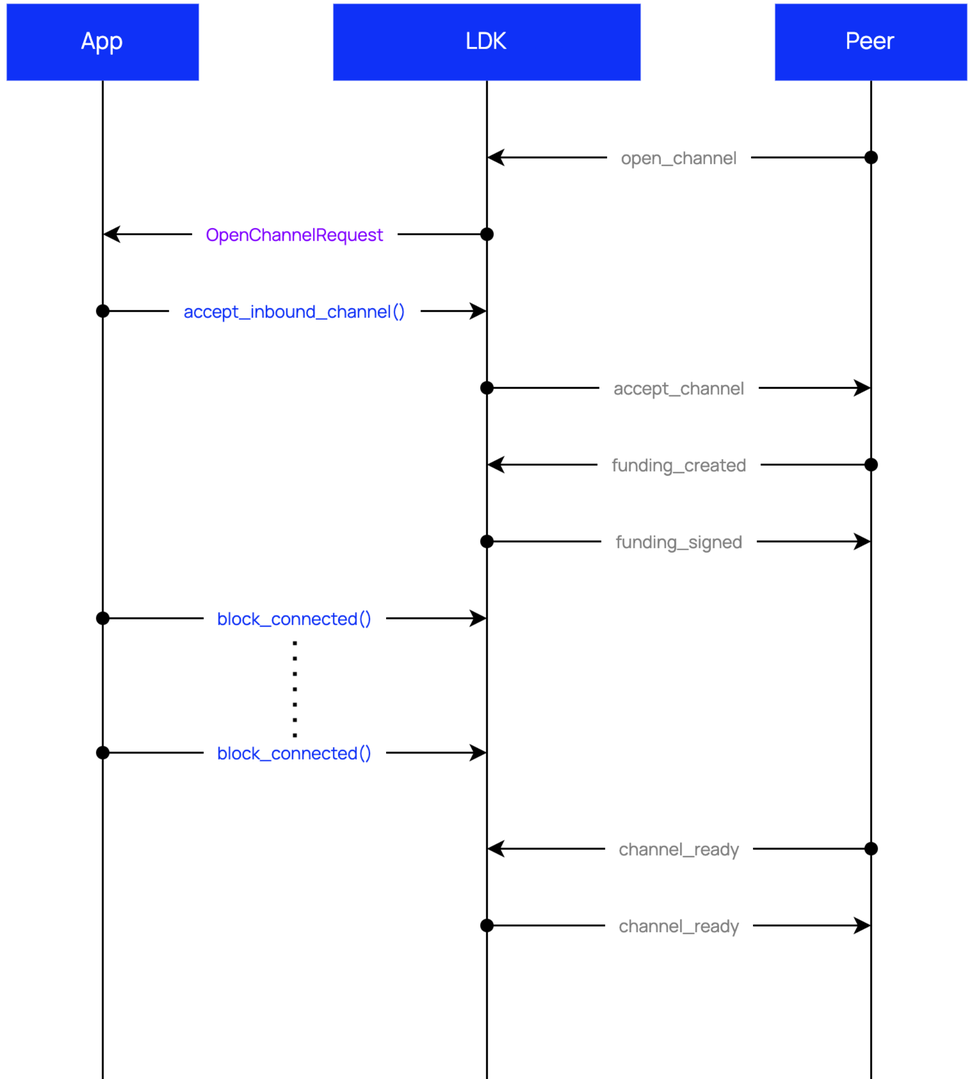
- FundingGenerationReady
- DiscardFunding
- ChannelClosed
- SpendableOutputs



Channel Management

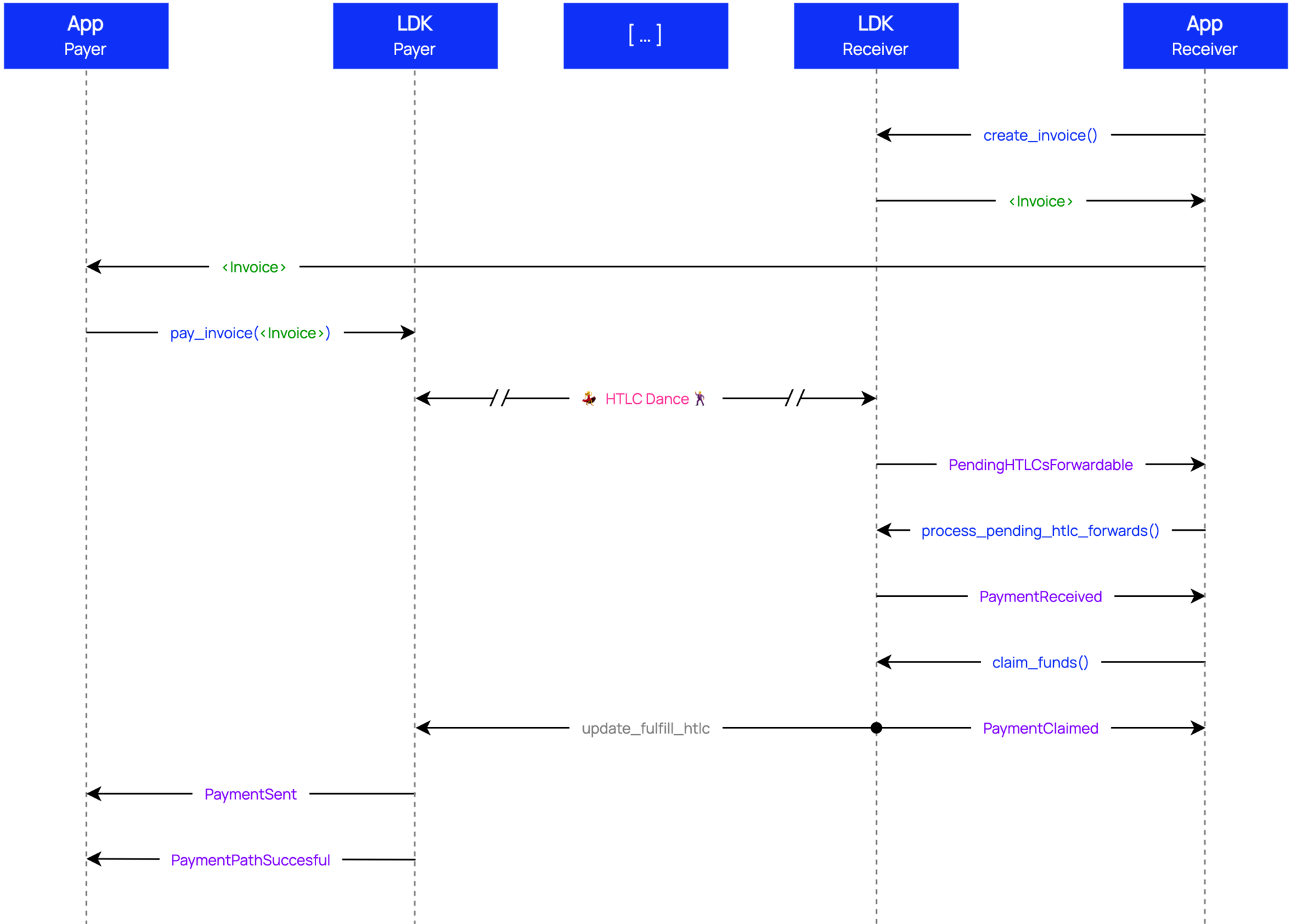
Event Handling: Inbound

- OpenChannelRequest
- DiscardFunding
- ChannelClosed
- SpendableOutputs



Payments

- Create & pay invoices
- Requirements
  - Scorer
  - Router
  - Payer
  - Event handling


App
Payer
LDK
Payer
[...]
LDK
Receiver
App
Receiver
create_invoice()
‹Invoice›
‹Invoice›
pay_invoice(‹Invoice›)
HTLC Dance
PendingHTLCsForwardable
process_pending_htlc_forwards()
PaymentReceived
claim_funds()
update_fulfill_htlc
PaymentClaimed
PaymentSent
PaymentPathSuccesful

Payments

Event Handling

- PendingHTLCsForwardable
- PaymentReceived, PaymentClaimed
- PaymentSent, PaymentFailed
- PaymentPathSuccessful, PaymentPathFailed

Payments

Gossip

- Choose how to obtain:
  - P2PGossipSync
  - RapidGossipSync
- Necessary for source-based pathfinding
- Not (strictly) necessary for invoice payment

Postscriptum

- Ubiquitous requirements
  - Logger
  - KeysManager
  - Event handling
- RapidGossipSync

- Default implementations:
  - BackgroundProcessor
  - ChainMonitor
  - P2PGossipSync
  - lightning-net-tokio

Inspiration

- Watchtower
- Multi-home backup
- Probing
- Gossip monitor
- Custom channel scoring

Weeds: PeerManager

- Socket: `trait SocketDescriptor`
- Socket open methods (called by user):
  - `fn new_outbound_connection(&self, their_node_id: PublicKey, descriptor: Descriptor, remote_network_address: Option<NetAddress>) -> Result<Vec<u8>, PeerHandleError>`
  - `fn new_inbound_connection(&self, descriptor: Descriptor, remote_network_address: Option<NetAddress>) -> Result<(), PeerHandleError>`
- Periodic calls (made by user):
  - `fn process_events(&self)`
  - `fn timer_tick_occurred(&self)`

Weeds: Chain Data

Full blocks

- Trait (pre-implemented): `trait Listen`
- Methods (called by user):
  - `fn block_connected(&self, block: &Block, height: u32)`
  - `fn block_disconnected(&self, header: &BlockHeader, height: u32)`

Weeds: Chain Data

Transactions

- Trait (pre-implemented): `trait Confirm`
- Methods (called by user):
  - `fn transactions_confirmed(&self, header: &BlockHeader, txdata: &TransactionData, height: u32)`
  - `fn transaction_unconfirmed(&self, txid: &Txid)`
  - `fn best_block_updated(&self, header: &BlockHeader, height: u32)`

Weeds: Chain Data

Transactions

- Trait (user-implemented): `trait Filter`
- Methods (called by LDK):
  - `fn register_tx(&self, txid: &Txid, script_pubkey: &Script)`
  - `fn register_output(&self, output: WatchedOutput) -> Option<(usize, Transaction)>`

Weeds: Chain Data

- Watchtowers: `trait Watch<ChannelSigner: Sign>`
- Alternatively, use `ChainMonitor<ChannelSigner: Sign, C: Deref, T: Deref, F: Deref, L: Deref, P: Deref>` remotely

Weeds: Persister

- Not to be confused with `Persist`!
- Responsibilities: `ChannelManager`, `NetworkGraph`, `Scorer`
- Trait (user-implemented): `trait Persister<'a, Signer: Sign, M: Deref, T: Deref, K: Deref, F: Deref, L: Deref, S>`
- Methods (called by LDK):
  - `fn persist_manager(&self, channel_manager: &ChannelManager<Signer, M, T, K, F, L>) -> Result<(), io::Error>`
  - `fn persist_graph(&self, network_graph: &NetworkGraph) -> Result<(), io::Error>`
  - `fn persist_scorer(&self, scorer: &S) -> Result<(), io::Error>`

Weeds: Persist

- Not to be confused with Persister!
- Responsibility: each and every `ChannelMonitor`
- Methods (called by LDK):
  - ```
    fn persist_new_channel(&self, channel_id: OutPoint, data:
    &ChannelMonitor<ChannelSigner>, update_id: MonitorUpdateId)
    -> Result<(), ChannelMonitorUpdateErr>
    ```
  - ```
    fn update_persisted_channel(&self, channel_id: OutPoint,
    update: &Option<ChannelMonitorUpdate>, data:
    &ChannelMonitor<ChannelSigner>, update_id: MonitorUpdateId)
    -> Result<(), ChannelMonitorUpdateErr>
    ```

Weeds: Fee Estimation

- Trait (user-implemented): `trait FeeEstimator`
- Method (called by LDK):
  - `fn get_est_sat_per_1000_weight(&self, confirmation_target: ConfirmationTarget) -> u32`

Weeds: Transaction Broadcasting

- Trait (user-implemented): `trait BroadcasterInterface`
- Method:
  - `fn broadcast_transaction(&self, tx: &Transaction)`

Weeds: Logging

- Trait (user-implemented): `trait Logger`
- Method (called by LDK):
  - `fn log(&self, record: &Record)`

Help us improve LDK.
Apply for a grant. Or don’t.