Running LDK on mobile ⚡

Ололо пыщь пыщь

Igor Korsakov / bluewallet.io / London / March, 2022

blue wallet

What is LDK?

Rust-lightning distributed in a developer-friendly packaging, developed by Spiral (ex Square-crypto)

blue wallet

LDK is written in Rust

Can cross-compile into different langs:

- Java JAR (desktop, server): https://github.com/lightningdevkit/ldk-garbagecollected
- Java AAR (android mobile): https://github.com/lightningdevkit/ldk-garbagecollected
- Swift .xcframework (MacOS desktop, iOS mobile): https://github.com/lightningdevkit/ldk-swift
- WASM (nodejs server, browser): https://www.npmjs.com/package/lightningdevkit

blue wallet

LDK is not a node, nor a shippable software

- Nuts and bolts: import LDK binaries in your project
- Write boilerplate code to make it do what YOU want!

blue wallet

LDK included batteries:

- Networking

LDK bring your own:

- Feed blockchain data to LDK (including reorgs), either per-transaction or per block; also current feerate
- Disk persistence
- Broadcast txhex yourself
- General-purpose bitcoin library (address manipulation, tx creation)

blue wallet

LDK objects

- ChannelManager
- ChannelMonitor (1 per channel)
- KeysManager -> provide entropy
- Persister (channels) / persister (channel manager) -> write implementation
- Filter -> write implementation
- Broadcaster -> write implementation
- Logger -> write implementation
- FeeEstimator -> write implementation
- ChainMonitor
- UserConfig & ChannelHandshakeConfig
- NetworkGraph & Scorer (optional)
- PeerManager
- PeerHandler (finally!)

blue wallet

LDK example (for a taste)

```
// INITIALIZE THE KEYSMANAGER ##################################################################
// What it's used for: providing keys for signing lightning transactions
keys_manager = KeysManager.of(
    hexStringToByteArray(entropyHex),
    starting_time_secs: System.currentTimeMillis() / 1000,
    (System.currentTimeMillis() * 1000).toInt()
)
```

blue wallet

LDK example (for a taste)

```
val channel_manager_persister = object : ChannelManagerConstructor.EventHandler {
    override fun handle_event(event: Event) {
        handleEvent(event);
    }

    override fun persist_manager(channel_manager_bytes: ByteArray?) {
        println("persist_manager");
        if (channel_manager_bytes != null) {
            File( pathname: "$homedir/$prefix_channel_manager.hex").writeText(byteArrayToHex(channel_manager_bytes));
        }
    }
}
```

blue wallet

LDK example (for a taste)

```
// loading from disk
channel_manager_constructor = ChannelManagerConstructor(
    hexStringToByteArray(serializedChannelManagerHex),
    channelMonitors,
    uc,
    keys_manager?.as_KeysInterface(),
    fee_estimator,
    chain_monitor,
    tx_filter,
    router,
    tx_broadcaster,
    logger
);
channel_manager = channel_manager_constructor!!.channel_manager;
channel_manager_constructor!!.chain_sync_completed(channel_manager_persister, scorer);
peer_manager = channel_manager_constructor!!.peer_manager;
nio_peer_handler = channel_manager_constructor!!.nio_peer_handler;
```

blue wallet

LDK: feed blockchain data

- Full blocks
- Specific transactions (electrum-style):

```
val tx_filter: Filter? = Filter.new_impl(object : FilterInterface {
  override fun register_tx(txid: ByteArray, script_pubkey: ByteArray) {
    println("ReactNativeLDK: register_tx");
    val params = Arguments.createMap()
    params.putString("txid", byteArrayToHex(txid))
    params.putString("script_pubkey", byteArrayToHex(script_pubkey))
    that.sendEvent(MARKER_REGISTER_TX, params);
  }

  override fun register_output(output: WatchedOutput): Option_C2Tuple_usizeTransactionZZ {
    println("ReactNativeLDK: register_output");
    val params = Arguments.createMap()
    val blockHash = output._block_hash;
    if (blockHash is ByteArray) {
      params.putString("block_hash", byteArrayToHex(blockHash))
    }
    params.putString("index", output._outpoint._index.toString())
    params.putString("script_pubkey", byteArrayToHex(output._script_pubkey))
    that.sendEvent(MARKER_REGISTER_OUTPUT, params);
    return Option_C2Tuple_usizeTransactionZZ.none();
  }
})
```

blue wallet

LDK: feed blockchain data

- Full blocks
- Specific transactions (electrum-style):

```
@ReactMethod
fun transactionConfirmed(headerHex: String, height: Int, txPos: Int, transactionHex: String, promise: Promise) {
  val tx = TwoTuple_usizeTransactionZ.of(txPos.toLong(), hexStringToByteArray(transactionHex))
  val txarray = arrayOf(tx);
  channel_manager?.as_Confirm()?.transactions_confirmed(hexStringToByteArray(headerHex), txarray, height);
  chain_monitor?.as_Confirm()?.transactions_confirmed(hexStringToByteArray(headerHex), txarray, height);

  promise.resolve(true);
}

@ReactMethod
fun transactionUnconfirmed(txidHex: String, promise: Promise) {
  channel_manager?.as_Confirm()?.transaction_unconfirmed(hexStringToByteArray(txidHex));
  chain_monitor?.as_Confirm()?.transaction_unconfirmed(hexStringToByteArray(txidHex));
  promise.resolve(true);
}
```

blue wallet

LDK: open channel: pure PSBT workflow

- Initiate (dont forget to connect to a peer):

```
val create_channel_result = channel_manager?.create_channel(
    peer_node_pubkey, channelValue.toLong(), push_msat: 0, user_channel_id: 42, uc
);
```

- Catch **FundingGenerationReady** event:

```
if (event is Event.FundingGenerationReady) {
    println("ReactNativeLDK: " + "FundingGenerationReady");
    val funding_spk = event.output_script;
    if (funding_spk.size == 34 && funding_spk[0].toInt() == 0 && funding_spk[1].toInt() == 32) {
        val params = WritableMap()
        params.putString( var1: "channel_value_satoshis", event.channel_value_satoshis.toString());
        params.putString( var1: "output_script", byteArrayToHex(event.output_script));
        params.putString( var1: "temporary_channel_id", byteArrayToHex(event.temporary_channel_id));
        params.putString( var1: "user_channel_id", event.user_channel_id.toString());
        temporary_channel_id = event.temporary_channel_id;
        storeEvent( eventsPath: "$homedir/events_funding_generation_ready", params)
        eventsFundingGenerationReady = eventsFundingGenerationReady.plus(params.toString())
    }
}
```

blue wallet

LDK: open channel: pure PSBT workflow

- Provide TXHEX to LDK:

```
val funding_res = channel_manager?.funding_transaction_generated(temporary_channel_id, hexStringToByteArray(txhex));
```

- Wait for your channel to appear:

```
val channels = channel_manager?.list_channels();
```

blue wallet

LDK: disk usage

- channe_manager: 5.5 kB
- channel_monitor: 8.78 kB (5 payments)
- Each new payment adds **< 200 bytes** of data
- Graph (optional): 80+ Mb

blue wallet

LDK: TODO to use LDK

- Choose your storage mechanism
- Choose if you're feeding blocks or transactions to LDK and source that data
- Put together (or copypaste) low-level boilerplate code
- Write event handlers (or pass events to higher-level code)
- Decide what to do with channels

LDK:

- Low-level boilerplate code: 1k LOC
- Higher-level code: 1k LOC

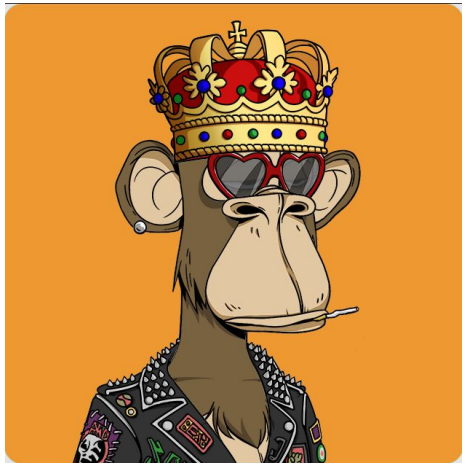
blue wallet

Result:

- https://github.com/BlueWallet/rn-ldk redy-to-use react-native lib (iOS/macOS(catalyst)/Android)
- https://github.com/BlueWallet/HelloLightning full node (very raw)

blue wallet

free NFT



please take a picture

“Not great, just ok”

“Only one coin”

“Just another wallet”

blue wallet

“Could be better”

i@bluewallet.io

“Meh”

“Some features”

follow me on twitter: @overtorment