# Interledger Web

A pragmatic ILP stack we can use today

Adrian Hope-Bailie

# Agenda

- Background and Motivations
- The Internet Hourglass-Architecture
- Getting Started with ILP
- Challenges with the current stack
- OPay (OAuth + Payments)
- Loopback Transport

# Interledger is AWESOME!

# Background and Motivations

Payments are regulated differently and therefor operate differently to data.
Let’s accept that fact, or not…

**Interledger is either going to internetwork existing payments networks, or be a new parallel payments network.**

Do we all have the same **vision** for Interledger?

# Background and Motivations

What have we *(Coil team in 🇿🇦)* learned over the last year:

- Building on the edges of an ILP network is unnecessarily HARD
- We thought it was ilp-connector that was the problem, it’s not
- ILP itself is simple to grok. But to use it you need to grok a complex stack that uses obscure technologies (and has unstable implementations)
- Hiding the complexity in library code is not a solution. The stack is not mature or well-documented enough for anyone to blindly trust the reference implementations (some of which are incomplete)

## Background and Motivations

What have we *(Coil team in 🇿🇦)* learned over the last year:

- Users can’t fix bugs themselves. We see the same requests for support on the forums and Slack over and over and over
- Using ILP from within a browser/cURL is a terrible experience (close to impossible without lots of tooling)

**ILP Enthusiast :** *[ Explains how ILP works ]*

**Interested Party :** *“Sounds cool, how do I use it?”*

**ILP Enthusiast :** *[ Explain the stack that’s required to actually do something ]*

# Background and Motivations

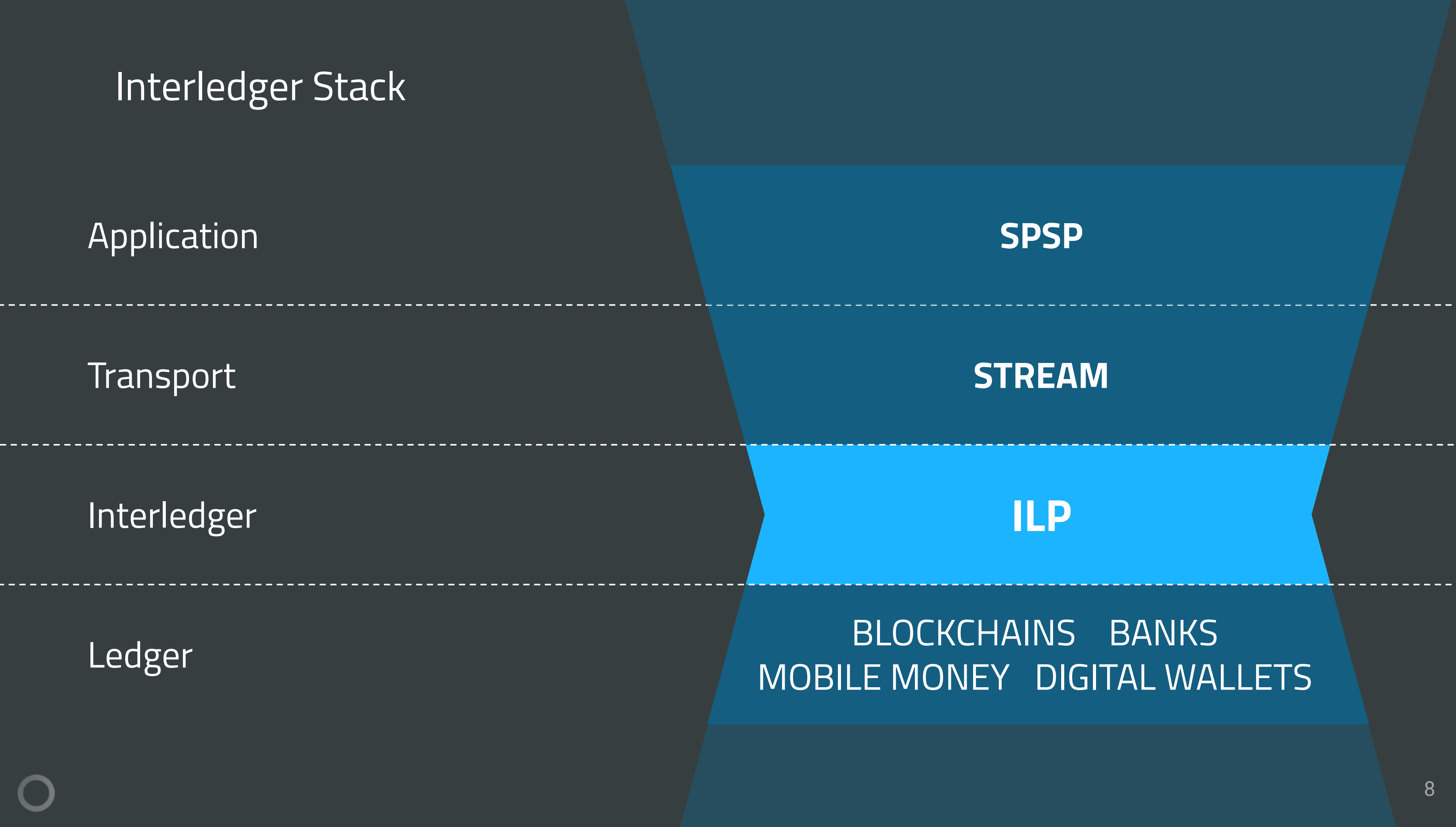
The current architecture is premised on the assumption that **one day ILP will be a core protocol in every platform** (like IP is for data) however, we are likely 20 years away from that, at least...

… but Interledger is usable (and useful) TODAY.  We need to focus on the things that make it AWESOME and spend less time trying to recreate the Internet:

**ILP Addresses:** A universal addressing scheme for value transfers

**Conditional Execution:** End-to-end transaction integrity over different networks

# Interledger Stack



Application — SPSP

Transport — STREAM

Interledger — ILP

Ledger — BLOCKCHAINS BANKS MOBILE MONEY DIGITAL WALLETS

# Interledger Stack (actually only has two layers)

| | | |
|---|---|---|
| ~~Application~~ Setup | No encapsulation. NOT a layer in a protocol stack | SPSP |
| ~~Transport~~ Payment | Business logic of a payment. Protocols encapsulated in ILP | STREAM |
| Interledger | | **ILP** |
| Ledger | No encapsulation. NOT a layer in a protocol stack | BLOCKCHAINS BANKS<br>MOBILE MONEY DIGITAL WALLETS |

# Interledger was inspired by the Internet...

| Interledger | | Internet | |
|---|---|---|---|
| **Type/Version** | `12 (Prepare)` | **Version** | `4` |
| **Address** | `g.wallet.alice.28y76fe` | **Address** | `127.0.0.1` |
| **Amount** | `1500` | | |
| **Expiry** | `20191001235959001` | **TTL** | `12` |
| **Condition** | `0x7AF57908BC6E...` | | |
| **Data** | `0x219CF5708C63...` | **Data** | `0x219CF5708C63...` |

# …but ended up looking exactly like a payments protocol

| Interledger | | ISO 8583 | |
|---|---|---|---|
| **Type/Version** | `12 (Prepare)` | **MTI** | `0200` |
| **Address** | `g.wallet.alice.28y76fe` | **PAN** | `4567 8910 8765 2345` |
| **Amount** | `1500` | **Amount** | `1500` |
| **Expiry** | `20191001235959001` | **Date/Time** | `20191001235959` |
| **Condition** | `0x7AF57908BC6E...` | | |
| **Data** | `0x219CF5708C63...` | **Other Data** | `...` |

# Interledger Stack

ILP is actually just an **upgrade of ISO 8583** (card payment protocols)

- ISO8583 is also agnostic of underlying settlement infrastructure
- ISO8583 also uses a hierarchical address space and routing infrastructure
- ISO8583 also uses request/response message pairs for each payment

ILP improves on ISO8583 by

- Separating use case specific business logic from the payment messages
- Not overloading addresses as identifiers
- Not being bound to a single payment instrument type
- Having a more flexible address space
- Using cryptography to provide certainty of end-to-end delivery

*Interledger is a payments protocol and money is not data*

*The Internet analogy is a stretch...*

# Getting Started (Current)

Front page of interledger.org

- 7 steps to send a payment (all shell commands, no code)
- Installation of 4 separate pieces of software
- Implicit dependencies (localtunnel) and other nuanced points of failure

End Result for the User:

- Sent a payment inside a closed local network (proved nothing)
- No concept of uplinks or other “realistic” topologies (learned nothing)
- No concept of settlement

# Getting Started (Simpler)

1. Create a test account on a test network with a prepaid balance
2. Use credentials to send a payment from inside a browser (or with cURL)
3. See balance change on account

```
const reply = await fetch('https://testnet.example/', {
  method: 'POST',
  headers: {
    'Authorization': 'Bearer 54ABCpj8HBSax1TImW+5JCeuQeRkm5NMpJWZG3hRuBG=',
    'ILP-Destination': 'test.alice',
    'ILP-Amount': '10',
    'ILP-Condition': '47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU=',
    'ILP-Expiry': '20190902235959999'
  }
})
if(reply.headers['ILP-Fulfilment']) {
  console.log("Payment sent!")
}
```

# Challenges with the Current Stack

**Simple** Payment Setup Protocol is too… simple.

- No way for entities to exchange **identities**
- Poor support for **alternative use cases**

  (e.g. pull payments)
- **Security** through obscurity

  (hard to guess URLs for “secured” payments)

# Challenges with the Current Stack

On the other hand STREAM is overly complex

- Connection establishment is **SLOOOOOW** (too many round-trips)
- Unnecessary **bi-directional payments** (no use cases)
- Unnecessary **multiplexing** of payments/data (no use cases)
- **Stateful** connections are hard to scale

STREAM is very clever but supports use cases that **don't exist yet**

(e.g. Java and Rust implementations don't even implement the full protocol)

Web Monetization **could** be done using a much simpler protocol

(that is implementable in the browser using just HTTP + JSON and optionally WS)

# Challenges with the Current Stack

**Octet Encoding Rules** are obscure

- Even the most seasoned Internet engineers have never heard of OER
- It is a massive barrier to understanding for newcomers to ILP
- The value of a canonical encoding has been deprecated

  (We previously generated the condition/fulfillment from the packet)
- No native support in ANY stack
- Limited and expensive tooling for both ASN.1 and OER
- Terrible developer experience

# Interledger Web

| | |
|---|---|
| Setup | **OPay** SPSP |
| Payment | **Loopback** **PSKv3** STREAM |
| Interledger | **ILP** |
| Ledger | Blockchains Banks<br>Mobile Money Digital Wallets |

# Payments

The flow of a payment is pretty standard irrespective of network or instrument

**Discovery :** Find the issuer of the payment instrument

**Setup:** Negotiate the terms of the payment with the issuer

**Authorization:** Get authorization of the payment from the account holder

**Clearing:** Reconcile and clear the payment among participants

**Settlement:** Settle obligations created during clearing

# Payments

## Cards

**Discovery :**

**Setup:**

**Authorization:**

For Card Payments these phases are completed together

Card Numbers are both an **Identifier** and an **Address**

**Clearing:**

**Settlement:**

Proprietary CLEARING and Settlement Systems owned by Networks

# Payments

## Digital Wallets

**Discovery :** Uses digital Identity or QR codes (usually network specific)

**Setup:**

**Authorization:** Proprietary Closed Networks (sometimes using Card Rails)

**Clearing:**

**Settlement:** Via Bank or Card Networks

# Payments

## Open Banking

**Discovery :** Done by AISP or PISP (often requires user selection)

**Setup:**

**Authorization:**

} OAuth 2.0 using FAPI profiles

**Clearing:** Bank APIs (numerous "standards")

**Settlement:** SWIFT/ISO 20022

# Payments

## Interledger

| | |
|---|---|
| **Discovery :** | Payment Pointers |
| **Setup:** | OAuth 2.0 using FAPI profiles |
| **Authorization:** | |
| **Clearing:** | Interledger (STREAM, Loopback or PSKv3) |
| **Settlement:** | Dealt with bilaterally |

# OPay - Payments on top of OAuth

Payments are inextricably bound to identity

↓

OAuth is the de-facto authN and authZ protocol for the Web

+

Open Banking uses OAuth and is doing the work to standardize and harden the protocols

↓

**Use OAuth 2.0 to setup Interledger payments**

# OPay - Payments on top of OAuth

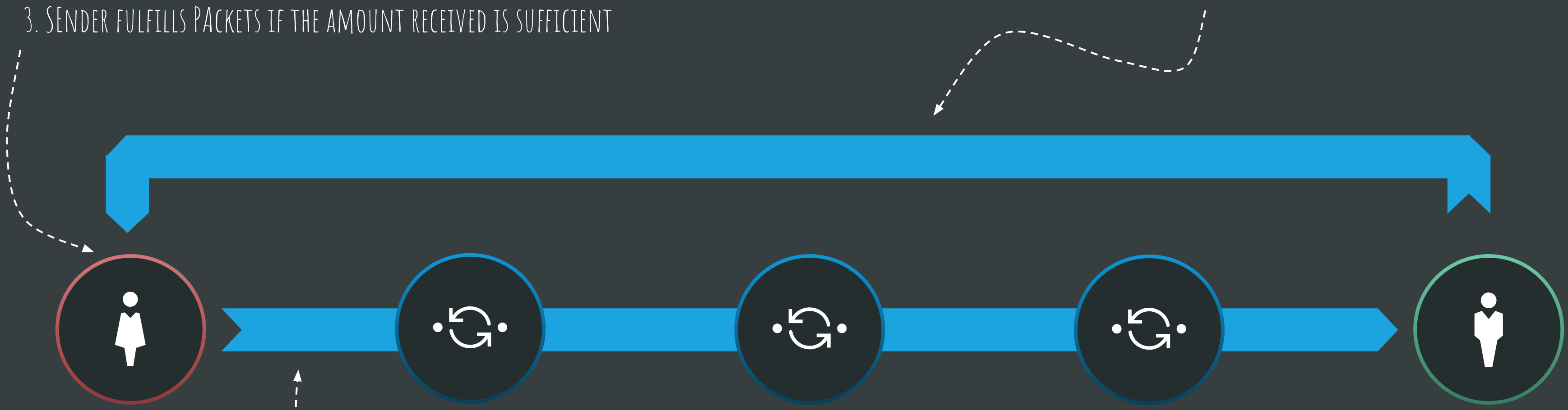
- OAuth 2.0 and OpenID Connect provide a framework for identity exchange
- Specifics of who shares identity and what is shared can be determined by parties to the payment and driven by the use case
- Using auth delegation (access tokens granted through OAuth 2.0) is a secure mechanism for providing third-parties limited access to an account
- Facilitates difficult use cases such as pull payments, debit mandates, subscriptions etc.

# OPay - Payments on top of OAuth

Flow:

1. Relying Party (payer/payee) discovers counterparty wallet (IdP) using Payment Pointer
2. RP creates payment intent/mandate on IdP
3. RP requests authZ of intent/mandate from counterparty via wallet (Either party may also request authZ to share identity data if required)
4. Using access token granted during authZ, RP connects to wallet
5. RP sends/receives packets from/to counterparty account to complete payment described by intent/mandate

# Loopback (Push Payments)

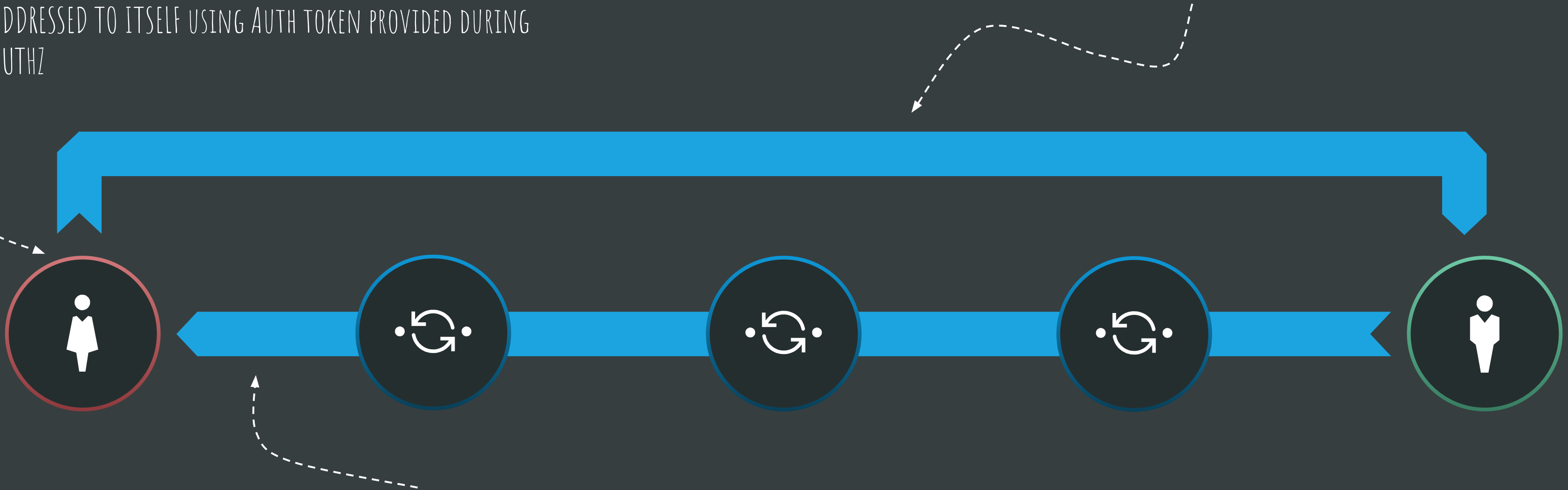


2. A temporary bilateral link adds a final hop from the receiver to the sender

3. Sender fulfills Packets if the amount received is sufficient

1. Sender sends prepare packets via their normal uplink.

# Loopback (Pull Payments)



1. Receiver sends PRepare Packets out of Sender's ACCOUNT ADDRESSED TO ITSELF using Auth token provided during AUTHZ

2. A temporary bilateral link is the first hop from the SENDER to the RECEIVER

3. RECEIVER gets prepare packets via their normal uplink and fulfills or rejects them

# PSKv3 - Simple STREAM

Re-use the best bits of STREAM

1. Auto-fulfilled packets using shared Key
2. End-to-End encryption
3. Minimum Acceptable Amount
4. Credit-based Flow Control

Simplify

1. Drop multiplexing
2. Drop bi-directionality

# Conclusions

ILP is a payment protocol

It is more like ISO8583 than IP (but much much better)

There is a HUGE opportunity to provide a universal clearing protocol for EXISTING payment networks

We should engage infrastructure providers and developers differently and provide end-to-end protocols that are **developer** friendly

We need end-to-end protocols that address common use cases (not just Web Monetization)

# Questions?