mojaloop

Cross Network Mojaloop

Sending payments between Mojaloop systems

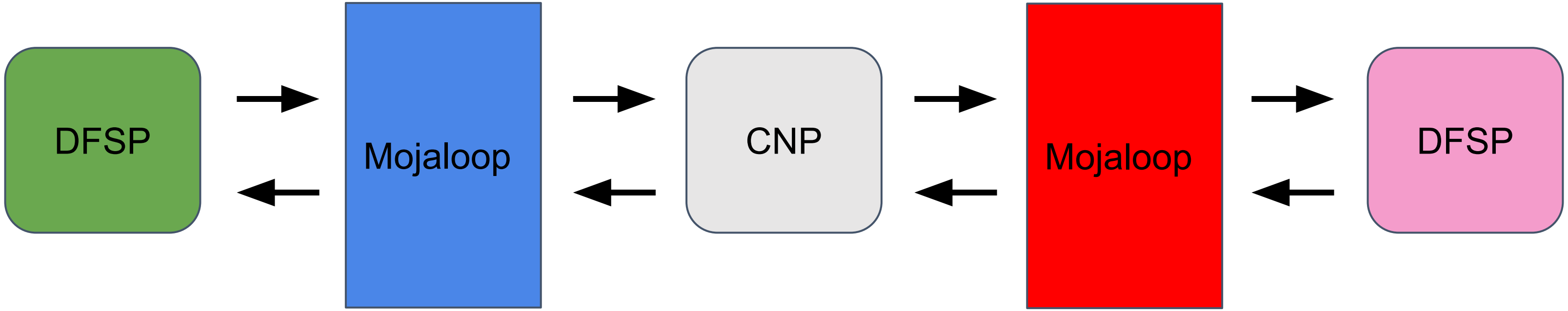
Proof of Concept - Part 1

Agenda

- Goals and Part 1 Scope
- Ecosystem
- Design Decisions
- Current Design
- Caveats
- How It Works
- Note on In-Country FX
- Community Contributions
- Proposed API changes
- Next Steps

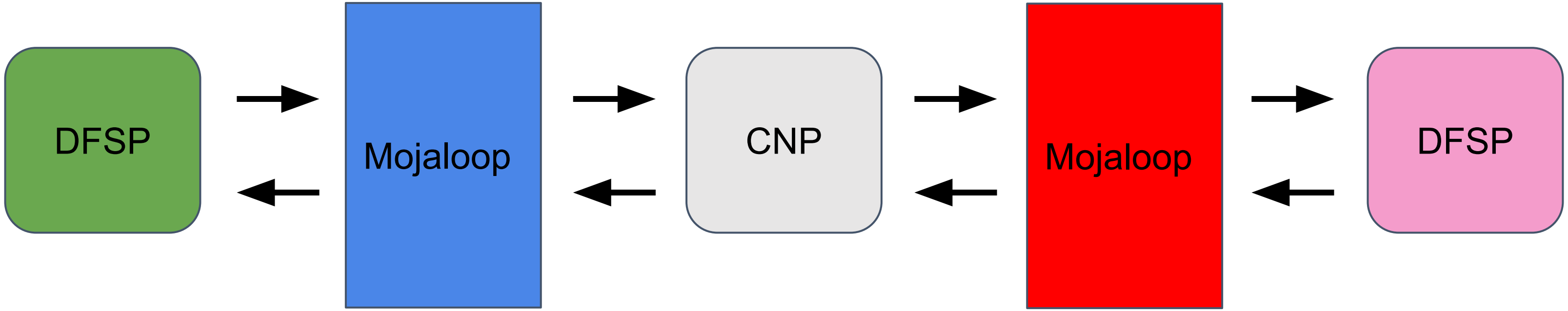
Goals of the POC

Demonstrate that a payment can be sent from a DFSP on one Mojaloop network to a DFSP on another Mojaloop network using a special DFSP (a **cross-network provider**) as a gateway.

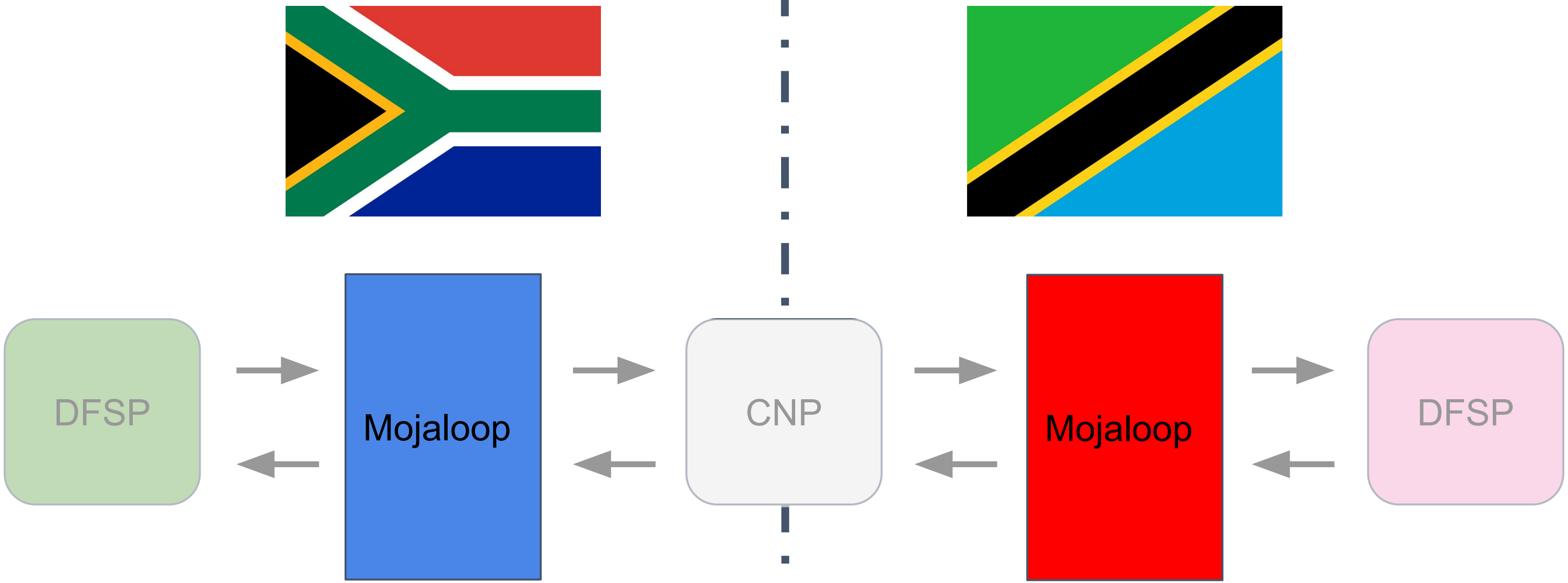


Part 1 Scope

1. Lookup payee (Moja Address)
2. Send a cross-network quote
3. Send a cross-network transfer (single currency)



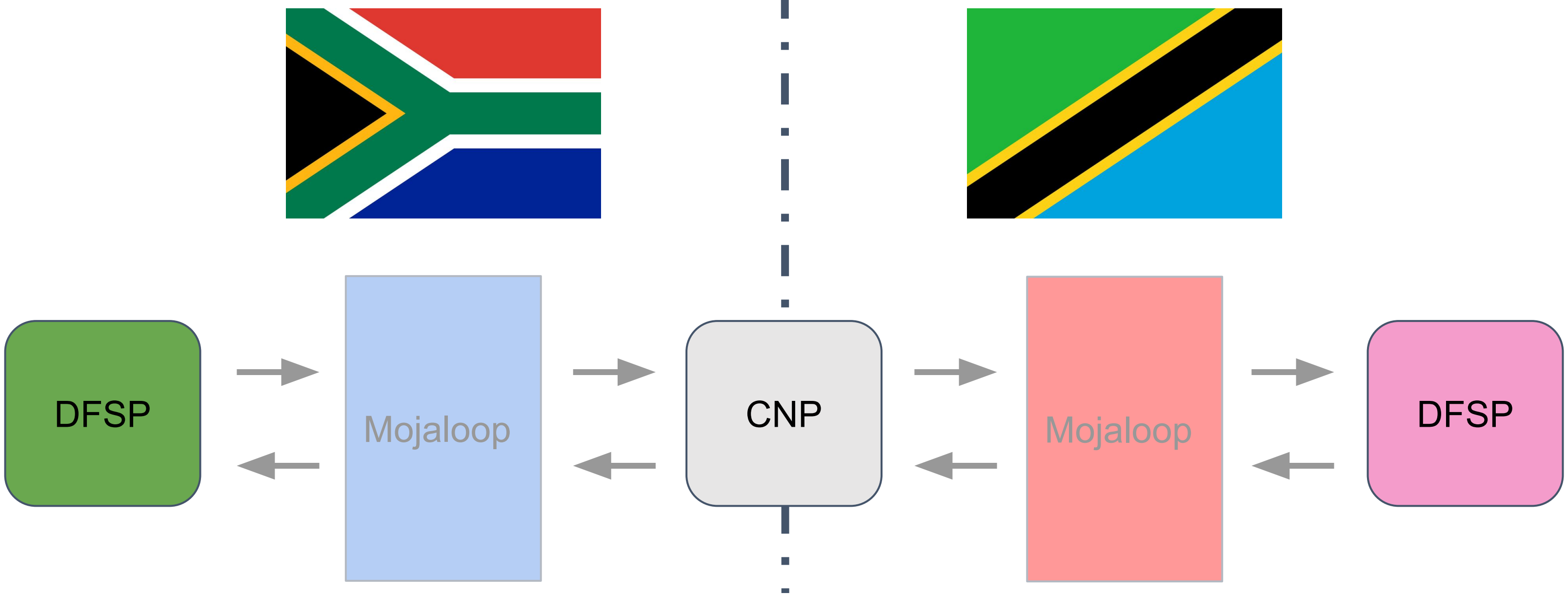
Ecosystem



Blue Moja
Domestic Mobile Money in South Africa

Red Moja
Domestic Mobile Money in Tanzania

Ecosystem



Green Mobile
DFSP in South Africa

Super Remit
Remittance provider operating in
South Africa and Tanzania

Pink Mobile
DFSP in Tanzania

Design Decisions

- No changes at the DFSP
- Use a **cross-network addressing scheme**
- **Routing logic** implemented at the centre

Current Design

- All API calls go via Mojaloop to be routed appropriately
- Interop Switch and Moja API Adaptor consult routing service to set correct destination headers
- CNP is an Interledger Connector with Moja API plugins (API calls mapped to ILP packets internally)

Current Design

- Use an Interledger Protocol-based addressing scheme for DFSPs
- Custom **moja** allocation scheme (Moja Addresses): e.g. `moja.tz.red.tzs.pink`
- Return *Moja Address* as `FspId` during lookup

Caveats

- Endpoint data for *interop-switch-js* (lookup and quote APIs) is stored in Central Ledger’s DB to re-use existing schema. Should be stored locally (in Interop Switch service DB) or accessed via admin API on Central Ledger.

- Routing logic is behind an API endpoint on the Central Ledger. Should be implemented in a stand-alone routing service which exchanges routing data with peers.

How It Works

1. Sending DFSP performs a **lookup** and gets a Moja Address as the value of the *FspId*

```
party: {
  partyIdInfo: {
    partyIdType: 'msisdn',
    partyIdentifier: '255222222222',
    fspId: 'moja.tz.red.tzs.pink'
  }
}
```

How It Works

2. DFSP sends a **quote** to interop-switch-js which resolves a route for the API call (based on `FspId`) and forwards the quote to the CNP

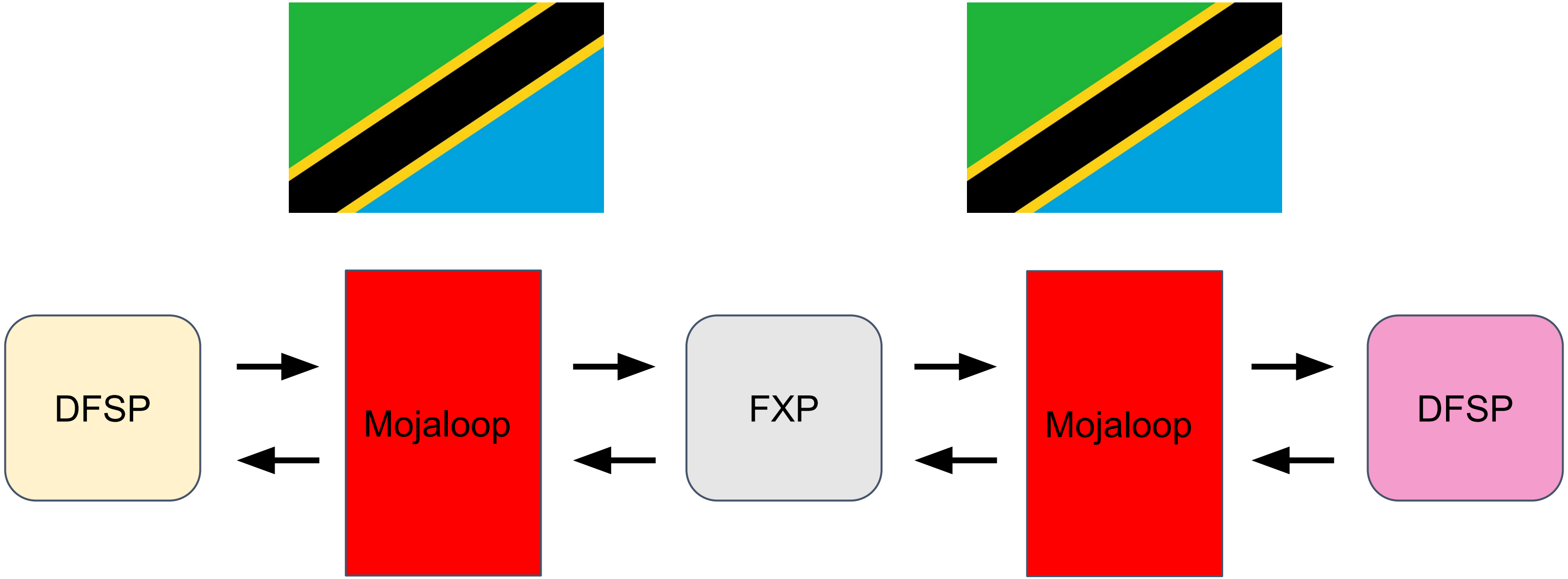
```
POST /transfers HTTP/1.1
Accept: application/vnd.interoperability.quotes...
Content-Type: application/vnd.interoperability.quotes…
Content-Length: 1820
Date: Mon, 28 Jan 2019 10:14:01 GMT
FSPIOP-Source: moja.za.blue.zar.green
FSPIOP-Destination: moja.tz.red.tzs.pink
```

How It Works

3. CNP forwards quote to the payee DFSP via `interop-switch-js` in payee’s network and routes response back

4. DFSP sends a **transfer** to `ml-api-adapter` which resolves the correct in-network participant via a routing API call and puts the transfer on the correct queue on the ledger

Note on In-Country FX



DFSP

Mojaloop

FXP

Mojaloop

DFSP

Yellow Mobile
DFSP in Tanzania

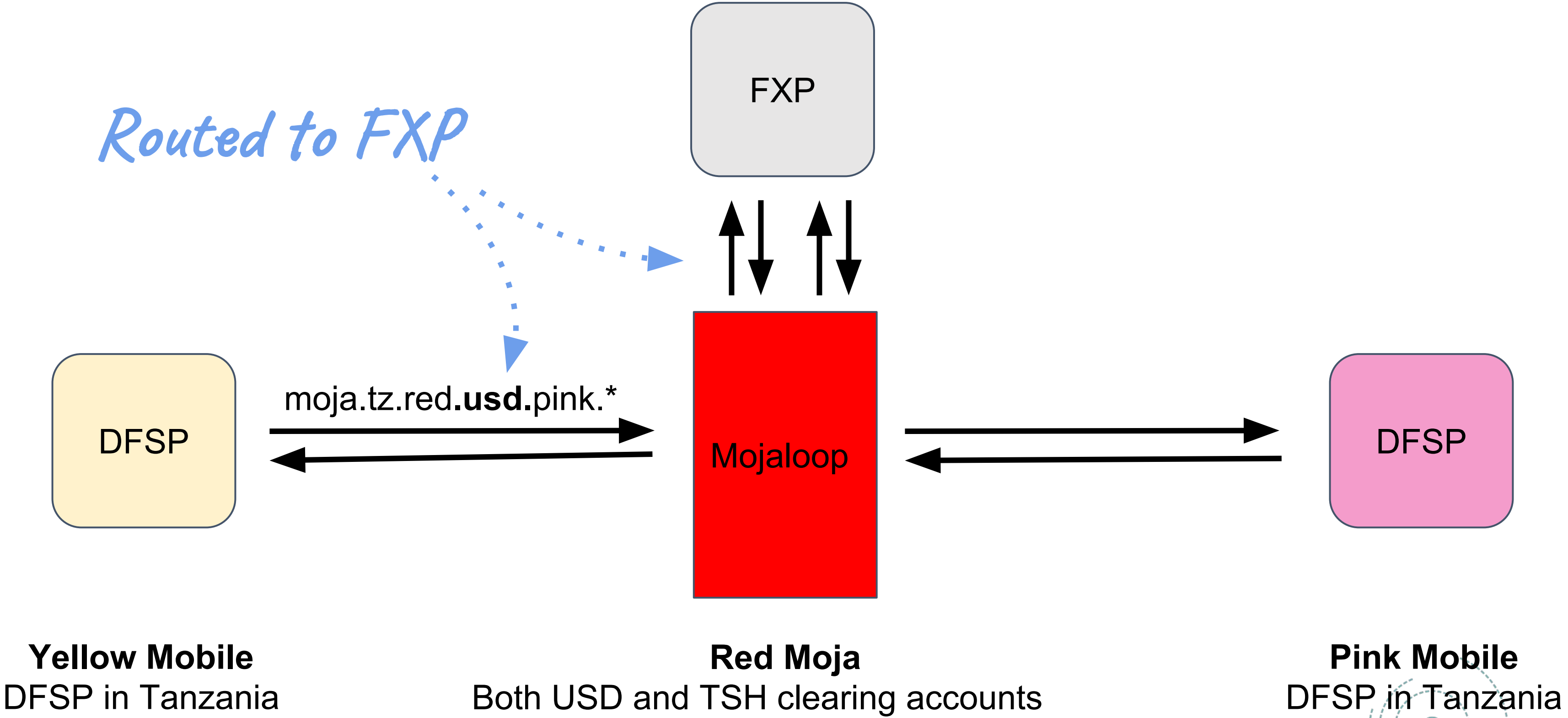
TZS

Super FX
FX provider

USD

Pink Mobile
DFSP in Tanzania

Route to FXP based on address space

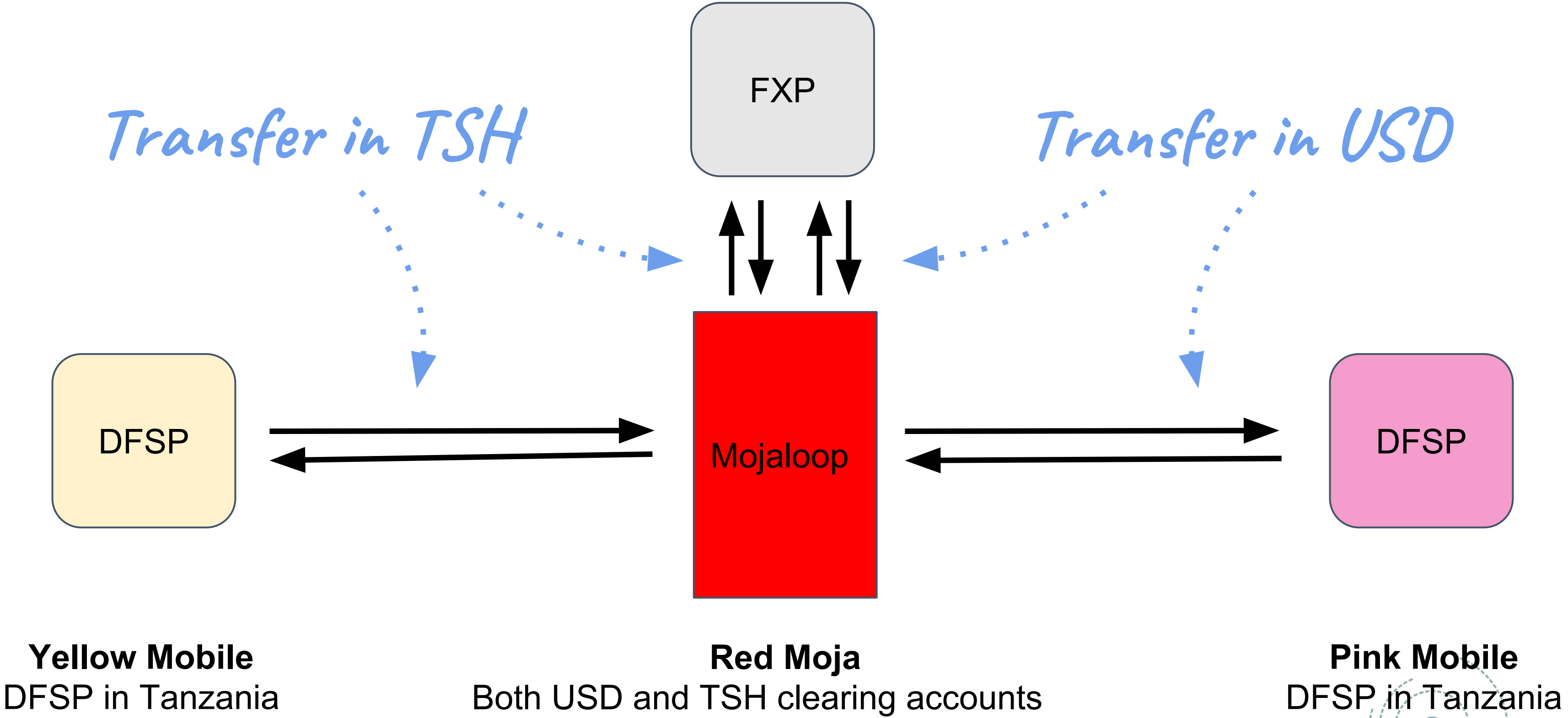


Yellow Mobile
DFSP in Tanzania

Red Moja
Both USD and TSH clearing accounts

Pink Mobile
DFSP in Tanzania

Two transfers

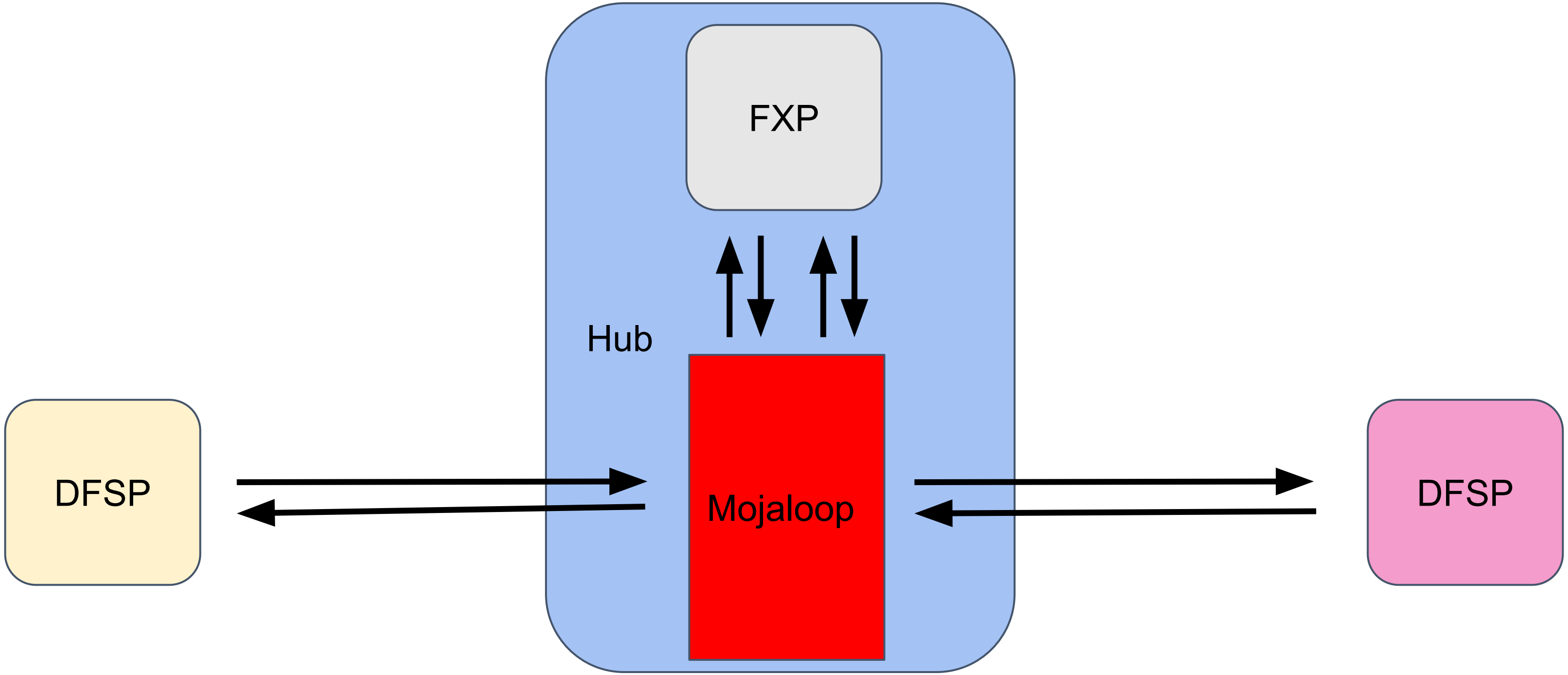


Yellow Mobile
DFSP in Tanzania

Red Moja
Both USD and TSH clearing accounts

Pink Mobile
DFSP in Tanzania

FX at hub



FXP

Hub

Mojaloop

DFSP

DFSP

Yellow Mobile
DFSP in Tanzania

Red Moja
Both USD and TSH clearing accounts

Pink Mobile
DFSP in Tanzania

Community Contributions

Mock DFSP (Deprecated)

- A payee service that responds to API calls as a mock payee DFSP

Interop Switch JS

- A Javascript implementation of the interop-switch

Visualizations (Not yet public)

- End-to-end transaction visualization dashboard

Community Contributions

ILP Plugin Moja

- An ILP connector plugin for Mojaloop API integration (will be contributed to Interledger.js project)

Document deployment on GCloud

- Step by step instructions to deploying Mojaloop stack on Google Cloud

Proposed API Changes (recap)

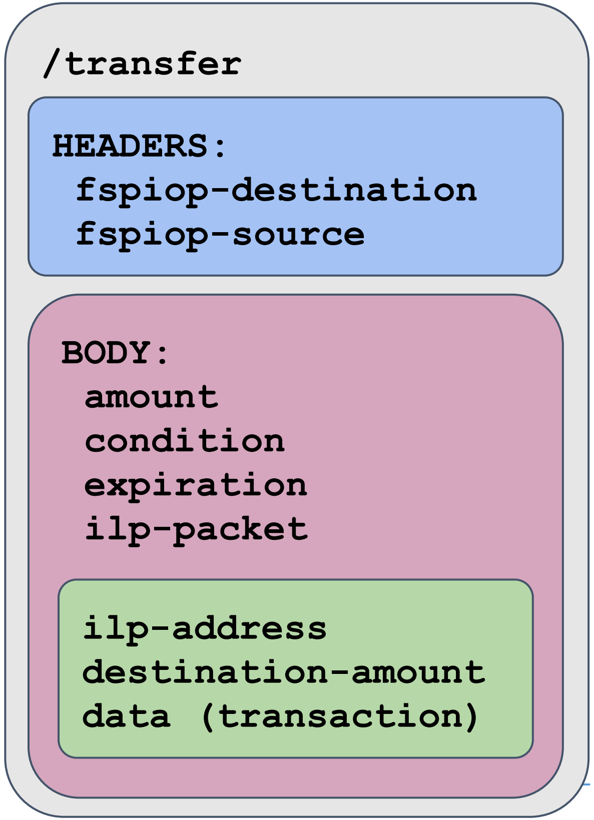
- The Interledger Protocol is a protocol for moving value
- The protocol defines:
  a. a two-phase flow for preparing and committing/aborting a distributed value transfer on any “ledger”
  b. an address space and routing protocol for nodes on the network
  c. a standardised commit signal
- An ILP **transaction** is a chain of **transfers** between nodes
- Mobile money has always used two-phase transfers
- ILP and the Mojaloop API have naturally converged (ILP v1 -> v4)

Proposed API Changes (drivers)

- Desire to align API with changes to ILP introduced between ILPv1 and ILPv4

- Recognition that the `/transfer` API fields already map directly to ILPv4 packet headers
  a. ILP Address (in headers as `FspId`)
  b. Transfer Amount (in `transfer` object)
  c. Condition (in `transfer` object)
  d. Expiry (in `transfer` object)
  e. Data (`transfer.transaction` is the end-to-end ILPv4 payload)

Proposed API Changes

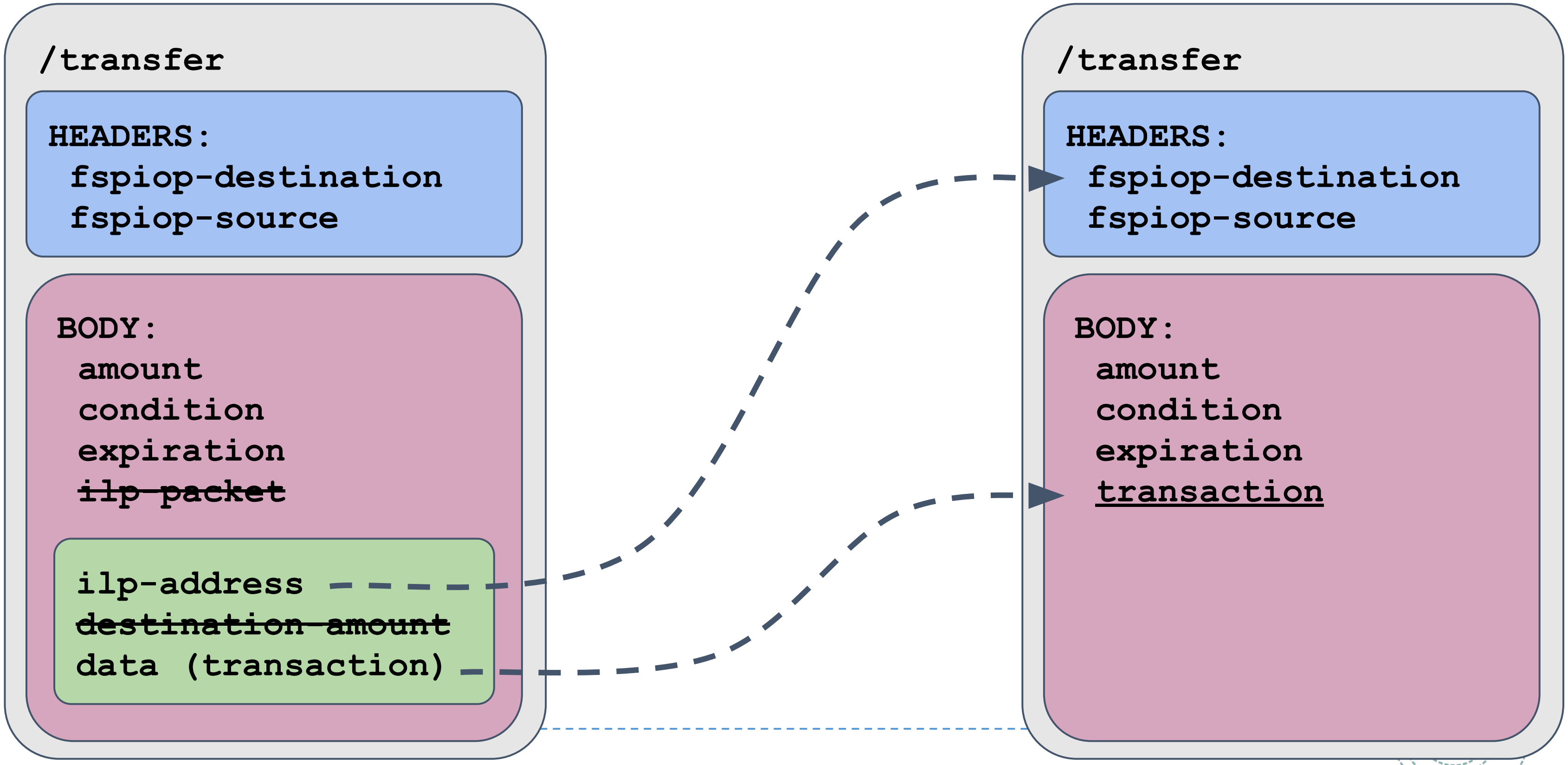


```
/transfer
HEADERS:
  fspiop-destination
  fspiop-source
BODY:
  amount
  condition
  expiration
  ilp-packet
  ilp-address
  destination-amount
  data (transaction)
```

Current API

- OER encoded ILP packet embedded in transfer
- ILP Address in ILP packet headers
- Transaction in ILP packet payload

Proposed API Changes



/transfer

HEADERS:
fspiop-destination
fspiop-source

BODY:
amount
condition
expiration
~~ilp packet~~

ilp-address
~~destination amount~~
data (transaction)

/transfer

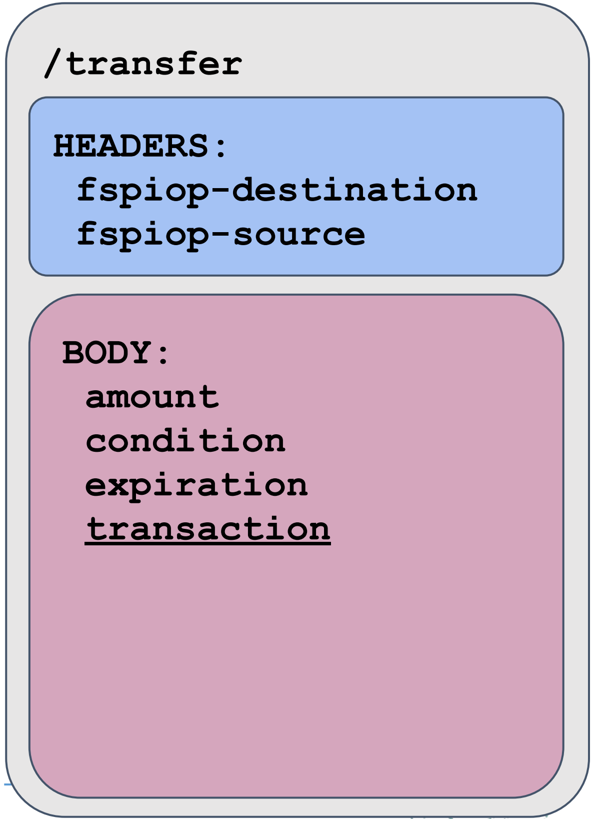
HEADERS:
fspiop-destination
fspiop-source

BODY:
amount
condition
expiration
transaction

Proposed API Changes

New API

- `ilp-address` is now in the transfer headers as `FspId`
- Elevate `transaction` object to be a field in the `transfer` object
- `destination-amount` is no longer required
- `ilp-packet` can be removed from transfer object



```
/transfer

HEADERS:
  fspiop-destination
  fspiop-source

BODY:
  amount
  condition
  expiration
  transaction
```

Proposed API Changes (impact)

- Simpler logic for derivation of `condition` and `fulfillment`
  - `fulfillment = SHA256-HMAC(transaction)`
  - `condition = SHA256(fulfillment)`

- However, logic for verification of fulfillment in Mojaloop stack is **unchanged**

Proposed API Changes (impact)

- Correlation between `/quote` and `/transaction` through `transfer.transaction.transactionId` and `transfer.transaction.quoteId`

- No OER encoding required by DFSPs

- **TODO**: Fully evaluate the risk of byte level changes when JSON encoding/decoding `transaction` object (noting that API calls are already signed and therefore this risk already exists)

Next Steps

- Model both fixed send and fixed receive amounts
- Dynamic routing and exchange of routing data between participants
- Include regulatory data exchange in the quote flow
- Demonstrate more than one CNP in the flow

Next Steps

- Model “multiple CNP” scenarios
  - Sending multiple quotes
  - Quote selection strategies
  - Routing transfers to follow best quote
- Model the “cross-currency, single Mojaloop” scenario
  - Deploy FX provider
  - Configure Mojaloop with multiple currencies

Questions

- Is this what you expected to see?
- How can we work more closely with the rest of the community?
- Are the API changes ready to propose to the CCB?
- How should the addressing scheme integrate with the Central Directory?

https://github.com/mojaloop/**cross-network**