mojaloop

Cross Network Mojaloop

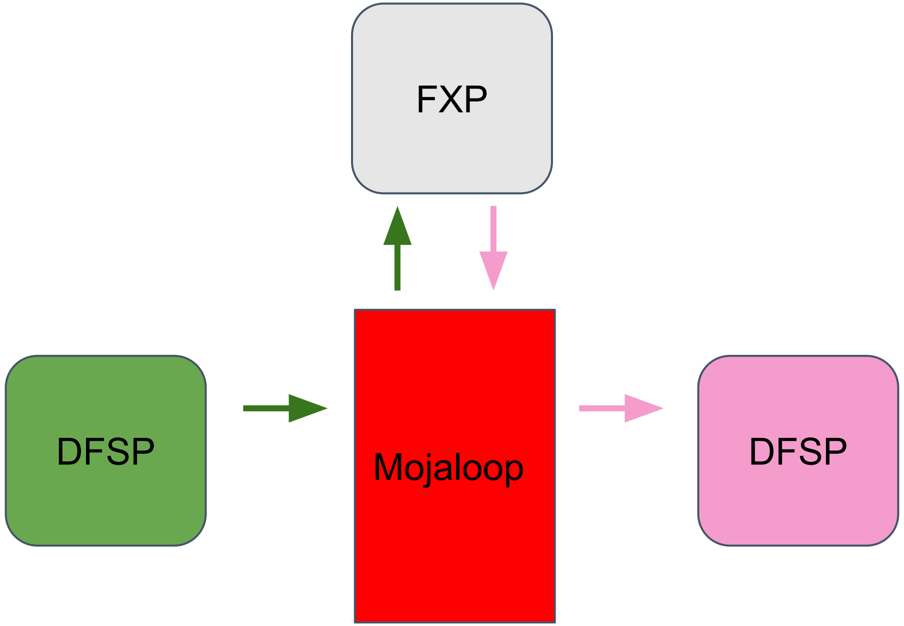
Cross-currency payments in a single Mojaloop systems

Proof of Concept - Part 2

Agenda

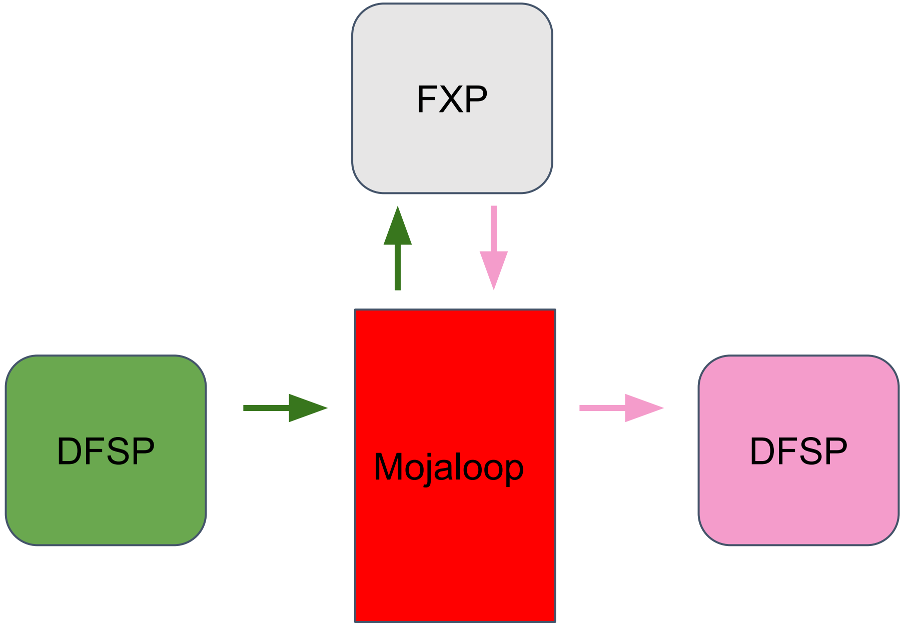
- Goals and Part 2 Scope
- Ecosystem (Part 1 vs Part 2)
- Design Constraints
- Design Decisions
- Current Design
- How It Works
- Community Contributions
- Next Steps

Goals of the POC



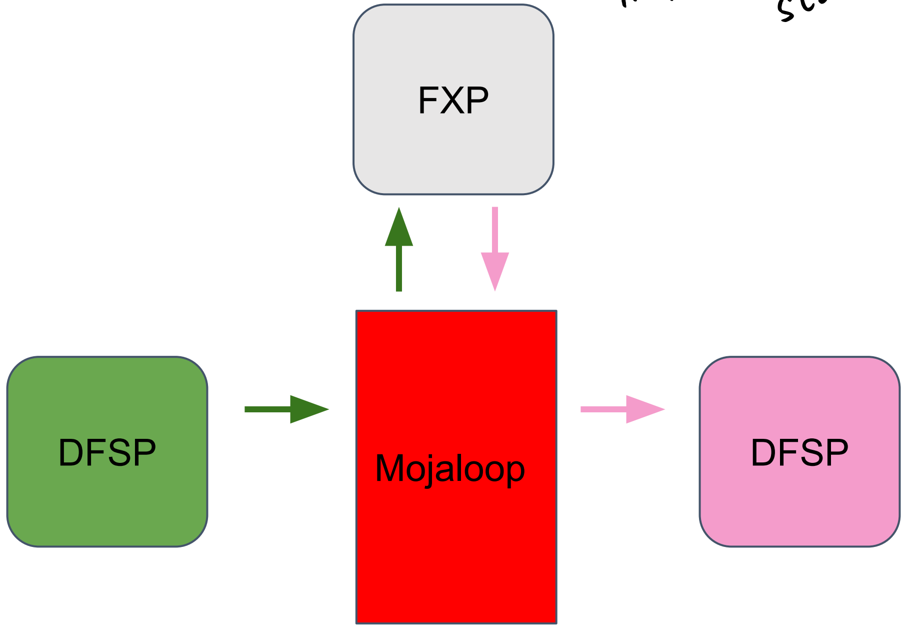
Demonstrate that a cross-currency payment can be sent between two DFSPs on a Mojaloop network using an FX provider as an intermediary providing the FX.

Part 2 Scope



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

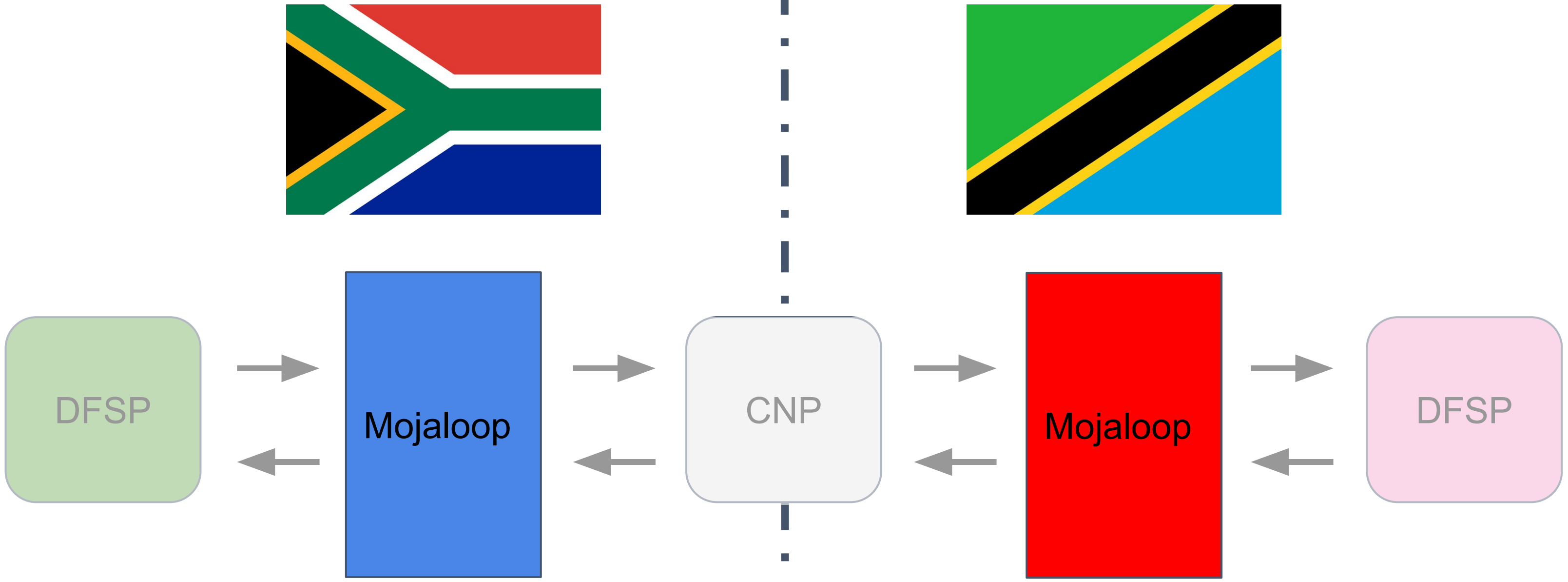
Part 2 Scope



Designed but not implemented until ALS stabilizes

1. ~~Lookup payee accounts (Moja Address)~~
2. Send cross-currency quote
3. Send cross-currency transfer

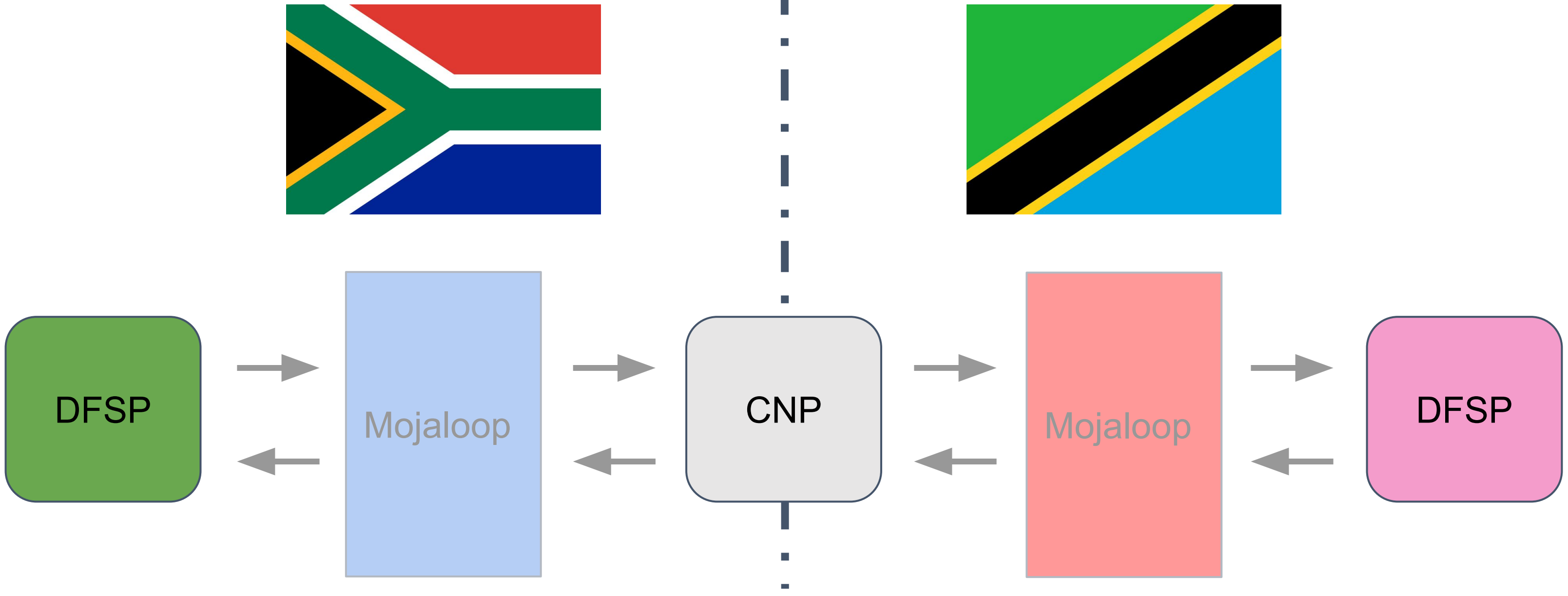
Part 1 - Ecosystem



Blue Moja
Domestic Mobile Money in South Africa

Red Moja
Domestic Mobile Money in Tanzania

Part 1 - Ecosystem

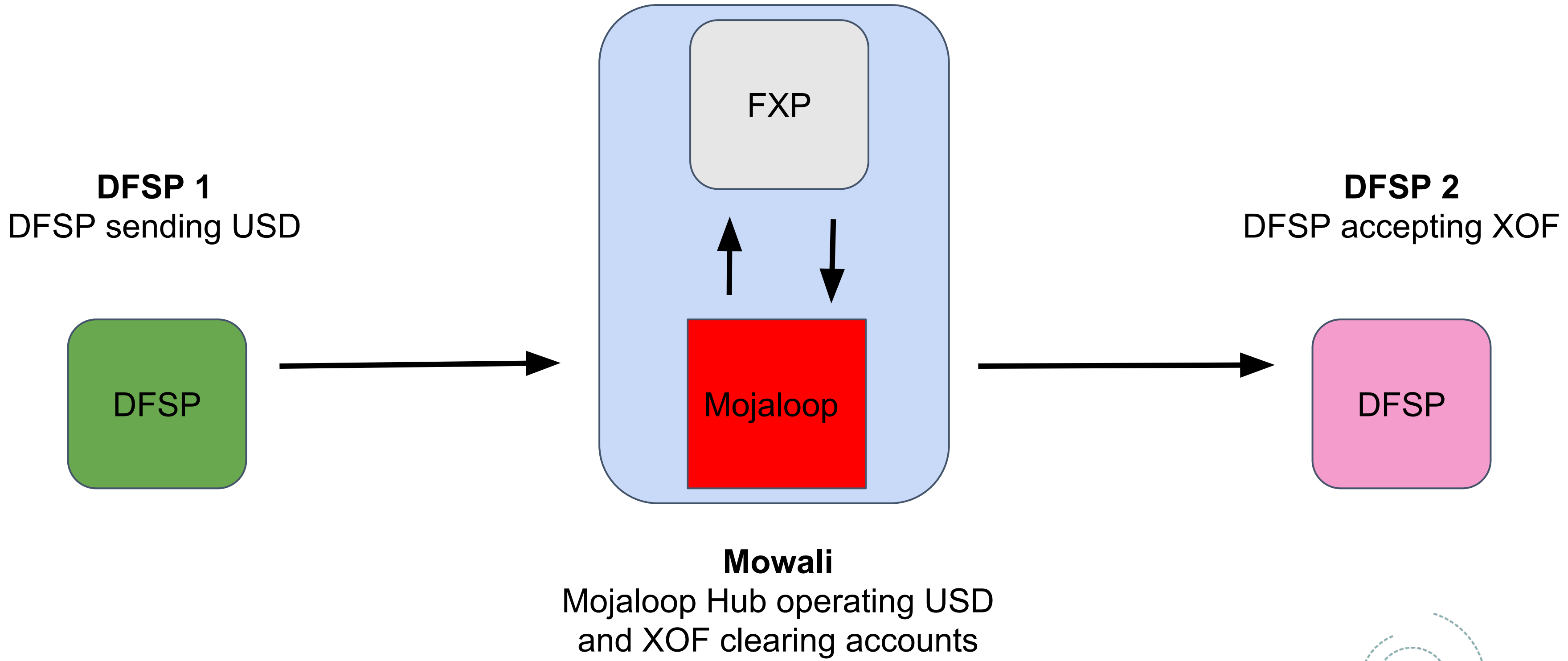


Green Mobile
DFSP in South Africa

Super Remit
Remittance provider operating in South Africa and Tanzania

Pink Mobile
DFSP in Tanzania

Part 2 - Ecosystem



Design Constraints

- Payer DFSP needs to know the receiver currency and receiver address (from **lookup**)
- Transfers must follow the same route as the quote
- The hub uses the same ***FspId*** for an FSP even if it has clearing accounts in multiple currencies

Design Decisions

- Used the same **addressing scheme** as part 1
- **Routing logic** implemented at the centre (interop-switch and FXP)
- Route is determined during **Quote** cycle only

Current Design

- All quote calls go via Mojaloop hub (interop-switch-js)
- Interop Switch consults **routing service** (1 required per currency) to set correct destination headers
- FXP wraps a **routing service**. Implements Open API endpoints and calculates exchange rates during **quote**

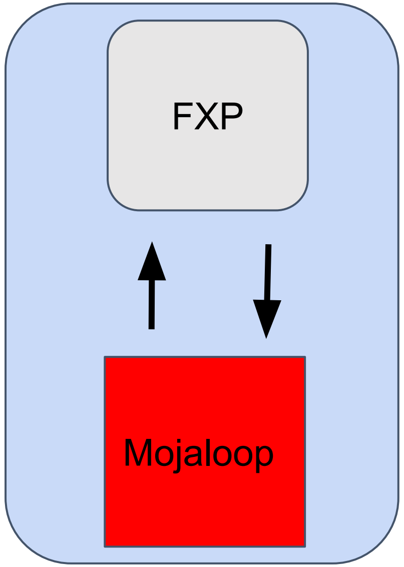
Current Design

- Use an Interledger Protocol-based addressing scheme for participants (DFSPs, the hub, FXP)

- Custom **moja** allocation scheme (Moja Addresses): e.g. *`moja.mowali.xof.dfsp2`*

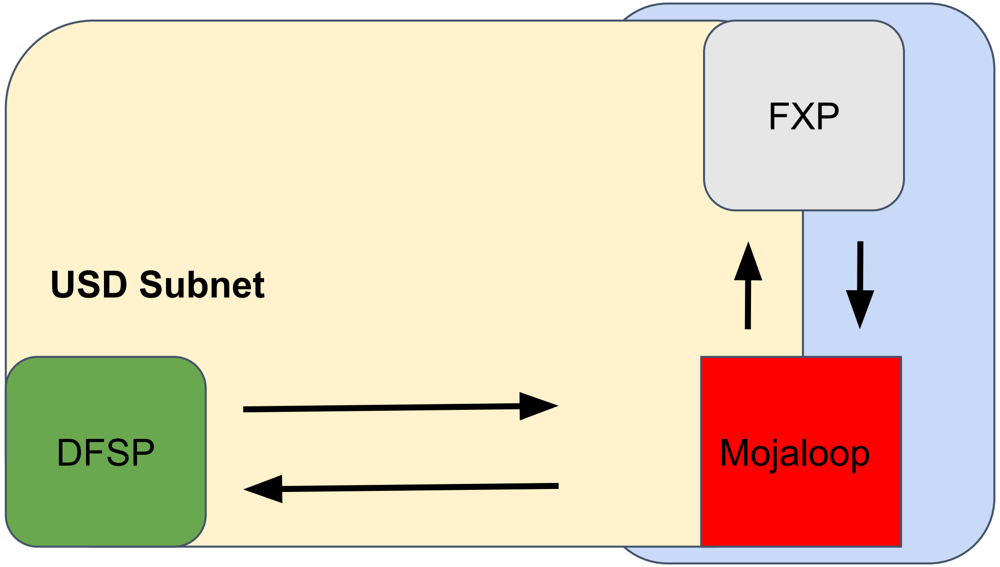
- Return *Moja Address* and **payee currency** during lookup as *`Party.PartyIdInfo.PartySubIdOrType`*

Address Space per Clearing Currency



moja.mowali.*

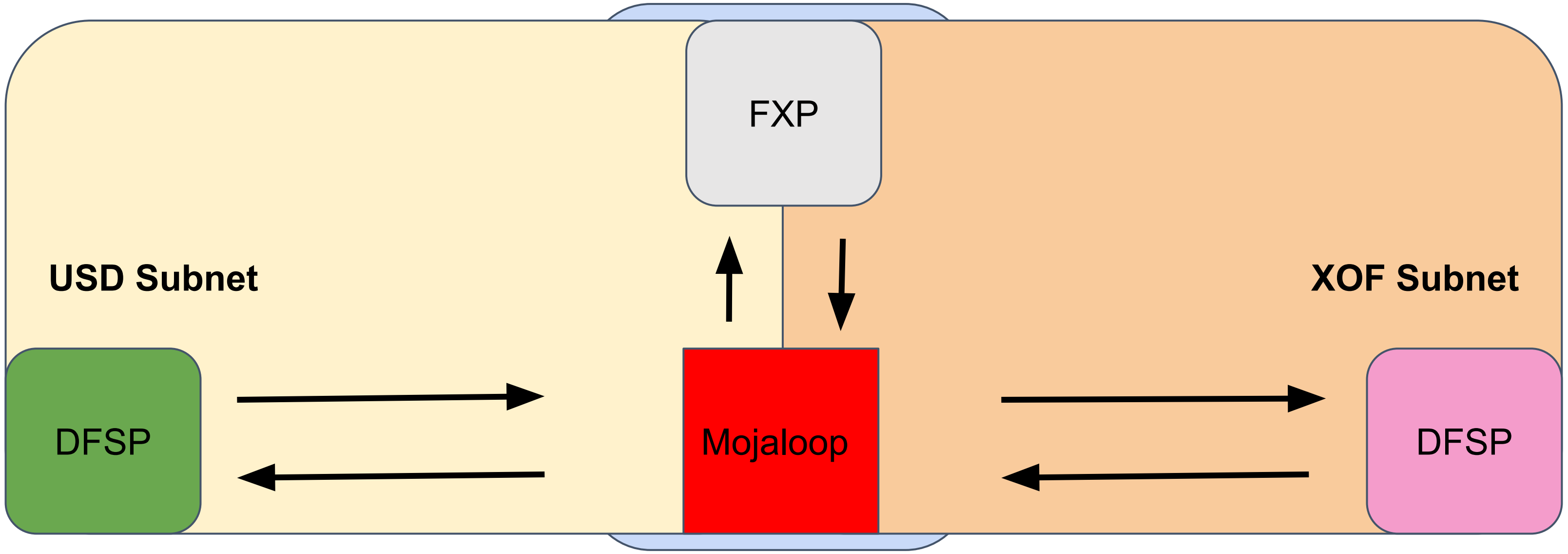
Address Space per Clearing Currency



moja.mowali.usd.dfsp1

moja.mowali.usd

Address Space per Clearing Currency



moja.mowali.usd.dfsp1

moja.mowali.usd
moja.mowali.xof

moja.mowali.xof.dfsp2

New and Changed Components

- Endpoint data for *interop-switch-js* (lookup and quote APIs) is still stored in Central Ledger’s DB to re-use existing schema.
- Routing logic deployed in a new stand-alone Routing micro-service. (Currently a service per currency, could be deployed behind a multi-currency facade if required.)

How It Works

Designed but not implemented until ALS stabilizes

1. Sending DFSP performs a **lookup** and gets a single Moja Address back and a receiving currency.

```
party: {
  partyIdInfo: {
    partySubIdOrType: 'XOF moja.mowali.xof.dfsp2'
  }
  ...
}
```

How It Works (alternative lookup proposal)

1. Sending DFSP performs a **lookup** and gets a **set** of Moja Addresses back, each associated with a receiving currency.

```
party: {
  addressList: [{
    currency: 'XOF',
    address: 'moja.mowali.xof.dfsp2'
  }]
  ...
}
```

How It Works

2. DFSP sends a **quote** to hub (*interop-switch-js)* which resolves a route and forwards the quote to the FXP

```
payee: {
   partyIdInfo: {
     partySubIdOrType: 'moja.mowali.xof.dfsp2'
   },
   transferCurrency: 'USD'
   ...
 }
```

New data element required in quote request

How It Works

3. FXP determines the **outgoing currency** based on the payee address. If the quote is for a **fixed send** amount it applies a currency conversion and routes the quote back to the hub (*interop-switch-js*)

4. Hub routes quote request to payee DFSP based on payee address

How It Works

5. Payee DFSP responds with quote to hub which routes back along same route (to FXP and then payer DFSP via hub). If the quote is for a **fixed receive** amount then the currency conversion is applied on the return path.

```
transferAmount: {
  ...
},
transferDestination: 'dfsp2'
...
```

New data element required in quote response

Quick Sidebar: Intermediary Data

Instead of ***TransferCurrency*** and ***TransferDestination*** data elements in quote, include a single ***Participants*** element.

An array of *Participant* items each containing an *FspId*, *TransferCurrency*, *Fees* and *Commissions*

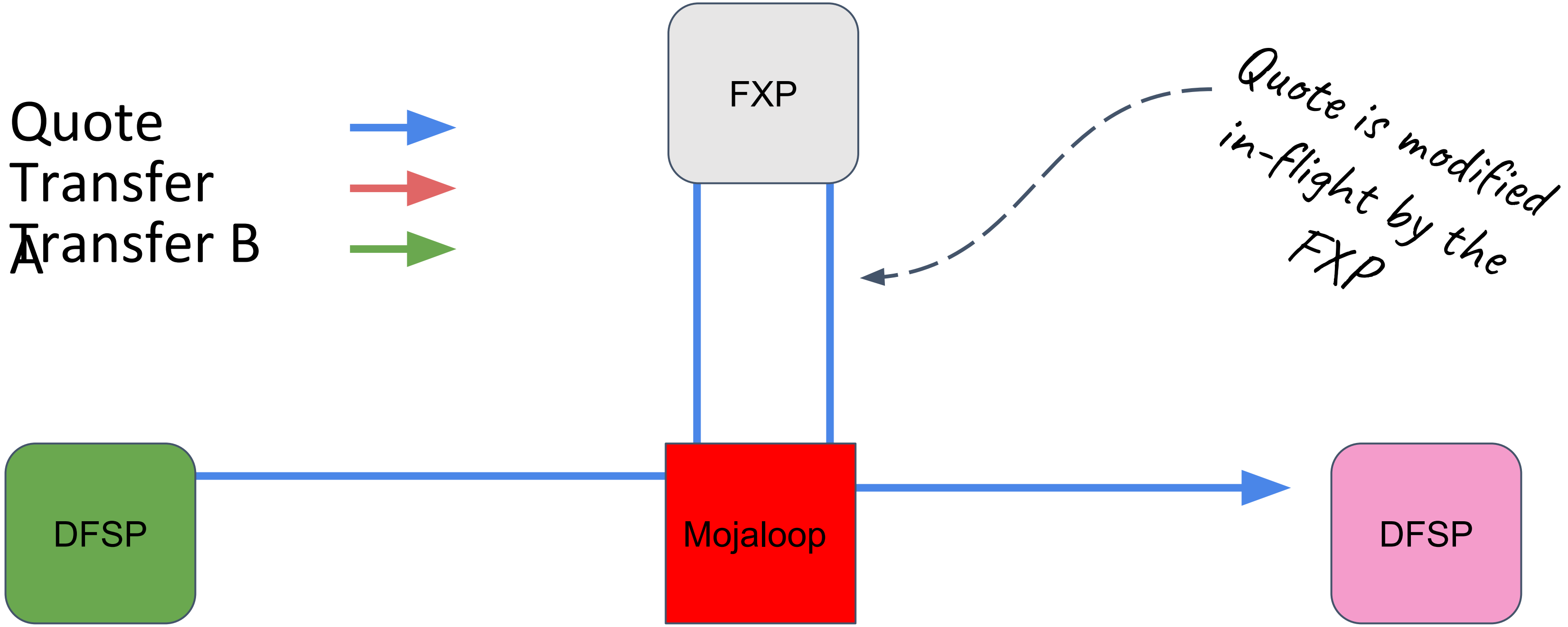
* To be explored further as may impact message signature.

How It Works

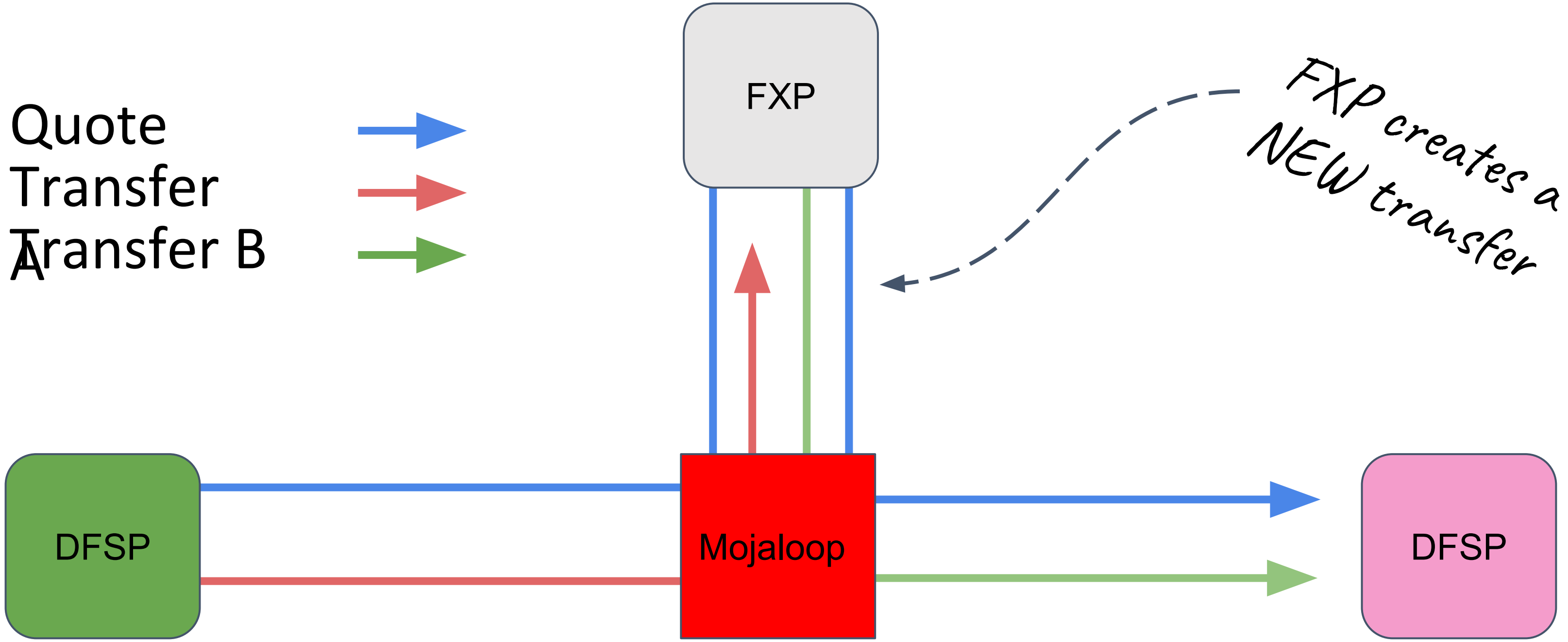
6. Payer DFSP sends a transfer to FXP (*TransferDestination*) for the amount quoted.
7. FXP uses the quoteId from the transaction to lookup the quote it provided previously and determine the exchange rate and fees to apply and where to send the next transfer.
8. FXP sends a transfer to the payee DFSP

Message Details (Quote vs Transfer)

Quote	🟦
Transfer A	🟥
Transfer B	🟩



Message Details (Quote vs Transfer)



Changing Quote

/quote

HEADERS:

source: dfsp1
destination: null

BODY:

payee: Bob@DFSP2
amountType: SEND
amount: USD 5
transferCurrency: USD

/quote

HEADERS:

source: **fxp**
destination: null

BODY:

payee: Bob@DFSP2
amountType: SEND
amount: XOF 500
transferCurrency: XOF

Two Transfers

```
/transfer
HEADERS:
 source: dfsp1
 destination: fxp

BODY:
 amount: USD 5
 condition: abuygew76f
 expiration: 9823...
 transaction: {
  quoteId: 1234
  ...
```

```
/transfer
HEADERS:
 source: fxp
 destination: dfsp2

BODY:
 amount: XOF 500
 condition: abuygew76f
 expiration: 5476...
 transaction: {
  quoteId: 1234
  ...
```

Message Impacts

- Different signatures between DFSP1 -> FXP and FXP -> DFSP2
- Dependency on previous API change proposals:
  - Drop ILP Packet
  - Raise ***Transaction*** object (need access to *quoteId*)
- Key material in ***Transaction*** object carried in ***Extensions***
- *Address in* *`Party.PartyIdInfo.PartySubIdOrType`*

Question: Does Transaction object need to be binary encoded?

Community Contributions

FXP App

- Implements Open API endpoints and ILP-like routing logic. Flexible rule configurations for FX and fees.

Routing Service

- Stand-alone micro-service hosts a routing table and route manager for dealing with route updates

Next Steps

- Dynamic routing and exchange of routing data between participants
- Include regulatory data exchange in the quote flow

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