Fabric Token SDK
From Token Transfer to Interoperability

From cryptocurrencies to CBDCs

Kaoutar Elkhiyaoui
IBM Reserarch

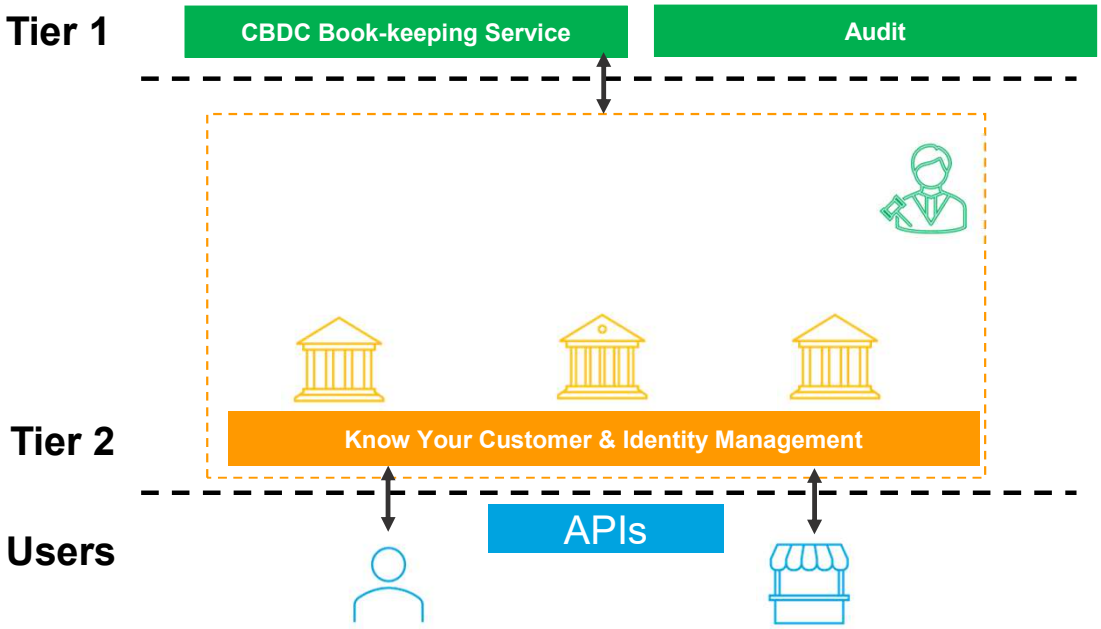
Organized jointly:

FDCI | Future of Digital Currency Initiative

ITU

An event of the Digital Currency Global Initiative

Central Bank Digital Currency: A Two-tiered Architecture



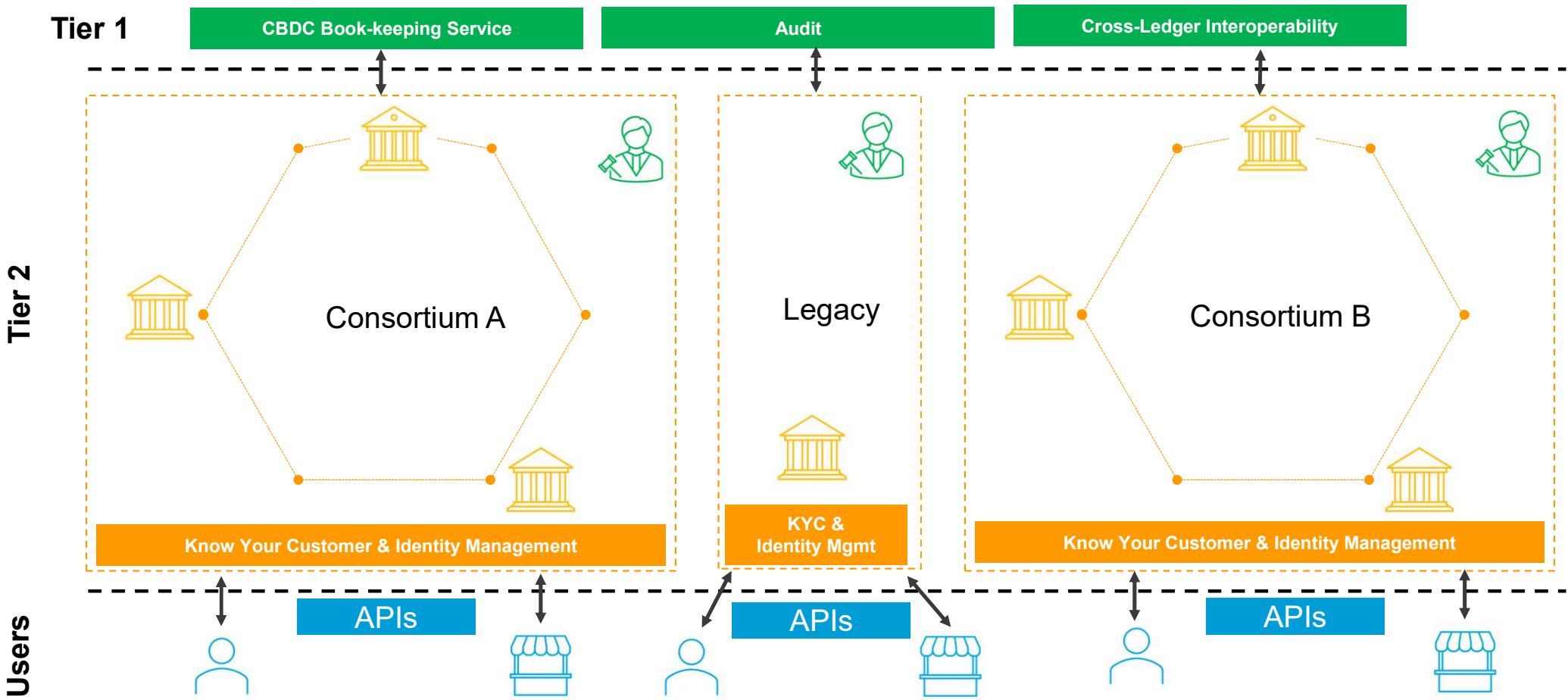
Tier 1 (Central Bank)

- ✓ Distribute CBDC to commercial banks
- ✓ Track CBDC in circulation
- ✓ Enforce regulatory compliance

Tier 2 (Commercial Banks)

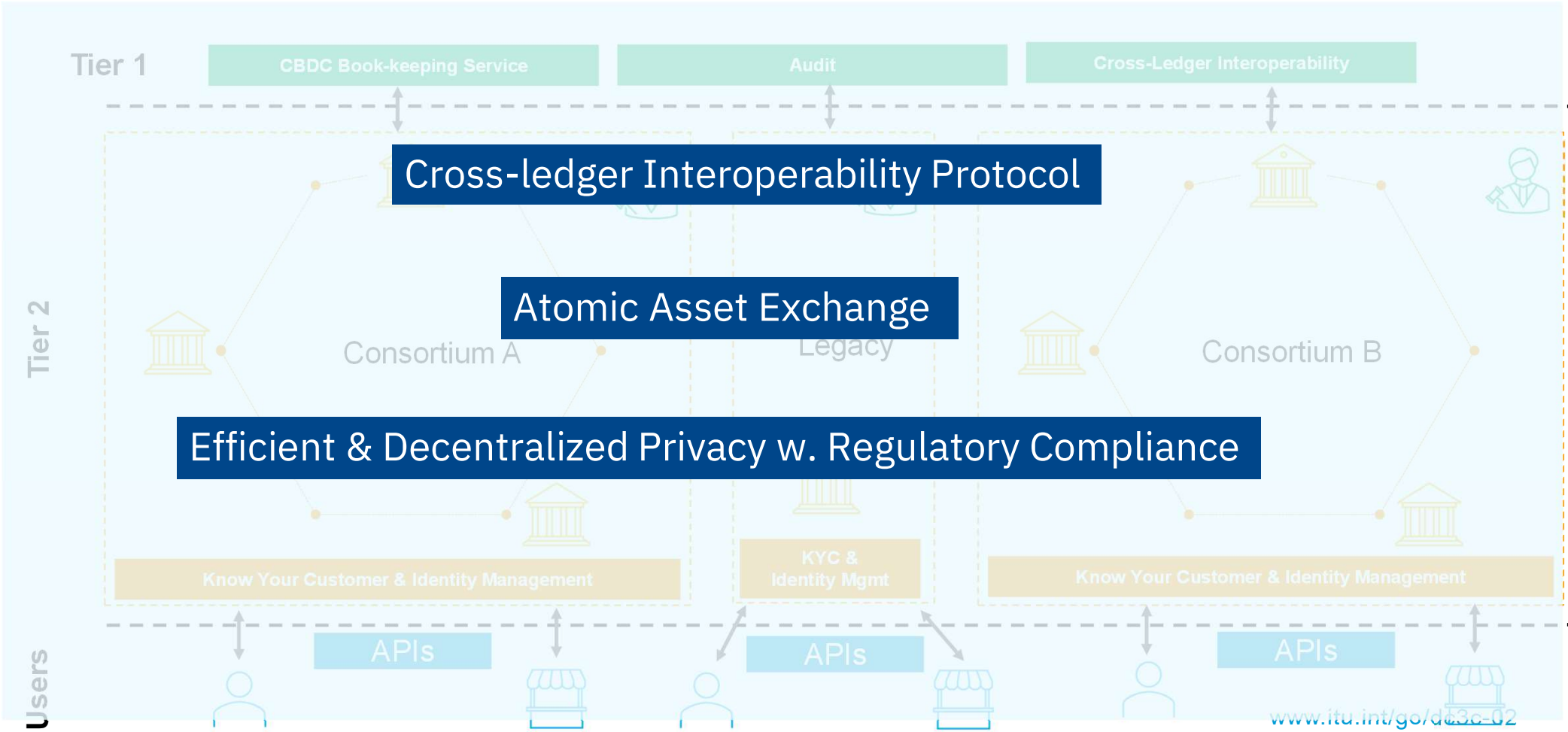
- ✓ Distribute CBDC to retail users
- ✓ On-board users (KYC)
- ✓ Execute retail payments

Central Bank Digital Currency: A Heterogeneous Architecture



www.itu.int/go/dc3c-02

Central Bank Digital Currency: A Heterogeneous Architecture



Fabric Token SDK

Scope & Features

The goal of Fabric Token SDK is to deliver a set of APIs that help developers create token-based distributed applications on Hyperledger Fabric

The Fabric Token SDK supports

- UTXO model
- Fungible and non-fungible tokens
- Key management via wallets
- **Multiple privacy** levels: from no privacy to zero-knowledge based instantiations that obfuscate the content of ledger while ensuring transaction verifiability
- **Auditability**
- **Interoperability thanks to Weaver**

A Simple and Effective Token Definition

A token consists of an

- **Owner**: The owner of the token

  It could be a public key, an anonymous identity or an *HTLC script*

- **Type**: The *denomination* of the token

  This is a string whose value is application specific

  Examples: The denomination of a digital currency or unique identifiers

- **Quantity**: The amount stored in the token.

  It is a positive number encoded as a string

Tokens are **fungible with respect to the same type**

Tokens with the same type can be merged and split, if not otherwise forbidden

Token Actions

Issue introduces a new token in the system

- An Issue action is defined by the newly created token
- An issue action is invoked by **Issue(Issuer_Wallet, Recipient, Type, Value)**

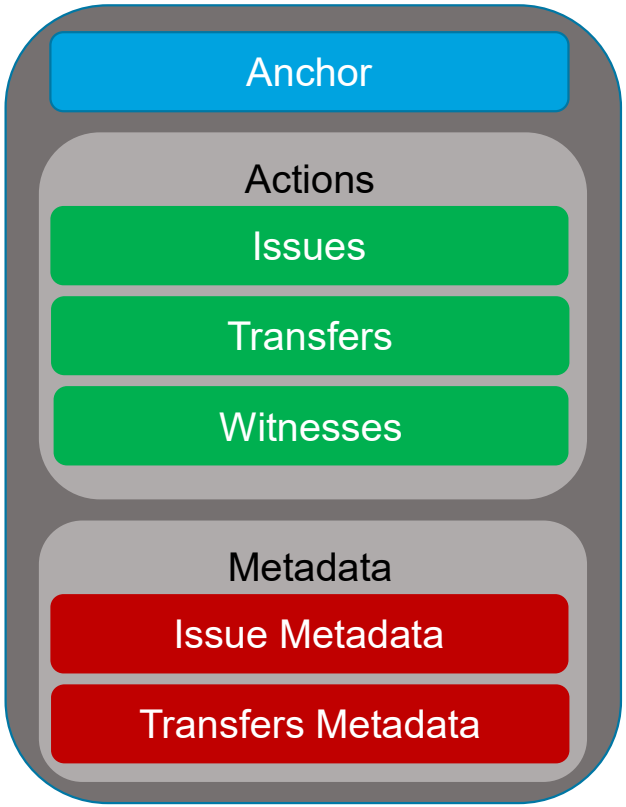
Transfer transfers tokens from a sender to a recipient

- A transfer action is defined by the inputs to be consumed and the outputs to be created
- A transfer action is invoked by **Transfer(Sender_Wallet, Type, Recipients, Values)**

Redeem removes tokens from circulation

- It is a transfer with an output with a **nil owner**
- A redeem action is invoked by **Redeem(Sender_Wallet, Type, Value)**

Anatomy of a Token Request



The **Anchor** is used to bind the Actions to a given transaction.
In Hyperledger Fabric, the anchor is the Transaction ID.

The **Actions** is a collection of Token Actions.

- **Issues** are used to **create new Tokens**
- **Transfers** are used to **transfer tokens**

Actions are accompanied by a set of **Witnesses** that attest to the consent of the issuers and/or the token owners to perform a certain action.

The **Metadata** is a collection of Metadata, one entry for each Token Action. This metadata is exchanged during the Token Request assembly. It contains secret information used by the parties to check the content of the Actions.

Metadata is not stored on the ledger.

Interoperability

Data Transfer

- Secure data transfer from ledger A to ledger B (i.e., ledger B uses ledger A as an oracle)
- Easy in centralized context, challenging in a decentralized one

Asset Transfer

- Atomic transfer of an asset from ledger A to ledger B
- Asset is redeemed in ledger A and introduced in ledger B atomically

Asset Exchange

- Asset X is transferred to Alice on ledger A while an asset Y is transferred to Bob on ledger B (e.g., DvP)

Interoperability

Data Transfer

- Secure data transfer from ledger A to ledger B (i.e., ledger B used ledger A as an oracle)
- Easy in centralized context, challenging in a decentralized one

Asset Transfer

- Atomic transfer of an asset from ledger A to ledger B
- Asset is redeemed in ledger A and introduced in ledger B atomically

Asset Exchange

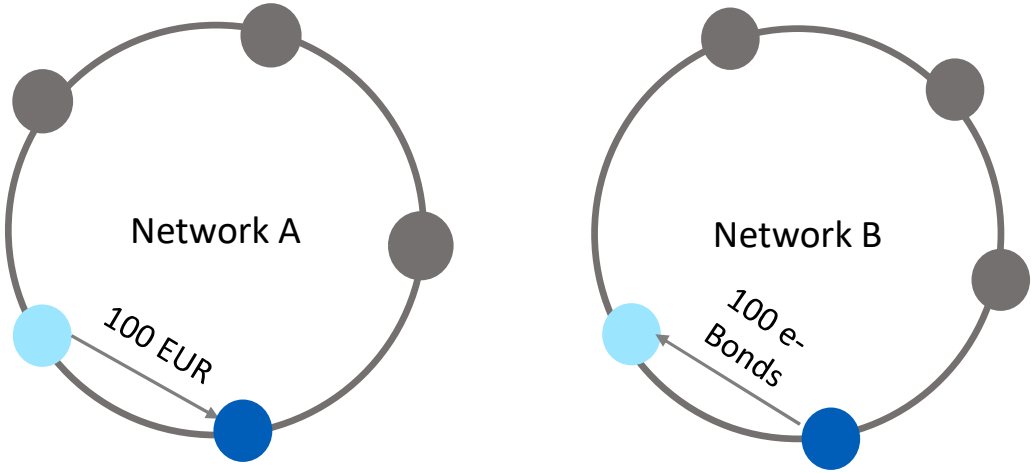
- Asset X is transferred to Alice on ledger A while an asset Y is transferred to Bob on ledger B (e.g., DvP)

Asset Exchange

Alice in Network A wants to transfer (100, EUR) to **Bob** in Network A

Bob in Network B wants to transfer (100, e-Bonds) to **Alice** in Network B

Alice and Bob are **rational and online**



Asset Exchange with HTLC

1) Alice and Bob agree on the terms of the HTLC contract
2) Bob picks a secret **x** and computes **H(x)**
3) Bob transmits H(x) to Alice
4) Bob submits a transaction that locks 100 e-Bonds for Alice in Network B
   - **Unlocking conditions:** Alice reveals **x** before **2*T** elapses; else Bob reclaims 100 e-Bonds
5) Alice submits a transaction that locks 100 EUR for Bob in Network A
   - **Unlocking conditions:** Bob reveals **x** before **T** elapses; else Alice reclaims 100 EUR

Asset Exchange in Fabric Token SDK

Pre-requisite: **x, H(x), T**

HTLC script is defined by **(Sender, Recipient, Hash, Timeout)**

Lock transfers a token to an **HTLC script**

- Lock is invoked by a call to **Lock(Sender_Wallet, Recipient, Value, Type, Hash, Timeout)**
- Lock creates a locked token whose ownership transfer is governed by the HTLC script

Claim transfers the ownership of a locked token from the HTLC script to the recipient

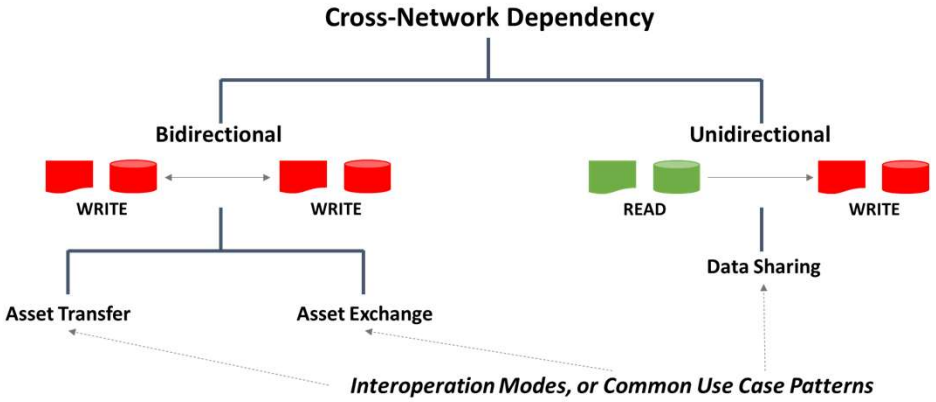
- Claim is invoked by a call to **Claim(Recipient_Wallet, Unspent_Token, Preimage)**
- Claim creates a token for the recipient

Reclaim transfers the ownership of a locked token from the HTLC script to the sender

- Reclaim is invoked by a call to **Reclaim(Sender_Wallet, Unspent_Token)**
- Reclaim creates a token for the sender

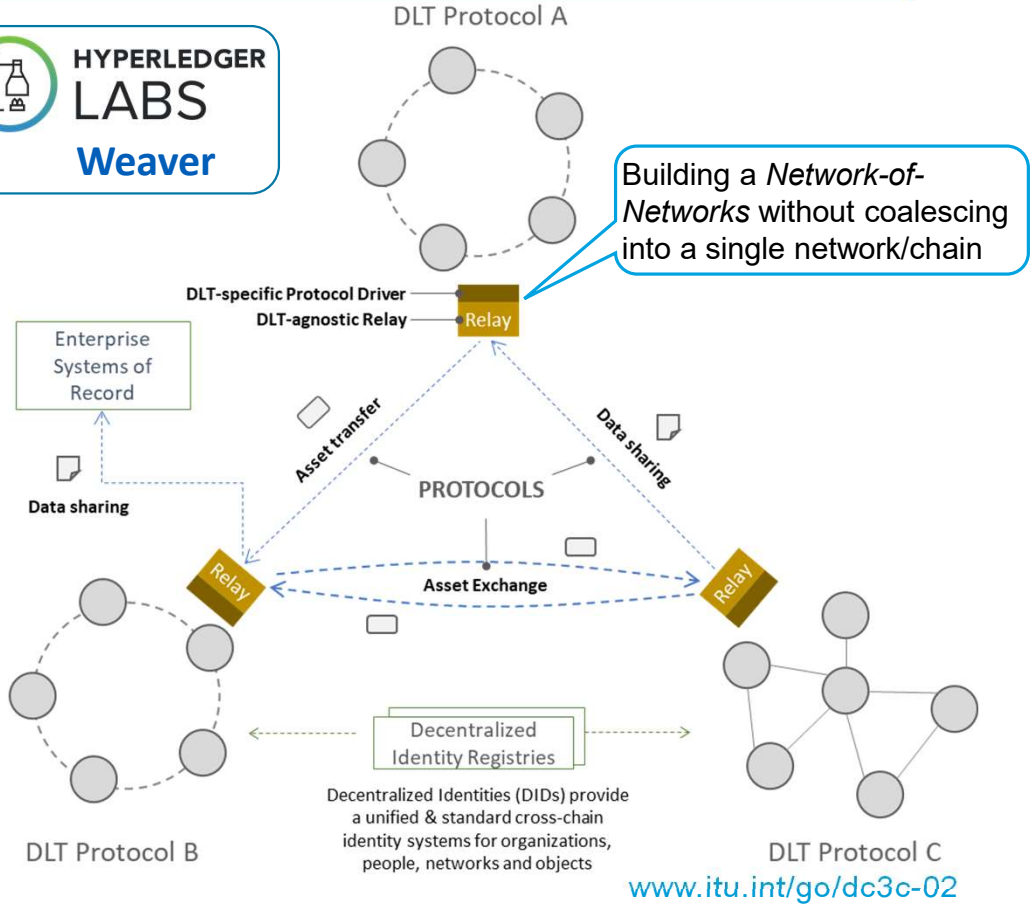
Weaver Interoperability Framework

Objective: manage fragmentation and increase scale and impact through cross-network transactions and business workflow linkages



Approach: direct on-demand controlled interactions among self-sovereign networks, using DLT-neutral protocols and internal network consensus, with no reliance on trusted third parties or intermediating chains

HYPERLEDGER LABS
Weaver



Building Blocks or Capabilities

Network

- × Access control
- × Generate ledger state proof
- × Validate ledger state proof
- × Lock or pledge asset
- × Claim or reclaim asset

Relay & Driver

- × Address remote views
- × Produce and expose view
- × Invoke contracts
- × Event pub/sub