Secure Asset Transfer Protocol (SATP)

Implementation in the Hyperledger Cacti Interoperability Framework

(draft-ietf-satp-core-02)
&
(draft-belchior-satp-gateway-recovery-00)

Zakwan Jaroucheh (Edinburgh Napier University)

Mentors: V. Ramakrishna (IBM), Rafael Belchior(Técnico Lisboa), Sandeep Nishad (IBM)

IETF 118: Secure Asset Transfer Working Group

Prague, Czech Republic

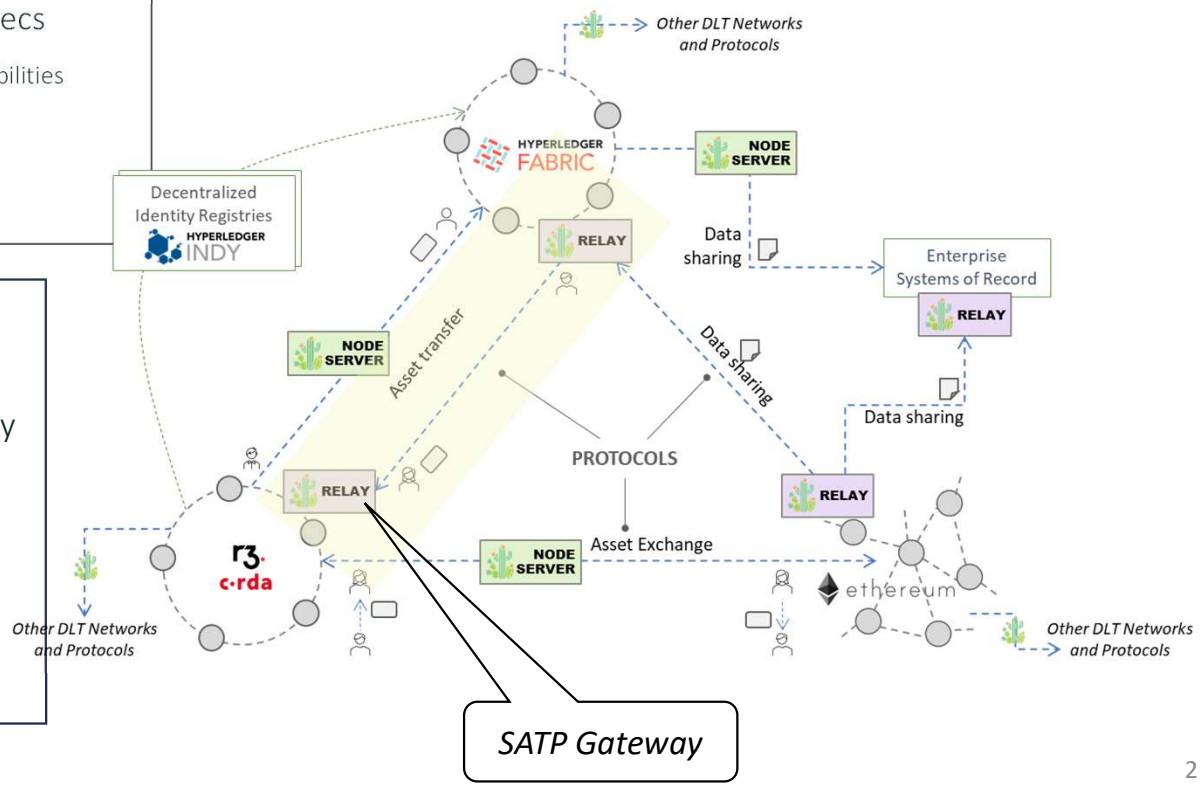
November 9, 2023

Reference Implementation of SATP in Hyperledger Cacti

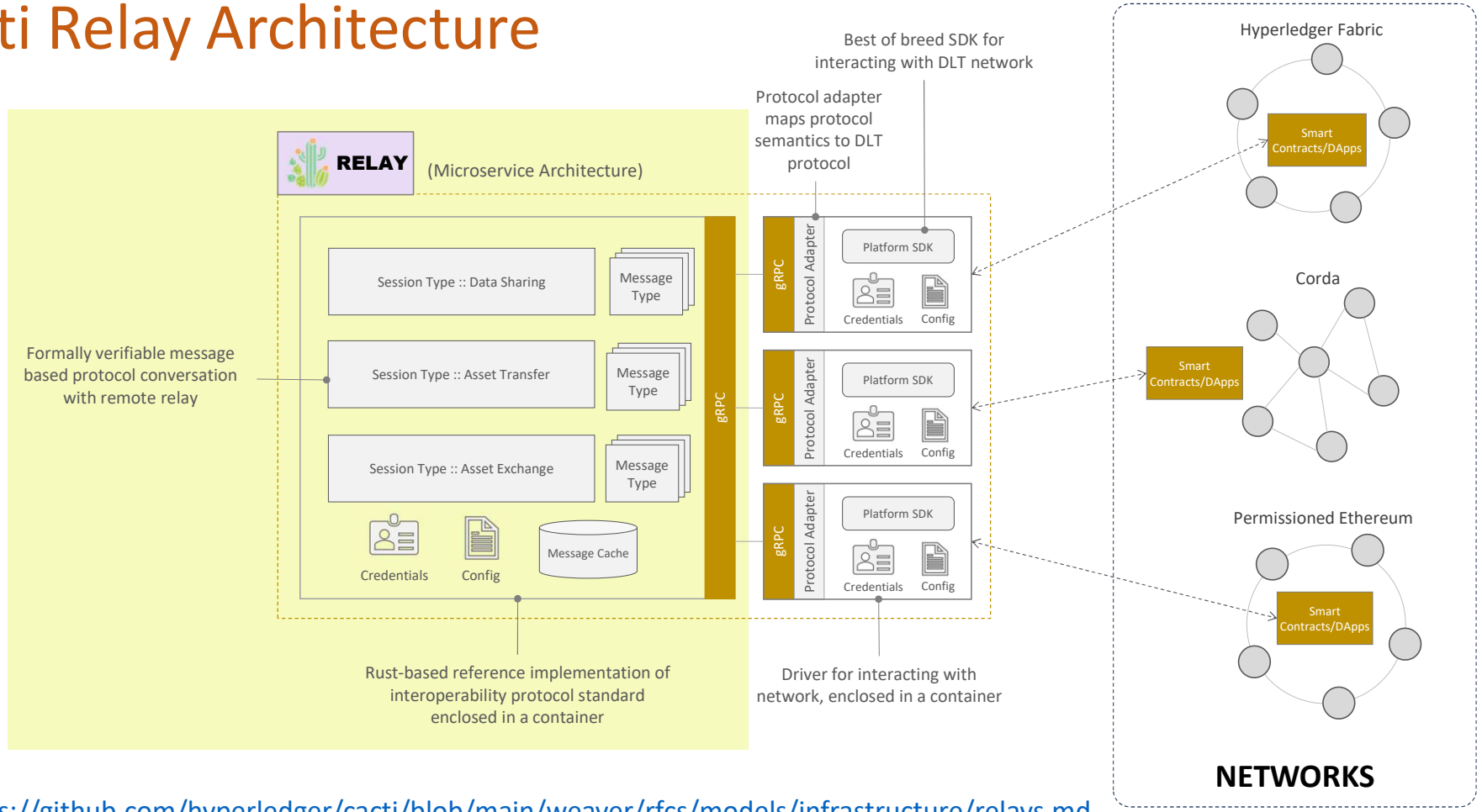
- Hyperledger Mentorship 2023 project 2023
  - https://wiki.hyperledger.org/display/INTERN/Cacti%3A+Implement+Standardized+Secure+Asset+Transfer+Protocol
  - Augment Cacti “relay” according to SATP draft specs
    - SATP-standard endpoints and SATP message parsing capabilities
    - Error handling and crash recovery support
  - ETA: end of November 2023

HYPERLEDGER CACTI https://github.com/hyperledger/cacti

- Relay is a configurable module running gRPC services built on Rust
  - Not built for any specific DLT; compatible with any
  - Fits the specification for an SATP gateway

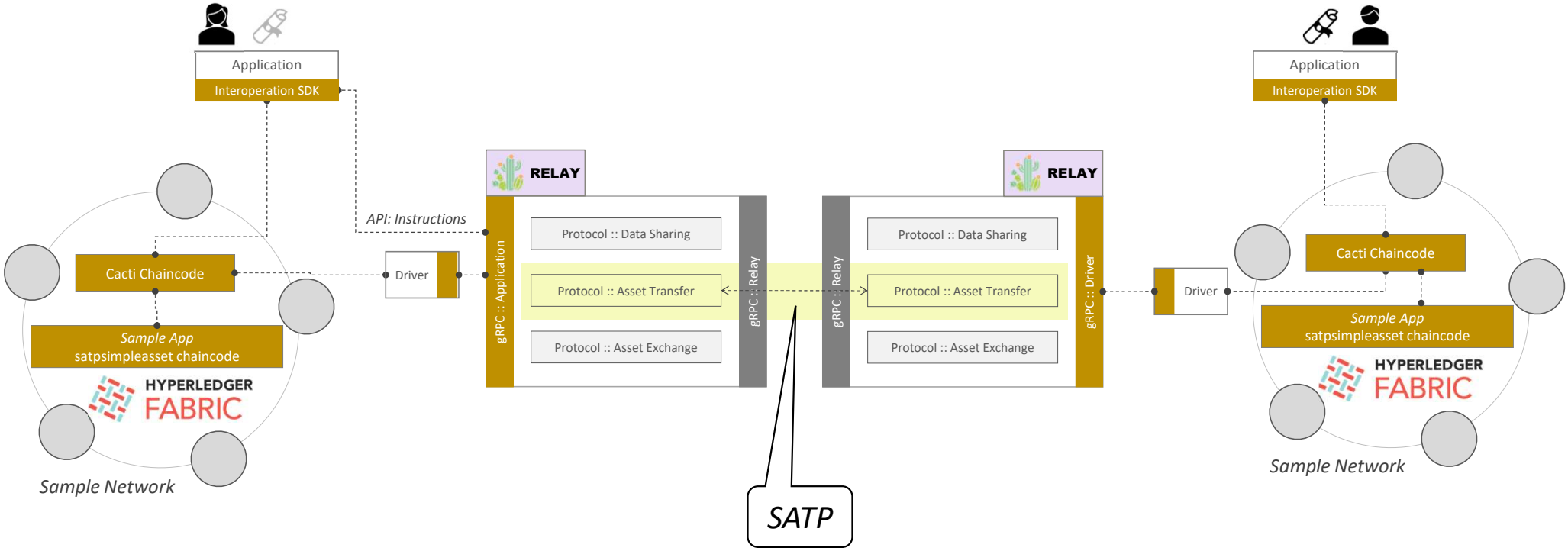


Cacti Relay Architecture



Spec: https://github.com/hyperledger/cacti/blob/main/weaver/rfcs/models/infrastructure/relays.md
Code: https://github.com/hyperledger/cacti/tree/main/weaver/core/relay

SATP Between Cacti-Augmented Networks

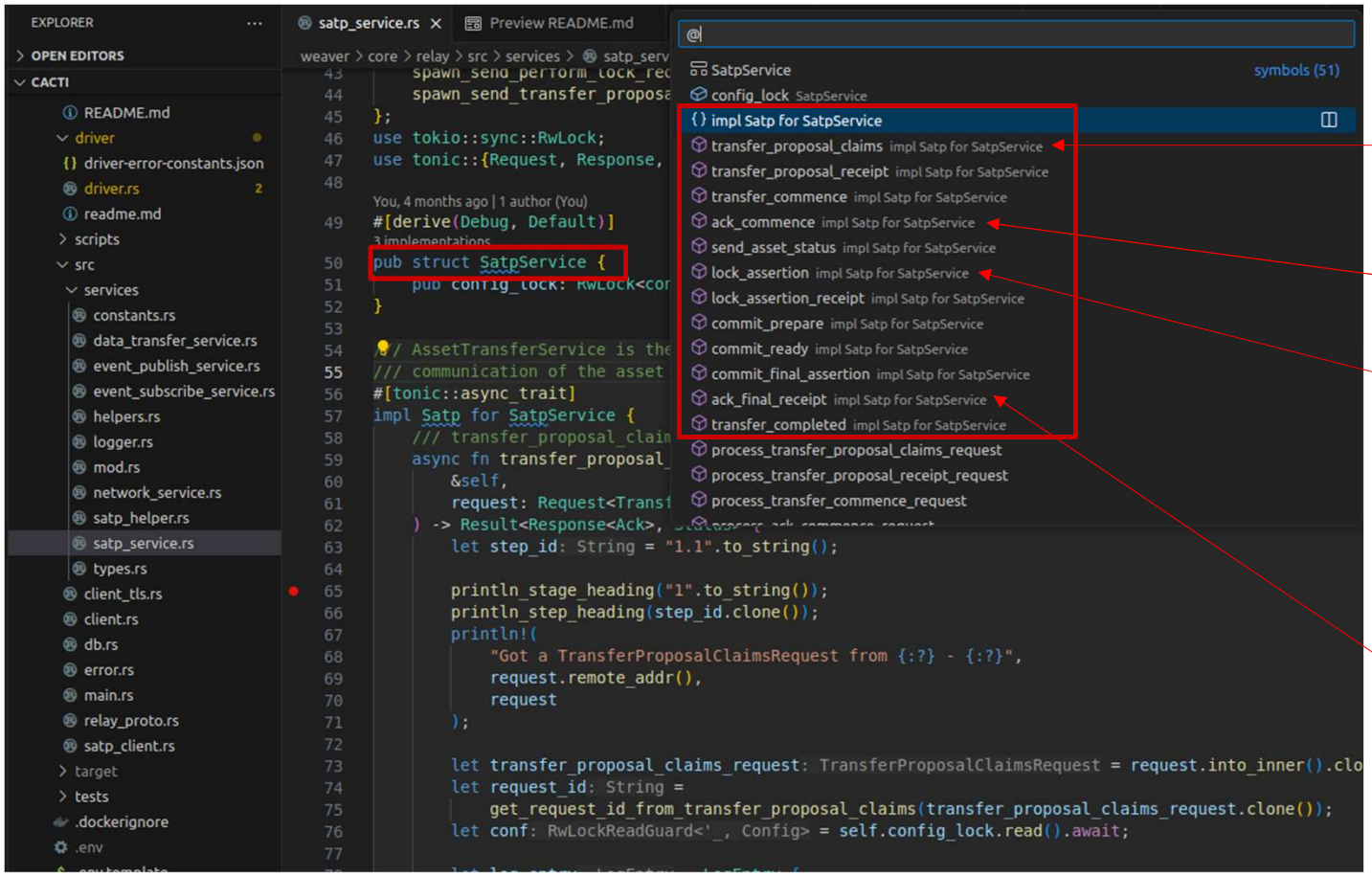


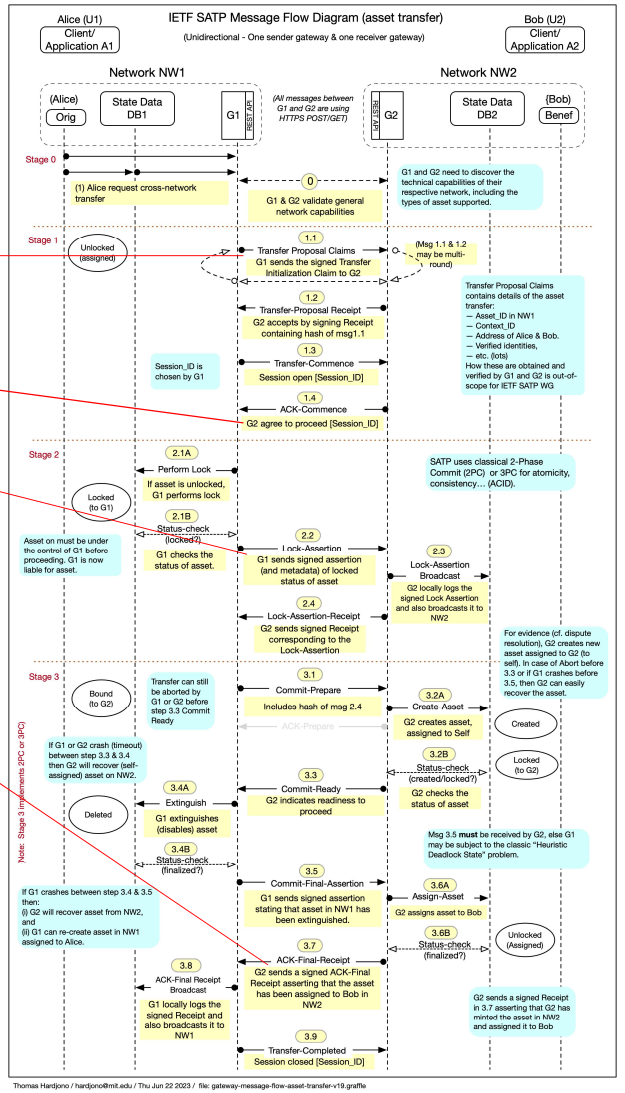
Augmenting Cacti Components for SATP

- Cacti Relay
  - Rust-based and runs gRPC services and clients
  - Added a SATP Service
    - Added the new service `satp_service.rs`
    - Added a new client `satp_client.rs`
    - Added/Changed the relevant library helper files
- SATP protobuf (service interface)
  - In `github.com/hyperledger/cacti/weaver/common/protos`, added the `satp.proto`
- Hyperledger Fabric App
  - In `github.com/hyperledger/cacti/weaver/samples/fabric`, added `satpsimpleasset` chaincode (`satpsimpleasset.go`), i.e., smart contract for Fabric
  - Augmented interoperation helper SDK for Fabric clients `github.com/hyperledger/cacti/weaver/core/fabric-driver/server/server.ts` to include the following functions: `performLock`, `createAsset`, `extinguish`, and `assignAsset`

- Reference: https://github.com/hyperledger/cacti/pull/2748

Relay Support for SATP Operations





Reference: draft-ietf-satp-core-02

SATP Protobuf (Service Interface for Relay)

```
service SATP {
  // Stage 1 endpoints

  // The sender gateway sends a TransferProposalClaims request to initiate an asset transfer.
  // Depending on the proposal, multiple rounds of communication between the two gateways may happen.
  rpc TransferProposalClaims(TransferProposalClaimsRequest) returns (common.ack.Ack) {};      You, 3 months ag

  // The sender gateway sends a TransferProposalClaims request to signal to the receiver gateway
  // that the it is ready to start the transfer of the digital asset
  rpc TransferProposalReceipt(TransferProposalReceiptRequest) returns (common.ack.Ack) {};

  // The sender gateway sends a TransferCommence request to signal to the receiver gateway
  // that the it is ready to start the transfer of the digital asset
  rpc TransferCommence(TransferCommenceRequest) returns (common.ack.Ack) {};

  // The receiver gateway sends a AckCommence request to the sender gateway to indicate agreement
  // to proceed with the asset transfe
  rpc AckCommence(AckCommenceRequest) returns (common.ack.Ack) {};

  // Stage 2 endpoints

  rpc SendAssetStatus(SendAssetStatusRequest) returns (common.ack.Ack) {};

  // The sender gateway sends a LockAssertion request to convey a signed claim to the receiver gateway
  // declaring that the asset in question has been locked or escrowed by the sender gateway in
  // the origin network (e.g. to prevent double spending)
  rpc LockAssertion(LockAssertionRequest) returns (common.ack.Ack) {};

  // The receiver gateway sends a LockAssertionReceipt request to the sender gateway to indicate acceptance
  // of the claim(s) delivered by the sender gateway in the previous message
  rpc LockAssertionReceipt(LockAssertionReceiptRequest) returns (common.ack.Ack) {};

  rpc CommitPrepare(CommitPrepareRequest) returns (common.ack.Ack) {};

  rpc CommitReady(CommitReadyRequest) returns (common.ack.Ack) {};

  rpc CommitFinalAssertion(CommitFinalAssertionRequest) returns (common.ack.Ack) {};

  rpc AckFinalReceipt(AckFinalReceiptRequest) returns (common.ack.Ack) {};

  rpc TransferCompleted(TransferCompletedRequest) returns (common.ack.Ack) {};
}
```

SATP Service Sample Function

Stage 1

Step 1.1

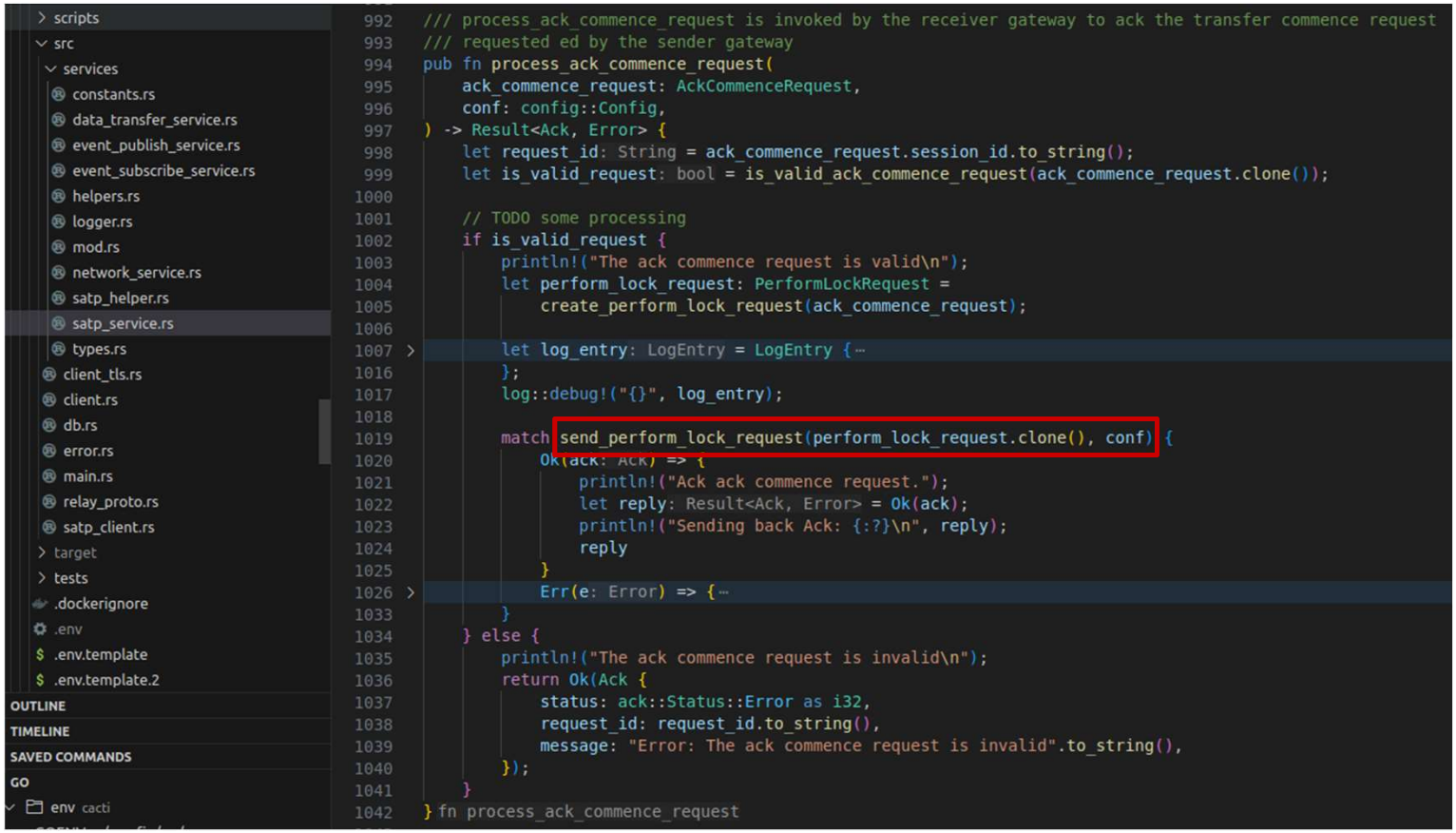
```
/// transfer_proposal_claims is run on the receiver gateway to allow the sender gateway to initiate an asset transfer.
async fn transfer_proposal_claims(
    &self,
    request: Request<TransferProposalClaimsRequest>,
) -> Result<Response<Ack>, Status> {
    let transfer_proposal_claims_request: TransferProposalClaimsRequest = request.into_inner().clone();
    let request_id: String =
        get_request_id_from_transfer_proposal_claims(transfer_proposal_claims_request.clone());
    let conf: RwLockReadGuard<'_, Config> = self.config_lock.read().await;

    let log_entry: LogEntry = LogEntry {…
    };
    log::debug!("{}", log_entry);

    match process_transfer_proposal_claims_request(
        transfer_proposal_claims_request.clone(),
        conf.clone(),
    ) {
        Ok(ack: Ack) => {
            let reply: Result<Response<Ack>, Status> = Ok(Response::new(ack));
            let log_entry: LogEntry = LogEntry {…
            };
            log::debug!("{}", log_entry);
            reply
        }
        Err(e: Error) => {
            let error_message: String = "Transfer proposal claims failed.".to_string();
            let reply: Result<Response<Ack>, Status> = create_ack_error_message(request_id.clone(), error_message.clone(), e);
            let log_entry: LogEntry = LogEntry {…
            };
            log::error!("{}", log_entry);
            reply
        }
    }
} fn transfer_proposal_claims
```

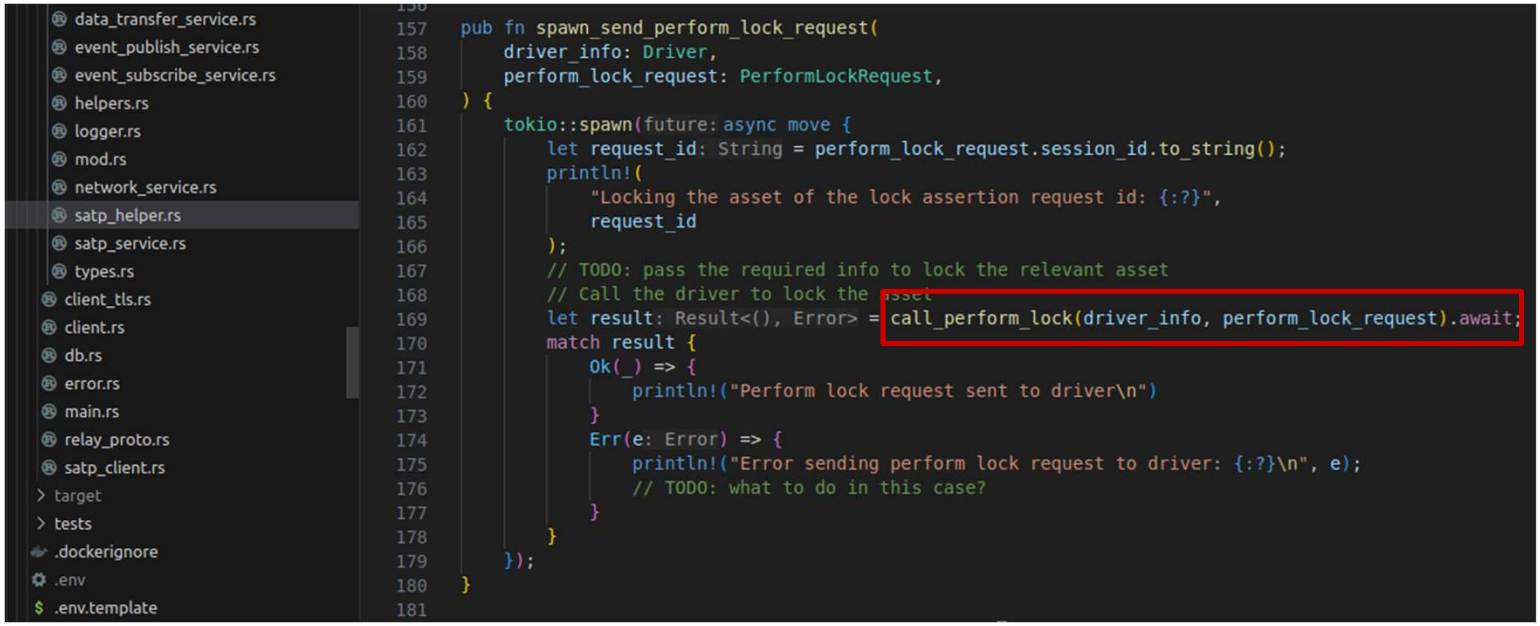
SATP Gateway Calls Fabric Driver

Step 1.1 → Step 2.1A



SATP Gateway Calls Fabric Driver

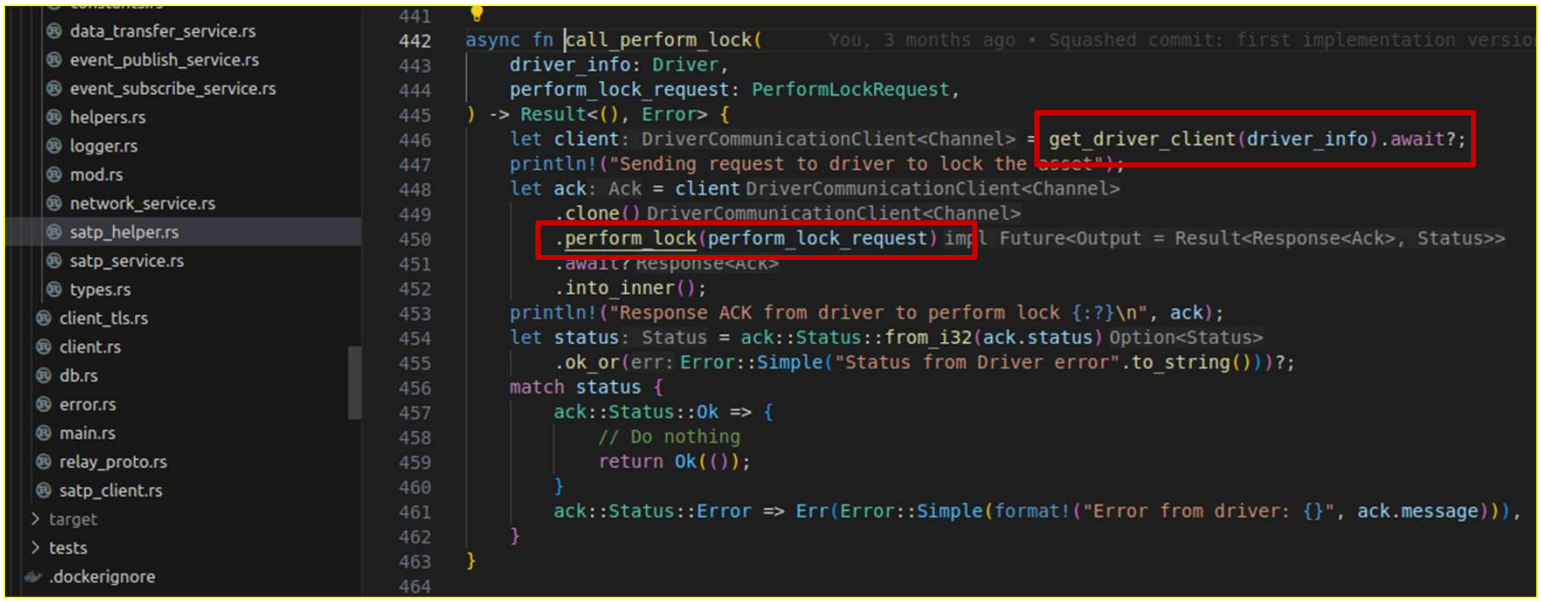
Stage 2
Step 2.1A



SATP Gateway Calls Fabric Driver

Stage 2

Step 2.1A

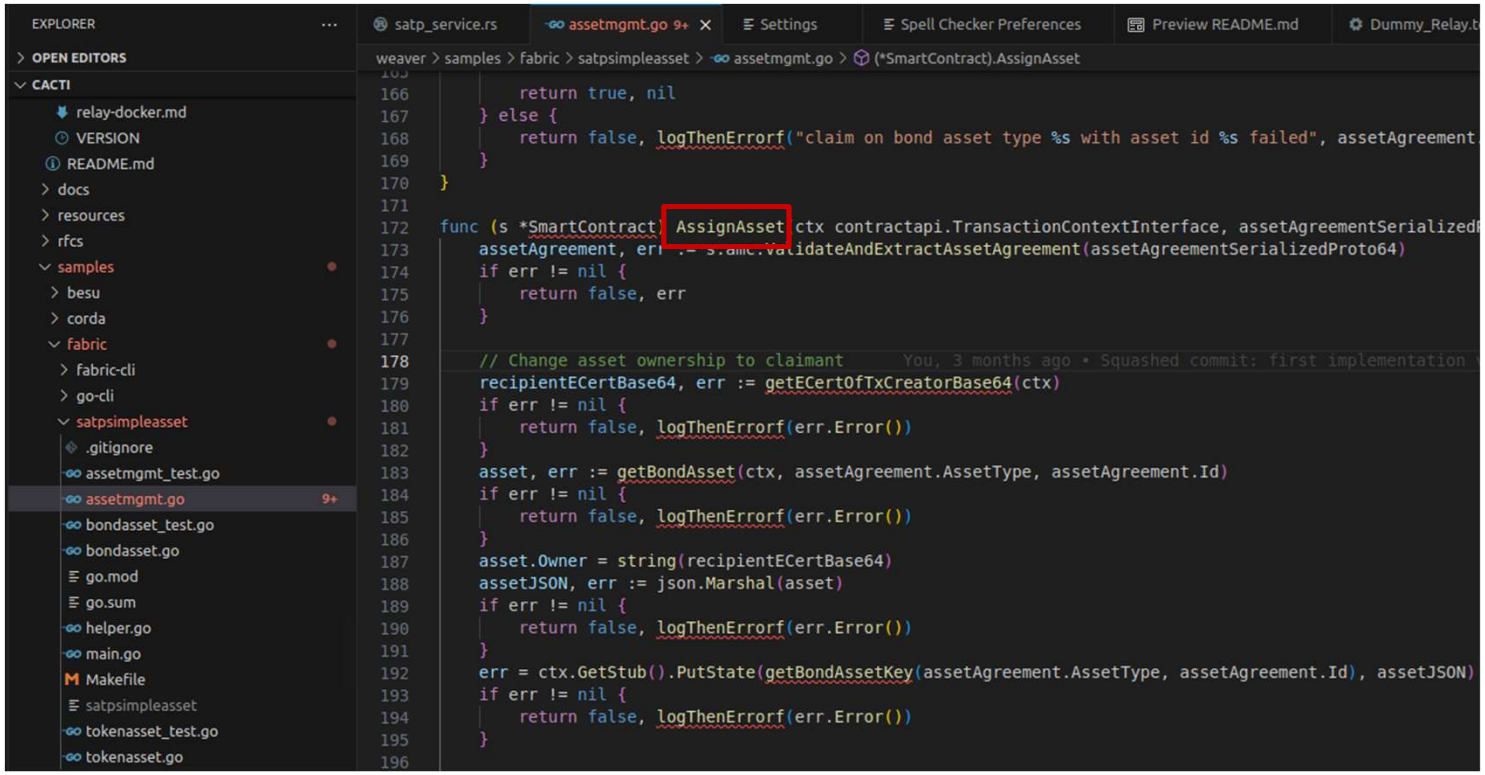


Fabric Driver Invokes `satpsimpleasset` Chaincode

A sample/reference implementation for networks built on Hyperledger Fabric that use SATP gateways

Function to process asset received from another network

Step 3.6A



Support for Other Kinds of Networks

- Relay is DLT-agnostic, so the SATP augmentation will work for any DLT (not just Fabric)
- Hyperledger Cacti supports connectivity to various kinds of DLTs: Hyperledger Fabric, Hyperledger Besu (permissioned Ethereum), Quorum, R3 Corda, Hyperledger Sawtooth, and others; more in the pipeline
  - Offers a connector/driver for each
    - Need to augment these to talk to the relay, just as we did for the Fabric driver in this project
  - Offers a client library/SDK for each
    - Need to add functions just like we did for the Fabric SDK in this project
  - Offers sample apps (smart contracts or DApps) for each
    - Need to add asset transfer endpoint functions (to lock, burn, mint, redeem, assets)

Logging (Crash Recovery)

```rust
#[derive(Serialize, Deserialize)]
3 implementations
pub struct LogEntry {
    pub request_id: String,
    pub request: String,
    pub step_id: String,
    pub operation: Operation,
    pub network_id: String,
    pub gateway_id: String,
    pub received: bool,
    pub details: Option<String>,
}
```

```rust
let log_entry: LogEntry = LogEntry {
    request_id: request_id.clone(),
    request: serde_json::to_string(&transfer_proposal_claims_request.clone()).unwrap(),
    step_id: step_id.clone(),
    operation: Operation::Init,
    network_id: "todo_network_id".to_string(),
    gateway_id: transfer_proposal_claims_request TransferProposalClaimsRequest
        .clone()
        .sender_gateway_network_id,
    received: true,
    details: None,
};
log::debug!("{}", log_entry);
```

Types: `Init`, `Exec`, `Done`

Reference: draft-belchior-satp-gateway-recovery-00

Logging (Crash Recovery)

```rust
fn log(&self, record: &Record) {
    if self.enabled(record.metadata()) {
        let conn: MutexGuard<'_, Connection> = self.conn.lock().unwrap();

        if let Ok(log_entry: LogEntry) = serde_json::from_str::<LogEntry>(&record.args().to_string()) {
            let mut details: String = "".to_string();
            if let Some(d: &String) = log_entry.details.as_ref() {
                details = d.to_string();
            }

            conn.execute(
                sql: "INSERT INTO log_entries (debug_level, timestamp, request_id, request, step_id, operation, gateway_id, received, details) VALUES
                (?1, ?2, ?3, ?4, ?5, ?6, ?7, ?8, ?9)",
                params: &[
                    &record.level().to_string(),          // Log level
                    &format!("{}", chrono::Local::now()), // Timestamp
                    &log_entry.request_id,
                    &log_entry.request,
                    &log_entry.step_id,
                    &log_entry.operation.to_string(),
                    &log_entry.gateway_id,
                    &log_entry.received.to_string(),
                    &details,
                ],
            ) Result<usize, Error>
            .expect(msg: "Failed to insert log entry");

        }
    }
} fn log
```

Storage Implementation: SQLite database

Sample Snapshot of Log Entries

log_entries | Enter a SQL expression to filter results (use Ctrl+Space)

	id	debu	timestamp	request_id	request	step	operation	gateway_id	received
128	128	DEBUG	2023-10-31 09:32:04.813467831 +00:00	session_id1	{"message_type":"message_type1","session_id":"sessi	1.4	Done	todo_gateway_id	true
129	129	DEBUG	2023-10-31 09:32:04.819427511 +00:00	session_id1			Done		false
130	130	DEBUG	2023-10-31 09:44:38.261779408 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.1	Init	sender_gateway_net	true
131	131	DEBUG	2023-10-31 09:44:38.301532806 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.1	Exec	sender_gateway_net	true
132	132	DEBUG	2023-10-31 09:44:38.307321484 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.1	Done	sender_gateway_net	true
133	133	DEBUG	2023-10-31 09:44:38.314183193 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.2	Init	sender_gateway_net	false
134	134	DEBUG	2023-10-31 09:44:38.326025051 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.2	Init	sender_gateway_net	true
135	135	DEBUG	2023-10-31 09:44:38.333864947 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","session_id":"to_b	1.2	Exec	todo_gateway_id	true
136	137	DEBUG	2023-10-31 09:44:38.364966704 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.2	Done	sender_gateway_net	true
137	139	DEBUG	2023-10-31 09:44:38.382240429 +00:00	to_be_calculated_session_id	{"message_type":"message_type1","session_id":"to_b	1.3	Init	todo_gateway_id	true
138	140	DEBUG	2023-10-31 09:44:38.394858645 +00:00	to_be_calculated_session_id	{"message_type":"message_type1","session_id":"sessi	1.3	Exec	todo_gateway_id	true
139	141	DEBUG	2023-10-31 09:44:38.413857783 +00:00	to_be_calculated_session_id	{"message_type":"message_type1","session_id":"to_b	1.3	Done	todo_gateway_id	true
140	142	DEBUG	2023-10-31 09:44:38.428883513 +00:00	session_id1	{"message_type":"message_type1","session_id":"sessi	1.4	Init	todo_gateway_id	true
141	144	DEBUG	2023-10-31 09:44:38.440126841 +00:00	session_id1	{"session_id":"session_id1"}	1.4	Exec	todo_gateway_id	true
142	145	DEBUG	2023-10-31 09:44:38.445342687 +00:00	session_id1	{"message_type":"message_type1","session_id":"sessi	1.4	Done	todo_gateway_id	true
143	147	DEBUG	2023-11-02 12:56:08.816127280 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.1	Init	sender_gateway_net	true
144	148	DEBUG	2023-11-02 12:56:08.832983294 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.1	Exec	sender_gateway_net	true
145	149	DEBUG	2023-11-02 12:56:08.838496872 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.1	Done	sender_gateway_net	true
146	150	DEBUG	2023-11-02 12:56:08.848544677 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.2	Init	sender_gateway_net	false
147	151	DEBUG	2023-11-02 12:56:08.865015914 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.2	Init	sender_gateway_net	true
148	152	DEBUG	2023-11-02 12:56:08.878016932 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","session_id":"to_b	1.2	Exec	todo_gateway_id	true
149	153	DEBUG	2023-11-02 12:56:08.881571873 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.2	Done	sender_gateway_net	true
150	154	DEBUG	2023-11-02 12:56:08.889036179 +00:00	to_be_calculated_session_id	{"message_type":"message_type1","session_id":"to_b	1.3	Init	todo_gateway_id	true
151	155	DEBUG	2023-11-02 12:56:08.894634330 +00:00	to_be_calculated_session_id	{"message_type":"message_type1","session_id":"sessi	1.3	Exec	todo_gateway_id	true
152	156	DEBUG	2023-11-02 12:56:08.898092023 +00:00	to_be_calculated_session_id	{"message_type":"message_type1","session_id":"to_b	1.3	Done	todo_gateway_id	true
153	157	DEBUG	2023-11-02 12:56:08.906376850 +00:00	session_id1	{"message_type":"message_type1","session_id":"sessi	1.4	Init	todo_gateway_id	true
154	158	DEBUG	2023-11-02 12:56:08.909971998 +00:00	session_id1	{"session_id":"session_id1"}	1.4	Exec	todo_gateway_id	true
155	159	DEBUG	2023-11-02 12:56:08.913557005 +00:00	session_id1	{"message_type":"message_type1","session_id":"sessi	1.4	Done	todo_gateway_id	true
156	160	DEBUG	2023-11-03 09:05:18.728351562 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.1	Init	sender_gateway_net	true
157	161	DEBUG	2023-11-03 09:05:18.760018150 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.1	Exec	sender_gateway_net	true
158	162	DEBUG	2023-11-03 09:05:18.777034518 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.1	Done	sender_gateway_net	true
159	163	DEBUG	2023-11-03 09:05:18.824966745 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.2	Init	sender_gateway_net	false
160	164	DEBUG	2023-11-03 09:05:18.846549066 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.2	Init	sender_gateway_net	true
161	165	DEBUG	2023-11-03 09:05:18.854730148 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","session_id":"to_b	1.2	Exec	todo_gateway_id	true
162	166	DEBUG	2023-11-03 09:05:18.869134800 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.2	Done	sender_gateway_net	true
163	167	DEBUG	2023-11-03 09:05:18.904405135 +00:00	to_be_calculated_session_id	{"message_type":"message_type1","session_id":"to_b	1.3	Init	todo_gateway_id	true
164	168	DEBUG	2023-11-03 09:05:18.913498099 +00:00	to_be_calculated_session_id	{"message_type":"message_type1","session_id":"sessi	1.3	Exec	todo_gateway_id	true
165	169	DEBUG	2023-11-03 09:05:18.938089153 +00:00	to_be_calculated_session_id	{"message_type":"message_type1","session_id":"to_b	1.3	Done	todo_gateway_id	true

Storage Implementation: SQLite database

Project Status

- SATP Service
  - All endpoints for Stage 1 to Stage 3 have been implemented
  - An endpoint has been added to allow the driver to update the status of an asset for the steps: 2.1B, 3.2B, 3.4B and 3.6B.
  - Placeholders have been added to validate each incoming requests
  - Placeholders have been added to derive the corresponding request object (that needs to be sent to the other gateway) based on the incoming requests
  - Demo how an asset can be transferred from one Fabric network to another Farbic network using SATP protocol implementation

- TODO
  - All endpoints related to Stage 0
  - Fill the above placeholders according to the SATP logic
  - Remove some hardcoded values used for demoing creating an asset, assign it and destroy it
  - Add a looping mechanism that enables the gateway to repeatedly inquire about the asset's status from the driver.

Feedback from Implementer

- It would be beneficial to have the format of these messages documented in a Git repository to serve as the definitive source of truth.
- Interface between gateway and driver (executor) is not clear (*Note*: out of scope)
- Unclear how to get a unique ID for each request
- How do gateways negotiate compatible signature algorithms? (*Note*: Stage 0)
- Ambiguity about asset state inference in the face of failure
  - If a gateway recovers after a crash and wishes to resume SATP but discovers that an asset is locked, how does it know that the asset was locked by the in-progress SATP instance and not by an unrelated process?
  - *Note*: do we need a generic interface for networks to expose the states of digital assets to third parties?
- Managing contention and avoid overhead: read asset state first before attempting to lock (atomic operation)

Thank you and Q&A

Zakwan Jaroucheh: zakwanj@gmail.com
Rafael Belchior: rafael.belchior@tecnico.ulisboa.pt
V. Ramakrishna: vramakr2@in.ibm.com
Sandeep Nishad: sandeep.nishad1@ibm.com

Zakwan's demo with voiceover: https://we.tl/t-dWH2vFeNt4