

IETF I17

Crash Fault Tolerance, Session Resumption, Update On SATP Implementations

draft-hargreaves-satp-core-02
(draft-belchior-satp-gateway-recovery-00)

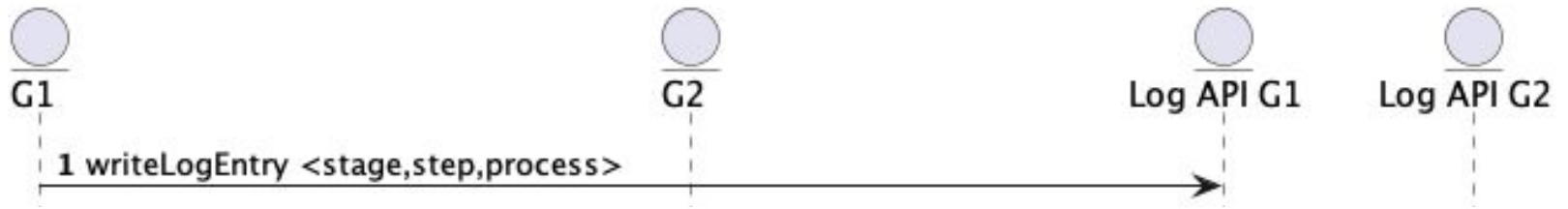
M. Hargreaves (Quant)

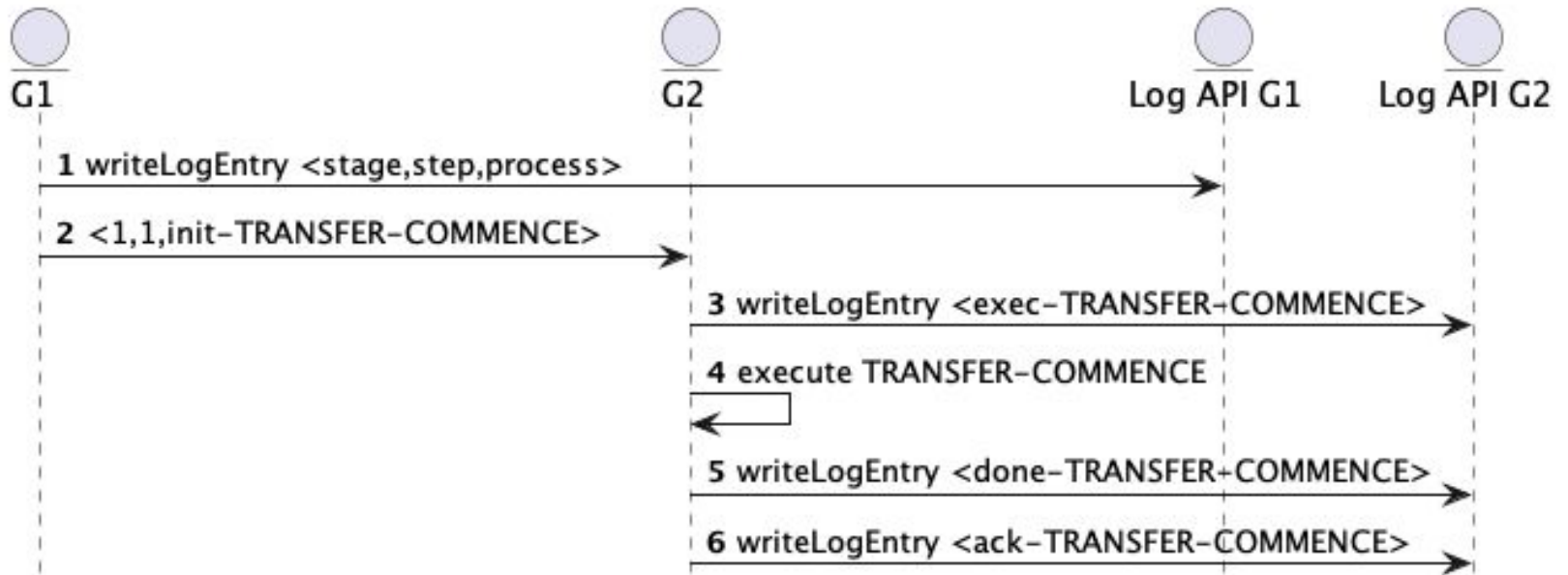
T. Hardjono (MIT)

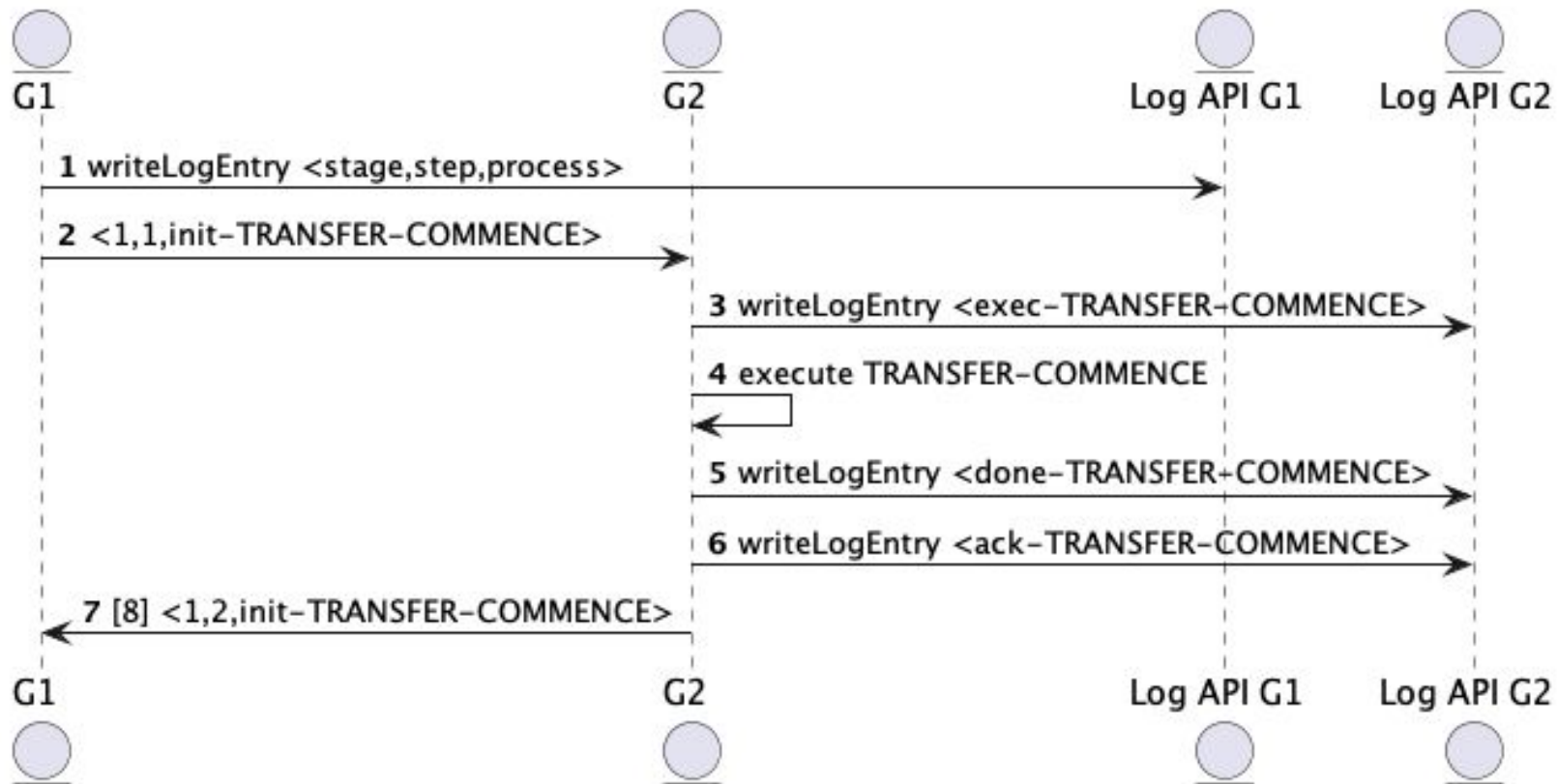
R. Belchior (Técnico Lisboa, Blockdaemon)

On faults, crashes, and errors

- Gateways can enter a faulty state: and produce errors (alerts to the user) or fail (crashes)
- We need a logging and recovery procedure: crash recovery draft.
- <https://datatracker.ietf.org/doc/draft-belchior-satp-gateway-recovery/>
- <https://github.com/CxSci/IETF-SATP/pull/9>
- Log Storage (each gateway has one):
 - State DB (safety: correct operation, liveness)
 - Decentralized, public DB (safety: non-repudiation and decentralization, used for retrieving logs upon crash)







Handling crashes

- How do we handle crash faults?
 - Let us assume two scenarios: faults in a stage of the protocol that does not imply changes to the state of external systems (non-critical); and faults in a stage where one would have changes to external systems (critical)
 - Gateways assume to recover by itself (self-healing) or by employing a primary-backup service.
- We handle crashes depending on the scenario at hand; by keeping a write ahead log of operations, and by enforcing either a session resumption protocol, abort, or rollback.
- We need to define the state for recovering from crashes

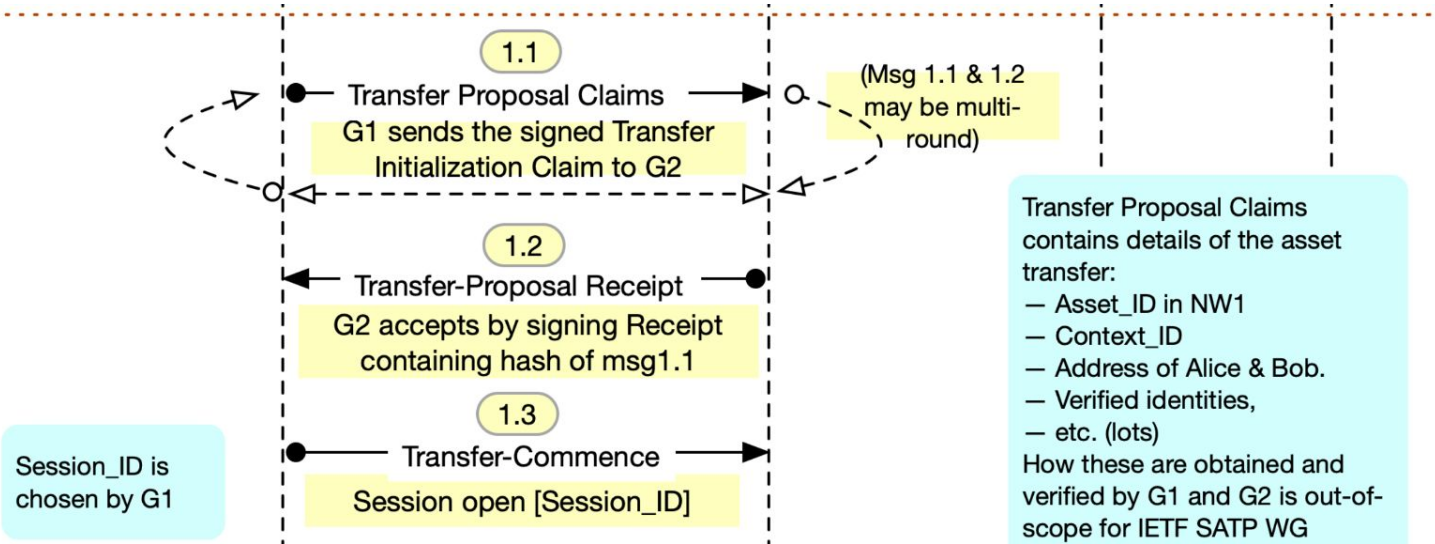
Crash recovery state

- See reference implementation:
<https://github.com/hyperledger/cacti/tree/main/packages/cactus-plugin-odap-hermes>
- RECOVERY_STATE: Session ID, gateways, transfer context, stage, step, operation (init, exec, done, ack, fail), and rollback logic (needs to be implemented in a per-phase basis)
- Different parts of RECOVERY_STATE may be used for an abort vs rollback (example for abort and rollback, one should invalidate session ID; rollback logic only for rollback).

Non-critical crash

Stage 1

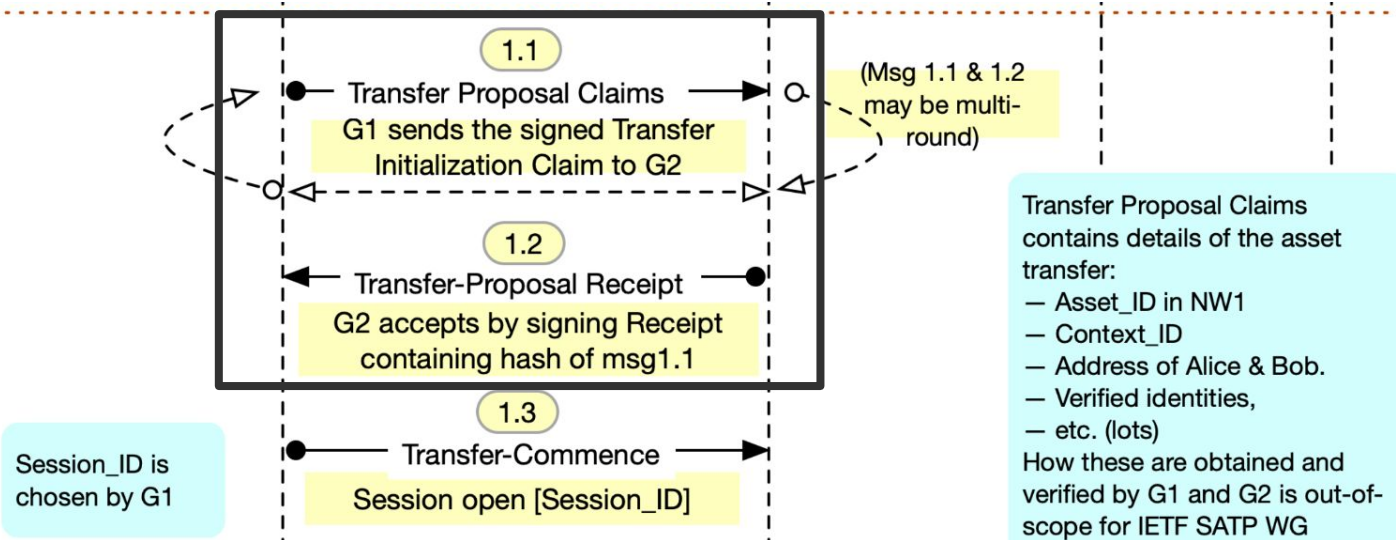
Unlocked
(assigned)



Non-critical crash

Stage 1

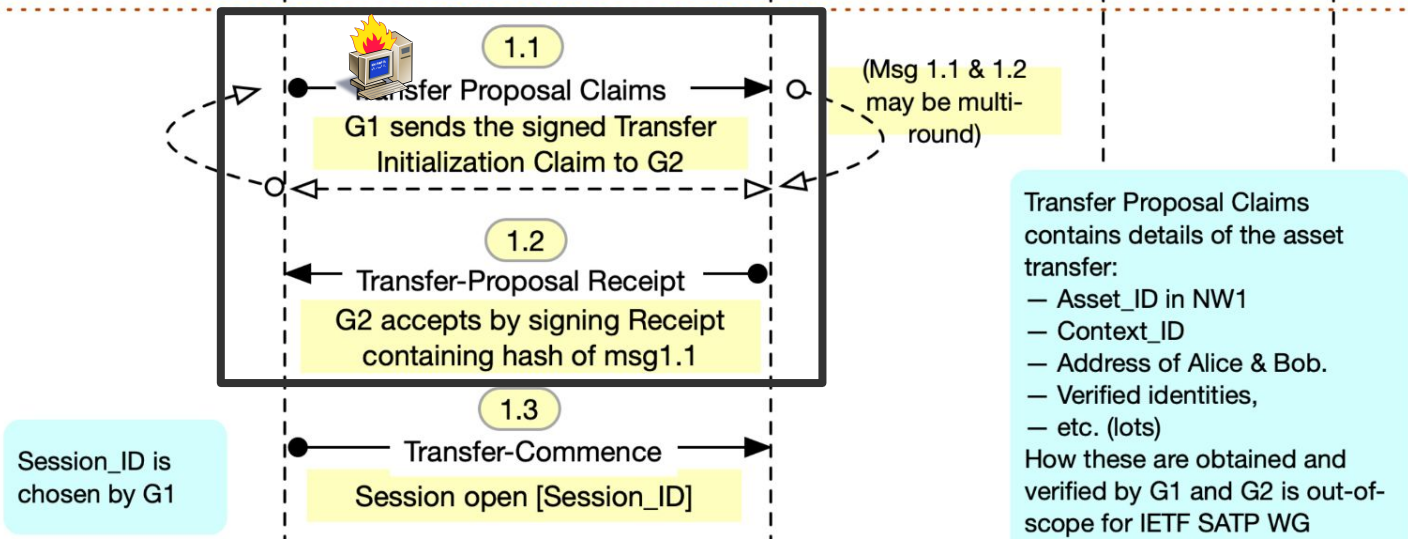
Unlocked
(assigned)



Non-critical crash

Stage 1

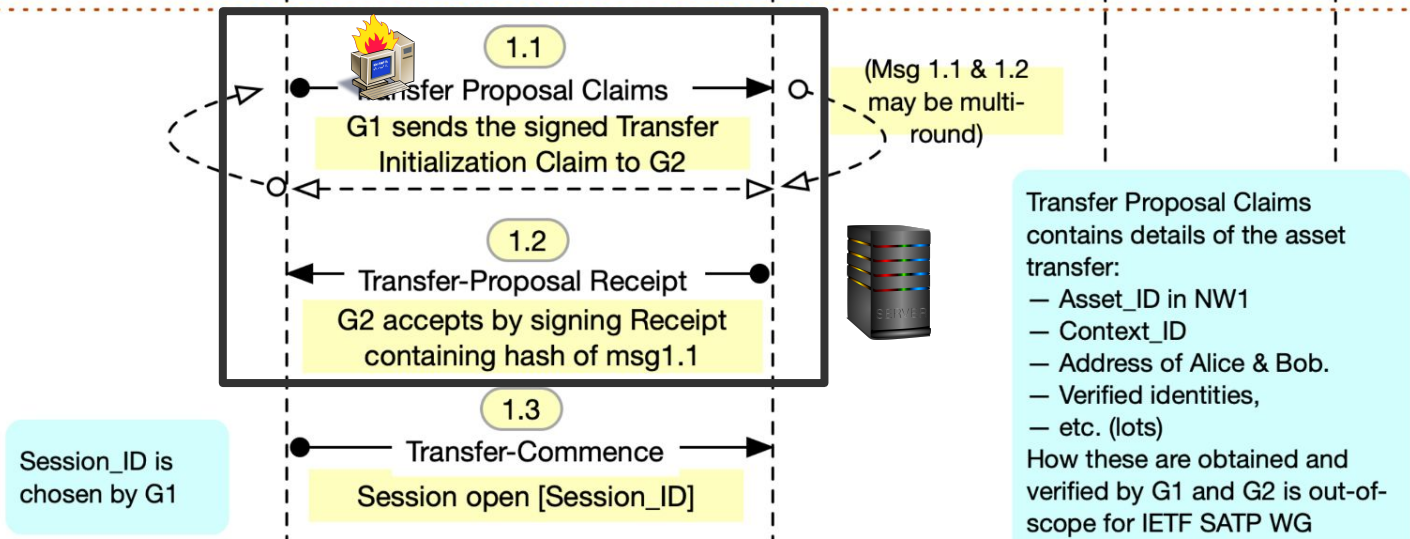
Unlocked
(assigned)



Non-critical crash

Stage 1

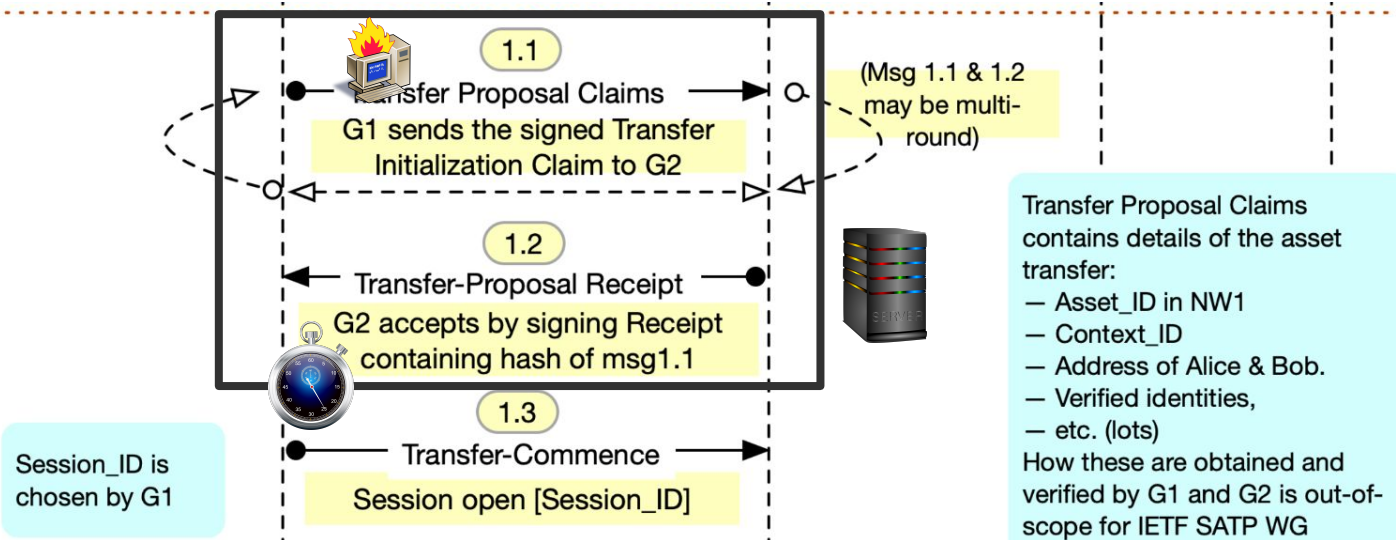
Unlocked
(assigned)

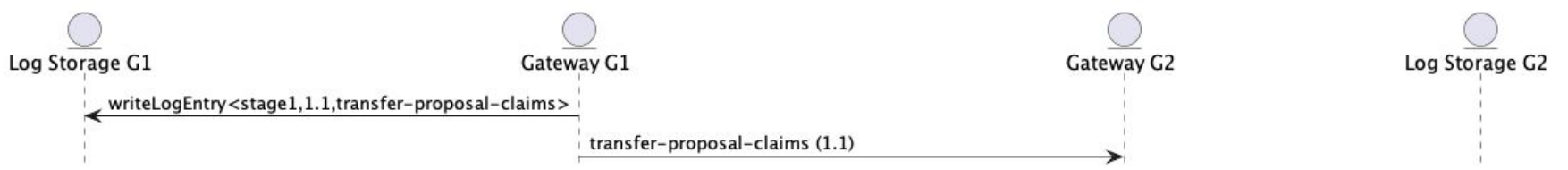


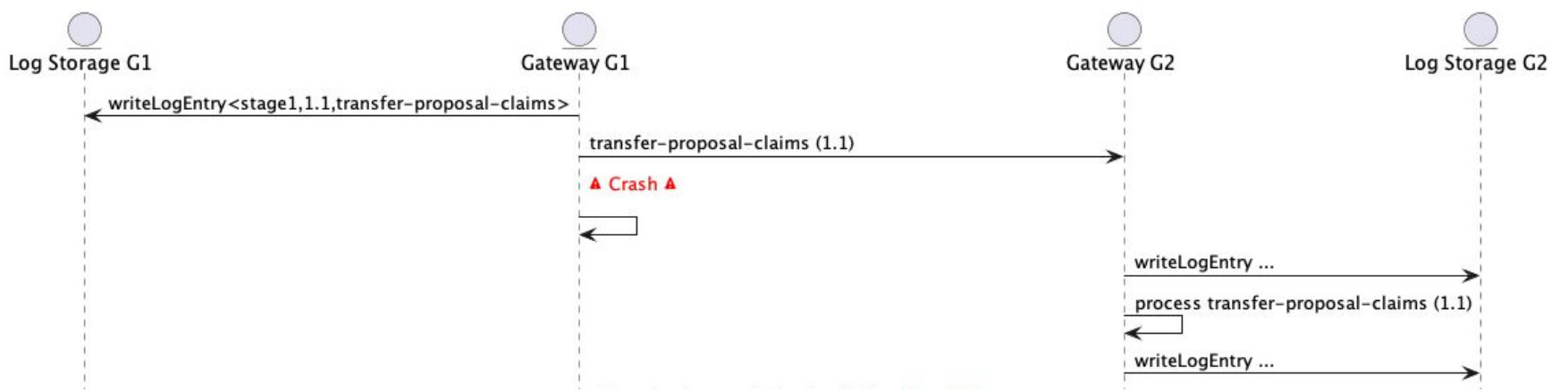
Non-critical crash

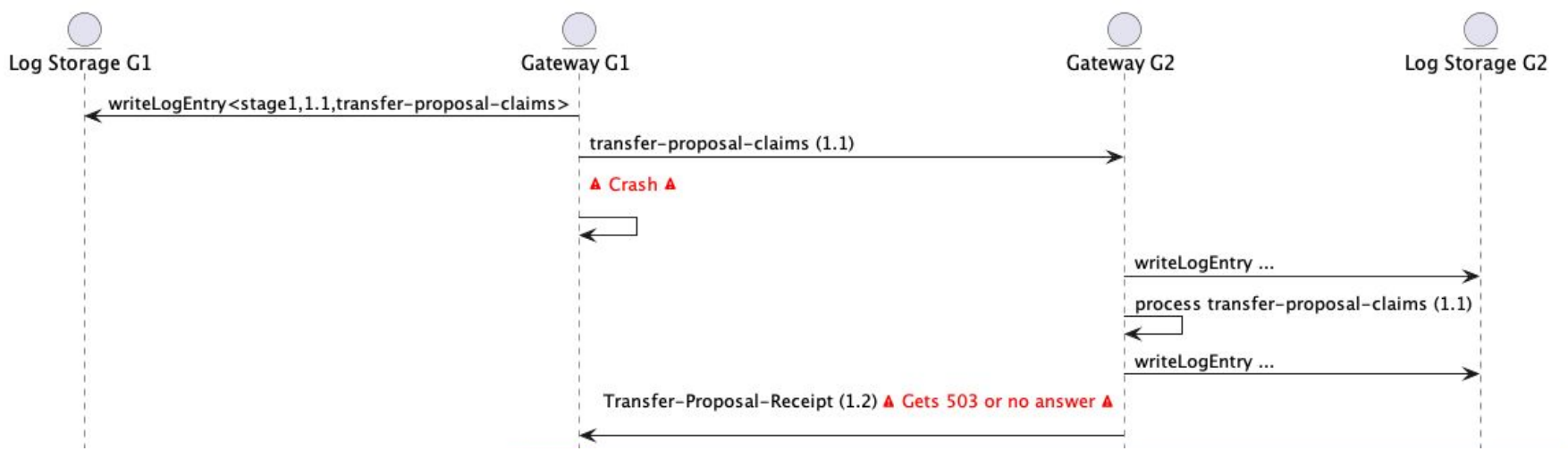
Stage 1

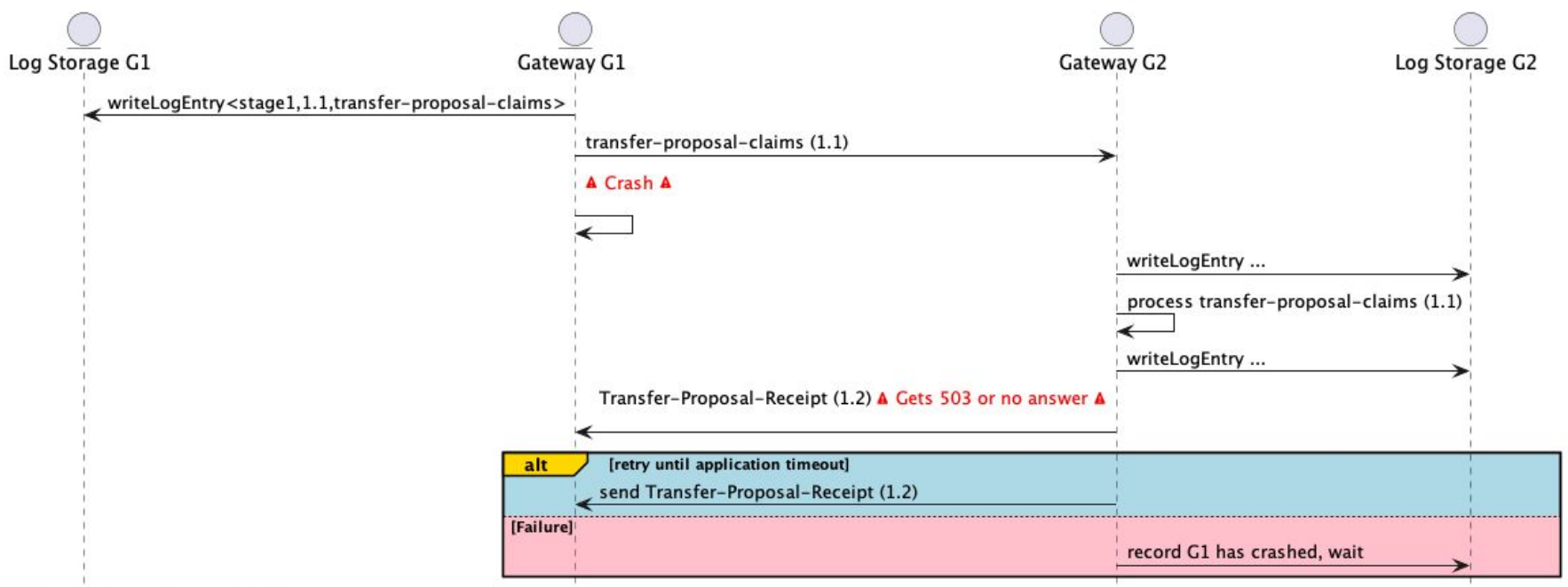
Unlocked
(assigned)

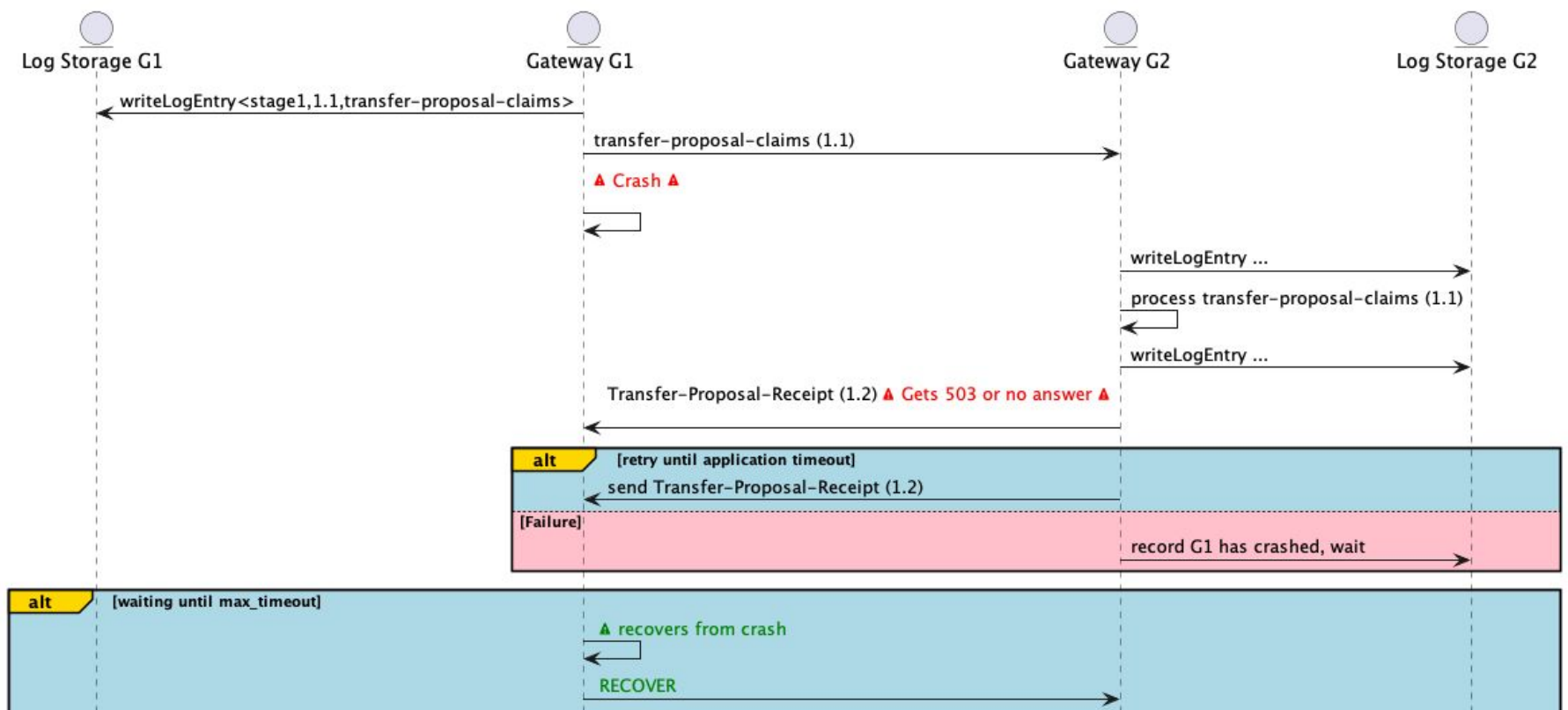


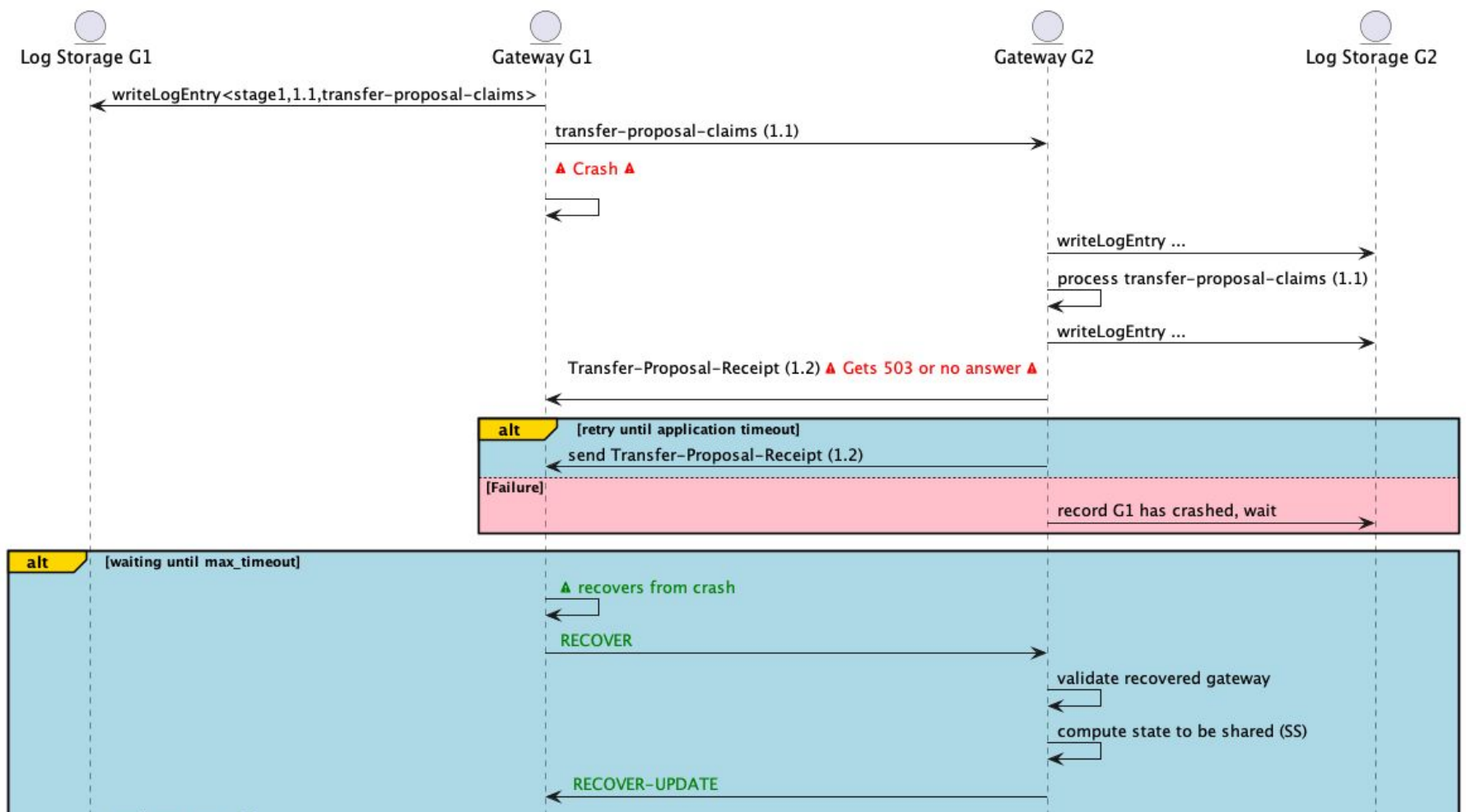


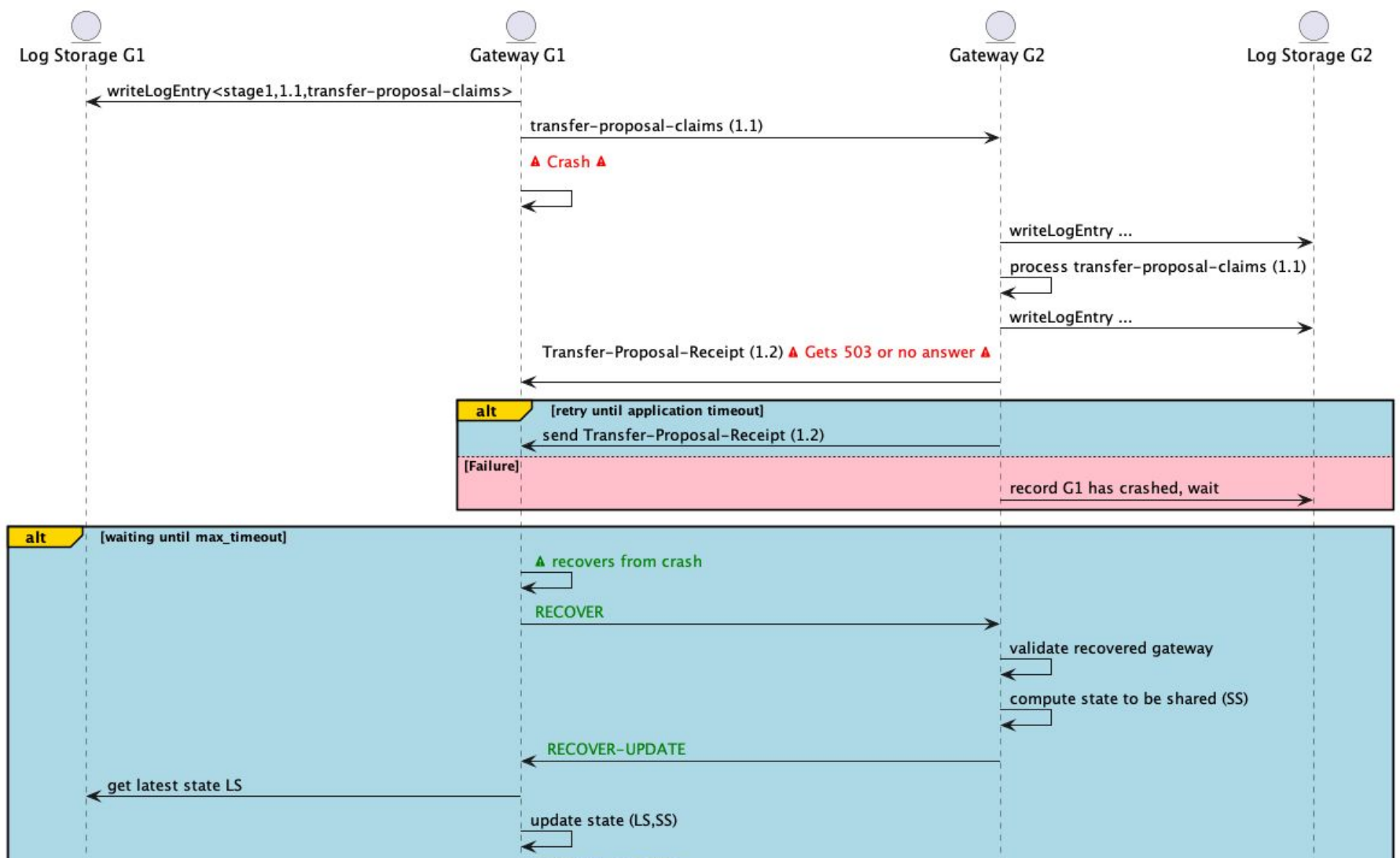


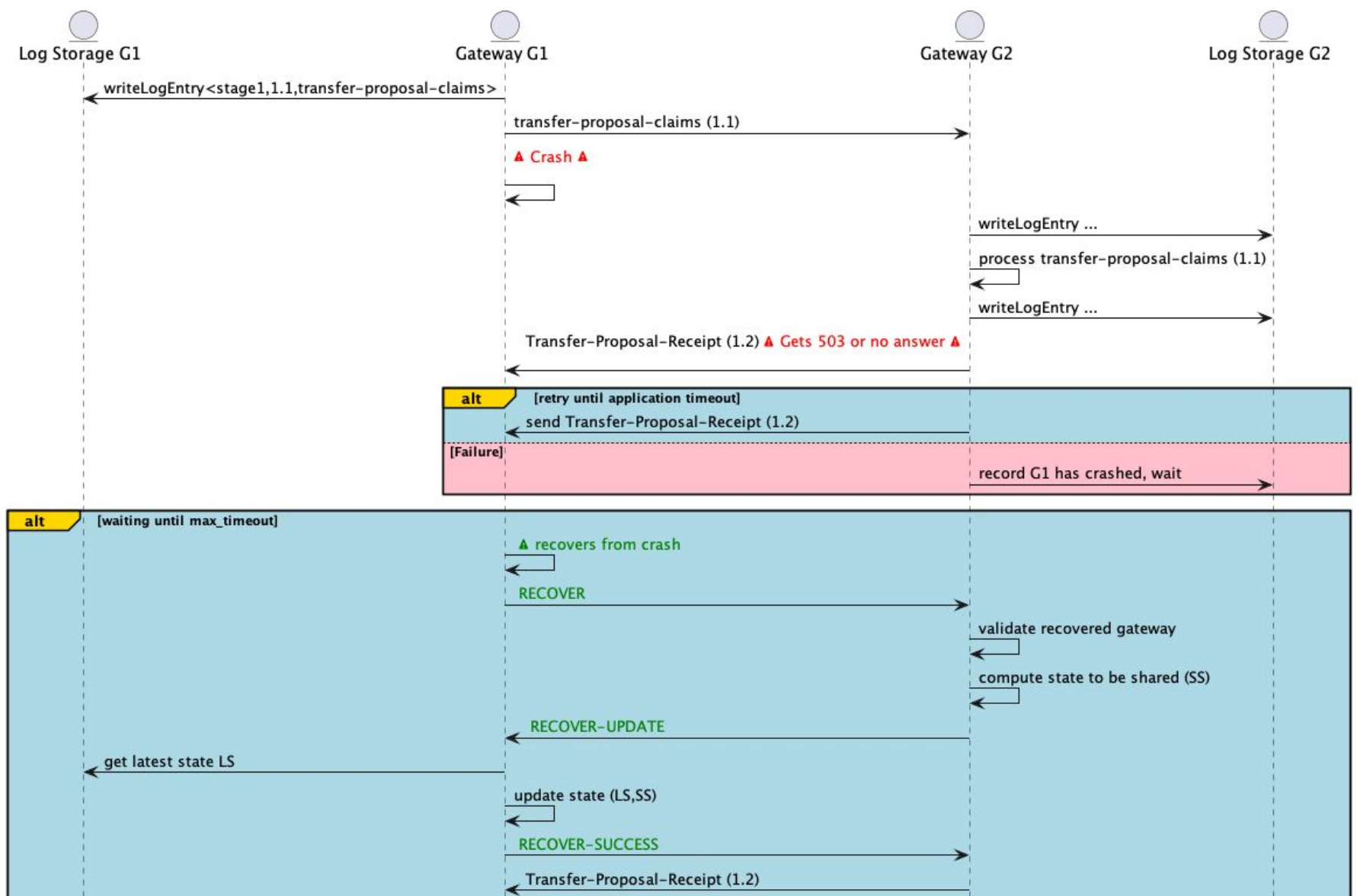






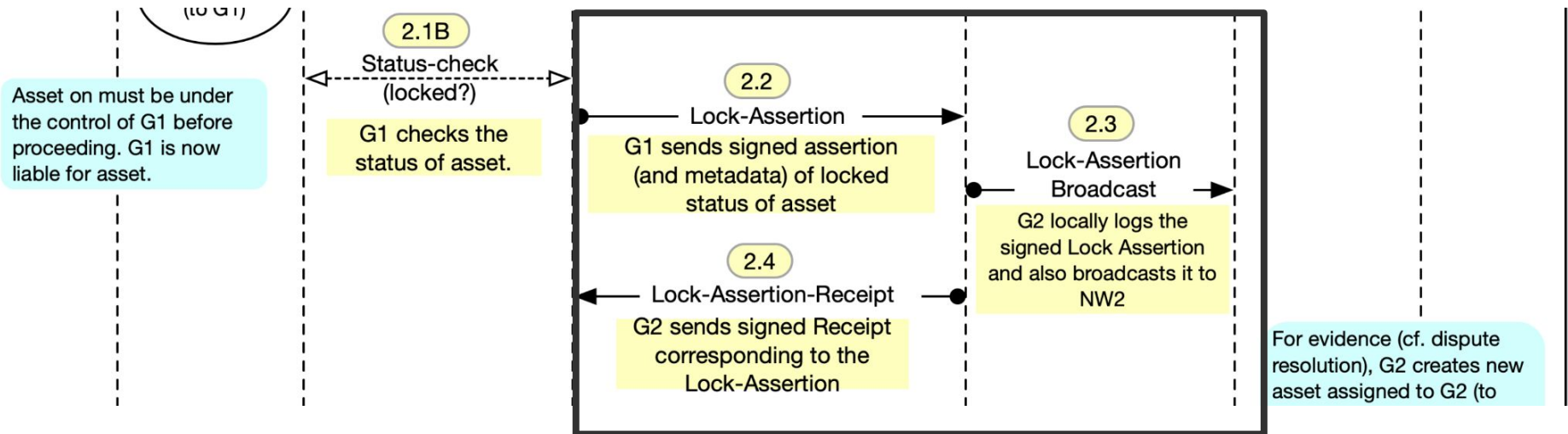




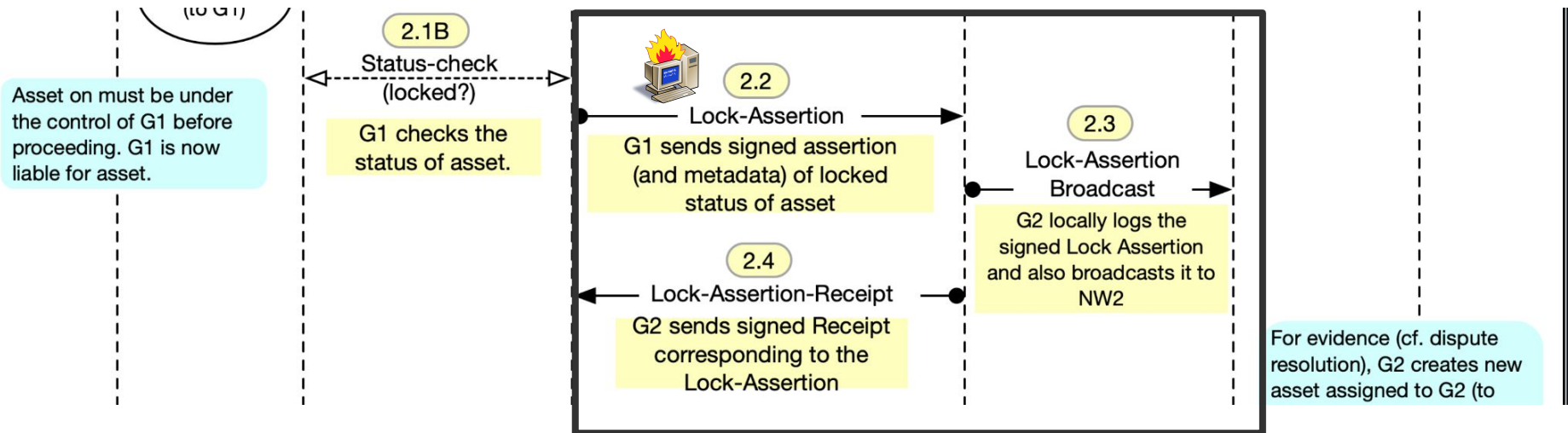




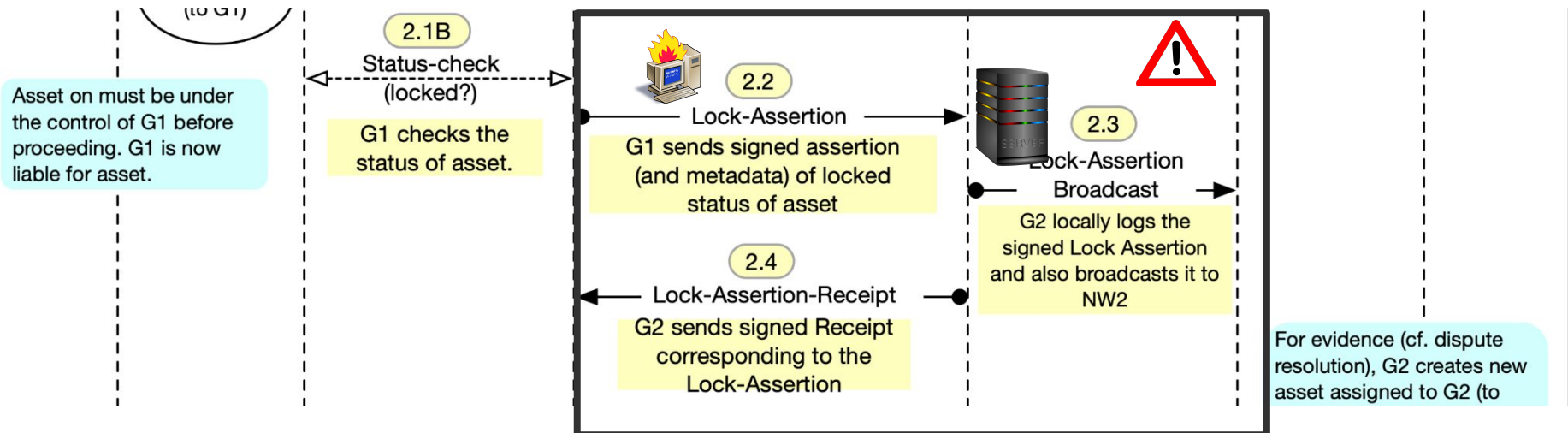
Critical crash



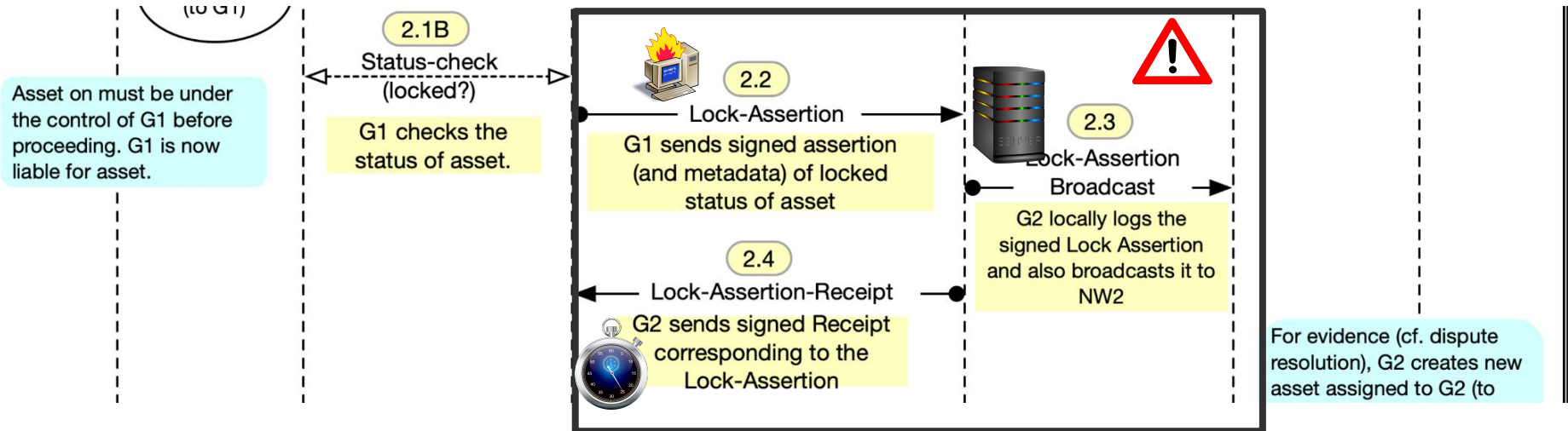
Critical crash

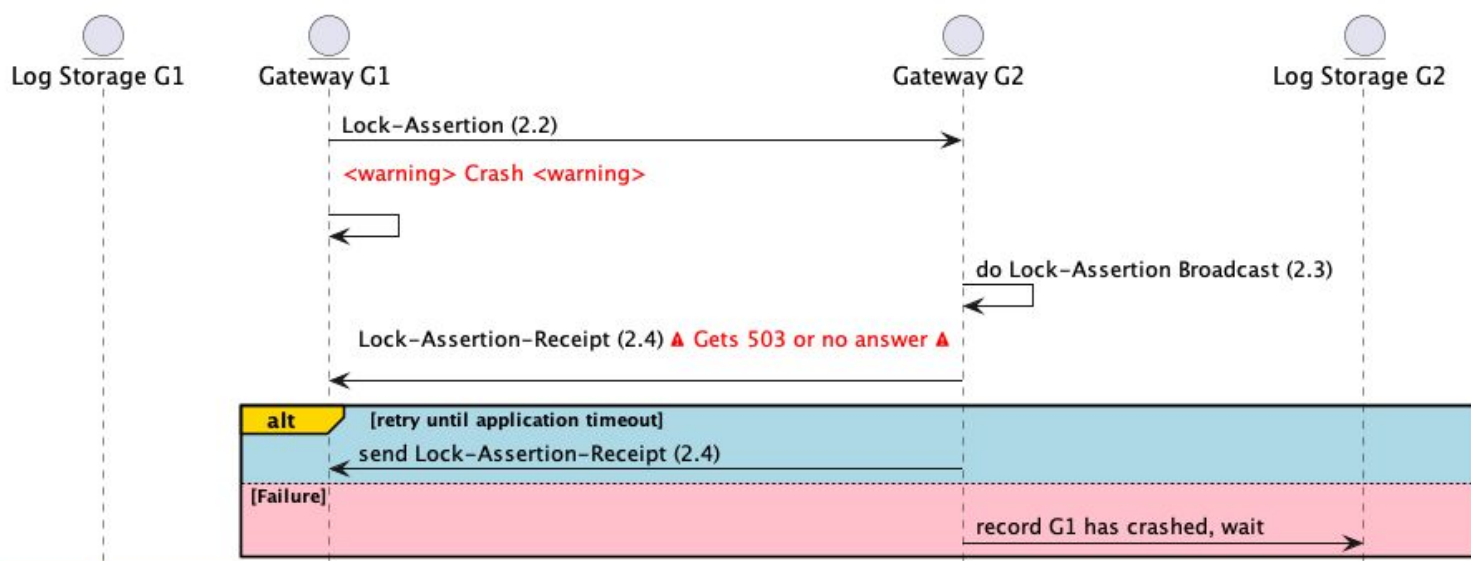


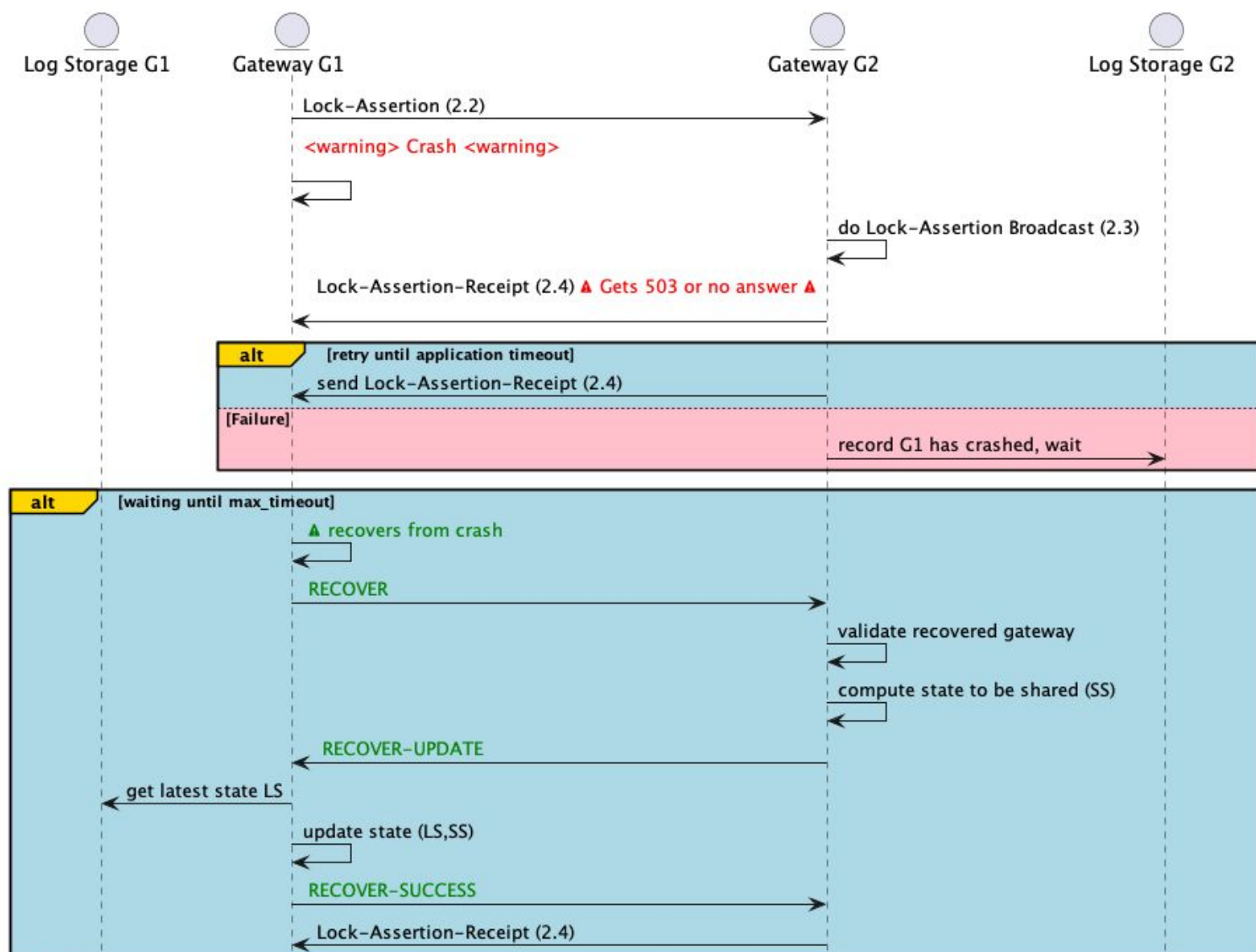
Critical crash

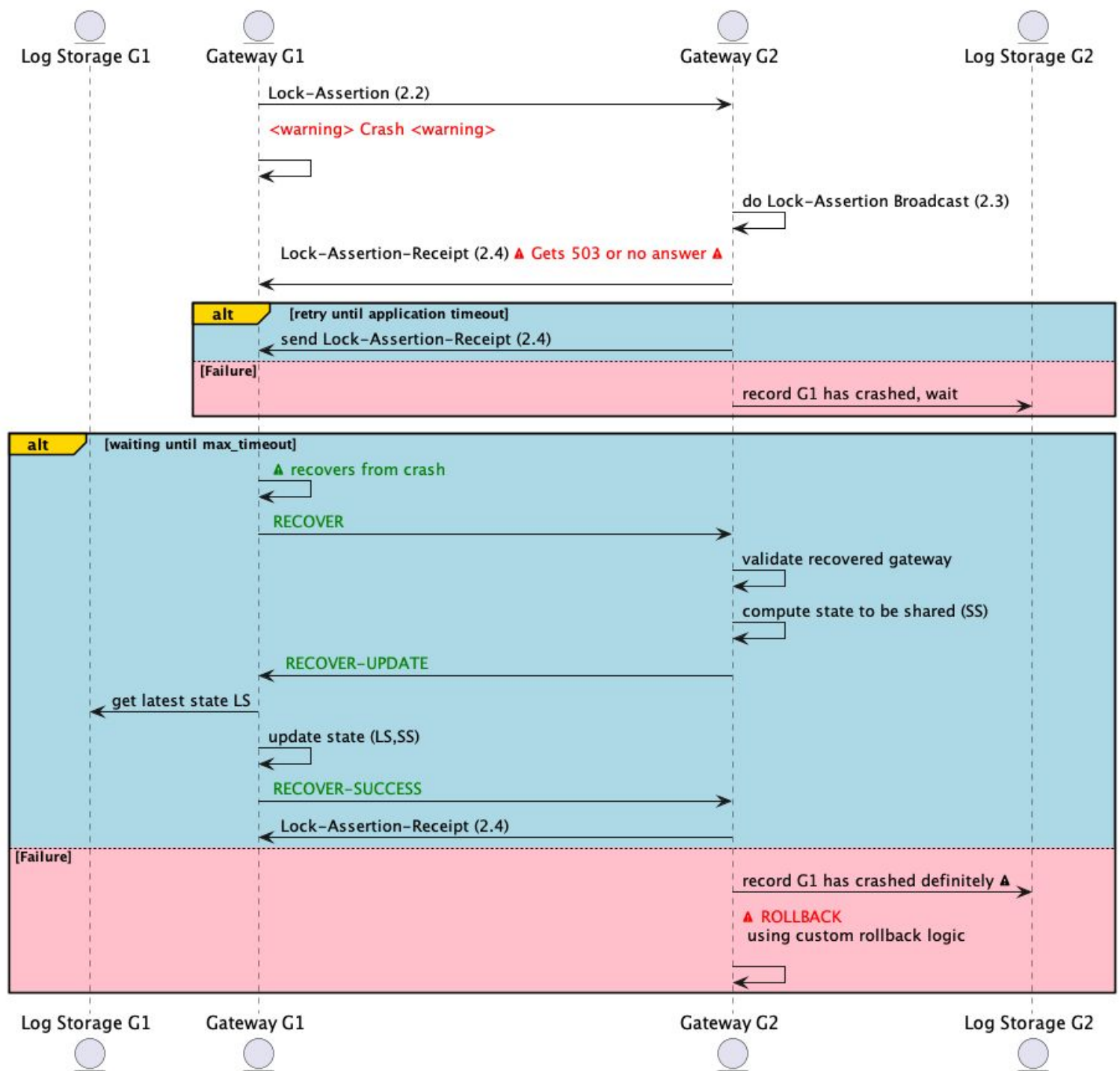


Critical crash









Example rollback logic for step Lock Assertion (2.2) - by G I

- EFFECT TO BE ROLLED BACK: X assets were locked/burned, we need to unlock it/mint it on the source chain
- INPUT: asset id and amount to unlock
- OUTPUT: success or fail (if fail requires manual intervention)
- PROCEDURE:
 - G I create transaction tx unlocking X assets
 - G I propagates transaction tx to source network and waits for confirmation
 - G I logs confirmation on the decentralized log

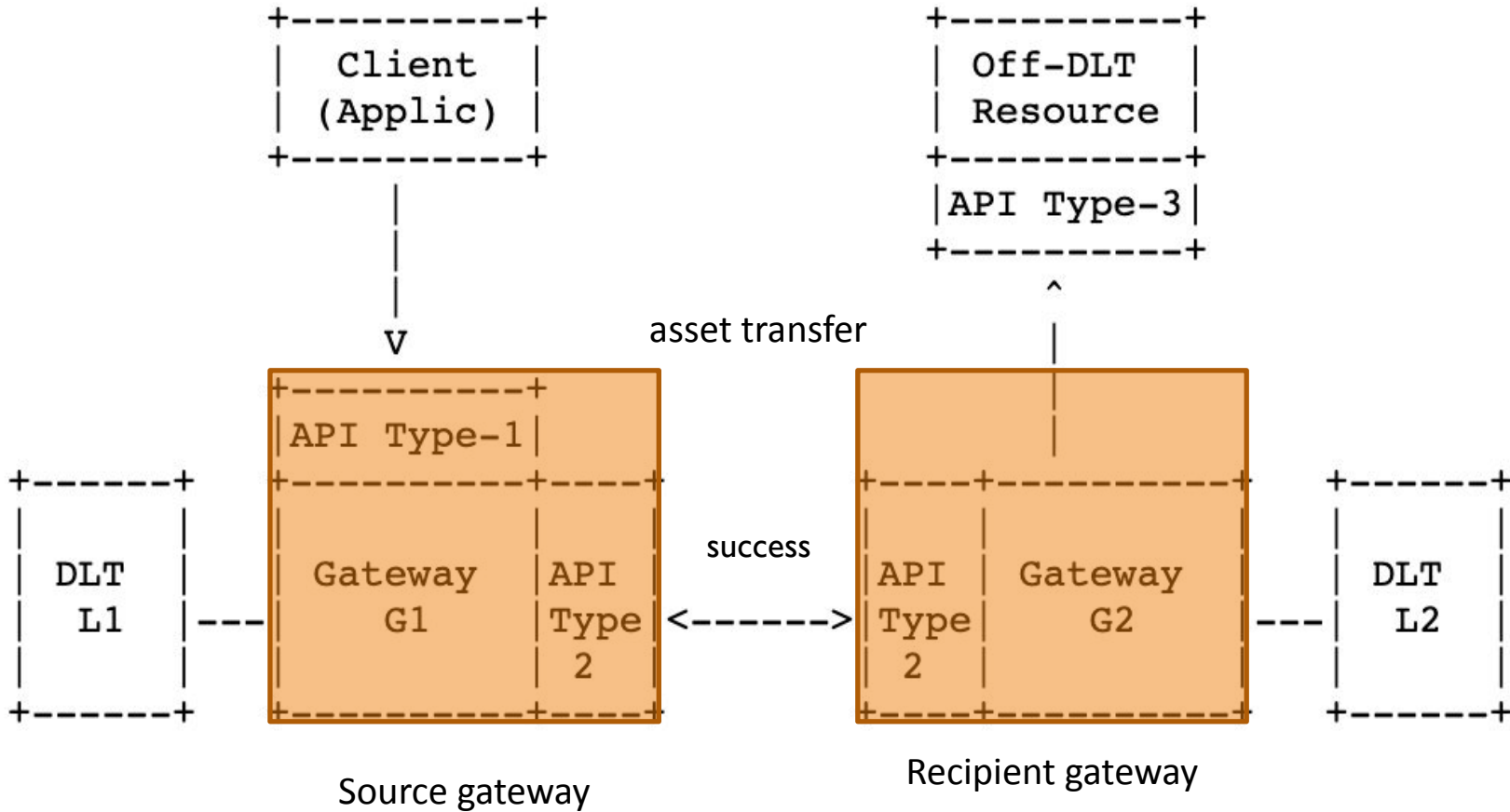
Session Resumption Discussion

- Should session resumption be explicit or implicit?
 - We are considering it implicit but may define interface to do it explicitly
- Timeout on recover message. what happens if a gateway G1 crashes, it recovers, and when is sending a recover message, G2 crashes, i.e., problem of nested failures?
 - Possible solution: timestamp ordering, and rollbacks with lower timestamp have priority?

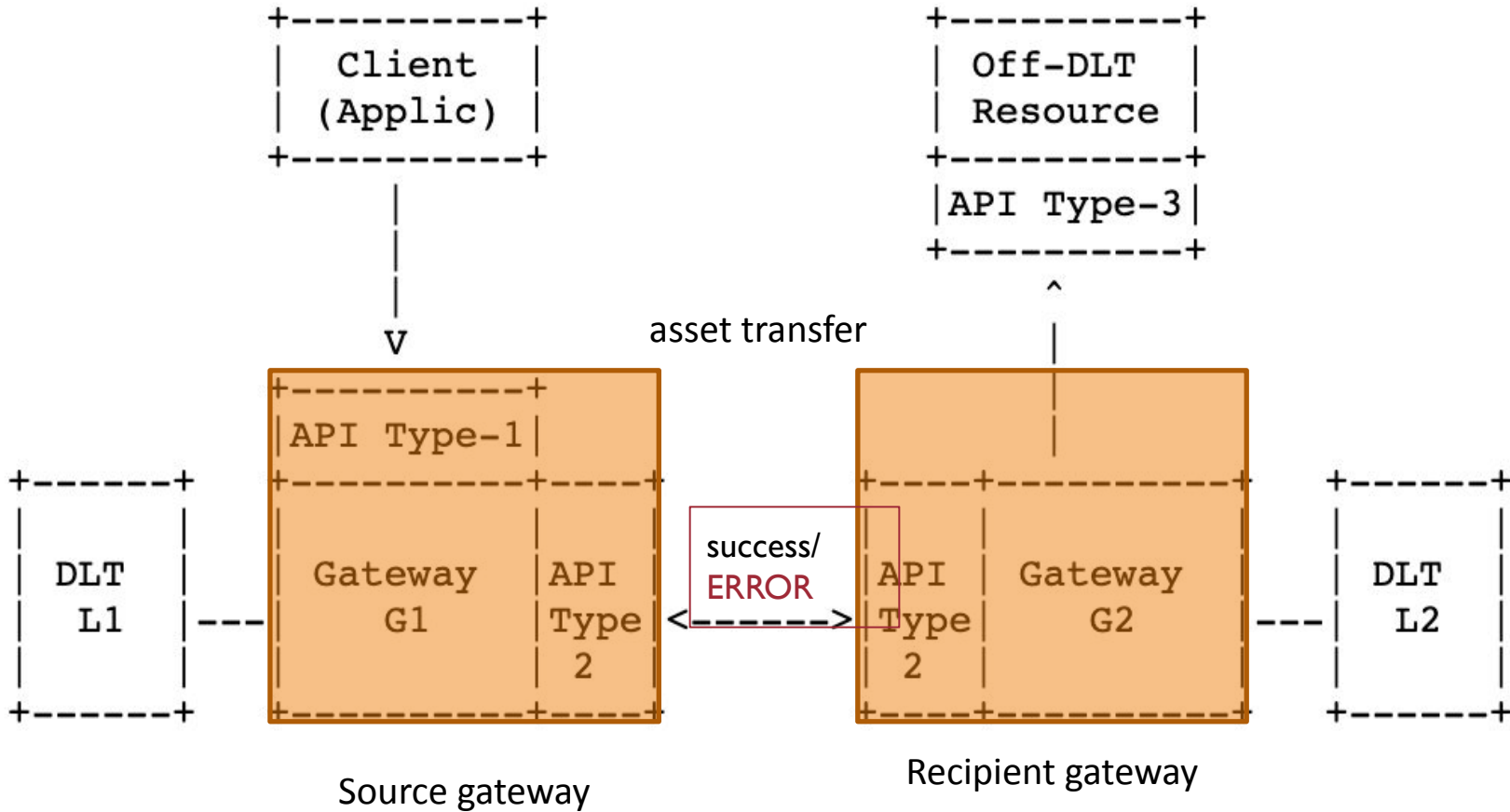
Rollback Discussion

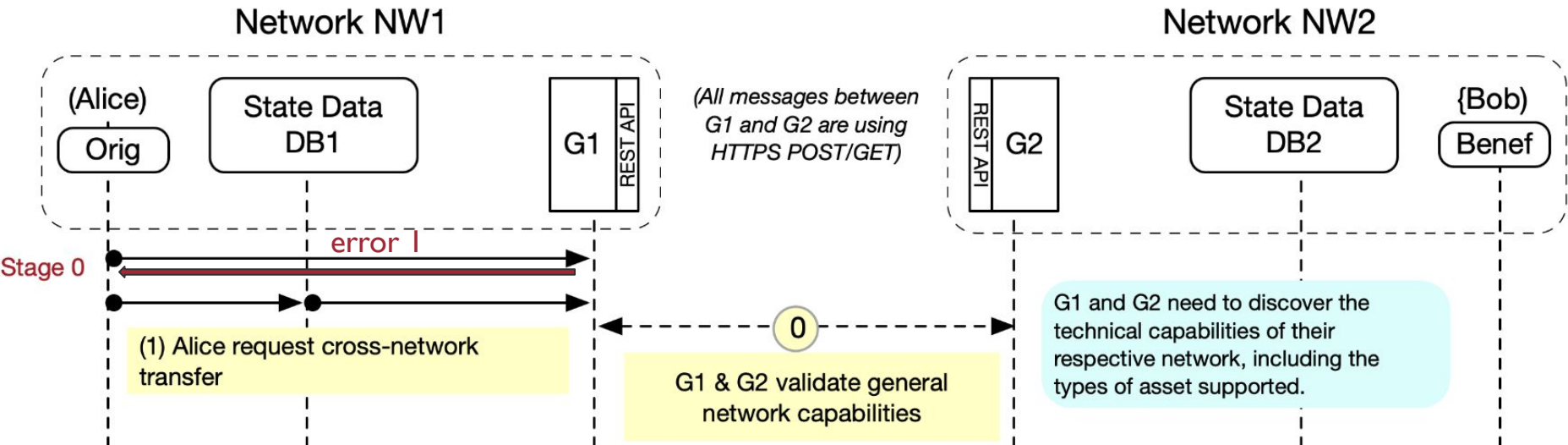
- Burn-Mint is more secure than Lock-Unlock because of honeypots. What are the tradeoffs in terms of rollbacks for the two paradigms?

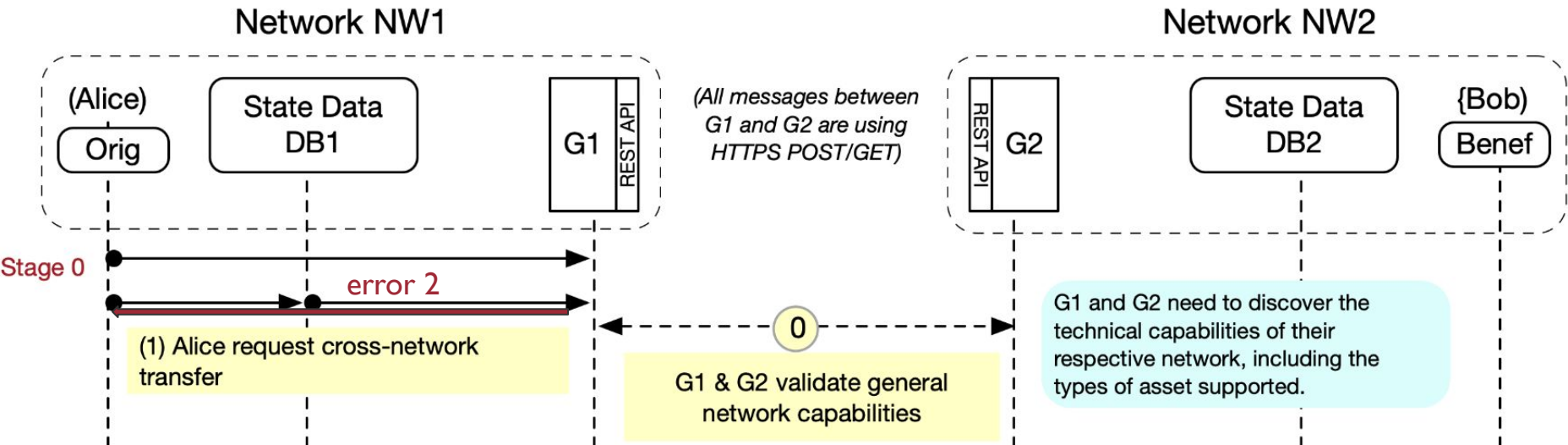
Error messages

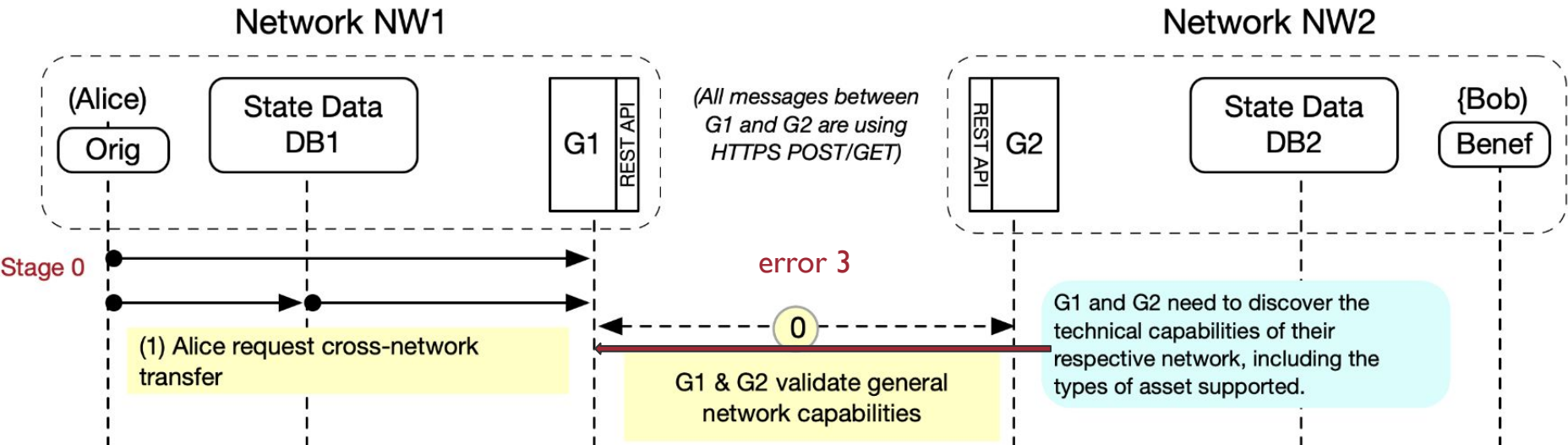


Error messages









Error schema

The Error message is sent by a gateway to a gateway or a gateway to a client to indicate a message error.

- error type - Code list containing a code that identifies the error condition.
- code - Specification of the error, in coded form.
- ...

Error schema

- description - Description of the error found. The description is a standardized message that is used to provide information about an error that occurred during the processing of a message. The error description message provides information about the type of error, the location of the error, and other relevant details.
- proprietary - Specification of the error, in free format (specific to the party that generates the error, and it is not standardized). The error proprietary message can be used to provide additional details that are not available in the standard error messages.

Error schema

A.3. Stage 2

Errors for messages 2.1 and 2.2

- [err_2.1] Badly formed message.
- [err_2.2] Incorrect parameter.
- [err_2.3] ACK mismatch.

Error reports for Messages 2.3.A and 2.3.B (Lock failed)

- [err_2.3.1] Asset already locked.
- [err_2.3.2] Insufficient funds to complete transaction.
- [err_2.3.3] Time-out on lock attempt.
- [err_2.3.4] Network consensus protocol error.

Error reports for Messages 2.3.A and 2.3.B (Read failed)

- [err_2.3.5] Gateway configuration error.
- [err_2.3.6] Insufficient read/write permission.

Error reports for Message 2.4 (Lock Assertion)

- [err_2.4.1] Badly formed message: badly formed Claim.
- [err_2.4.2] Badly formed message: bad signature.
- [err_2.4.3] Badly formed message: wrong transaction ID.
- [err_2.4.4] Badly formed message: Mismatch hash values.
- [err_2.4.5] Expired signing-key certificate.
- [err_2.4.6] Expired Claim.

Error schema

A.4. Stage 3

Error reports for Message 3.1 (Commit Prepare):

- [err_3.1.1] Badly formed message: wrong transaction ID.
- [err_3.1.2] Badly formed message: mismatch hash value.
- [err_3.1.3] Incorrect parameter.
- [err_3.1.4] Message out of sequence

Error reports for Message 3.2 (ACK-Prepare):

- [err_3.2.1] Badly formed message: wrong transaction ID.
- [err_3.2.2] Badly formed message: mismatch hash value.
- [err_3.2.3] Incorrect parameter.
- [err_3.2.4] Message out of sequence

Error reports for Message 3.3A and 3.3B (Create asset fail):

- [err_3.3.1] Asset already exist.
- [err_3.3.2] Insufficient funds to complete transaction.
- [err_3.3.3] Time-out for transaction finality.
- [err_3.3.4] Network consensus protocol error.

Update on open-source SATP implementations

- Stable SATP implementation, trusted, centralized node, with crash recovery (outdated):
<https://github.com/hyperledger/cacti/tree/main/packages/cactus-plugin-odap-hermes>
- On-going implementation, relay, decentralized node (with Rama):
<https://github.com/outsidethecode/cacti/pull/>

Next steps

- Confirm error scope
- Define various errors across four stages of SATP
- Find a meaningful message for each error code
- Update/continue developing open-source implementations

Thank you and Q&A

rbelchior@blockdaemon.com