iSCSI Error Recovery

Mallikarjun Chadalapaka

Randy Haagens

Julian Satran

London, 6-7 Aug 2001

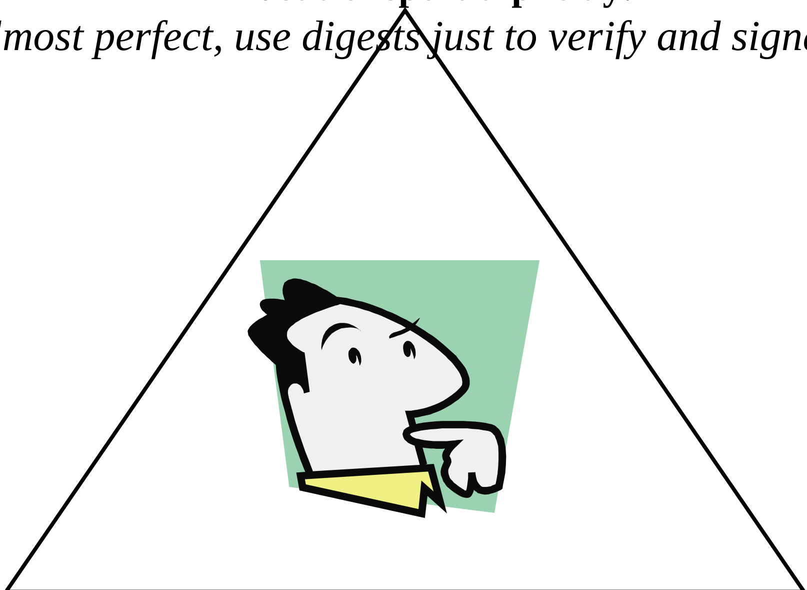
Why do we care?

- Error statistics – an attempt to extrapolate (innovatively) from an experiment conducted at Stanford:
  - Indicate errors are quite possible with data
  - Less frequent with headers
  - Enough to worry
  - Not enough to build complex recovery
- The basic mechanisms built for detection are the expensive part (counting).
- Two major sources of errors (together: ***transport path***) –
  - Unknown TCP checksum "escape" performance
  - Unknown Proxy performance (TCP *and* iSCSI)

Error Management Design challenges in iSCSI

Three different camps of thinking on ***transport path*** performance….

Trust transport explicitly!
(*transport is almost perfect, use digests just to verify and signal failure to SCSI***)**



Trust transport implicitly!
(*transport is perfect, iSCSI digests aren't necessary***)**

Can't trust transport!
(*transport is non-deterministic, do full recovery***)**

Current analysis and experimental evidence points to reality being somewhere between "Trust transport explicitly" and "Can't trust transport" camps.

Error Recovery Philosophy in Rev07 Draft

- Mandate only the baseline session recovery mechanism, but with four defined levels recovery.
  - Within-command, to handle dropped PDUs but no command restart.
  - Within-connection, to handle dropped command/status but no connection restart.
  - Within-session (aka connection), to handle TCP connection failures in the same session context.
  - Session recovery, the worst-case and minimally required recovery, terminates all I/Os and ends the session.

Error Recovery Philosophy in Rev07 Draft (contd.)

- Ensure interoperability between any two implementations supporting different levels of error recovery.

- Define the error recovery mechanisms to ensure command ordering even in the face of errors, for initiators that demand ordering.

- Command counting is needed for ordering and flow control.

- Status sequence tracking and data sequence tracking (StatSN and DataSN) can be dispensed with for only-session recovery implementations.

How much does it cost to do Error Recovery?

- No addition on the fast path (counting needed for other reasons)
- Logic on the slow path with a moderate complexity (in comparison, certainly less than security…)
- Mechanisms seem to be now well understood.

iSCSI's Error Management Tools

- Header and Data digests
- Selective negative acknowledgement (SNACK)
- Recovery R2T (if allowed by "DataSequenceOrder=no")
- Unsolicited NOP-IN
- Three flavors of "retry"
  - Command replay (retry on the same connection after status delivery)
  - Command failover (retry of a command on new connection)
  - Command plugging (retry when a gap is suspected in command sequence)

Issue #1: Should iSCSI define SNACK?

Cons

- SNACK purports to recover "dropped" PDUs, but itself is susceptible to digest failures, and currently not architected to do timers/retransmissions for a robust recovery.
  - Options:
    - a) Assign a CmdSN (may lead to resource deadlocks!).
    - b) Accept the non-determinism (since the odds are very low).
    - c) Leave it to implementations to retransmit SNACKs (if they can deal with potential duplicate data PDUs).
    - d) Define timer-based SNACK retransmissions in the protocol (*more and more* complexity!)
    - e) Drop SNACK!

Issue #1: Should iSCSI define SNACK? (contd.)

- Through SNACK, iSCSI assumes traditional “transport” functions, even when it is an application layer protocol in reality.
  - Options:
    - a) Keep it since TCP’s checksum escape rate is uncertain.
    - b) Rely on IPSec always for data integrity (expensive!)
    - c) Drop SNACK to consider for iSCSI-02 (TCP checksum could conceivably be adequate as well).

- Optimizing the demands on memory and the back-end for targets supporting SNACK requires data ACKs!
  - Options:
    - a) Mandate data ACKs whenever SNACK is supported.
    - b) Assume that medium can be accessed to satisfy SNACKs (doesn’t work for non-idempotent devices!).
    - c) Mandate I/O replay buffer support for SNACK (expensive!).

Issue #1: Should iSCSI define SNACK? (contd.)

Pros

- ✓SNACK retrieves lost status PDUs, which would otherwise force a connection recovery resulting in several SCSI I/O errors.
- ✓Since the draft allows the notion of a command retry, SNACK can be considered merely a special case of command retry (partial I/O).
- ✓Partial I/O recovery was considered a requirement for tape support in Networked Storage (the FC-TAPE effort in Fibre channel), and SNACK delivers it.
- ✓SNACK enables a swift recovery of lost PDUs closer to the source of error, as opposed to propagating the error up the stack resulting in a longer error recovery time.

Issue #1: Should iSCSI define SNACK? (contd.)

Bottomline:

What do we gain if we drop SNACK?

Less complex implementations, Less complex specification.

What do we lose if we drop SNACK?

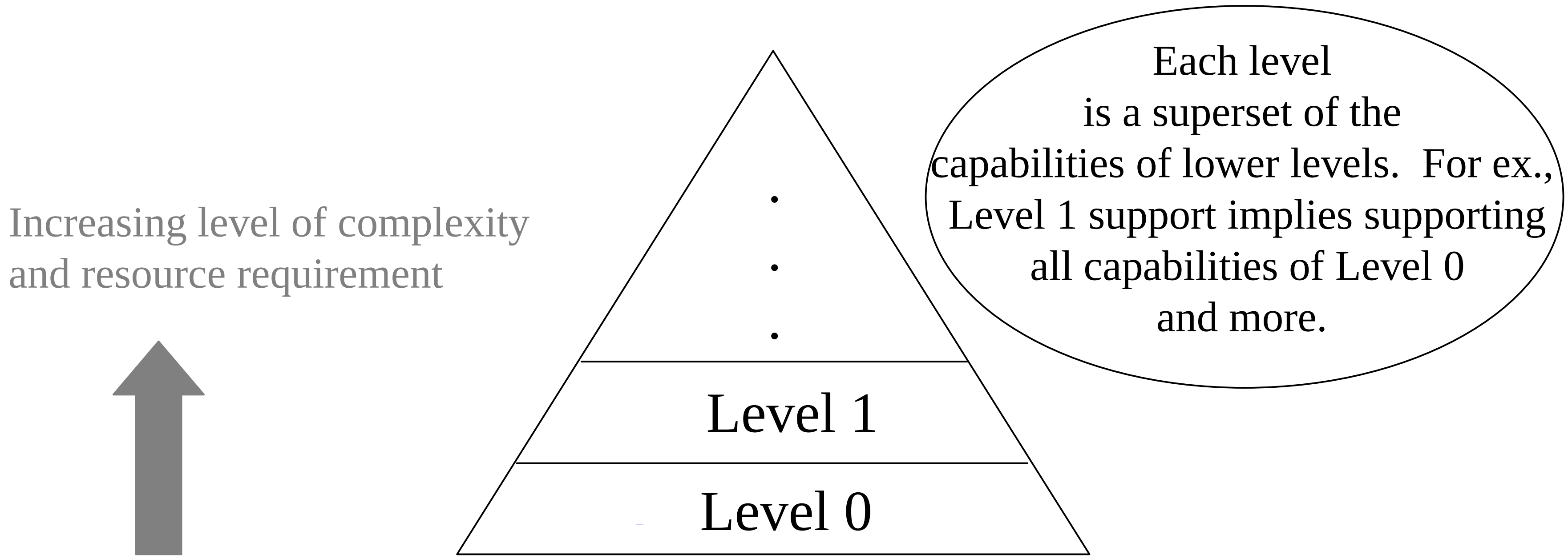
If ***transport path*** failure rates are *extremely low*: nothing!

If failure rates are *moderately high*: a capable specification that saves link & back-end bandwidth (by allowing partial I/Os).

If failure rates are *too high*: not much since SNACK isn't architected to be robust!

- **Proposal is to continue to define SNACK for iSCSI-01. Assumption is that tapes supporting queueing (very few, if any!) must support I/O replay buffer for SNACK during iSCSI-01.**

Issue #2: How to layer error recovery capabilities for simplicity?



❖ Proposal is to create a hierarchy.

- ✓ One text key - "ErrorRecoveryLevel=n" - to advertise/negotiate ALL error recovery capabilities.
- ✓ Ability to distinguish a transient recovery attempt failure from that of the absence of the recovery capability.
- ✓ Fewer choices of implementation, significantly reducing the test matrix (from 2^{n-1} to n).

Issue #3: What is a reasonable Error Recovery hierarchy?

Recovery layering can be reasoned as:

Command replay ***4***

Connection Recovery ***3***

Within-command Recovery ***2***

Within-connection Recovery ***1***

Session Recovery ***0***

wants a guarantee that a redoing an I/O would deliver the exact same data, even on conn. failures.

wants connection failures not to cause any SCSI errors.

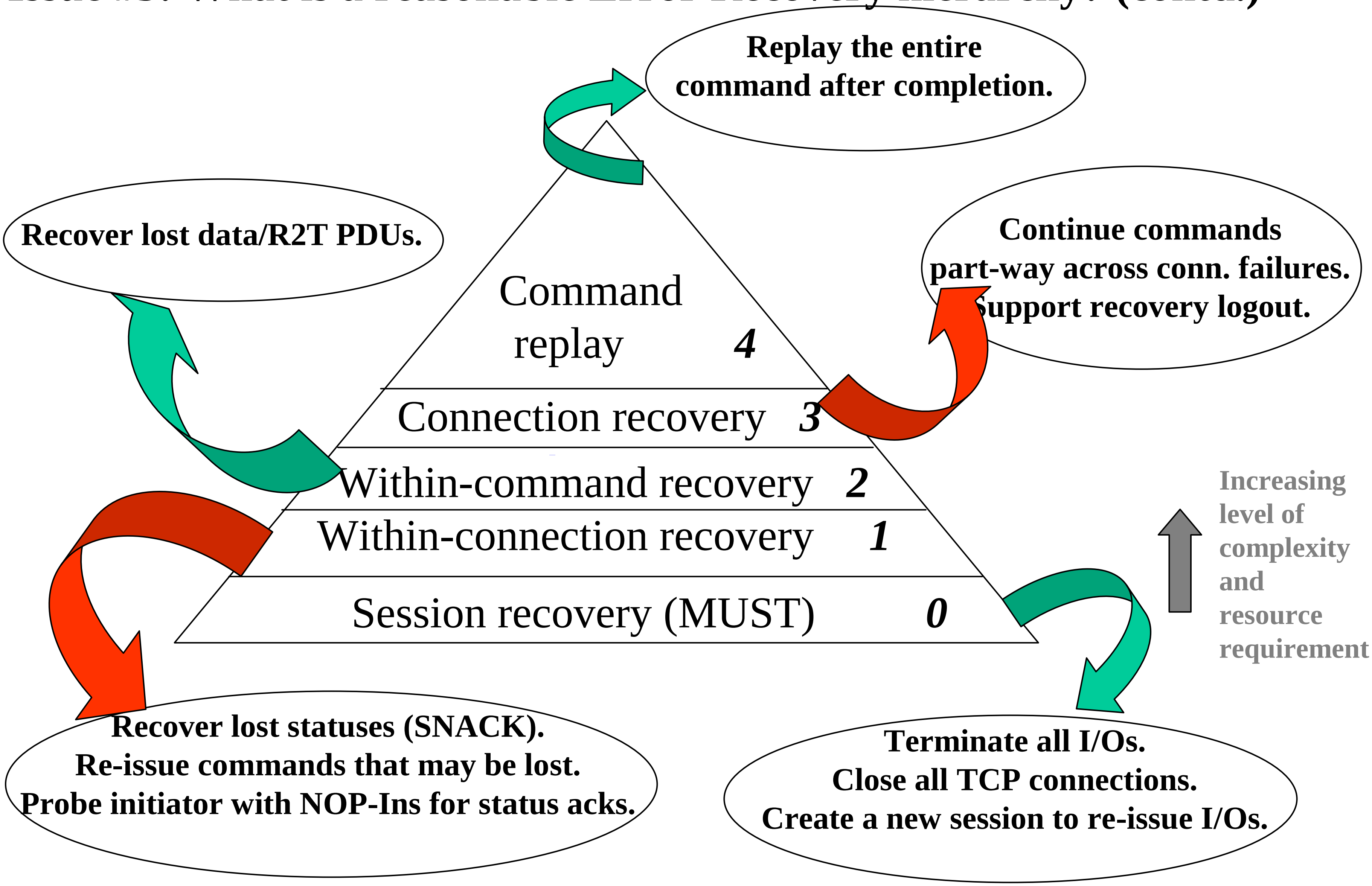
wants digest errors not to cause any task failures.

wants to prevent digest errors from destroying the session/connection.

don't care if any errors destroy the session, SCSI/wedge drivers take care of all recovery.

Since incremental aspirations are most likely to be -

Issue #3: What is a reasonable Error Recovery hierarchy? (contd.)



Issue #3: Why this model?

✓Incremental book-keeping & resource requirements.

Recovery Level transition	Incremental requirement
[0] Session	Mandatory to support.
[0⇨1] Session ⇨Within-connection	Atmost one PDU retransmission per task.
[1⇨2] Within-connection ⇨Within-command	Retransmit possibilities include data PDUs.
[2⇨3] Within-command ⇨ Connection	Retransmission across connections.
[3⇨4] Connection ⇨ Command replay	Replaying the entire command (all PDUs).

Issue #3: Why this model? (contd.)

✓Rev07 already defines part of the proposed hierarchy, by mandating data/status PDU retransmission support for Connection Recovery support (currently via the CommandFailoverSupport key).

✓Command replay with most resource requirements (with a replay buffer) and highest implementation complexity is positioned at the top.

✓This model maintains the current idea that implementations supporting only Level 0 do not have to keep track of any sequence numbers (except CmdSN), since <u>any</u> digest failure would lead to session recovery.

❖**Proposal is to adopt this model into iSCSI.**

So, to summarize the proposals…

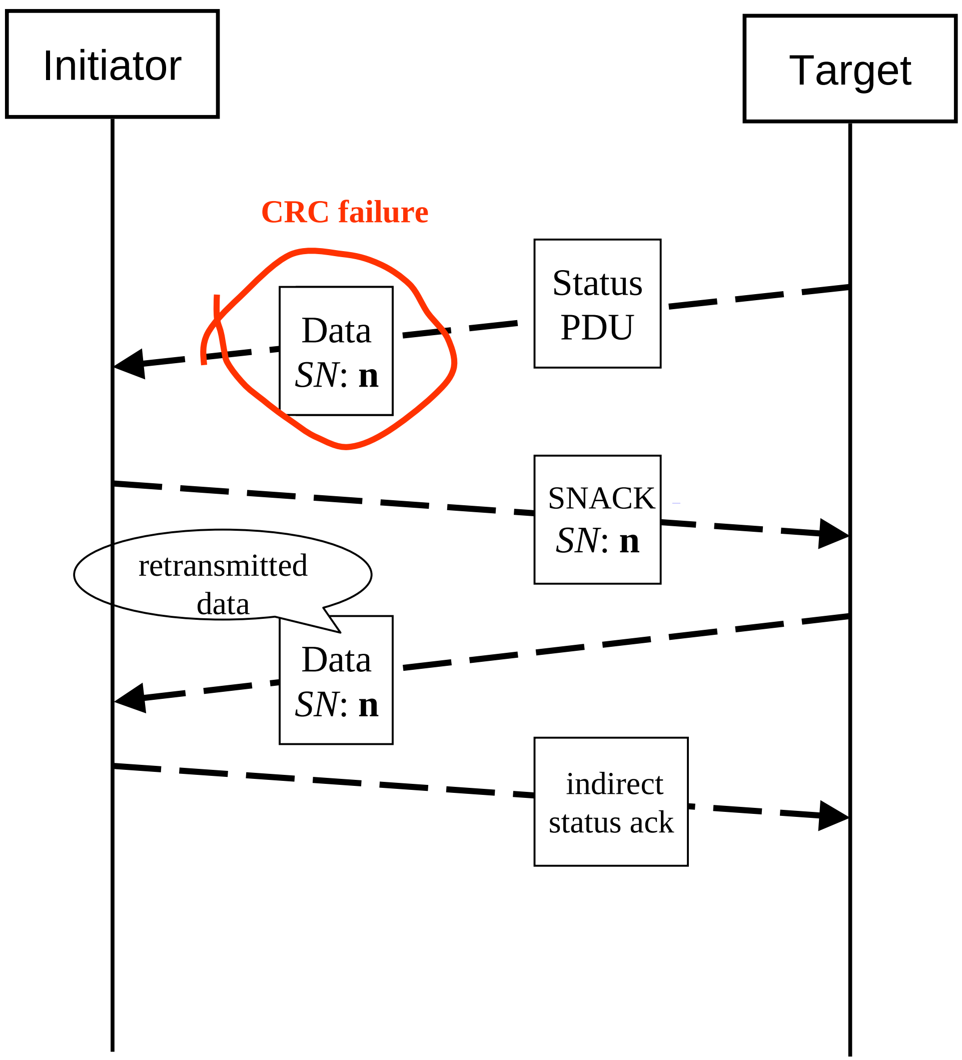
- **Continue to define SNACK.**

- **Layer the error recovery capabilities and create a new single text key to summarize all capabilities – "ErrorRecoveryLevel=n".**

- **Adopt the proposed error recovery hierarchy into iSCSI.**

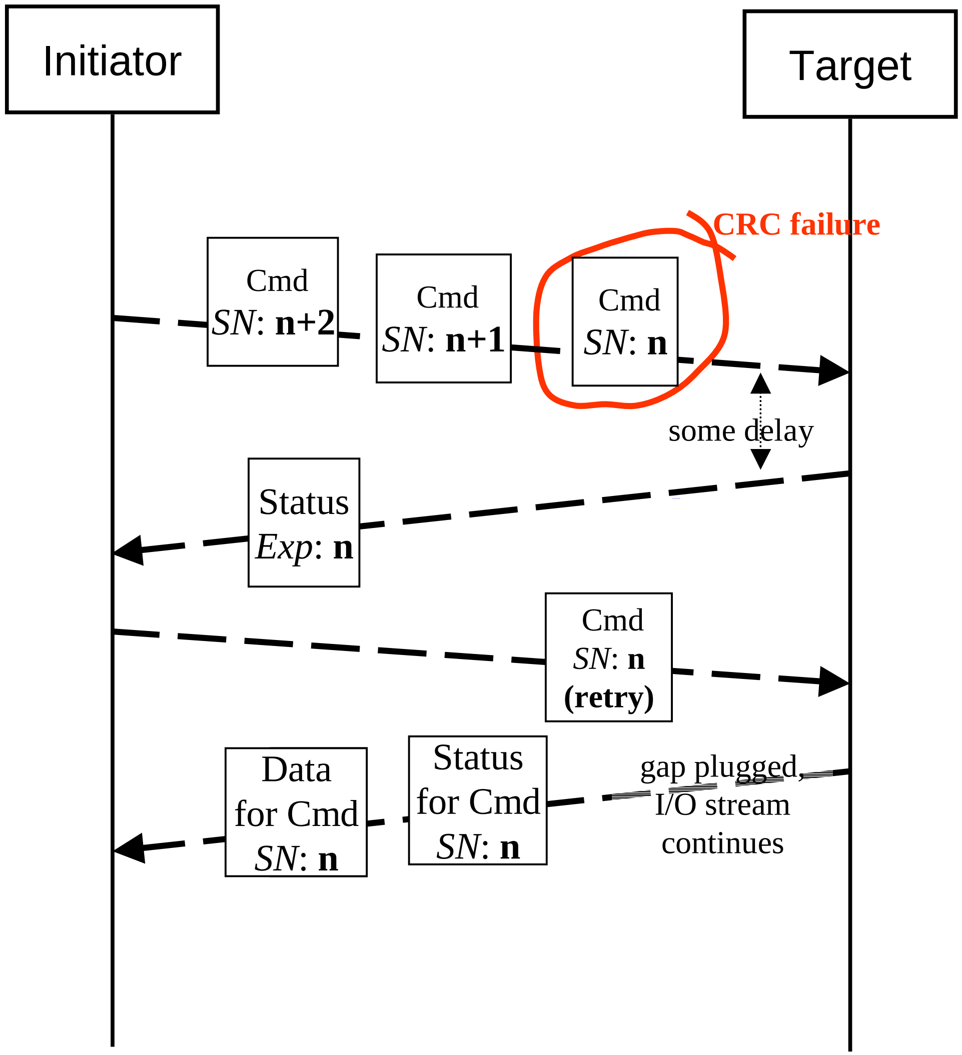
Backup

Within-command recovery example (dropped data PDU)



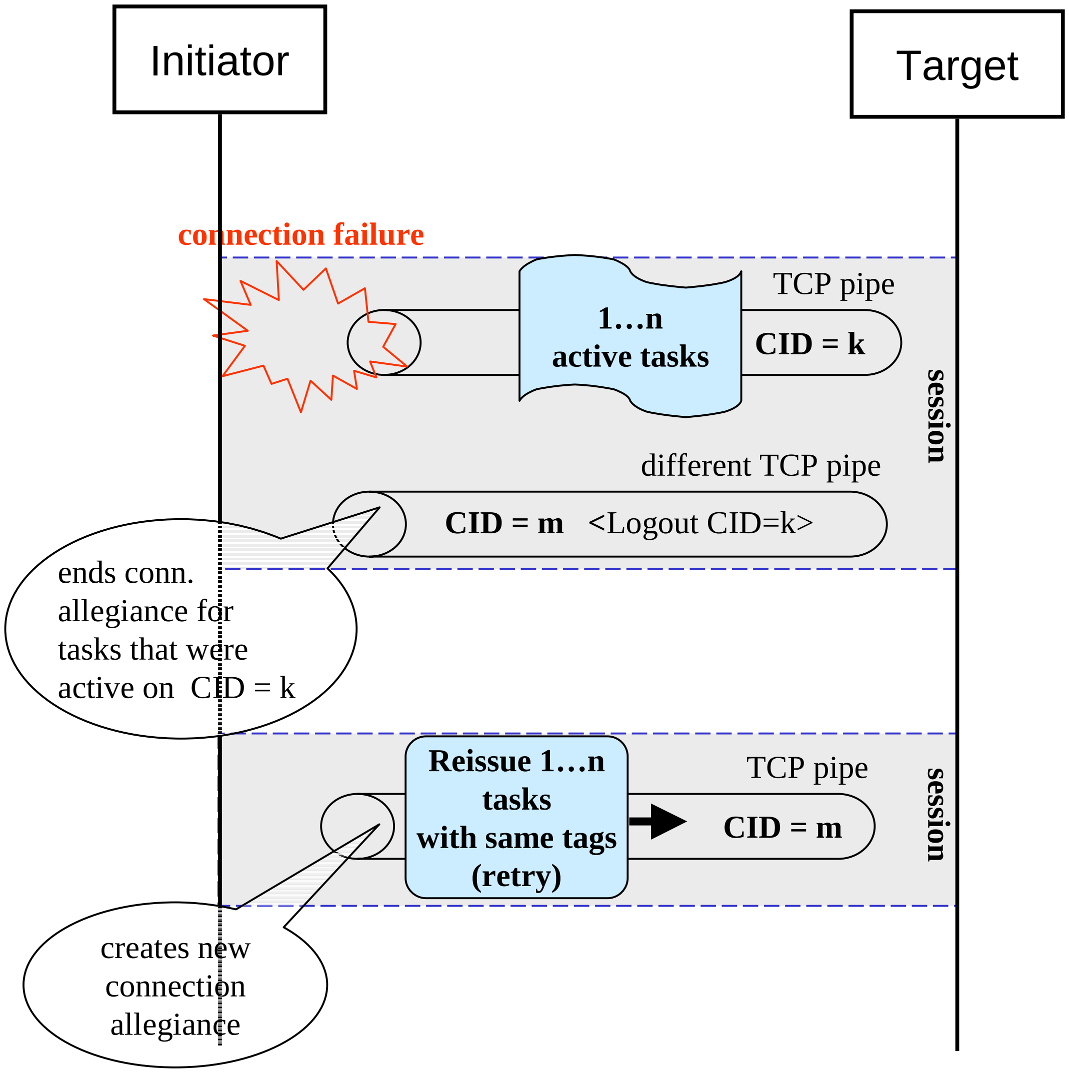
- Data PDU is dropped due to iSCSI CRC failure.
- Status PDU contains EndDataSN that indicates a gap.
- SNACK message sent to request data retransmission.
- Data PDU retransmitted.
- Status acknowledged through ExpStatSN mechanism.

Within-connection recovery example (dropped command/status)



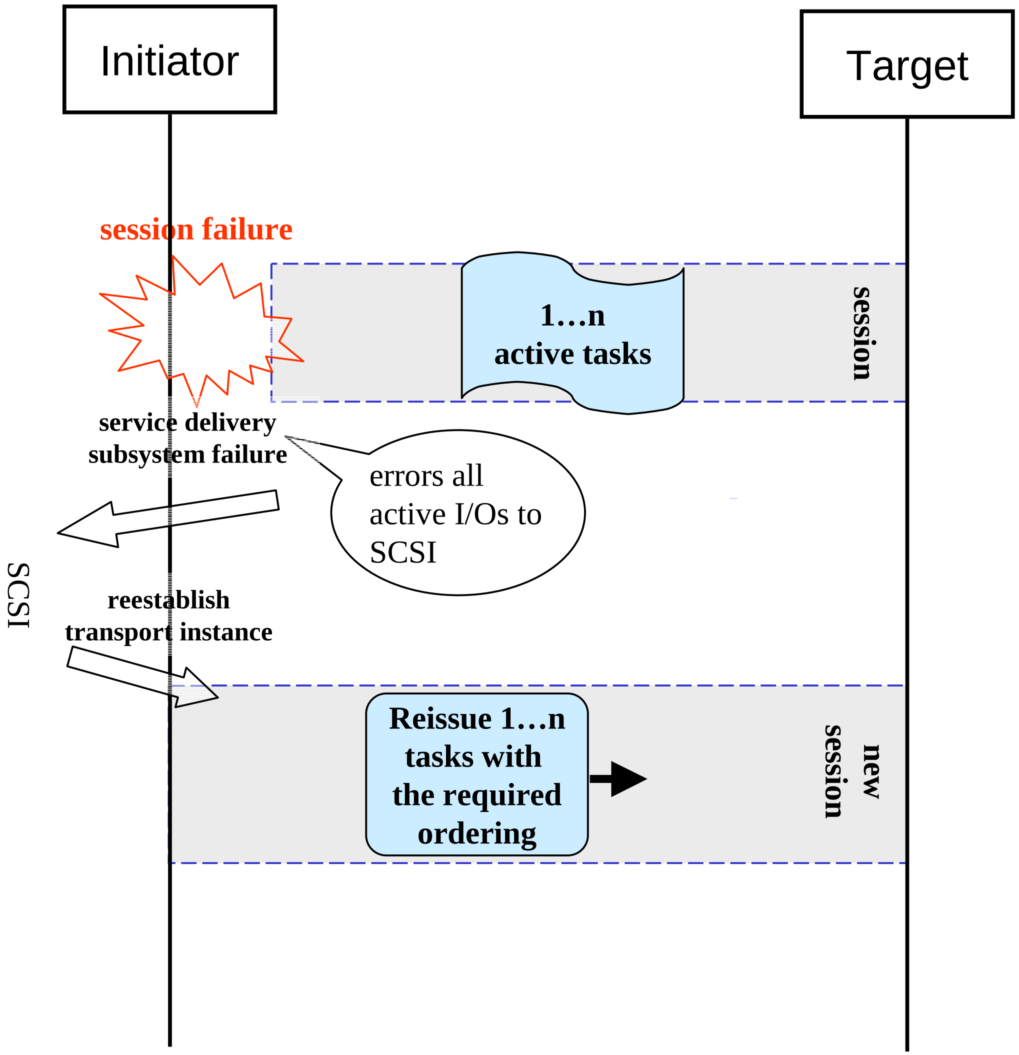
- Command PDU is dropped due to iSCSI CRC failure.
- An unrelated status PDU indicates the expected command using the ExpCmdSN.
- Command PDU is retransmitted, with "retry" bit set.

Within-session recovery example (failed TCP connection)



- Connection failure is detected at initiator.
- Initiator issues Logout for CID = k on a different connection in the same session.
- All active tasks are reissued on the other connection(s).

Session recovery example (all connections failed)



- Session failure is detected by initiator.
  - All active I/Os are errored back to SCSI layer within initiator.
- SCSI layer in initiator reestablishes iSCSI session.
- SCSI layer in initiator reissues failed tasks with the required ordering.