Data Integrity

iSCSI-R Data Integrity

version 1d

10/4/02

Jim Wendt, Pat Thaler, Julian Satran, Ilan Shimony, Vadim Makhervaks

Data Integrity

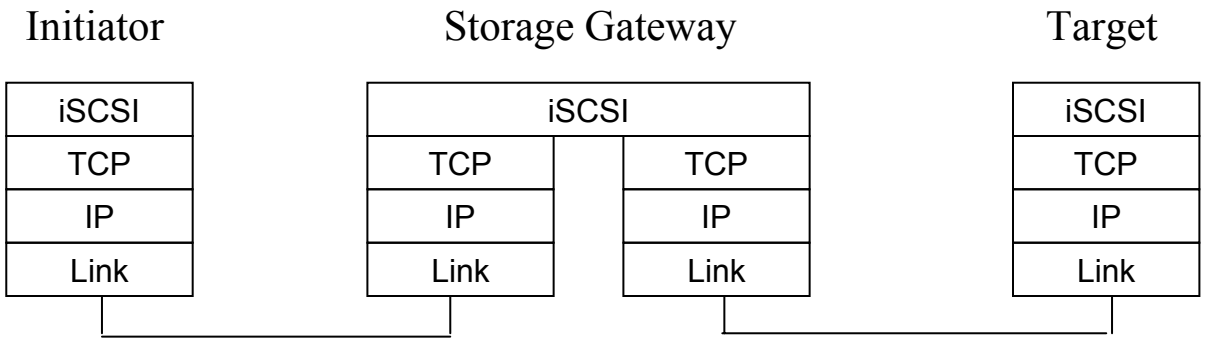
RFC3347 – Original Requirements on iSCSI Specification

- Several research papers suggest TCP checksum calc allows a certain number of bit errors to pass undetected
- To protect against data corruption - the iSCSI protocol MUST support a data integrity check format for use in digest generation.
- The iSCSI protocol MAY use separate digests for data and headers.
- In an iSCSI proxy or gateway situation, the iSCSI headers are removed and re-built, and the TCP stream is terminated on either side. This means that even the TCP checksum is removed and recomputed within the gateway.
- To ensure the protection of commands, data, and status the iSCSI protocol MUST include a CRC or other digest mechanism that is computed on the SCSI data block itself, as well as on each command and status message.
- Since gateways may strip iSCSI headers and rebuild them, a separate header CRC is required.
- Two header digests, one for invariant portions of the header (addresses) and one for the variant portion would provide protection against changes to portions of the header that should never be changed by middle boxes (eg, addresses).
- The iSCSI header format SHOULD be extensible to include other digest calculation methods.
- The iSCSI protocol SHOULD NOT preclude use of additional data integrity protection protocols (IPSec, TLS)

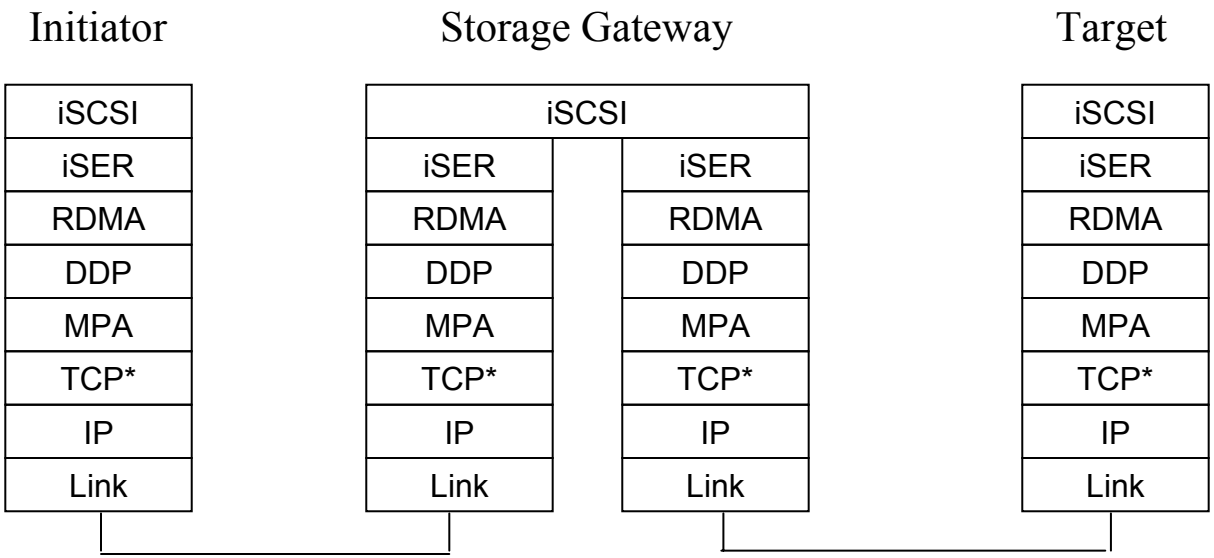
Data Integrity

Storage Gateway Protocol Layering

iSCSI v1.0



iSCSI-R



Data Integrity

iSCSI-R Data Integrity Alternatives

1. MPA level Data Integrity Check
   - The MPA CRC covers each sub-segment of data (each DDP segment)
2. iSCSI level e2e Data Integrity Check
   - CRC at iSCSI level using standard iSCSI Header Digest & Data Digest
     - iSCSI-specific NIC calculates Header & Data Digests
3. RDMAP level e2e Data Integrity Check
   a. CRC carried in RDMA Read Response / Write messages
      - CRC field added to each RDMA Read Response and Write Message Header
      - CRC rides with RDMA read/write data through GW
      - CRC calculated using OOO Incremental CRC
      - CRC is checked after all message segments have been received and placed
   b. CRC passed in Send message following RDMA Read / Write message
      - CRC passed in Send message following RDMA Read Response and Write messages
      - Must correlate CRC from Data Source (in Send message) to CRC calculated by Data Sink RNIC
      - CRC calculated using OOO Incremental CRC
      - CRC is checked after all message segments have been received and placed

Data Integrity

Opt 1 - MPA Data Integrity Check

- MPA CRC covers each sub-segment of data (e.g. each DDP segment)
- Pros:
  - No additional verbs / RI changes
  - Provides e2e data protection through robust-memory / trusted GWs (that don't manipulate data blocks).
- Cons:
  - No e2e data protection through non-robust memory / untrusted GWs

Data Integrity

Opt 2 - iSCSI E2E Data Integrity Check

- CRC at iSCSI level using Header Digest & Data Digest
  - iSCSI-specific NIC/RNIC calculates Header & Data Digests
  - CRC calculated using OOO Incremental CRC
- Pros:
  - Provides e2e data protection through non-robust memory / untrusted GWs (that don't manipulate data blocks)
- Cons:
  - Requires iSCSI-specific NIC/RNIC (e.g. iSCSI on RNIC)
  - CRC (data digest) calculation during DMA/RDMA to local memory requires rototilling of iSCSI & RDMAP

Data Integrity

Opt 3a - CRC carried in RDMAP Read/Write messages

- CRC carried as field in RDMA Read/Write messages
  - CRC field added to RDMAP Read Response and Write message headers
  - CRC travels with RDMA read/write data through GW
  - CRC calculated using OOO Incremental CRC
  - CRC is checked after all message segments have been received and placed
- Pros:
  - Provides e2e data protection through non-robust memory / untrusted GWs (that don't manipulate data blocks)
- Cons:
  - Requires RDMAP wire protocol changes (addition of CRC field & possible MsgID)
  - (GW) Requires extensive changes to RI for Read/Write Operations:
    - On Write Data Sink - passes incoming Write message data and CRCs from RI to ULP (for forwarding)
    - On Read Data Source – passes incoming Read Requests to ULP - since RDMAP Read requests must be 1:1 for Target-to-GW and GW-to-Initiator to preserve e2e data blocks boundaries and CRCs
    - On Read Data Source - must pass response data and CRC from ULP to RI - to generate outgoing Read Response (for forwarding)
    - Dual-mode RNIC must – a) verify CRCs in Writes and Read-Responses on end-nodes – b) pass up/down on intermediate boxes

Data Integrity

Opt 3b - CRC passed in Send message after RDMAP Read/Write message

- CRC passed in Send message after Read/Write message
  - CRC passed in Send message following an RDMA Write or Read Response message
  - Must match CRC from Data Source (in Send message) to CRC calculated by Data Sink RNIC
  - CRC calculated using OOO Incremental CRC
  - CRC is checked after all message segments have been received and placed
- Pros:
  - Provides e2e data protection through non-robust memory / untrusted GWs (that don't manipulate data blocks)
  - Doesn't require RDMAP wire protocol changes
- Cons:
  - (GW) Requires either:
    - ULP generated CRC Send message – with extensive RI changes for passing up incoming Write data/CRC and passing down outgoing Read Response data/CRC (for forwarding)
    - RNIC generated CRC Send message – with extensive functionality on RNIC

Data Integrity

Arguments for/against e2e CRC reqmt

Arguments for requiring e2e CRC:

- Don't have to trust the GW
- Can build cheaper Storage GWs using a separate data memory with lesser data protection

Arguments against requiring e2e CRC:

- Customers who care about data integrity will pay for GWs that can be trusted
- Storage GWs typically have only one (robust) memory subsystem
- Data integrity is already compromised for GWs that are susceptible to iSCSI header (block address) corruption
- Some GW functions modify iSCSI data blocks and crack CRC anyway
- GWs must read/write CRC to memory – Is the CRC algorithm's Pue preserved given memory error characteristics?
- Original iSCSI CRC was to compensate for TCP checksum's low Pue - but the MPA CRC addresses this weakness
- GW RI differs from end-node RI (to pass CRC up/down through GW ULP) –or- CRC forwarding functions run on the RNIC
- Requires either unique Stag per RDMAP Read/Write message (issue: multiple Writes per buffer) –or- CRC per-buffer –or- added DDP segment MsgID field

Data Integrity

RDMAP over MPA CRC

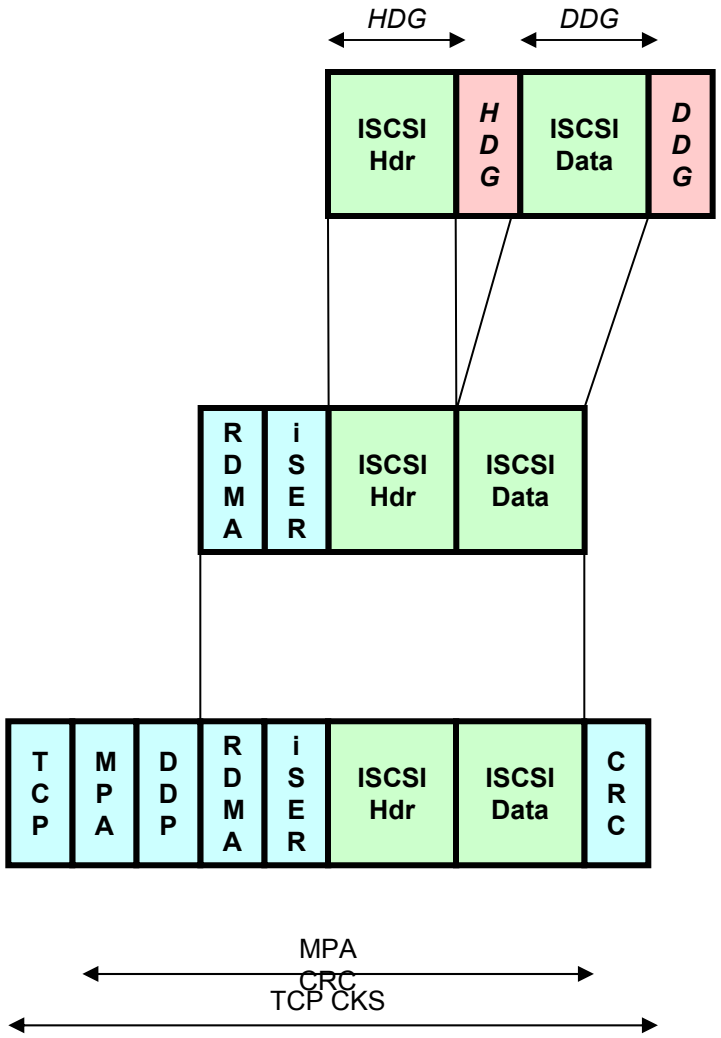
Data Integrity

MPA CRC

- CRC is E2E:
  - through L4 (TCP) middleboxes
  - through robust-memory storage GW (i.e. GW has protected buses and memory subsystem
- CRC isn’t E2E:
  - through non-robust-memory storage GW

Data Integrity

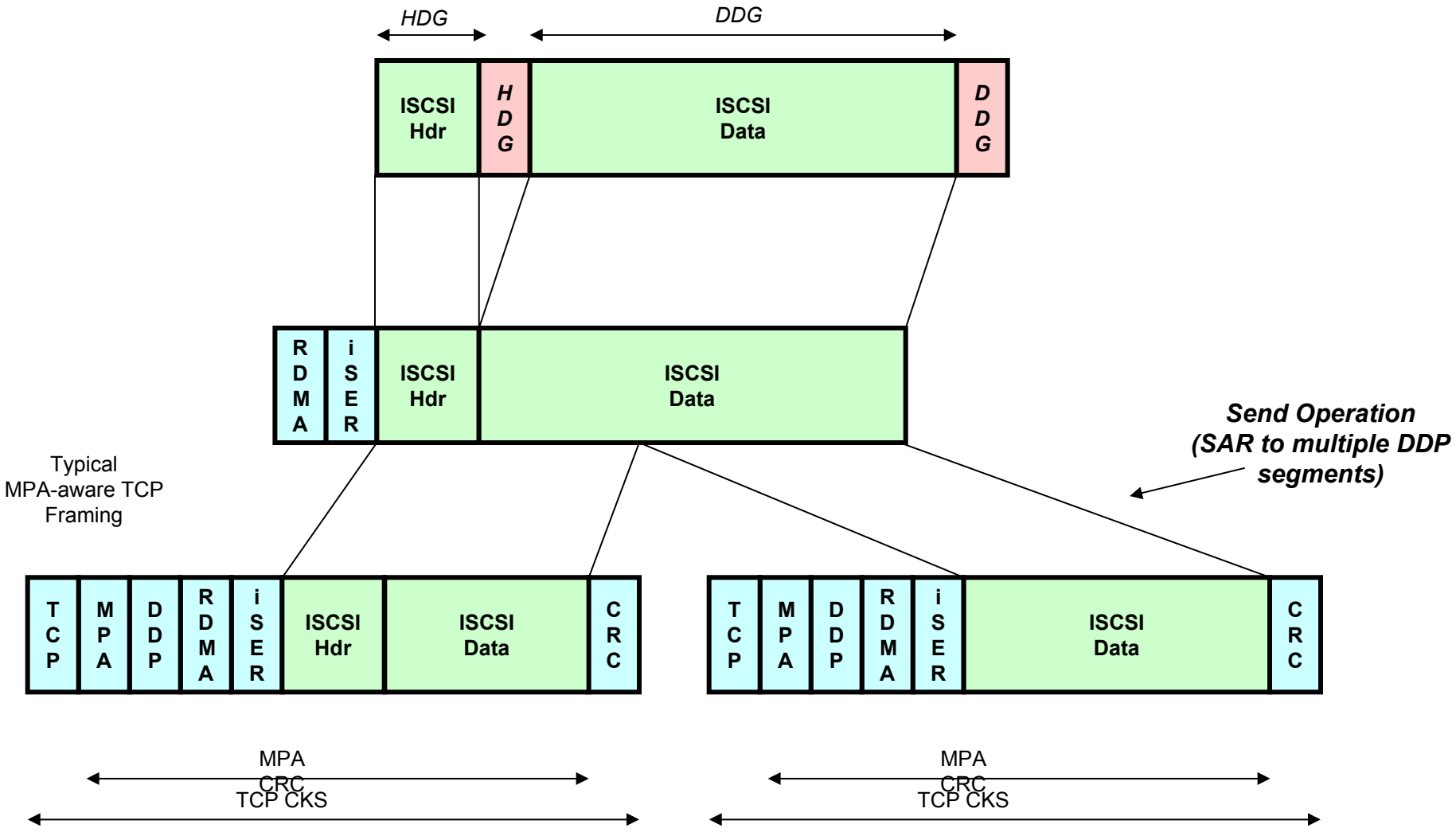
iSCSI-R – CMD PDU - Checks w/o SAR



Typical MPA-aware TCP Framing

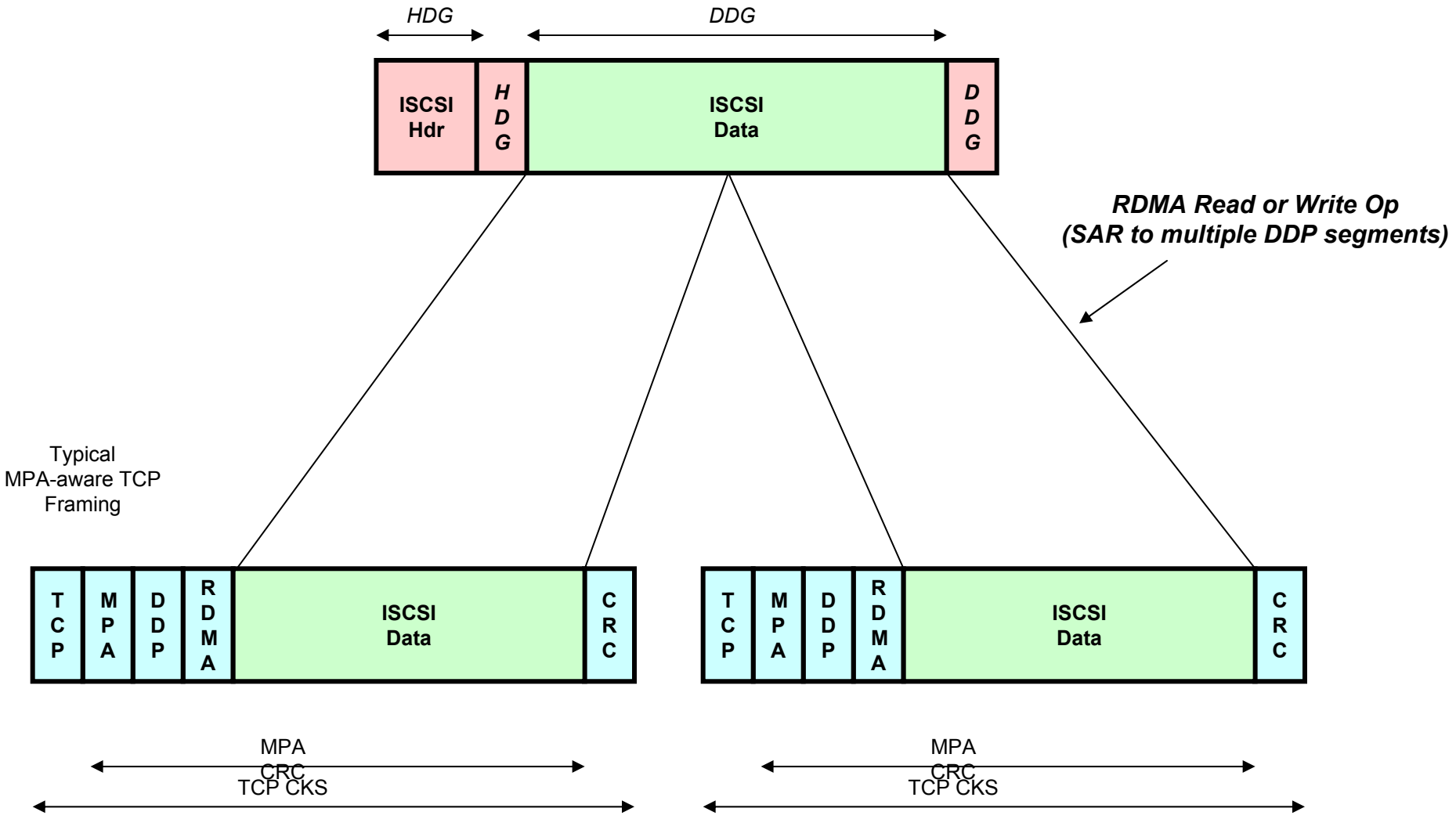
Data Integrity

iSCSI-R – CMD PDU - Checks w/ SAR



Data Integrity

iSCSI-R – Data Xfer - Chk w/RDMA SAR



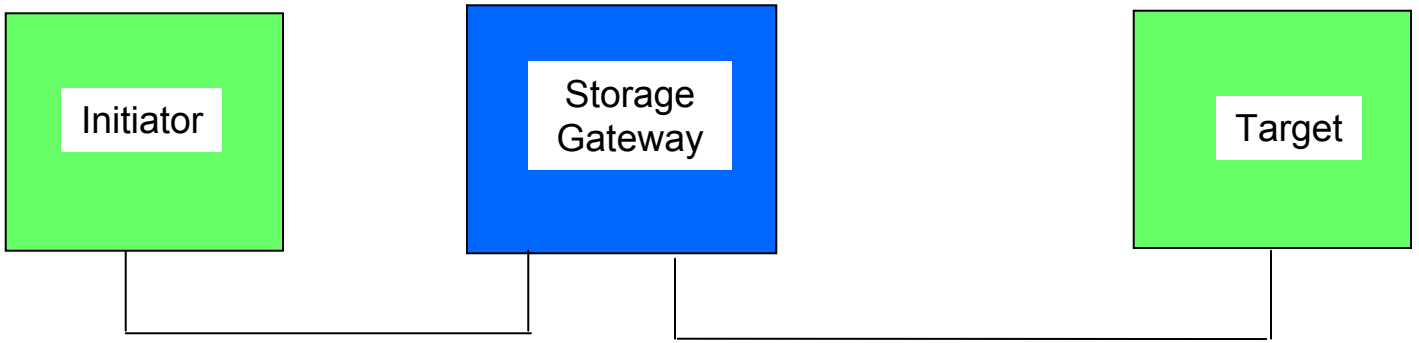
Data Integrity

End-to-End Data Integrity v0.2

(IBM)

30 Sept 2002

Configuration



- Purpose: Maintain and validate end-to-end data integrity
- Applicable to any ULP requiring E2E data integrity
  - iSCSI-R used as an example
- Assumption: All three nodes use standard RNICs
  - Requires additional functionality from the RNIC
- Basic configuration includes a single Target
  - Same approach is applicable for multiple target configurations

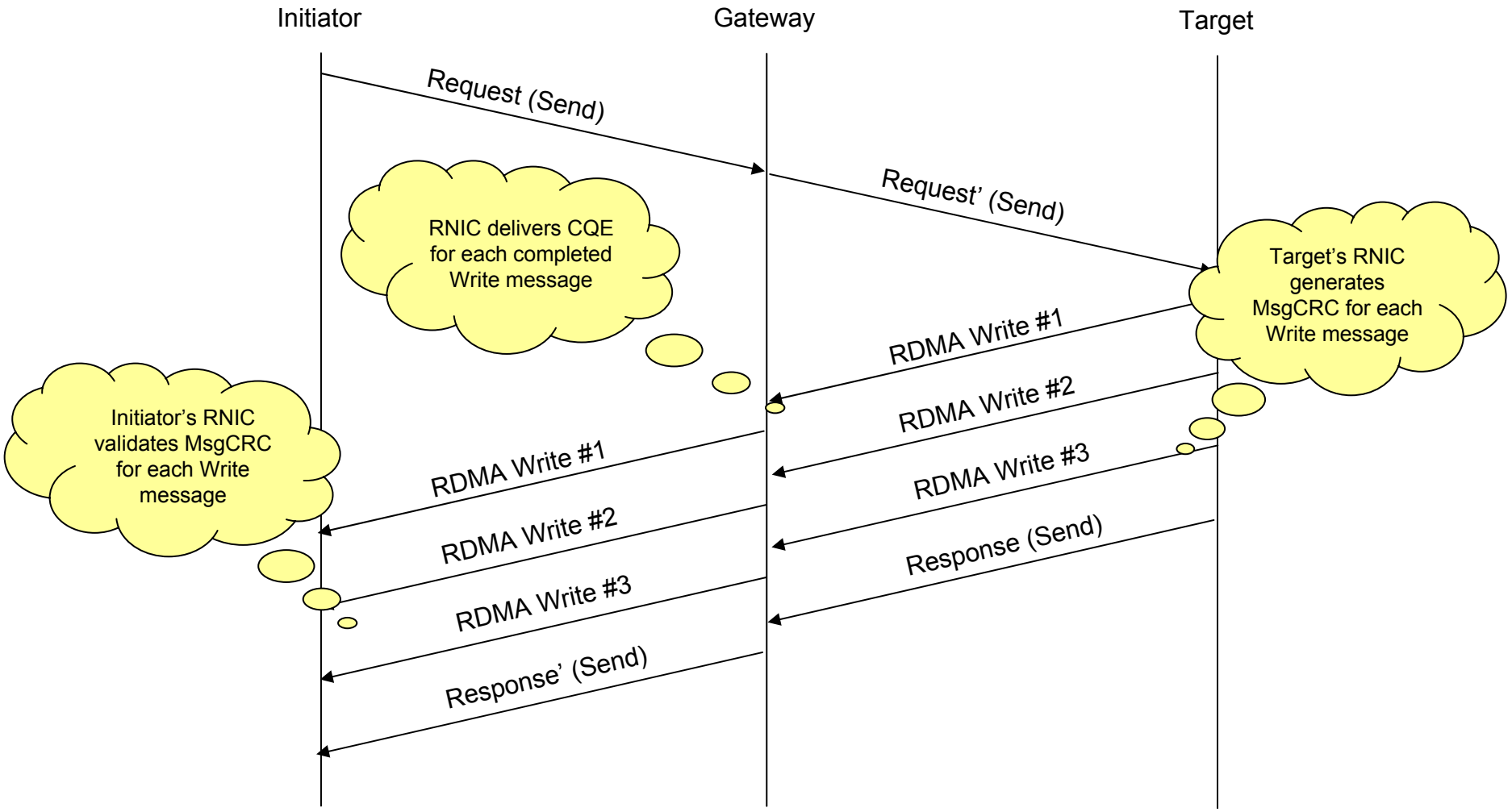
Data Integrity

Proposed Concept

- Transaction consist of
  - Request (via Send)
  - One or more data movement RDMA messages
  - Response or Status (via Send)
- Payload of data movement message is protected by MsgCRC.
  - MsgCRC is generated by Data Source
  - MsgCRC is validated by Data Sink
- Gateway does not modify MsgCRC
- Gateway uses a standard Verb-like interface
  - No dedicated hardware beside RNIC
- During Login phase all parts agree on End-to-End data integrity option
  - Behavior of RNIC depends on this option

Data Integrity

Disk Read Flow Diagram (option A)



Data Integrity

Disk Read Flow Description (option A)

- Target may generate several Write messages
  - Order is not necessarily preserved
- For each Write message RNIC generates MsgCRC
  - MsgCRC covers message payload only
  - MsgCRC travels with the last DDP segment of the message
- Gateway's RNIC
  - Places Write's payload to the memory
  - Has to keep the information required to transmit the same message as the one that was received from the target. Therefore an information per message have to be kept on the Gateway
  - Adds new **CQE** to the special completion queue
    - CQE includes MsgCRC, Stag, TO and Length
      - Note that TO and Length should cover the whole Write message
- Gateway
  - Peaks CQE and generates Write message toward Initiator
    - Note that some fields need to be modified back (Stag, LUN, etc.)

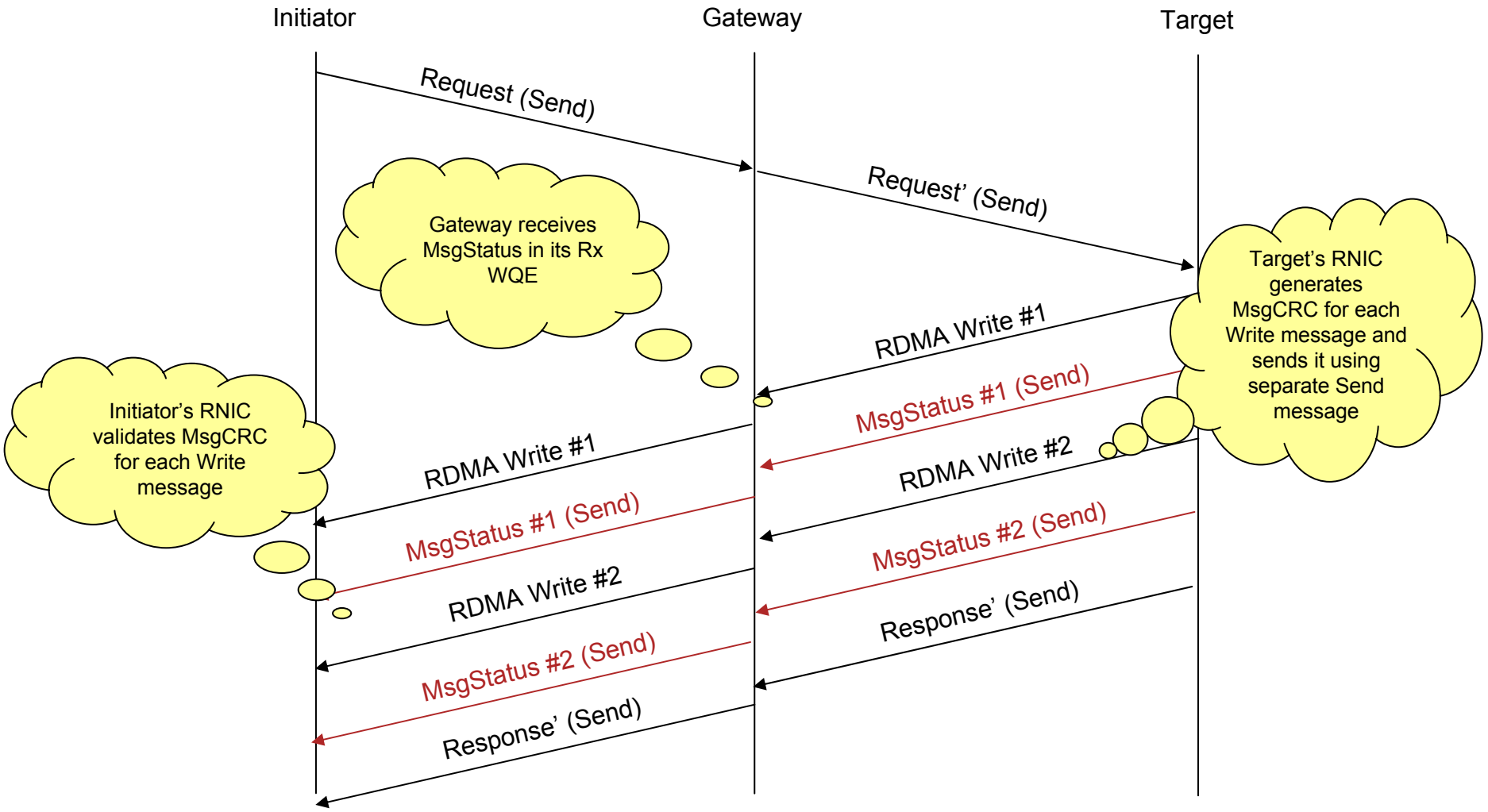
Data Integrity

Disk Read Flow Description (option A)

- Initiator's RNIC
  - Places Write's payload to the memory
  - Calculates and validates MsgCRC
    - Note that either message reassembly or incremental CRC calculation is required
    - Incremental CRC would require additional MsgLength field in the RDMA/DDP header
  - Invalid CRC should be reported using Async Error Notification mechanism

Data Integrity

Disk Read Flow Diagram (option B)



Data Integrity

Disk Read Flow Description (option B)

- Target's RNIC
  - Generates MsgCRC for each Write message
  - Generates Send which carries
    - MsgCRC, Stag, TO and Length corresponding to the Write message
    - This Send follows Write message
- Gateway's RNIC
  - Places Write's payload to the memory
  - Places Send's payload to the Rx WQ
    - Uses already existing mechanism
    - Gateway needs to know the max number of Writes to post sufficient amount of WQEs to Rx WQ.
- Gateway
  - Peaks Rx WQE and generates Write message toward Initiator
    - Note that some fields need to be modified back (Stag, LUN, etc.)
    - For each Write message, the Send carrying MsgCRC, Stag, TO and Length is sent.

Data Integrity

Disk Read Flow Description (option B)

- Initiator's RNIC
  - Places Write's payload to the memory
  - Calculates and validates MsgCRC
    - Same problem with message reassembly or incremental CRC calculation as in Option A
    - Original MsgCRC travels with Send.
  - Invalid CRC should be reported using Async Error Notification mechanism

Summary

- Both Options
  - MsgCRC calculation by Initiator's RNIC requires either message reassembly or incremental CRC
  - Either
    - E2E option should be associated with all Reads and Writes of the given connection
  - Or
    - New verb should be added to Post RDMA Read or RDMA Write with E2E CRC option
    - New bit should be added to RDMA Read Request header indicating E2E CRC option
  - RNIC should be able to calculate CRC for outgoing Write or Read Response message
  - Invalid CRC Asynchronous Error Notification
- Option A
  - New completion queue and completion mechanism for inbound Write/Read Response messages
- Option B
  - Additional Send message on the wire per Write/Read Response message

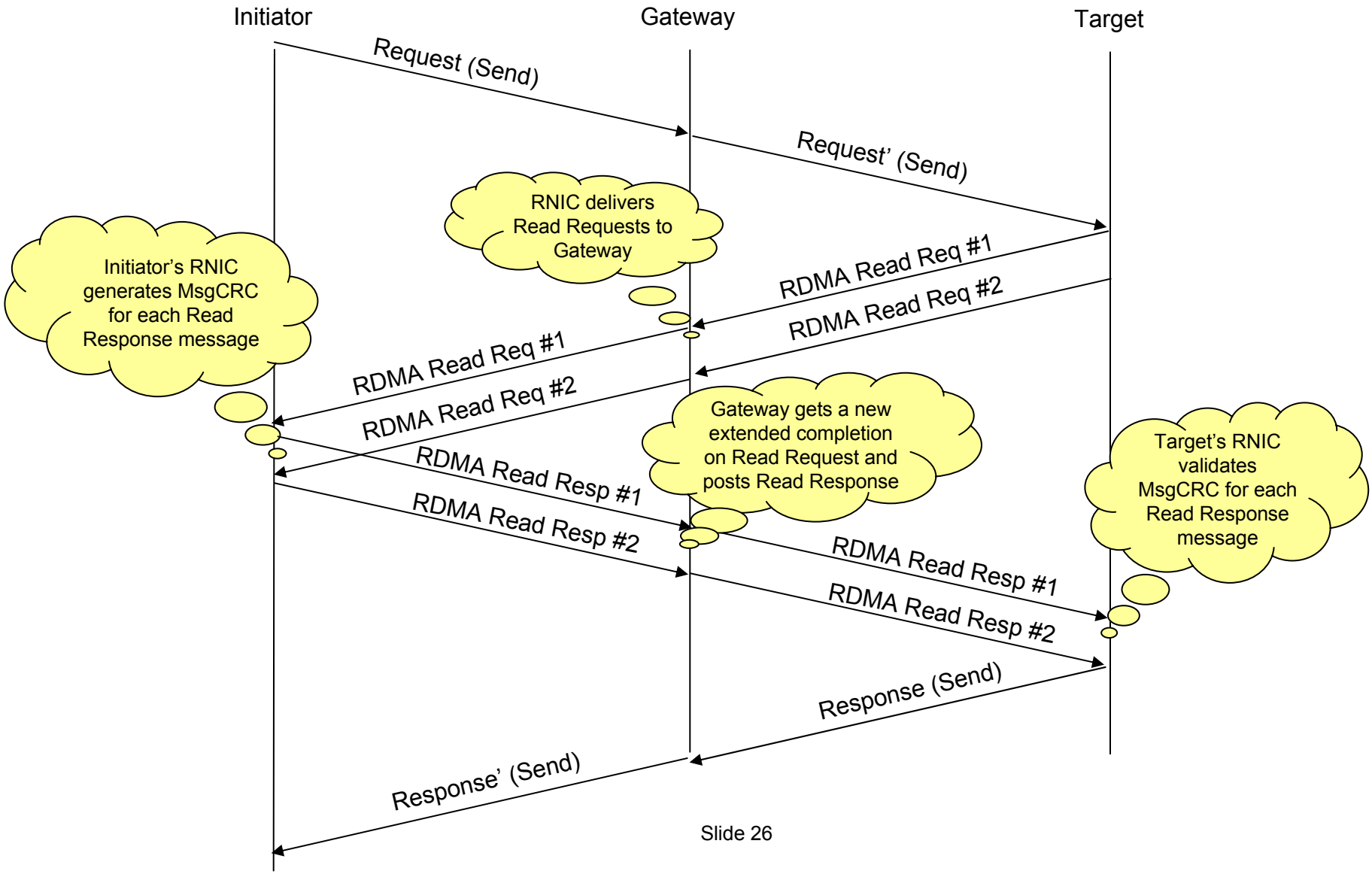
Data Integrity

Disk Read Flow

- Requires additional modification to Gateway's RNICs
  - Delivery of inbound Read Requests to the Gateway's software
    - New Receive Queue for inbound Read Requests
  - Ability to post Read Response using Verb interface
- Both Options are applicable
  - Following diagram shows flow of Option A
- Read Response is generated and treated similarly to the Write message in Disk Write Flow

Data Integrity

Disk Write Flow Diagram (option A)



Incremental CRC Information

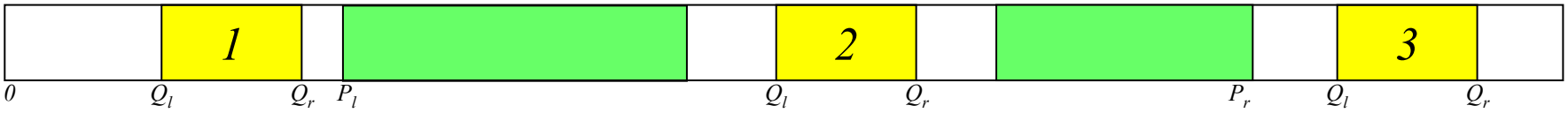
Assumptions

- A message is built on the sender side, and a 32 bit CRC is computed for it
- The message concatenated with the CRC is then chopped into segments
- Each segment is includes a header that contains a 64 bit address (or offset), and segment length.
- Segments may arrive at the receiving side in any order
- There is no overlap between segments

Notation

- P , $P(x)$ — message/polynomial received up to now - holes are treated as zeros.
- Q , $Q(x)$ — current received segment/polynomial
- $g(x)$ — generator polynomial of degree 32
- $deg\{f(x)\}$ — degree of polynomial $f(x)$
- P_{len} — length of message $P == deg\{P(x)\}$
- P_l — left offset message P
- P_r — right offset of message $P == P_{len}+P_l$
- $crc\{P(x)\}$ — CRC function: $r(x)=P(x) \bmod g(x)$
- $dcrc\{\}$ — Delta CRC function

Incremental segment arrival



segments received up to now

arriving segment

Arriving segment can be either before, contained within, or after the segments received up to now. The new incremental polynomial is:

$$P'(x) = \begin{cases} x^{P_r - Q_r} Q(x) + P(x) & Q_r < P_l \\ x^{P_r - Q_r} Q(x) + P(x) & P_l \le Q_r \le P_r \\ Q(x) + x^{Q_r - P_r} P(x) & Q_r > P_r \end{cases}$$

The incremental CRC of all arriving data is calculated as follows:

$$P'(x) \bmod g(x) = \begin{cases} \left(x^{P_r - Q_r} Q(x)\right) \bmod g(x) + P(x) \bmod g(x) = dcrc\{P_r - Q_r, Q(x), g(x)\} + crc\{P(x)\} & Q_r \le P_r \\ Q(x) \bmod g(x) + \left(x^{Q_r - P_r} P(x)\right) \bmod g(x) = crc\{Q(x)\} + dcrc\{Q_r - P_r, P(x), g(x)\} & Q_r > P_r \end{cases}$$

$$dcrc\{k, P, g\} \equiv \left(x^k P(x)\right) \bmod g(x)$$

CRC Calculation

- The delta CRC function (as defined before) can be calculated as a function of crc{P} in O(1) time/space
- For each incoming segment a new incremental CRC is calculated according to the new minimum offset

Data Integrity

Misc Slides

Data Integrity

Storage Gateway Models

- GW Functions
  - Data Manipulation GW
    - Typical Functions: ???
    - E2E data protection not maintained through GW
  - Data Pass-Thru GW
    - Typical Functions: ???
    - E2E data protection flows through without CRC recalculation
- GW Buffering Models
  - Store and Forward GW
  - Flow-Thru GW
  - Out-of-band virtualizers
    - Data flows directly between Initiator and Target
    - Every client connects to virtualizer
- Functions
  - (snapshot backups, data replication, caching, others…)

Data Integrity

Storage Gateway HW Path