Architecture for End to End Data Protection

Robert Snively
Principal Engineer, Technology
Brocade Communications Systems, Inc.
www.brocade.com

BROCADE
The intelligent platform for networking storage

August 29,2003

Goals

- Allow application client to mark data with integrity information during writes.
- Allow device server to examine integrity information received during writes for consistency.
  - Allows device server to verify receipt and consistency of data.
- Require device server to return integrity information unchanged.
- Allow application client to verify integrity of returned data.
  - Allows application client to verify receipt of data and consistency with original transmission of data.
- Allow marking of data with some special meta-data.
  - The marking is outside the scope of standardization

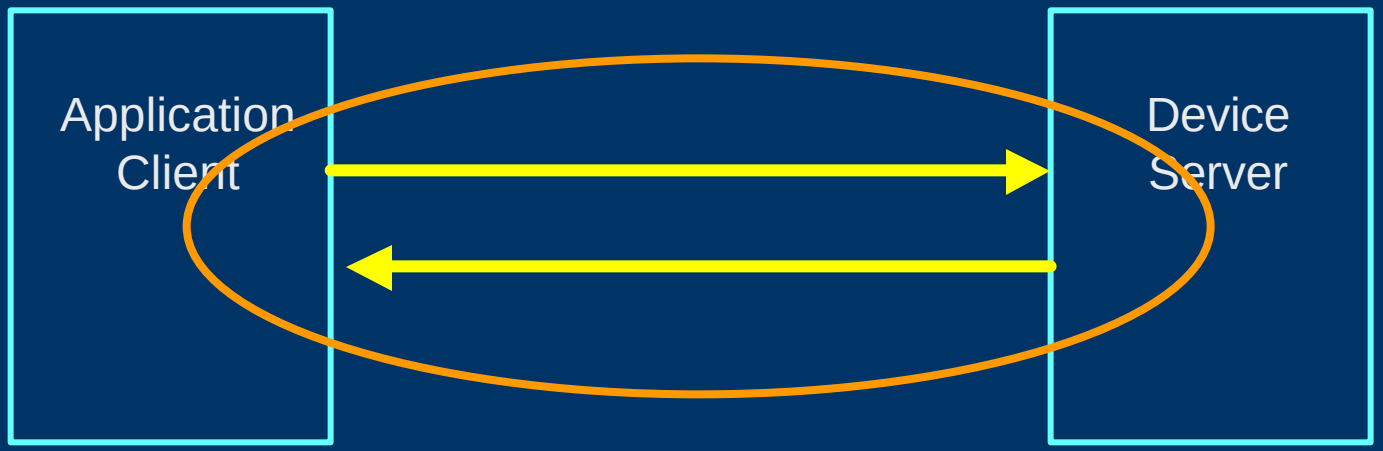
Usage Models

- Direct Attachment
- RAID Attachment
- Intelligent Fabric Attachment

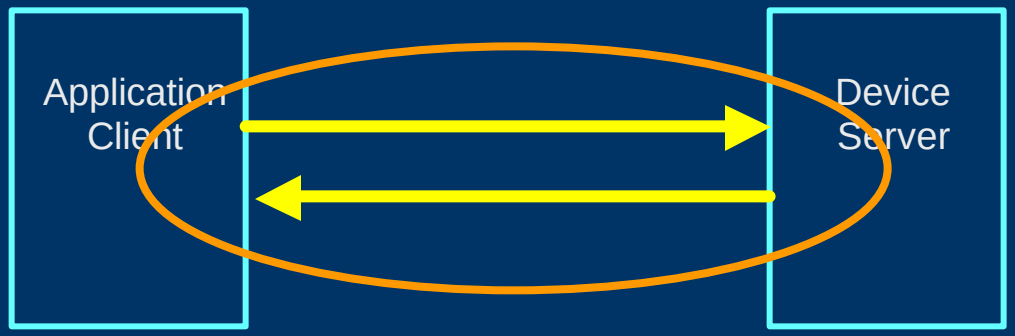
- There are probably many variations on these models, but they all shall match the simplified concept.

Simplified Concept

The only context for SCSI commands



How it works for WRITE



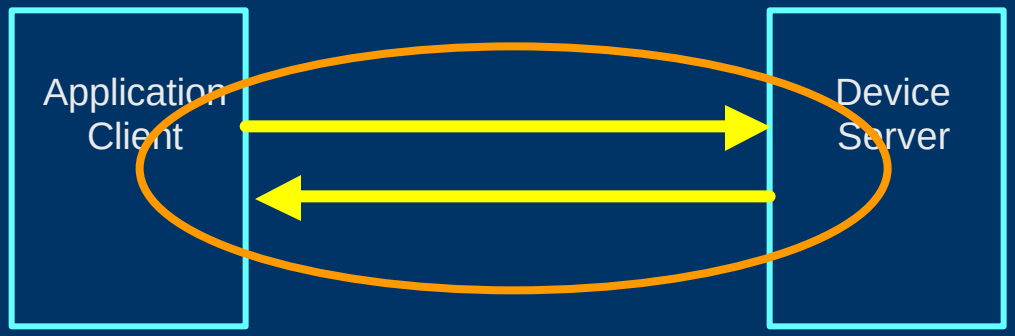
WRITE:

- Command transmitted to DS
  - Command indicates DIF context
- Data transmitted to DS
  - DIF can be verified at DS using context

Allowed Write Contexts

- Write without DIF (legacy command)
  - Drive requiring DIF inserts a special DIF indicating no DIF was provided.
- Write with DIF, check LBA against CDB
- Write with DIF, check LBA against E-DIF
  - E-DIF (Expected DIF) provided in CDB
- Write with DIF, don’t check LBA

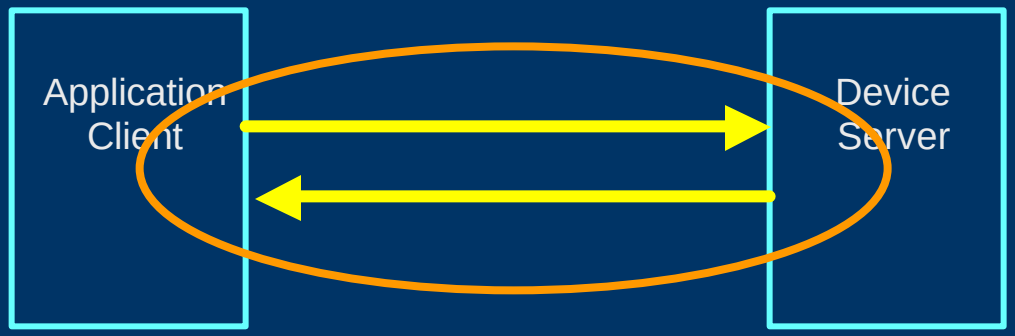
How it works for READ



Read:

- Command transmitted to DS
  - Command requests DIF or no DIF
- Data transmitted to Application Client
  - DIF is returned unchanged from value sent.
  - DIF can be verified at Application Client
  - If no DIF is requested, data is returned w/o DIF

Apply Simplified Concept to Direct Attach



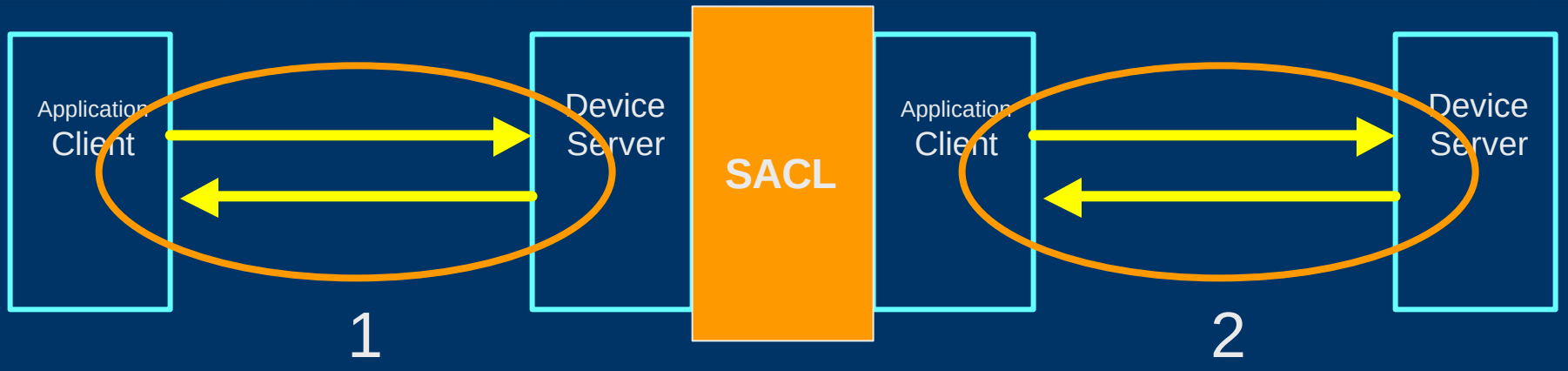
Write

1) Application client develops data protected by DIF
   Example, Checkable portion of DIF = LBA = 200 for first block)
2) Application client sends WRITE CDB indicating DIF is to be checked
3) Data is delivered to device server
4) Device server verifies that Checkable portion of DIF (C-DIF) being received matches the values of the LBA, starting with 200.

Read

1) Application client prepares CDB to perform READ
2) Application client sets up internal checker to verify expected DIF matches expected LBA values, starting with 200.
3) CDB is transmitted to Device Server, requesting transfer with DIF.
4) Device server verifies that Checkable portion of DIF being received matches the values of the LBA, starting with 200.

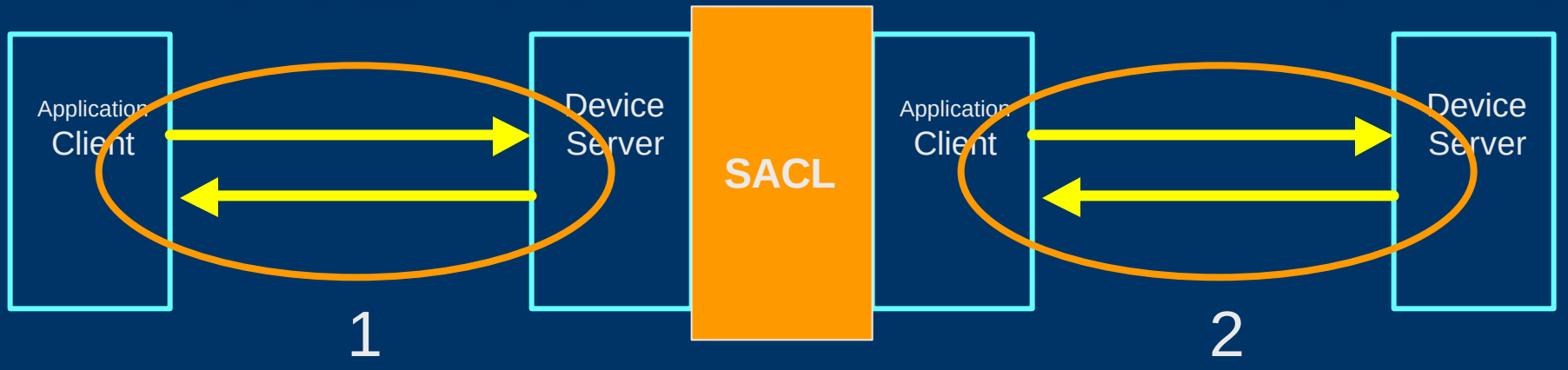
Apply Simplified Concept to RAID



WRITE:

AC1 -> CDB (LBA=200)
AC1 -> Data (C-DIF = 200)
DS1 Rcv CDB LBA=200
DS1 Chk Data C-DIF = 200
SACL maps CDB and data C-DIF to LBA =400
AC2 -> CDB (LBA mapped to 400)
AC2 -> Data (C-DIF mapped to 400
DS2 Rcv CDB LBA=400
DS2 Chk Data C-DIF = 400

Apply Simplified Concept to RAID



READ:

AC1 -> CDB (LBA=200)

DS1 Rcv CDB LBA = 200

SACL maps CDB LBA =400

AC2 -> CDB (LBA mapped to 400)

DS2 Rcv CDB LBA=400

DS2 Snds Stored Data

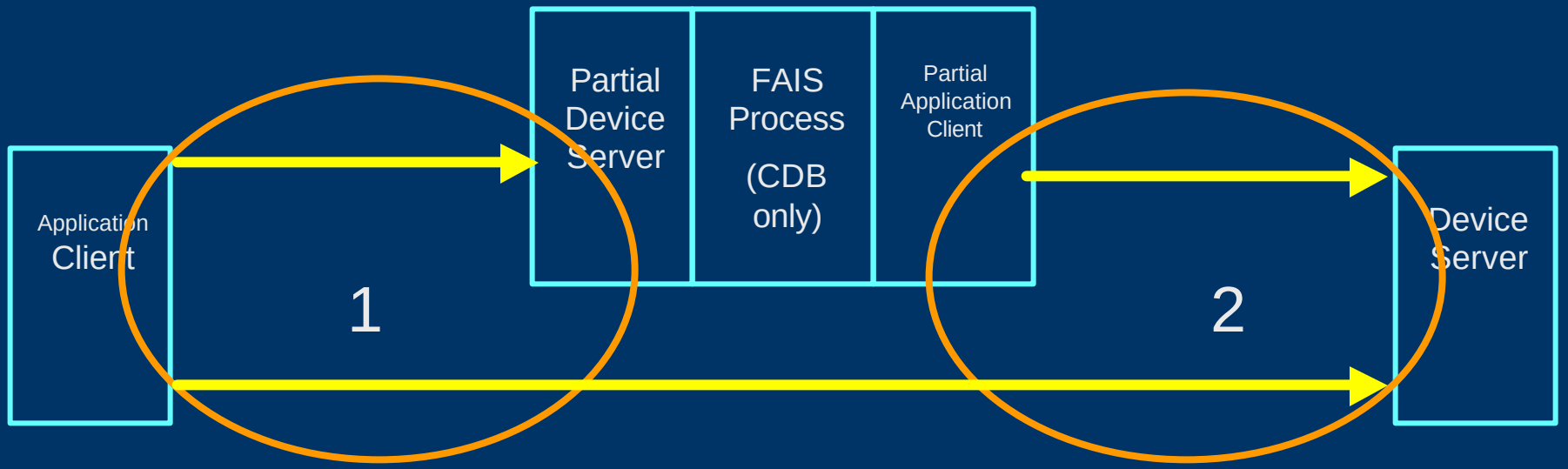
AC-2 Chks recvd C-DIF = 400

SACL maps C-DIF 400 -> 200

DS1 Sends converted data

AC1 Chks Rcvd C-DIF = 200

Apply Simplified Concept to Intelligent Fabric Attachment

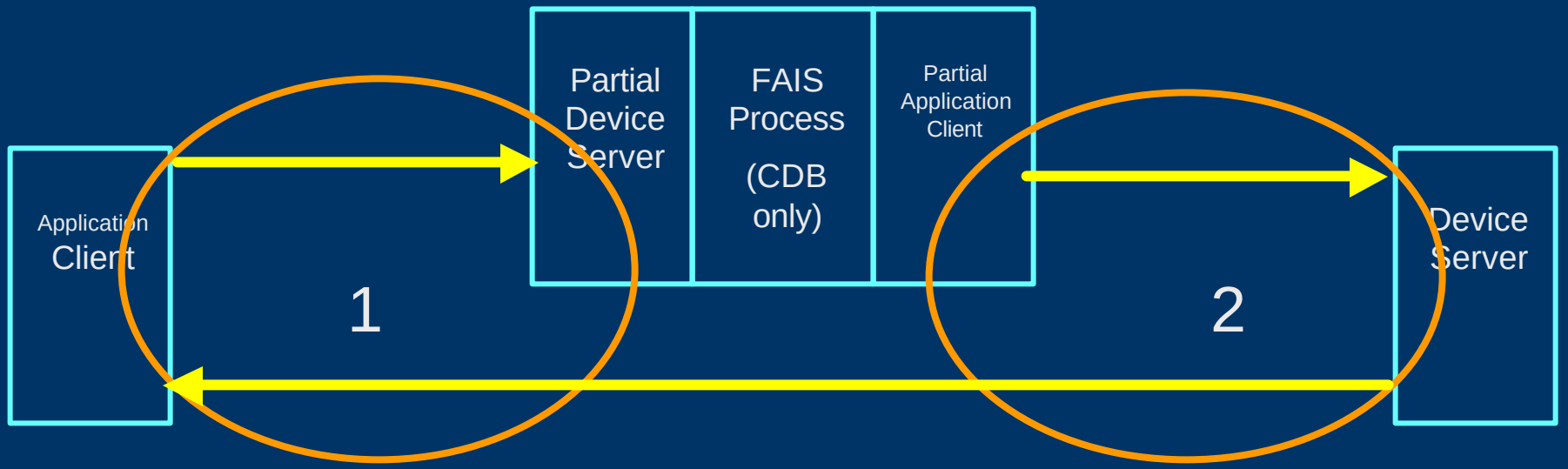


WRITE
AC1 -> CDB (LBA=200)
DS1 Rcv CDB LBA=200
FAIS makes CDB LBA=400, E-DIF=200
AC-2 -> CDB (LBA=400, E-DIF=200)
DS2 Rcvs CDB

AC1 -> Data (C-DIF=200)

DS2 Rcvs Data, checks C-DIF using E-DIF, places at 400

Apply Simplified Concept to Intelligent Fabric Attachment



READ
AC1 -> CDB (LBA=200)
DS1 Rcv CDB LBA=200
FAIS makes CDB LBA=400
AC-2 -> CDB (LBA=400)
DS2 Rcvs CDB
DS2 Snds Stored Data

AC1 <- Chks Rcvd C-DIF = 200

Conclusions (1)

- LBA and C-DIF mapping is outside the standard
  - Exceptions:
    - Device server's checking algorithm for Checkable portion of DIF against LBA must be defined.
    - Device server's checking algorithm for Checkable portion of DIF against Expected DIF must be defined.
- Meta-data mapping is outside the standard
  - Meta data portion of DIF is assumed to be opaque, unchecked by device server, verified when returned to application client.
- Encapsulation of [block+DIF] as a block inside another [block+DIF] is allowed, but not generally necessary. It is outside the standard.

Conclusions (2)

- There must be a mechanism for defining E-DIF for WRITE commands.
  - Best location is probably in the CDB
  - Probably need only one or two formats.
- No mechanism for carrying E-DIF is required for READ commands.
- Application clients expect that a READ of a block that was written with DIF information shall return the exactly the written DIF information. Any other value of DIF information is an error.

Conclusions (3)

- All device servers that support end to end data integrity SHALL support
  - C-DIF to LBA checking during WRITE
  - C-DIF to E-DIF checking during WRITE
  - Writing of C-DIFS with checking disabled
  - Legacy WRITEs of data streams containing no DIF information, automatically placing a special “no content” meta-data and a device server generated C-DIF in the stored data.
  - Legacy READs of stored data containing C-DIF information with the C-DIF information not transmitted in the data stream.