

Data Integrity from Application to Storage

William Martin **Emulex**

Fred Knight **NetApp**

Agenda

- ❑ What is data corruption
- ❑ Dealing with data corruption
- ❑ Protection Information model
- ❑ Data Integrity Architecture
- ❑ Data Integrity API

What Is Data Corruption

- ❑ *Defined as the non-malicious loss of data resulting from component failure or inadvertent administrative action*
- ❑ Frequency and impact
 - ❑ Frequency low
 - ❑ Cost very high!
- ❑ Causes of data corruption
 - ❑ Hardware
 - ❑ Software
 - ❑ Administrative error

What Is Data Corruption

- ❑ At the storage level, there are two types of data corruption
 - ❑ Latent sector errors (application cannot read once valid data)
 - ❑ This type of error is automatically corrected by RAID devices
 - ❑ Silent data corruption (data read by application is not what was last written)
- ❑ Silent data corruption returns invalid data on a read operation, rather than a “failed I/O operation”
- ❑ SNIA’s Data Integrity TWG focus is silent data corruption

Data Corruption (What)

- ❑ There are four general types of data corruption
 - ❑ *Data Misplacement Errors*
 - ❑ Data is stored or retrieve from the wrong location or device
 - ❑ *Data Content Errors*
 - ❑ Data content is changed during its life
 - ❑ *Lost I/O Operations*
 - ❑ An apparent write operation is lost, but signaled complete
 - ❑ *Administrative Errors*
 - ❑ Sysadmin makes an error leading to destroyed data

Data Corruption (When)

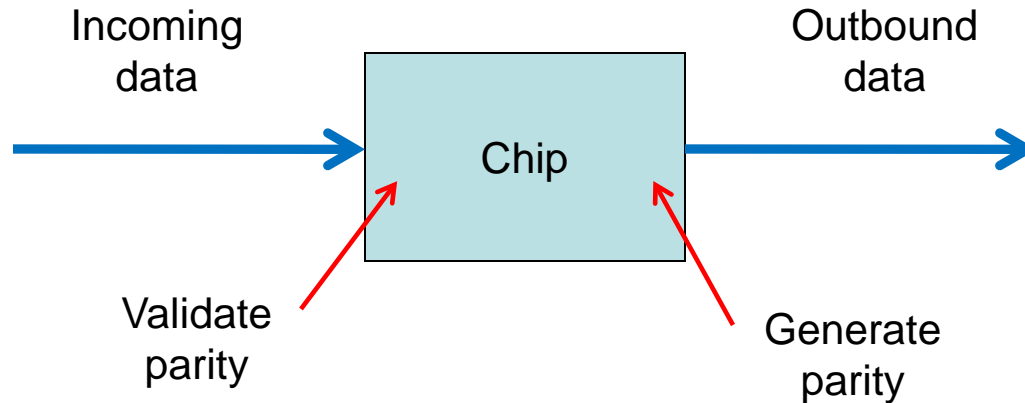
- ❑ The event of data corruption occurs at one of three stages in the life of data
 - ❑ Corruption can occur during the process of writing data
 - ❑ Corruption can occur during the process of reading data
 - ❑ Or corruption can occur while data is at rest
- ❑ It is usually not possible to know when and where corruption occurred

Data Corruption (Where)

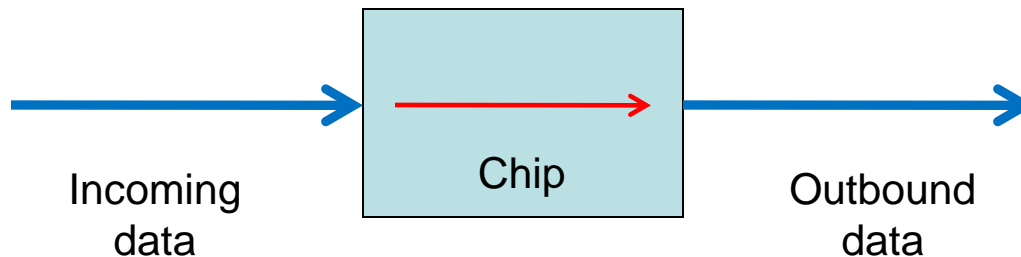
- Data corruption can occur at many places in the storage stack
 - Application layer
 - Operating System
 - Host Bus Adapter (or any storage interface)
 - Storage Fabric
 - Storage Array
 - Hard Disk Drive



Data Corruption (examples)



Parity passed through



Data Corruption (examples)

□ Examples

- O/S memory map failure leading to data going to the wrong LBA
- Lost write caused by storage array firmware
- Admin error formatting wrong volume
- O/S failure writing dump to wrong device on system crash
- O/S memory mapping failure leading to data being read from the wrong device

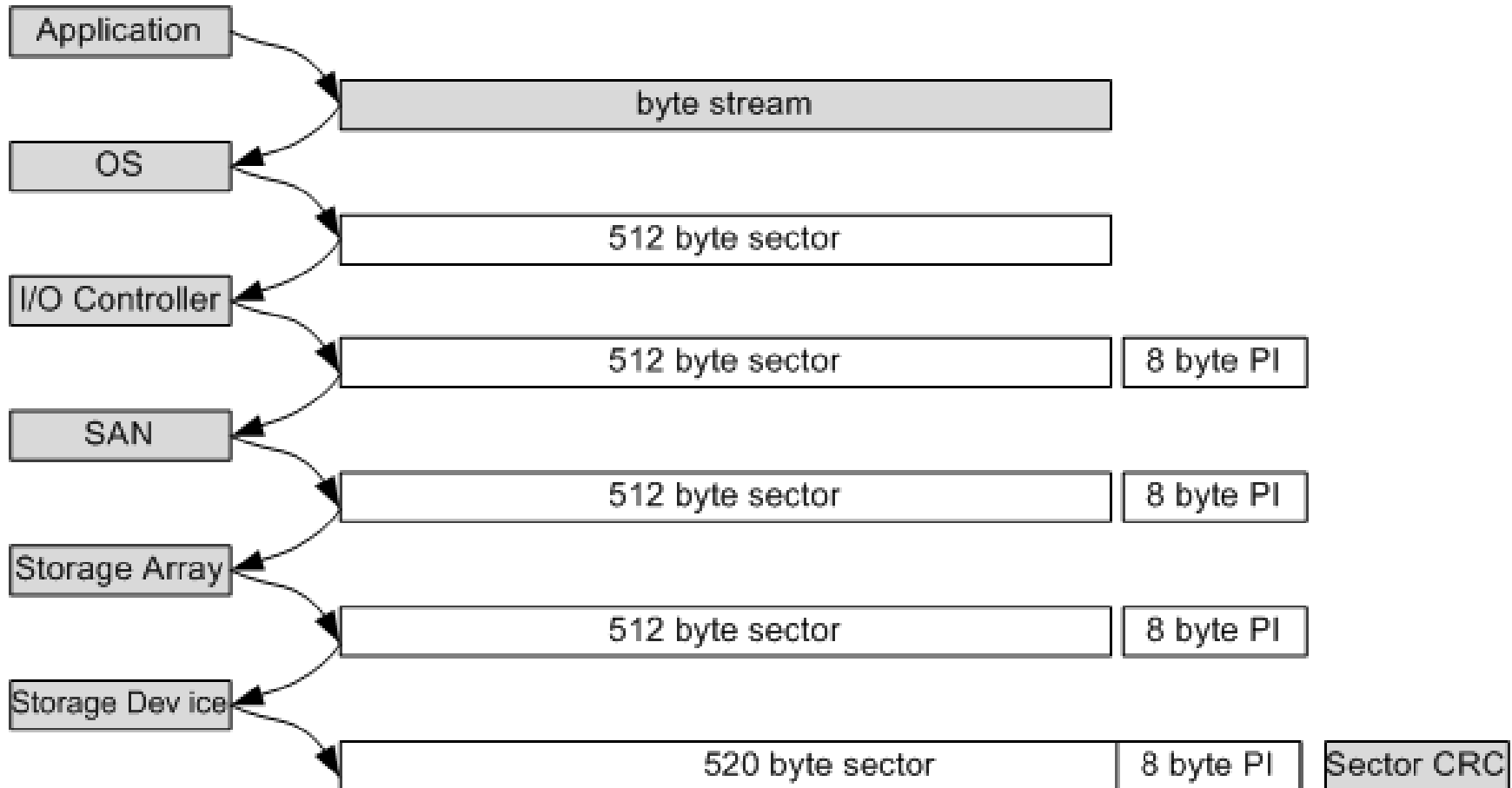
Dealing With Data Corruption

- ❑ Detection versus prevention (early detection)
 - ❑ An example of a detection mechanism is the REF tag in type 2 T10 DIF. By itself, the REF tag only enables the HBA to detect an error, during a read operation, when the data block has been corrupted somewhere in the storage stack from the HBA to the storage medium.
 - ❑ An example of prevention is if the 16 bit CRC in T10 DIF is checked by the storage array or storage medium and corrupt data is prevented from being written to permanent storage.
- ❑ Both prevention and detection are useful together.
 - ❑ This is the concept behind the SNIA Data Integrity architecture.

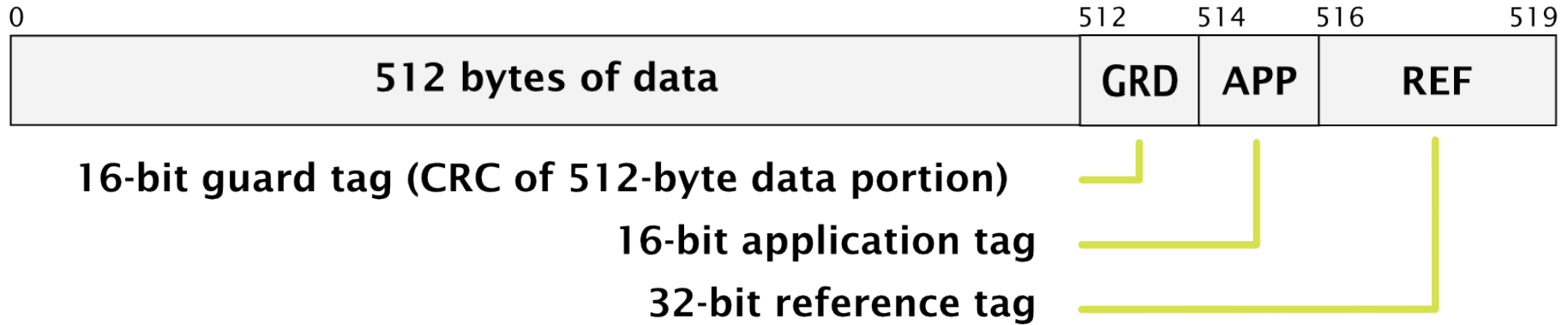
Data Integrity Architecture and DIF

- ❑ Drives use 512-byte sectors or 4096-byte sectors
- ❑ Each sector is protected by a proprietary cyclic redundancy check internal to the drive
- ❑ Enterprise drives support 520-byte sectors or 4160-byte sectors
- ❑ Sector sizes that are not a multiple of 512 bytes have seen limited use because operating systems deal with units of 512 bytes
- ❑ RAID arrays make extensive use of 520 byte or 4160 byte sectors

Normal I/O (512 byte sector)



T10 Data Integrity Field



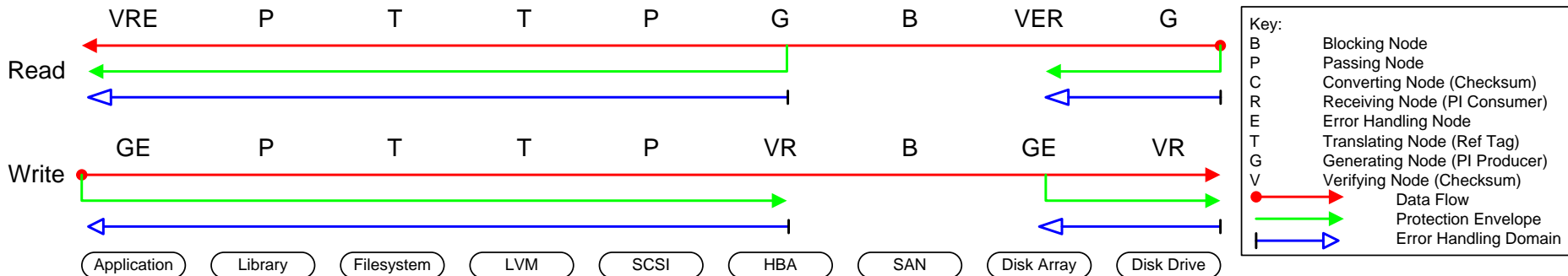
- ❑ Only protects between HBA and storage medium
- ❑ PI contiguous with data sectors on the wire
- ❑ Three protection schemes
 - ❑ All have a 16-bit CRC guard tag
 - ❑ Type 1 reference tag is lower 32 bits of target sector
 - ❑ Type 2 reference tag seeded in CDB

Data Integrity Architecture

- ❑ Extends data protection all the way up to the application, enabling true end-to-end data integrity protection
- ❑ Does not change the current SCSI architecture or wire protocols
- ❑ Architecture that allows protection information to be passed between layers of the storage stack
- ❑ Data Integrity Architecture:
 - ❑ Enable transfer of protection information to and from host memory
 - ❑ Separate data and protection information buffers
 - ❑ Allows different protection information at different levels
 - ❑ Provide a set of commands that tell HBA how to handle I/O:
 - ❑ *To be defined in a Data Integrity API*

Data Integrity Model definitions

- ❑ Data Integrity Metadata
- ❑ Protection envelope
- ❑ Properties of nodes
 - ❑ Blocking > Passing
 - ❑ Converting > Receiving
 - ❑ Error handling > Translating
 - ❑ Generating > Verifying
- ❑ Handoff
 - ❑ Protected > Unprotected

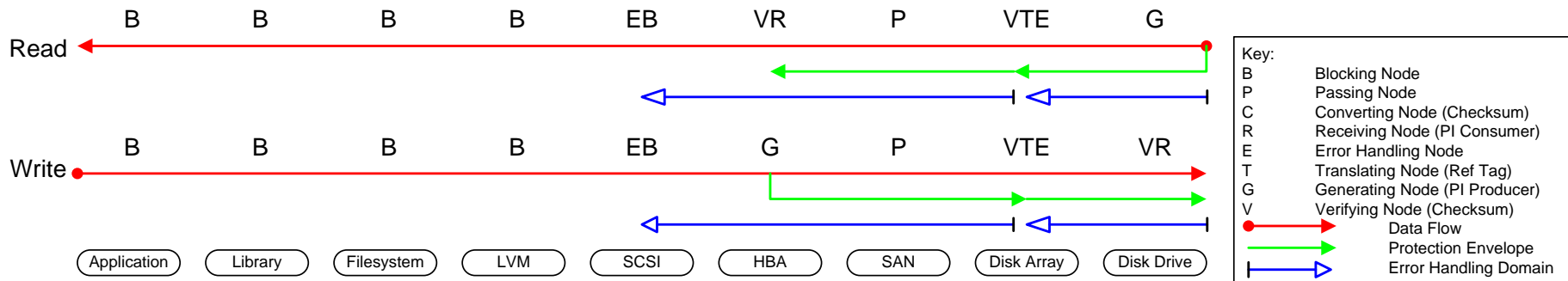


Data Integrity Metadata

- **Data Integrity Metadata:** consists of Data Stamp, Application Stamp, and Address Stamp.
 - **Data Stamp:** the information (e.g., checksum, CRC, or hash code) that validates the data within the protection envelope. As applied to the T10 Data Protection model this is the Logical Block Guard.
 - **Address Stamp:** the information that validates the locale of the data within a specific domain. As applied to the T10 Data Protection model this is the Logical Block Reference Tag.
 - **Application Stamp:** information that is assigned by the application to validate the originator of the data. As applied to the T10 Data Protection model this is the Logical Block Application Tag.

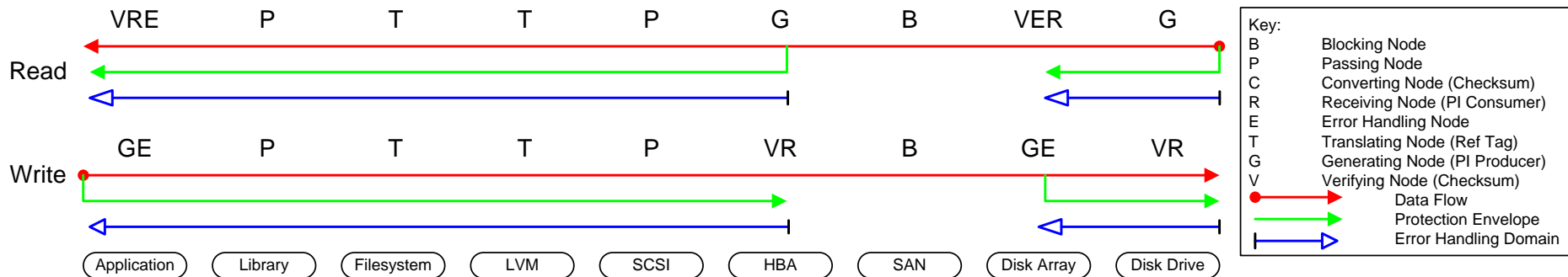
Protection Envelopes

- **Protection Envelope:** all of the components that allow data and associated Data Integrity Metadata to flow through them without any modification of the Data Stamp & Application Stamp.
- **Combined protection envelope:** multiple contiguous protection envelopes that allow data and associated Data Integrity Metadata to flow through them while allowing modifications of the Data Integrity Metadata.



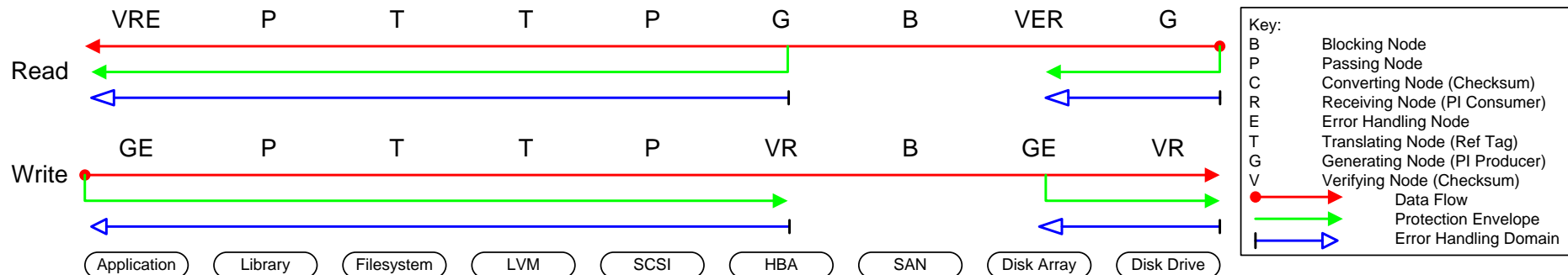
Properties of nodes

- ❑ **Blocking:** a property of a node in which the node, by design, does not pass Data Integrity Meta Data.
- ❑ **Passing:** a property of a node in which the node passes data and Data Integrity Metadata un-modified.
- ❑ **Converting:** a property of a node in which the node modifies the Data Stamp received to a different Data Stamp that is transmitted.
- ❑ **Receiving:** a property of a node in which the node receives data and Data Integrity Metadata and is the termination of the Combined protection envelope.



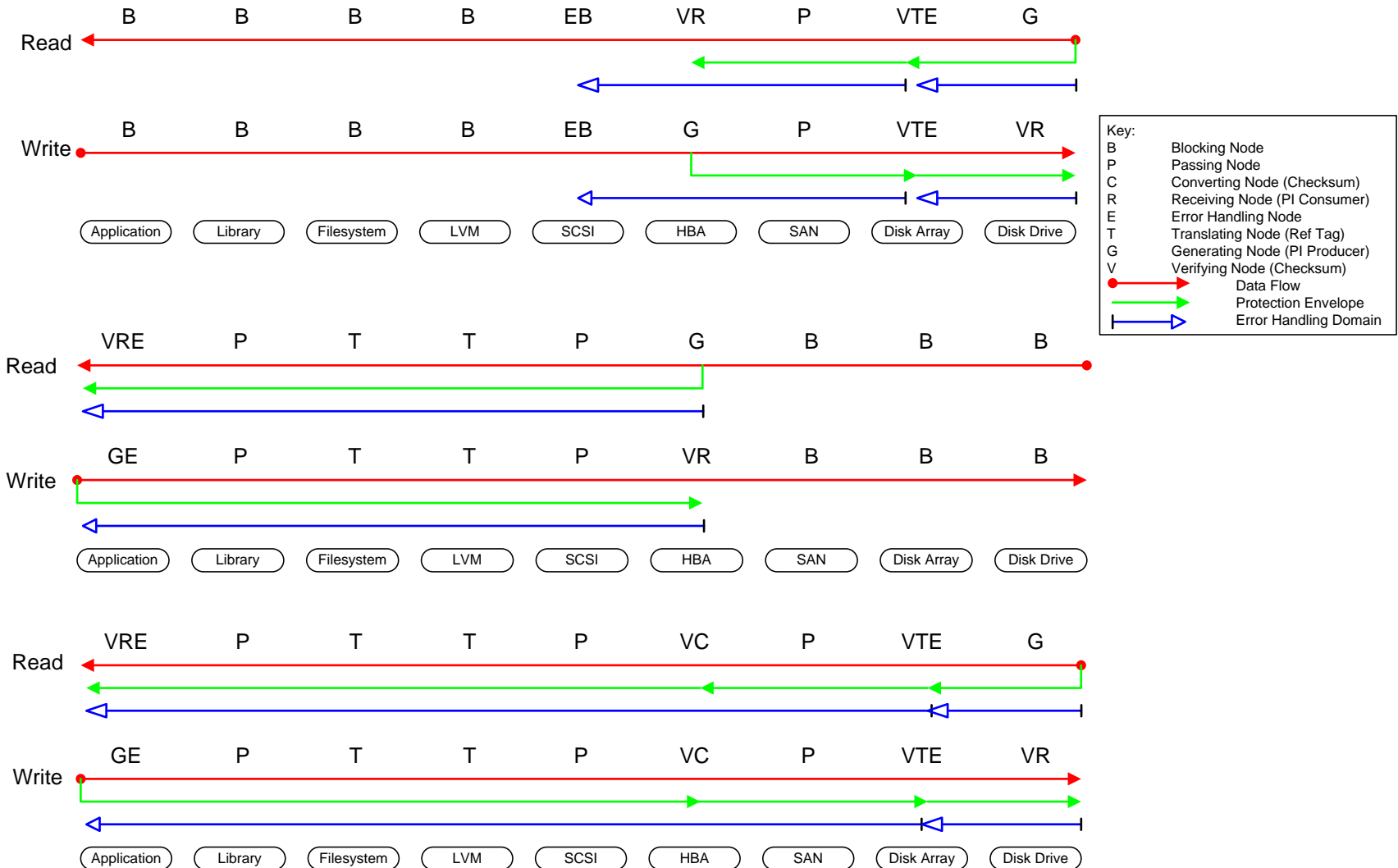
Properties of nodes (continued)

- ❑ **Error Handling:** a property of a node in which the node receives error notification from a Verifying Node and processes data integrity errors.
- ❑ **Translating:** a property of a Node in which the node modifies the Address Stamp in the Data Integrity Metadata received to a different Address Stamp in the Data Integrity Metadata that is transmitted.
- ❑ **Generating:** a property of a node in which the node generates Data Integrity Metadata.
- ❑ **verifying node:** a node that verifies the Data Integrity Metadata that is associated with the data and passes errors to the Error Handling Node of the Error Handling Domain.



- ❑ **handoff:** the process that occurs with the Data, Address, and Data Integrity Meta Data of an I/O transaction in a node that is both a receiving node and a generating node (i.e., converting node or translating node).
- ❑ **protected handoff:** a handoff where Data Integrity Meta Data that is to be transmitted is generated before the validation of the received Data Integrity Meta Data in such a way that any intervening data corruption is detected.
- ❑ **unprotected handoff:** a handoff where it is possible that neither of the protection envelopes detects an error introduced during the handoff (e.g., the data is outside the protection envelope during the handoff).

Examples



Different protection information

- ❑ Separate protection scatter-gather list
 - ❑ 520-byte sectors are hard to deal with in a general purpose OS
 - ❑ <512, 8, 512, 8, 512, 8, ...> does not perform well
- ❑ Checksum conversion
 - ❑ CRC16 is slow to calculate
 - ❑ IP checksum is fast and cheap
 - ❑ Strength is in data and protection information buffer separation

Data Integrity API

- ❑ The next step for the DI TWG is to define this application interface
- ❑ Define capability negotiation between elements of storage stack
 - ❑ Capability information needs to be negotiated both directions in the storage stack
 - ❑ Protection algorithm
- ❑ Define error handling in the storage stack
 - ❑ Error handling layer and communication mechanism
 - ❑ Error handling information passed during capability negotiation

Data Integrity API (continued)

- ❑ Data and protection information needs to be passed between layers
- ❑ Define interfaces necessary to embed data integrity into File System
- ❑ Mechanism to communicate location of protection information between layers

Data Integrity API goal

Make data integrity as transparent as possible to applications