Techniques for Device-Managed Error handling

Eliel Louzoun
Intel Israel

Disclaimer

Presentation Disclaimer: All opinions, judgments, recommendations, etc. that are presented herein are the opinions of the presenter of the material and do not necessarily reflect the opinions of the PCI-SIG®.

Agenda

- Overview
- Motivations
- OS Error Handling Mechanisms
- Error Containment
- New Problems
  - ✓ Malicious VF or User space driver handling
  - ✓ TLP source involvement
  - ✓ Peer to Peer
- Summary

Overview

- The legacy PCI™, PCI Express® and AER capabilities define ways to report PCIe® error events
- These errors are routed as messages to the Root Complex and handled by the PCIe bus driver of the OS
- OS calls the device driver of the offending device
- The driver may then apply device specific solutions that may limit the impact of the error event and may solve the root cause of the error
- This presentation reviews existing mechanisms for error processing, possible usages of such mechanisms and possible extensions

Motivations

Better Handling of Errors

- Some of the PCIe errors are Uncorrectable Nonfatal:
  - ✓ ECRC
  - ✓ Poisoned
  - ✓ Completion Timeout
  - ✓ Unsupported Request
  - ✓ Completer Abort
  - ✓ Unexpected Completion
  - ✓ Reception of a completion with CA/UR (not reported by receiver)
- Most of these errors are recoverable by device level actions that would limit the impact of the error:
  - ✓ Limit to the specific PCI function
  - ✓ Limit to specific sub function (queue/thread)
- Can be done by combination of OS logging and reporting of the error together with the device driver capabilities to identify and handle the root cause
  - ✓ E.g. identify the queue to which the transaction was directed and reset it

Untrusted Agents

- We identified two types of untrusted agents:
  - Virtual Function drivers
  - User space drivers
- These agents have direct access to HW resources. Wrong programming or malicious behavior can harm other agents or the entire system.
- VF drivers
  - In physical systems, one machine can't stop the devices of another machine. The same behavior is expected between virtual machines.
- User Space drivers
  - Must not affect Kernel driver services and other user space drivers
- The PF/Kernel driver should be able to identify and isolate the offending agent

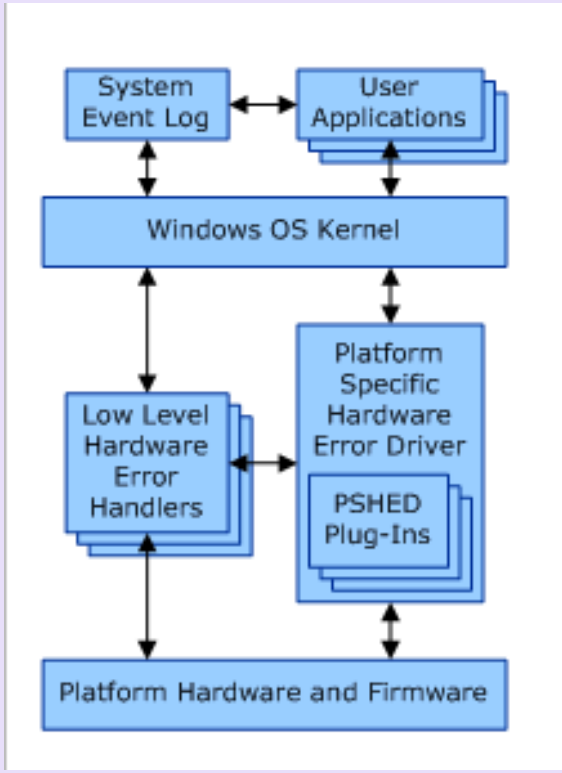
Debug Capabilities

- During development, a device may expose a buggy behavior on the PCIe that may trigger PCIe errors
- Such errors may reset the device or ignore the error, preventing further analysis of the error
- A mechanism allowing redirection of errors including details of the transaction that caused the error to the driver may help in the debug process

OS Error Handling

Windows Hardware Error Architecture

- The PCI bus driver contains a corresponding *low-level hardware error handler* (LLHEH) that performs the following tasks:
  - ✓ Acknowledges the hardware error
  - ✓ Captures the available error information related to the hardware error
  - ✓ Reports the hardware error condition to the operating system
- LLHEHs and the Windows kernel draw upon the services of the *platform-specific hardware error driver* (PSHED) to collect platform-specific error information
- Platform vendors can supplement the default PSHED functionality by providing PSHED plug-ins that take advantage of platform-specific capabilities

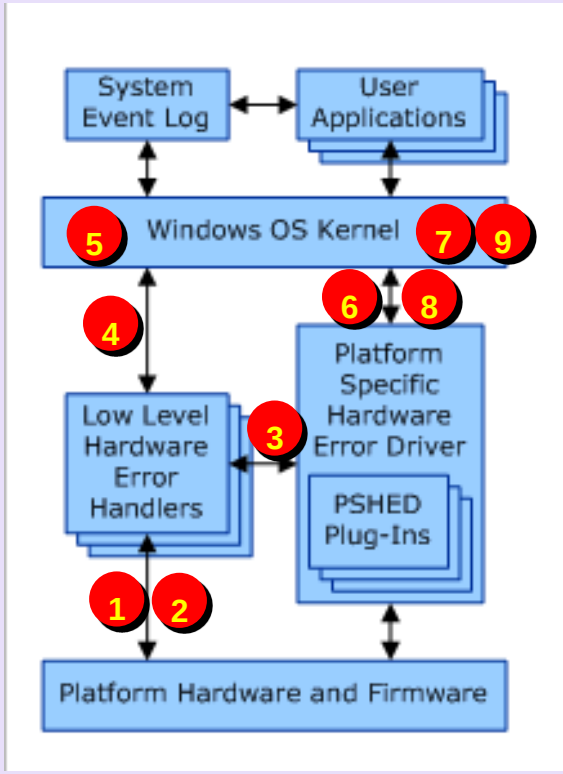


Source: http://msdn.microsoft.com

Nonfatal Uncorrected Error Recovery

1. The LLHEH is notified about the presence of the hardware error condition
2. The LLHEH retrieves hardware error information from the error source
3. The LLHEH calls into the PSHED (and plug-ins) to retrieve any platform-specific hardware error information
4. The LLHEH calls the Windows operating system kernel, passing it the error packet
5. The Windows kernel creates an error record with the information from the error
6. The Windows kernel calls into the PSHED to allow the PSHED (and plug-ins) to add sections to the error record
7. The Windows kernel attempts to recover from the error by trying to correct the hardware error condition
8. The Windows kernel then calls into the PSHED (and plug-ins) to give it an opportunity to perform any required recovery operations
9. If the hardware error was successfully corrected, the Windows kernel generates an ETW* event and logs the error information in the system event log

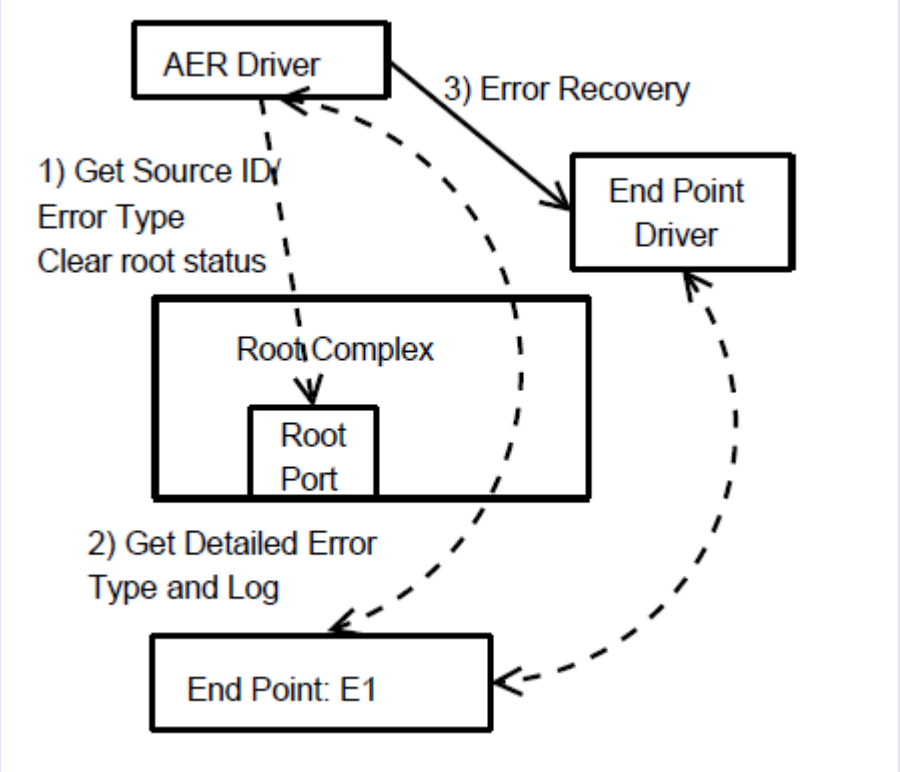
   If the hardware error was not corrected, the Windows kernel calls into the PSHED (and plug-ins) to save the error record. After the error record has been saved, the Windows kernel generates a bug check.



* ETW = Event Tracing for Windows

Error Reporting in Linux

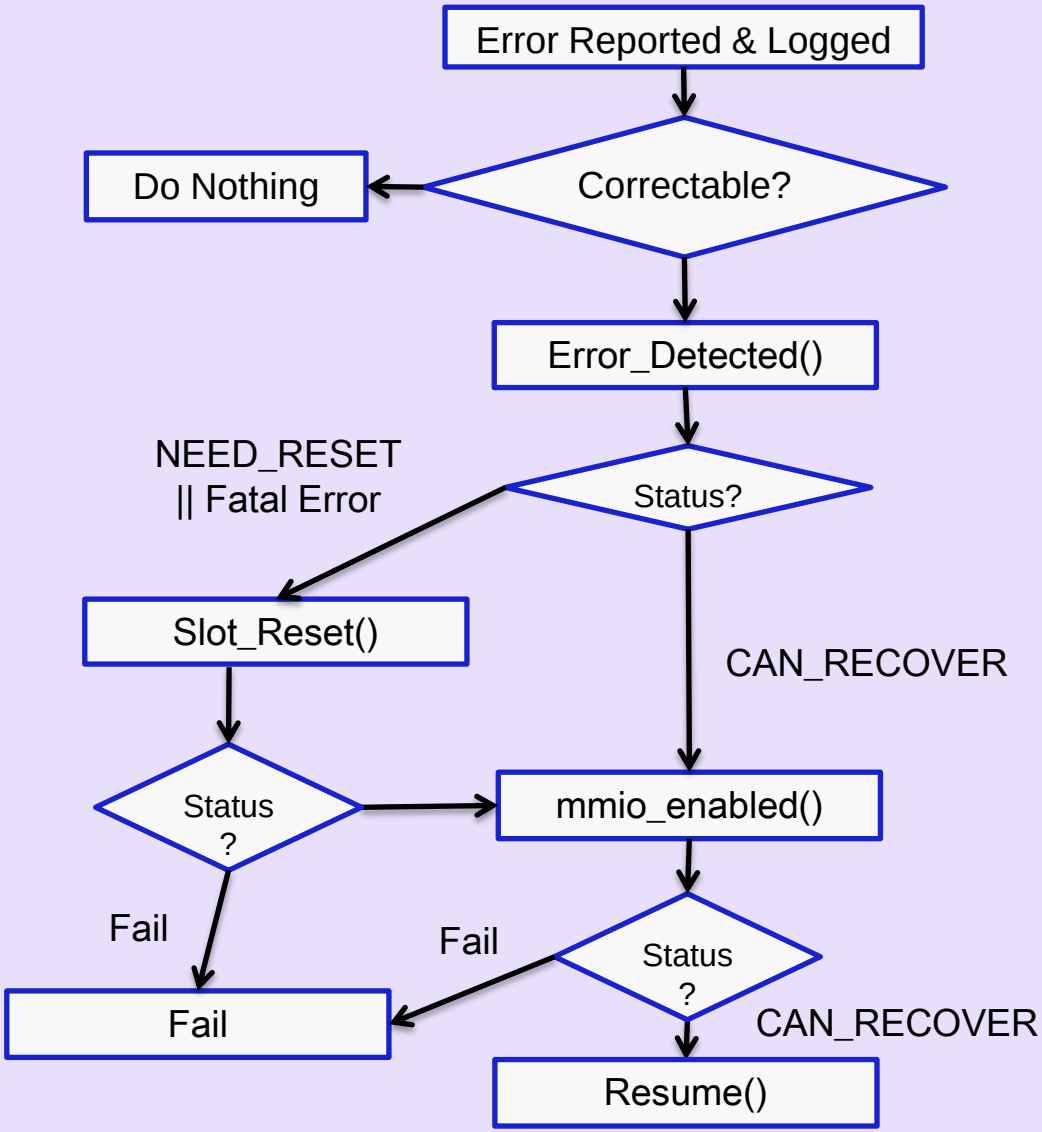
- PCIe Error Reporting:
  - End point (E1) sends an error message to Root Port
  - Root port logs the source of error and interrupts AER driver
- AER Error Logging:
  - AER driver queries Root Complex to get the source ID and Error Type and clears the Error Status in the Root Port (Step 1)
  - AER access end point to extract AER log and logs the error to system console (Step 2)
- AER Error Recovery (in case of non correctable errors):
  - AER Driver attempts error recovery with the help of the end point drivers. In case of fatal errors, the link must be reset as part of the recovery (Step 3)



Source: http://www.kernel.org/doc/ols/2007/ols2007v2-pages-297-304.pdf

Error Recovery in Linux

- Error is reported & logged (previous slide)
- If error is not correctable call error_detected. No access to device allowed.
- If returned status is NEED_RESET or error severity is fatal, reset the link
- If returned status is CAN_RECOVER, call mmio_enabled. Device can now be accessed.
- If returned status is CAN_RECOVER, call resume(). Recovery is done, Otherwise – recovery fails.



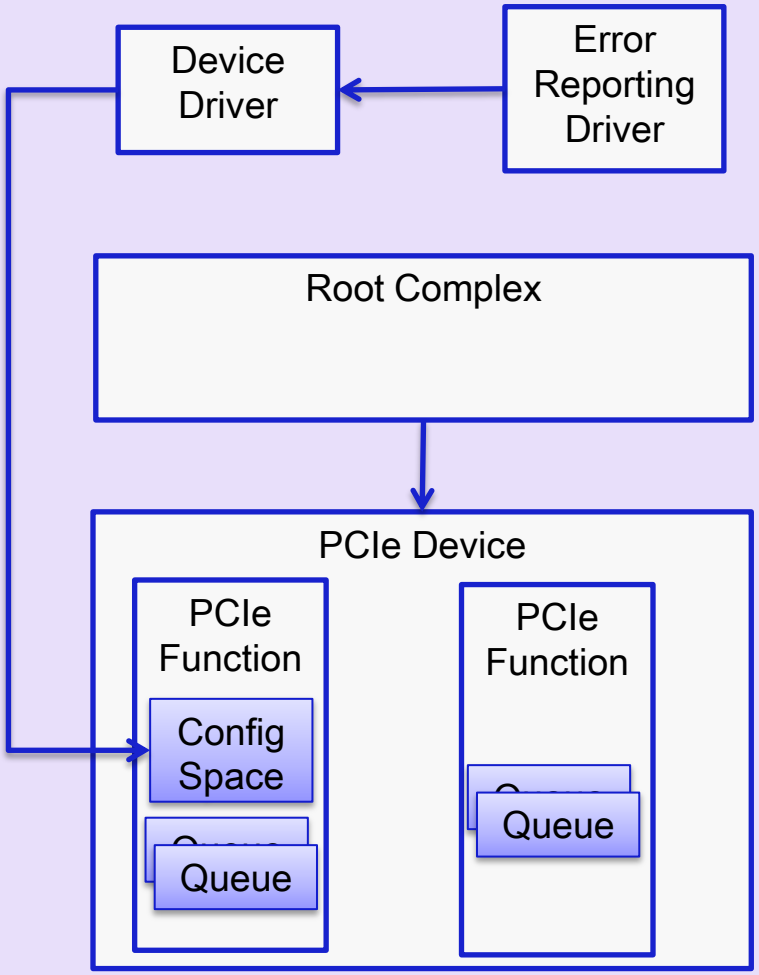
Error Containment by Driver

Error Containment by Driver

- As shown in previous slides, device driver can be involved in error recovery
- Can implement different behavior according to type of error
  - ✓ Transient Errors – errors that may have been caused by soft errors in memories (ECRC, Poisoned, Completion Timeout)
  - ✓ Non Transient Errors – errors caused by miss-programming or hardware failures (malformed, UR, CA, Unexpected completion)

Transient Error Handling

- Driver gets an error indication from the OS
- Collects information for AER log and statuses and from the device. Detects if the error is transient.
- Current Behavior – Reset the Device
- Option 1: Internally reset the affected function
- Option 2: Find the affected queue from the tag in the header log/internal device logging and restart only this queue



Non Transient Error Handling

- Non transient errors are not expected during regular operation
  - ✓ The case of untrusted agents will be covered later
- Can occur during SW/HW development or due to Firmware error
- Possible flow:
  - ✓ HW reports an error and freeze the device or log the state
  - ✓ Driver gets an error indication from OS/device
  - ✓ Collects information for AER report
  - ✓ Collects debug information from the frozen HW
  - ✓ Device is reset
- This flow is useful for debug

New Problems

New Problems/Issues

- Malicious User Mode driver or VF driver
- Sender involvement in recovery
- Peer to Peer

Malicious VM Driver

- A VF with device DMA capabilities is assigned to a VM
- The VF driver programs a DMA engine to use unmapped Guest Physical Addresses (GPA)/MSI-X vector
  - ✓ Unmapped write or unmapped MSI-X vector – write is discarded and error logged. ATC (Address Translation Cache) driver must make sure this error does not stall other VMs. May forward to PF device driver.
    - Current driver ignores the error – PF driver is not notified.
  - ✓ Unmapped read – ATC returns an unsupported request. May stall DMA in device.
    - HW DMA should be able to continue normal operation
    - PF Device driver should handle (not through OS handling)
    - Recent Intel NICs support this functionality.

Sender Involvement in Recovery

- Error is usually not detected by the device that created it
  - ✓ E.g. Malformed TLP is reported by receiver, but generated by another device
- The driver better suited to handle the error is the driver of the sender
- Need to allow redirection of the error to the sender driver
  - ✓ This is currently not supported by error handlers.

Peer to Peer

- Peer to Peer transaction involves two devices
  - ✓ An error in P2P TLPs may not impact the regular activity of the system.
  - ✓ There is no special handling of P2P transactions in current error handlers.
- Example:
  - ✓ MCTP (Management Control Transport Protocol) is a protocol defined by the DMTF that uses peer to peer VDMs (RID addressing) to transport data between PCIe endpoints
  - ✓ May choose to ignore transient uncorrectable errors as higher level protocol will handle packet drops
  - ✓ In case of non transient errors, the sender should be reset – not the receiver

Summary

- An extensive Error Recovery mechanism exists both in hardware (PCIe) and in the operating systems
- Driver has the ability to contain the error recovery to the specific unit impacted (specially for transient errors)
- New problems like non trusted agents and peer to peer usages were exposed that may require improved OS support
- Should consider opportunities to involve the agent that created the error in recovery process

Thank you for attending
PCI-SIG Developers Conference
Israel 2011

For more information please go to
www.pcisig.com

Backup

Linux Error Log Example

+—— PCI Express Device Error ——+
Error Severity : Uncorrected (Non-Fatal)
PCIE Bus Error type : Transaction Layer
Unsupported Request : First
Requester ID : 0500
VendorID=14e4h, DeviceID=1659h, Bus=05h, Device=00h, Function=00h
TLB Header:
04000001 0020060f 05010008 00000000
Broadcast error_detected message
Broadcast slot_reset message
Broadcast resume message
tg3: eth3: Link is down.
AER driver successfully recovered

Linux Error Recovery Statuses

- PCI_ERS_RESULT_RECOVERED: Everything is OK
- PCI_ERS_RESULT_CAN_RECOVER: Driver can recover if granted access to device
- PCI_ERS_RESULT_NEED_RESET: Driver requires a slot reset to recover
- PCI_ERS_RESULT_DISCONNECT: Driver returns this if it doesn't want to recover at all