

DAIMLER

Error Handling Strategies for a MOST Application Framework

Torsten Pech

Alexander Leonhardi

Andreas Vallentin

Overview

- Motivation
- Causes of errors in a MOST system
- Device model and state chart for error handling
- Error handling mechanisms on different layers
- Further work and conclusion

Motivation: Importance of Error Handling

MOST systems usually not safety relevant

However, MOST systems are highly interactive systems:

- errors can lead to application freezes, unavailability of functions, or display of wrong data
- effective error handling improves perceived quality

Why specific error handling for MOST?

- error handling from data base systems or safety critical systems too complex or requires too many resources
- error handling should not influence responsiveness of system
- specific characteristics of MOST bus and MOST application framework

Introduction: Scope of Error Handling

Goals/restrictions:

- define uniform error handling strategy:
still specific error handling necessary on application level
- only complement to robust system design
- error handling for synchronous and MHP connections not discussed here

Basic considerations:

- error responses need to be defined and implemented correctly
(MOST specification defines error responses on application level)
- error handling can only observe symptoms not actual cause
- important to consider error causes and their probabilities
- errors are detected and handled on different layers

Causes of Errors in a MOST System (1/3)

Message losses:

- result: one or multiple (burst) transmitted message are not received
- cause: usually a full receive buffer, sometimes device internal processing
- probability: single message losses very common, multiple common

Transmission errors:

- result: messages with wrong content are received
- cause: bit errors on MOST
- probability: very uncommon (bit error rate: 10^{-9})

Interrupted communication:

- result: observed as a burst of message losses
- cause: non-critical unlock
- probability: common during start-up and cranking

Causes of Errors in a MOST System (2/3)

Configuration errors:

- result: communication does not reach (correct) device
- cause: configuration changes, device resets
- probability: common during start-up and cranking

Unavailability of device functions:

- result: device does not respond or responds with error response
- cause: high processing load, external influences, sequence errors
- probability: quite common (application dependent)

Duplicate messages:

- result: messages are received twice
- cause: error handling mechanisms in combination with message losses
- probability: uncommon (low probability of lost messages)

Causes of Errors in a MOST System (3/3)

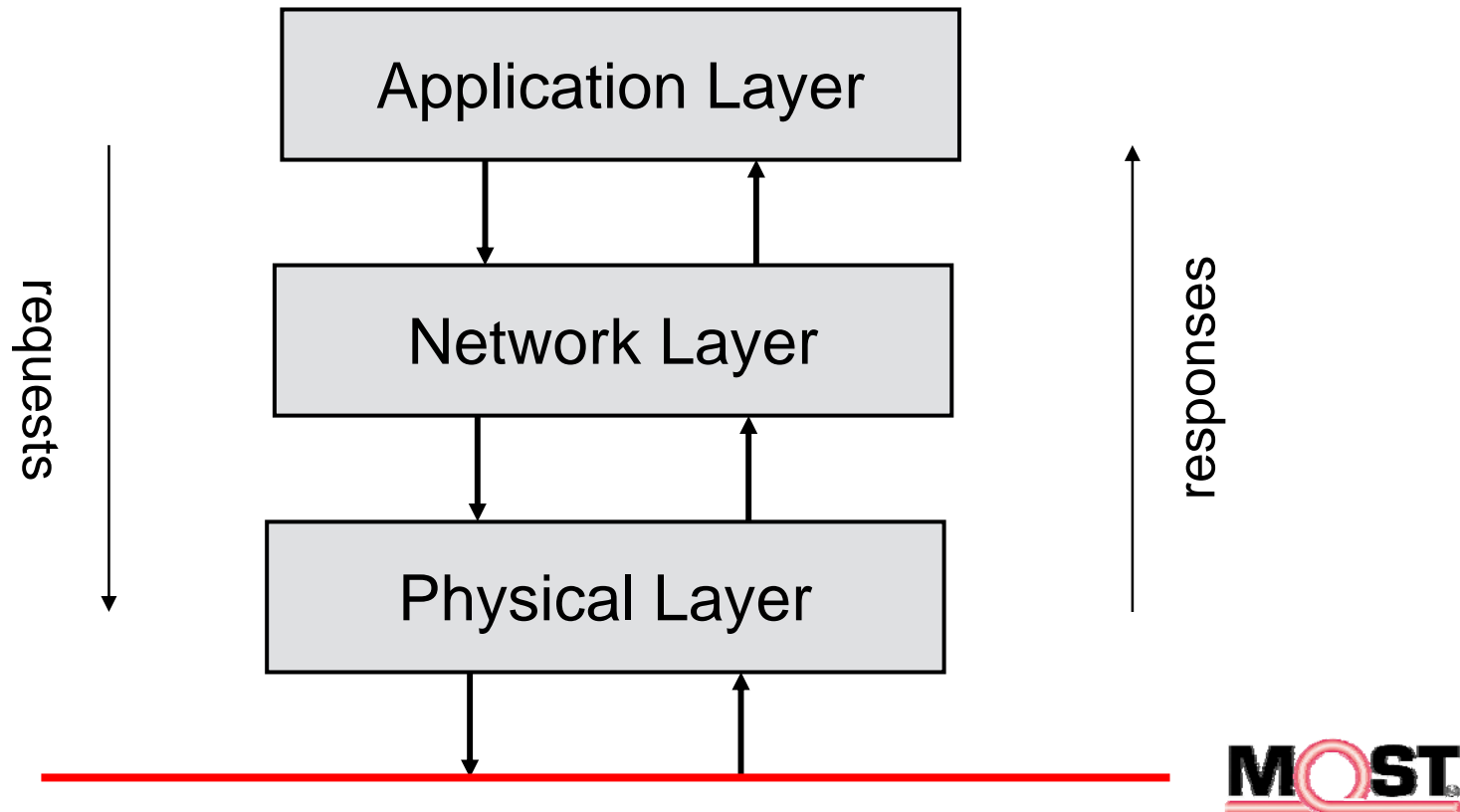
Message sequence errors:

- result: messages are received in wrong order
- cause: message passing each other because of retries
- probability: uncommon (single message queues)

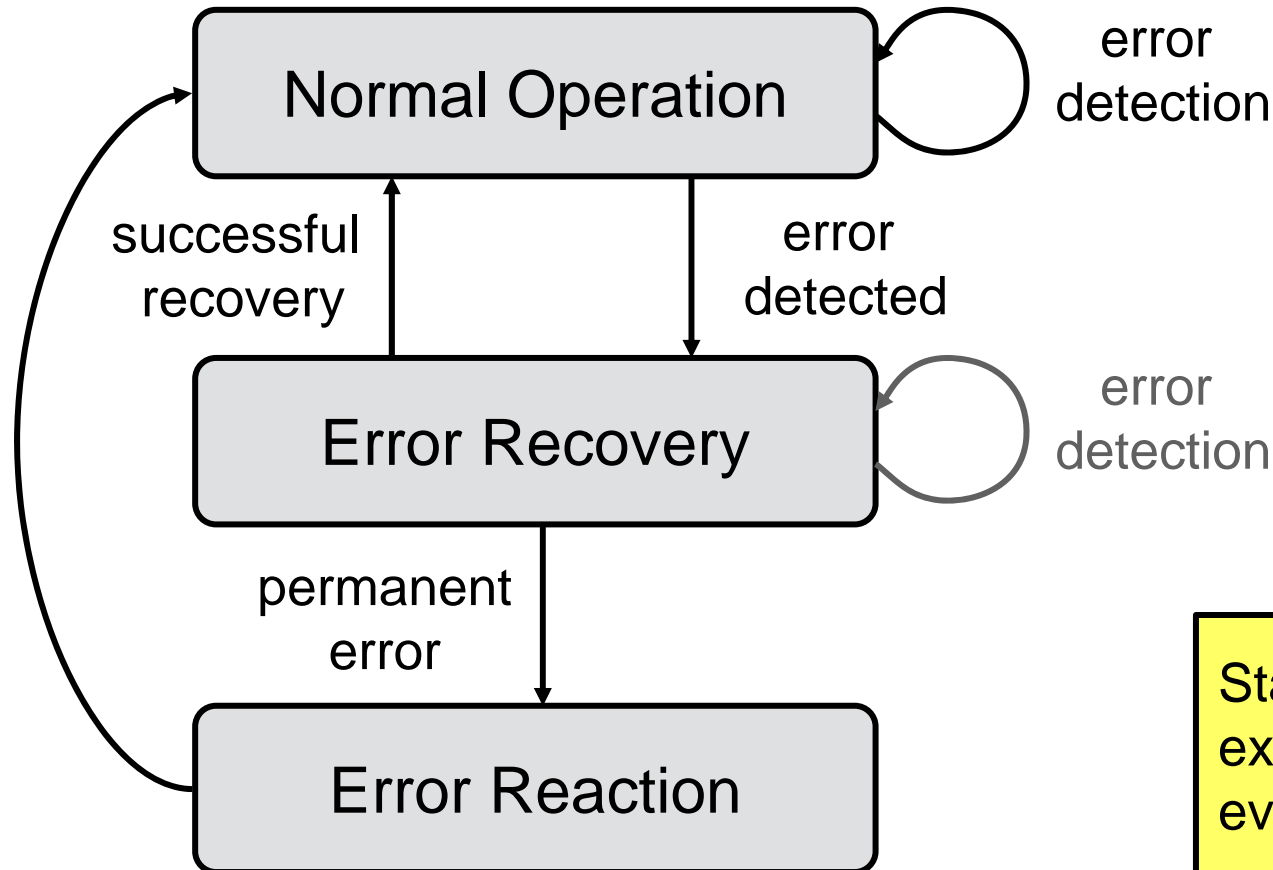
Device failure and implementation errors:

- result: correct function not executed, other errors observed
- cause: implementation errors (HW and SW)
- probability: uncommon, have to be found during system integration

Error Handling: 3-layered Model



Error Handling: State Chart



Error Handling on Physical Layer

Low level retries:

- ACK/NAK mechanism detects messages that have not been received
- low level retries automatically generated by MOST INIC
- number and interval of low level retries configurable
(Note: large number of retries reduce probability of message loss but cause high message load and block sender)

CRC mechanism:

- CRC mechanism detects bit errors in control messages
- further higher level handling of bit errors usually not necessary

Detection of critical unlock:

- after 70 ms of unlock a critical unlock is detected and a reset of the MOST system is caused
- higher level error handling has to deal with 70 ms bursts of message losses

Error Handling on Network Layer

Error handling based on error reports from NetServices

Network Master:

- detects configuration changes and initiates network scan
- handles configuration errors (e.g., duplicate addresses)
- responsible for distributing consistent configuration in MOST system

Mid level retries:

- can extend time interval covered by low level retries and while reducing load caused on network
- low and mid level retries should cover time of non-critical unlock

Error Handling on Application Layer (1/2)

Error handling according to classes:

- ignored errors: no retries or error reaction
- non-fatal errors: continue with assumed reasonable results after retries
- fatal errors: indicate unavailability of function block

Error handling for timeouts and error responses:

- missing responses of properties and methods:
non-fatal error with 1 retry
- error codes 0x01 or 0x02: FBlockID or InstID not available:
fatal error, 1 retry and another retry after checking central registry
- error codes 0x03 to 0x07: syntax or parameter errors:
non-fatal error with 1 retry

Error Handling on Application Layer (2/2)

Error handling for timeouts and error responses (cont.):

- error code 0x20: function specific:
application specific, default: non-fatal error with 1 retry
- error codes 0x40 to 0x42: temporary unavailability (e.g., Busy):
non-fatal error with 5 retries covering longer time interval
- errors responses that can't be mapped to a particular request:
ignored error

Handling of Permanent Errors

Permanent errors should not happen too often after successful system integration phase

- Amount and timing of retries shall be dimensioned in such a way that all typical error causes (e.g. high processor load) are covered
- Errors caused by implementation, specification or configuration issues should be solved

Individual handling necessary to achieve optimum trade off between

- Long term system stability
- Visibility to the customer

Conclusion and Further Work

Overview of error handling in MOST systems given

- possible errors with their result, cause and probability
- generic error handling strategy
- errors have to be handled on suitable layer
- applications have to be aware of error handling

Desirable: more awareness and support for error handling

Define error handling strategies in MOST Cooperation:

- analysis of error causes to quantify probabilities
- application recommendation for building robust MOST systems
- application recommendation for error handling strategies

Contact

Thank you very much for your attention!

Questions?

Contact:

Torsten Pech, Alexander Leonhardi, Andreas Vallentin

HPC: X908

71059 Sindelfingen

Tel.: +49 7031 90 83503

Email: torsten.pech@daimler.com