802.1CB Failure Mode Considerations

Johannes Specht	johannes.specht AT uni-due.de	**Univ. of Duisburg-Essen**
Soheil Samii	soheil.samii AT gm.com	**General Motors**
Helge Zinner	helge.zinner AT de.bosch.com	**Robert Bosch GmbH**

Contents

802.1CB

- 802.1CB is intended to provide robust and fault-tolerant communication in safety-critical systems
- Provides network availability in case of link or bridge failure (“fail silent”). Building block to avoid single-point failures in a safety-critical system.

Goals of this slide set

- Shows fault scenarios that may not be covered by the current 802.1CB Draft (1.0)
- Presents potential countermeasures to address or avoid these scenarios for discussion
- Relate to current industry standards on functional safety (e.g., ISO-26262 and IEC-61508)

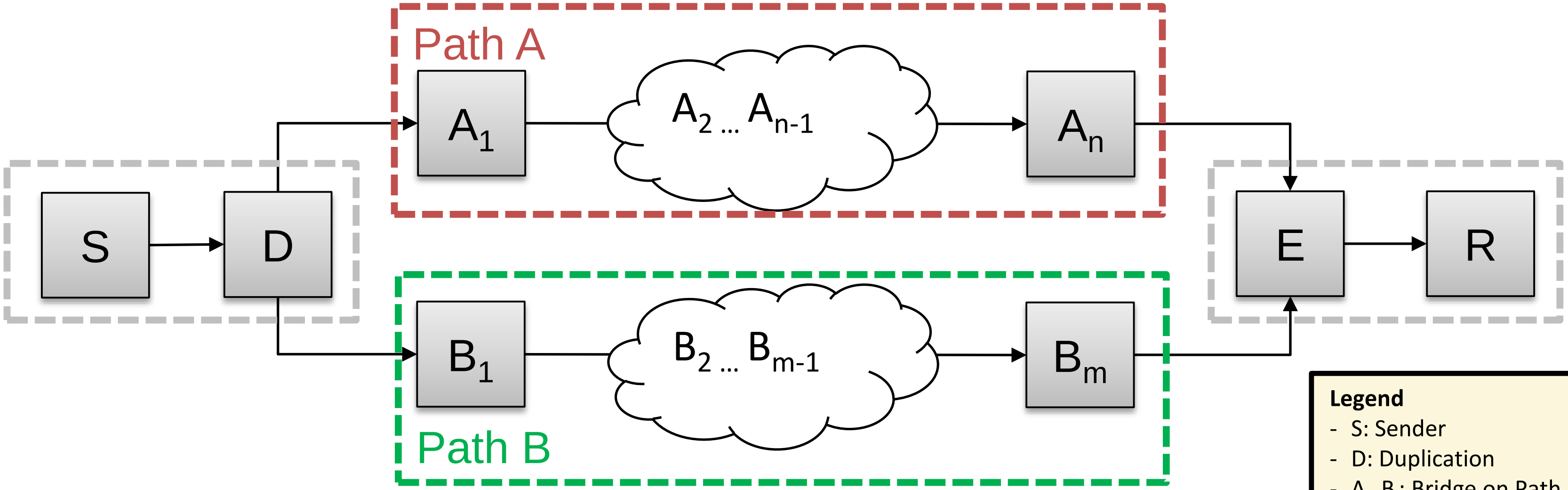
Functional Safety

- Covering Functional Safety requirements at 802 level instead of at system level
  1. Can be done by specification (i.e., not affecting implementations)
  2. Can be done by additional mechanisms in end-stations and bridges (e.g., Qci, CB)
  3. Must be done in bridges when it is technically infeasible to fulfil the requirements at system level
- Automotive OEMs, Tier-1 suppliers, and semiconductor manufacturers need to fulfil Functional Safety requirements for most in-vehicle digital systems involved in a safety-critical feature (e.g., suspension, breaking, steering, ADAS, automated driving, and active safety)
- Depending on the *safety integrity level*, certain diagnostic coverage of failure modes is required

Assumption on desired behavior and faulty components

- Faulty components shall be fail silent instead of delivering wrong data
- Don‘t expect that components recover from a failure, nor try to bring them back as fast as possible

Basic Scenario



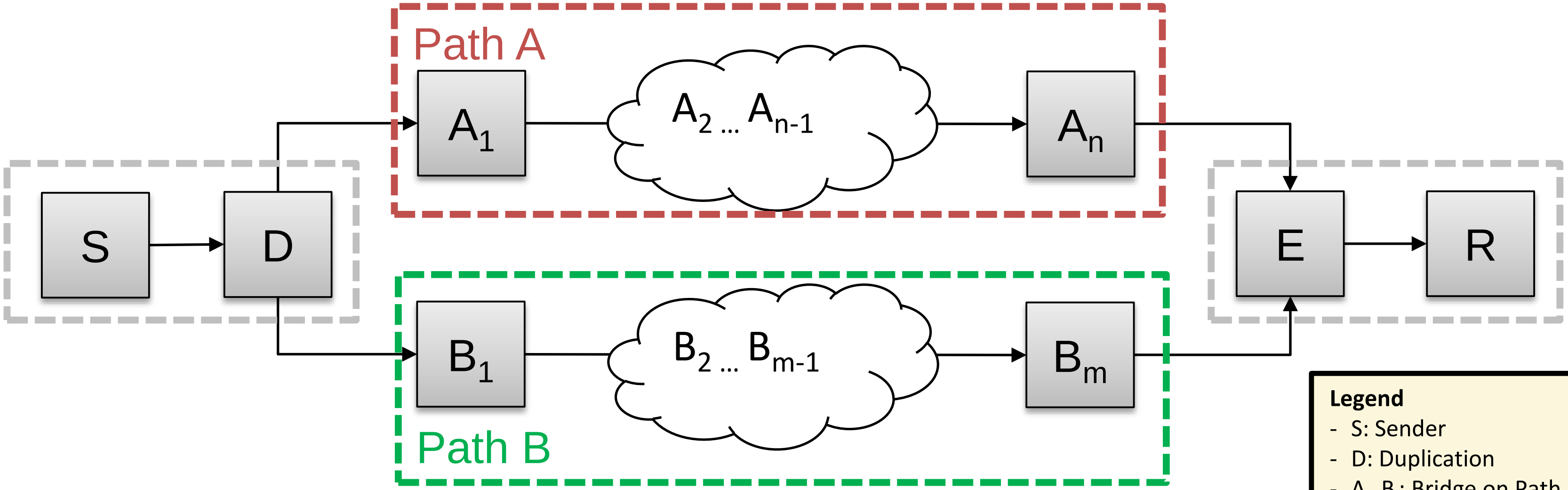
Legend

- S: Sender
- D: Duplication
- A_i, B_j: Bridge on Path A/B
- E: Duplicate elemination
- R: Receiver

- **One sender (S) with a duplication function/bridge (D), one receiver (R) with a duplicate elimination function/bridge (E)**
  - S+D and/or E+R may be physically combined
  - S and R may be bridges or end stations

- **2 Paths (A and B) used by 802.1CB**
  - Path A: *n* bridges and *n+1* wires
  - Path B: *m* bridges and *m+1* wires

Basic Scenario



Legend
- S: Sender
- D: Duplication
- A_i, B_j: Bridge on Path A/B
- E: Duplicate elemination
- R: Receiver

Reliability

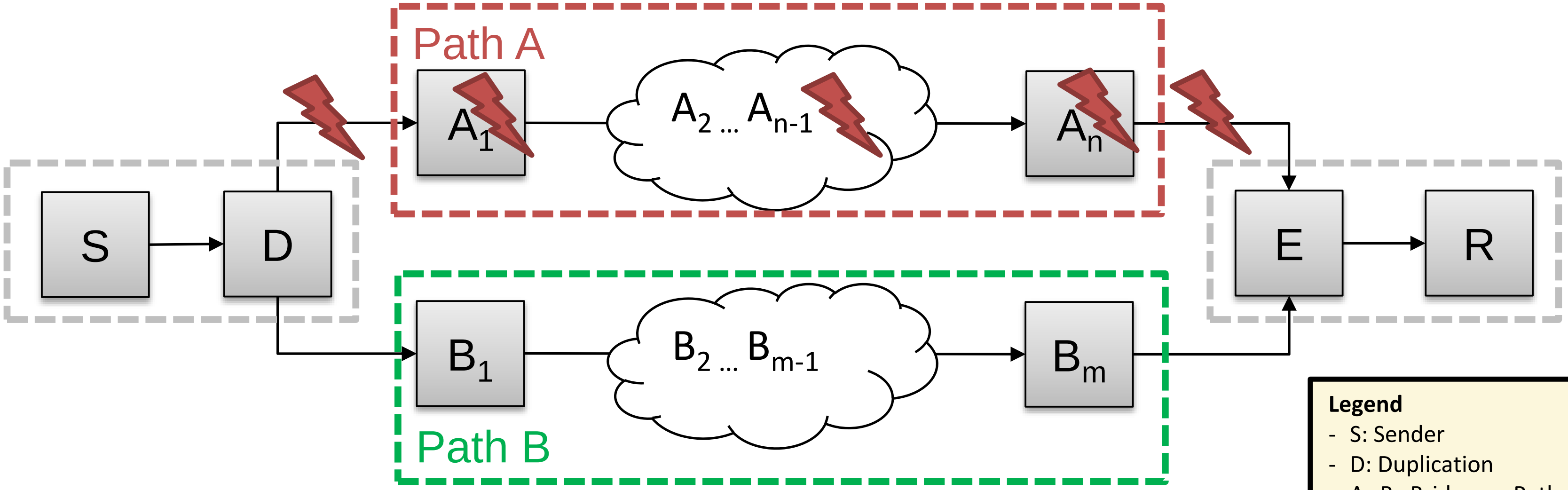
- **Paths A and B are parallel**
  - INDEPENDENT failure probabilities (<< 1) of both paths are multiplied (→ <<< 1)
  - If a component on path A fails, path B is still delivering data
- **S, D, E, and R and wires in between are non-redundant, i.e. single points of failure (?)**
  - Failure probabilities (<< 1) are “nearly“ added (→ < 1)
  - Other measures are to be taken by system engineers to make sure that S, D, E, or R are not single-point failures
  - **NOT in scope of 802.1CB**

Failure Modes

<table>
<tr><th></th><th>Failure Mode</th><th>Interpretation</th></tr>
<tr><td rowspan="2">Adressed by <b>802.1CB</b>
(not in scope of this slide set)</td><td>Failure of communication peer</td><td>Failure that results in that a box stops communicating (“fail-silent”)</td></tr>
<tr><td>Message Loss</td><td>Packet is dropped (e.g., by FCS verification or buffer overrun)</td></tr>
<tr><td rowspan="2">Job of <b>802.1Qci</b>
(not in scope of this slide set)</td><td>Insertion of Message</td><td>New packets are “spawned”, existing packets are forwarded incorrectly</td></tr>
<tr><td>Masquerading</td><td>Packet gets a wrong SA, DA, Tag, etc.</td></tr>
<tr><td>Not yet considered
<b>(802.1CB does not guarantee in order delivery)</b></td><td>Resequencing</td><td>Out-of-order delivery</td></tr>
<tr><td rowspan="3">Initially addressed in this slide set</td><td>Message Corruption</td><td>Bitflips, bad octets, oversized packets, etc.</td></tr>
<tr><td>Unintended Message Repetition</td><td>The same packet is transmitted repeatedly</td></tr>
<tr><td>Message Delay</td><td>Packets remains longer than expected in a queue</td></tr>
</table>

Cmp. e.g. ISO-26262 „Road vehicles — Functional safety” Part 5, Annex D, Table D.1

Issue 1: Cut-Through and Corrupt Data

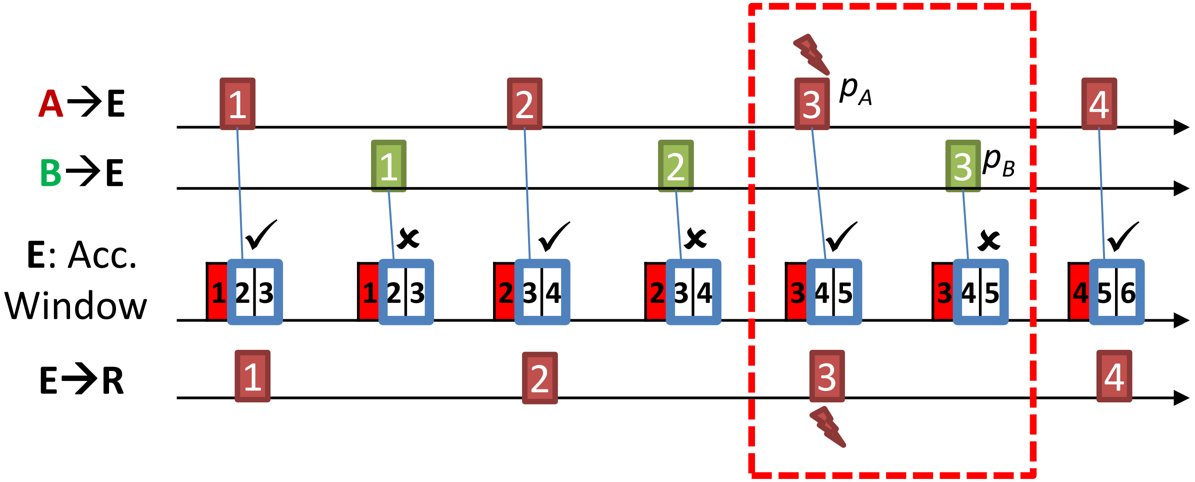


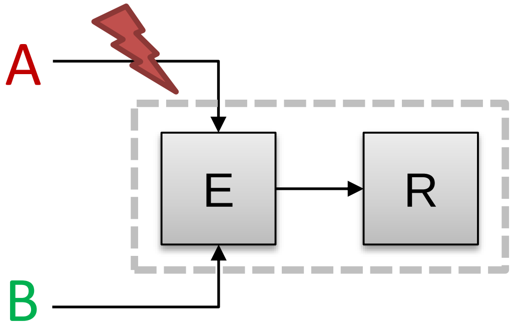
Legend
- S: Sender
- D: Duplication
- A_i, B_j: Bridge on Path A/B
- E: Duplicate elemination
- R: Receiver

Description

- *Message corruption* on path A causes data error in a packet p_A.
- At least E is performing cut-through forwarding.
- Failing components:
  - (n+1)*wire + n*Bridge

Issue 1: Cut-Through and Corrupt Data



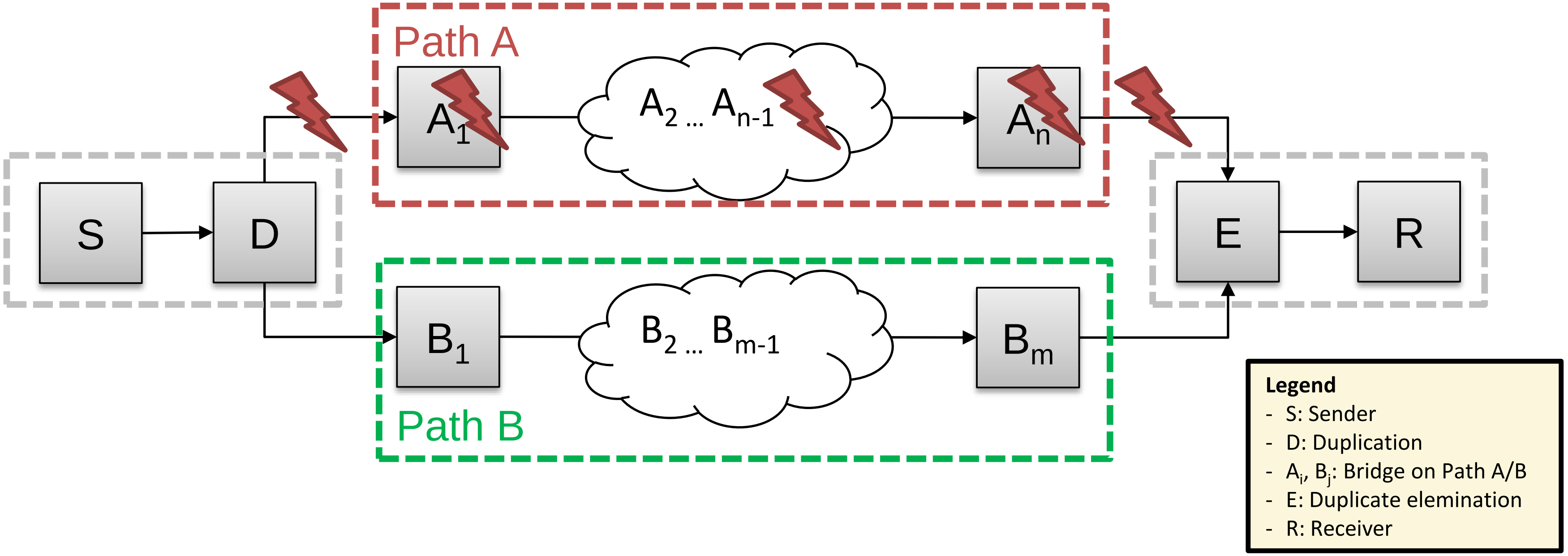


Legend
- S: Sender
- D: Duplication
- A_i, B_j: Bridge on Path A/B
- E: Duplicate elemination
- R: Receiver

Description

- *Message corruption* on path A causes data error in a packet p_A. At least E is performing cut-through forwarding.
- Consequence:
  - p_A is accepted by E, elimination of (fault free) duplicate p_B from channel B
- Present countermeasures:
  - **None:** FCS check in E is performed at the end of transmission

Issue 2: Cut-Through and Corrupt Sequence Numbers

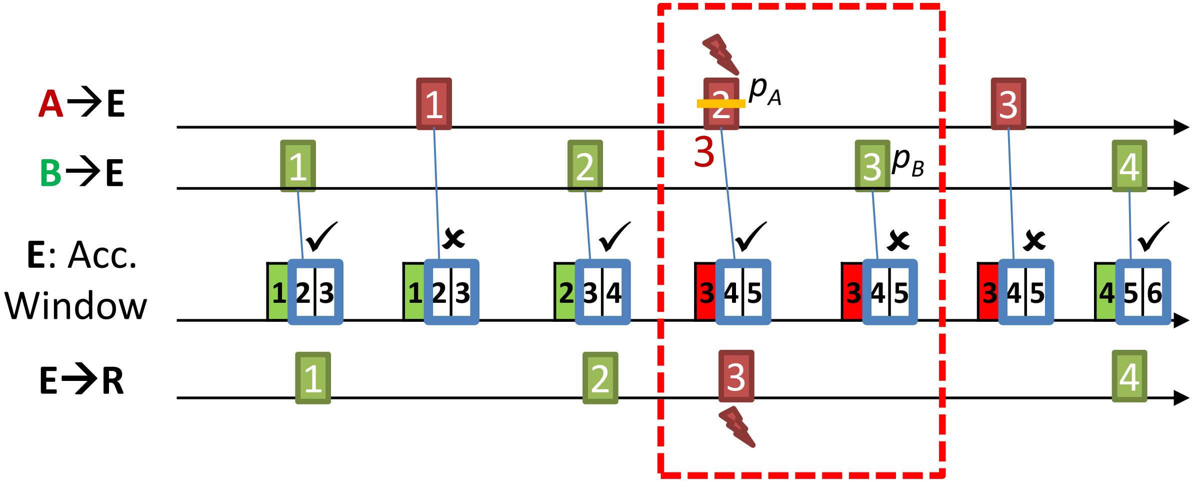


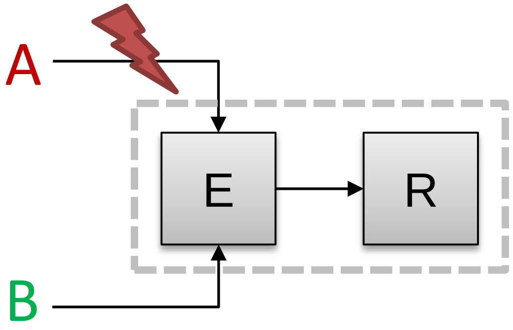
Legend
- S: Sender
- D: Duplication
- A_i, B_j: Bridge on Path A/B
- E: Duplicate elemination
- R: Receiver

Description

- *Message corruption* on path A causes erroneous sequence number in a packet p_A. At least E is performing cut-through forwarding.
- <u>Failing components:</u>
  - (n+1)*wire + n*Bridge

Issue 2: Cut-Through and Corrupt Sequence Numbers



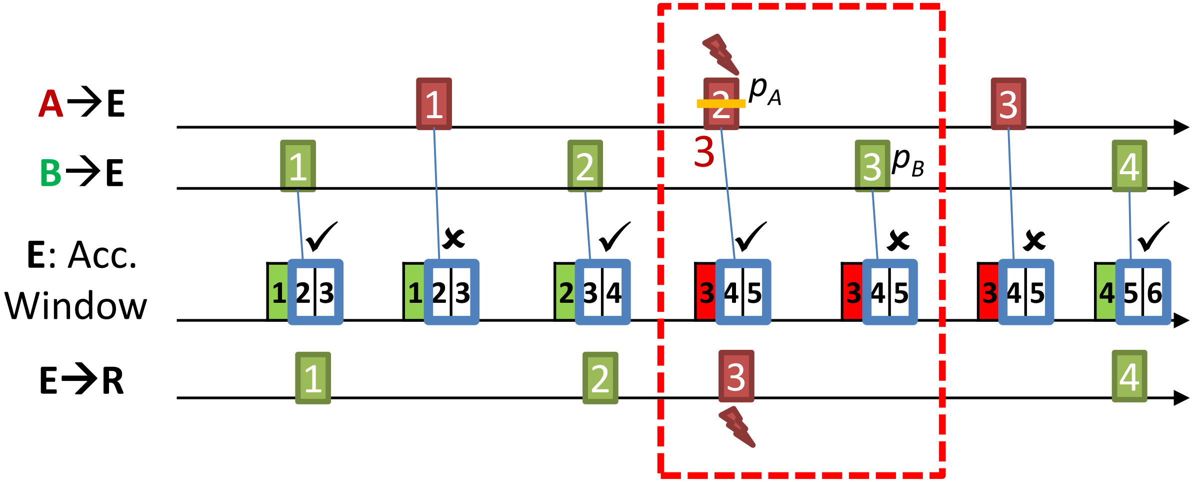


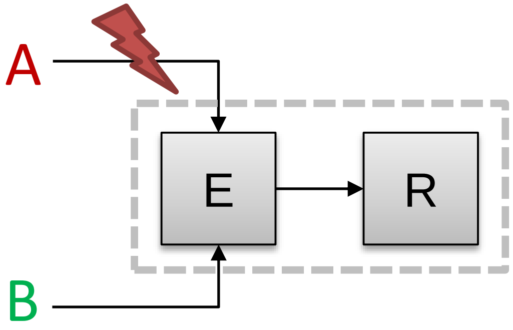
Legend
- S: Sender
- D: Duplication
- A_i, B_j: Bridge on Path A/B
- E: Duplicate elemination
- R: Receiver

Description

- *Message corruption* on path A causes erroneous sequence number in a packet p_A. At least E is performing cut-through forwarding.
- Consequence:
  - p_A is accepted by E, elimination of (fault free) duplicate p_B from channel B
  - In case of multiple broken sequence numbers, path A can (falsely) take over sequence number alignment in E
- Present countermeasures:
  - **None:** FCS check in E is performed at the end of transmission

Issues 1 & 2: Potential Countermeasures



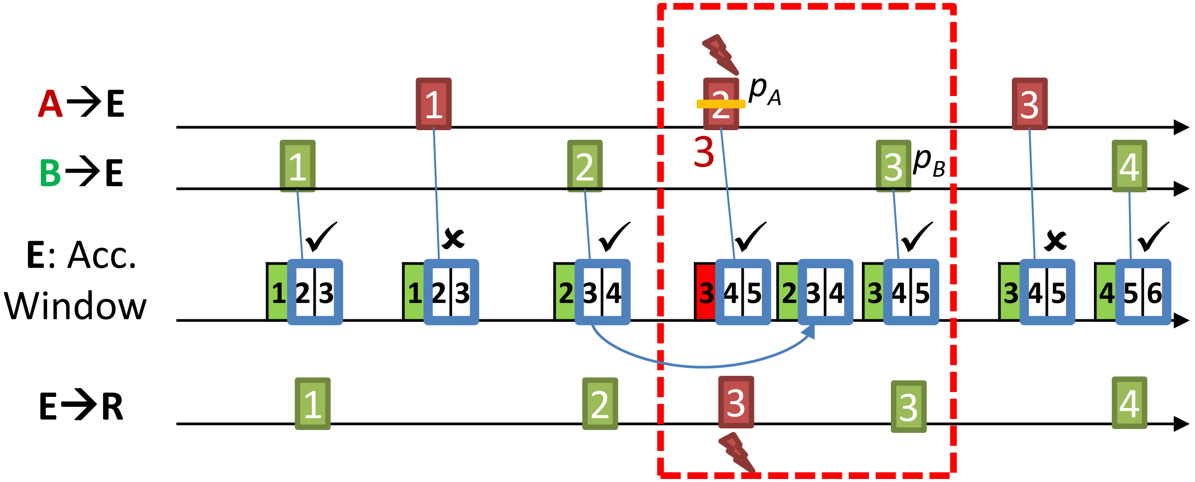


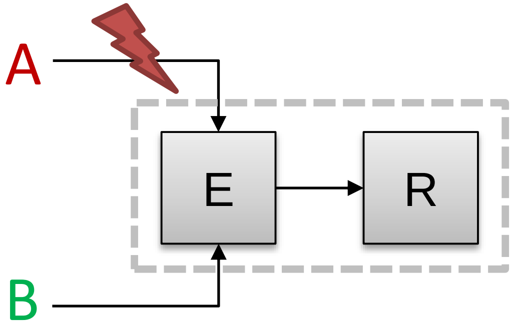
Legend
- S: Sender
- D: Duplication
- A_i, B_j: Bridge on Path A/B
- E: Duplicate elemination
- R: Receiver

Specification option

- *Don‘t do cut-through in E!*
  Enforce finished FCS check before reaching 802.1CB FSMs at egress
- Describe the impact of cut-through and store & forward on reliability in 802.1CB
- Non-duplicating bridges shall treat the sequence number as data (i.e., not recompute it in any way).

Issues 1 & 2: Potential Countermeasures





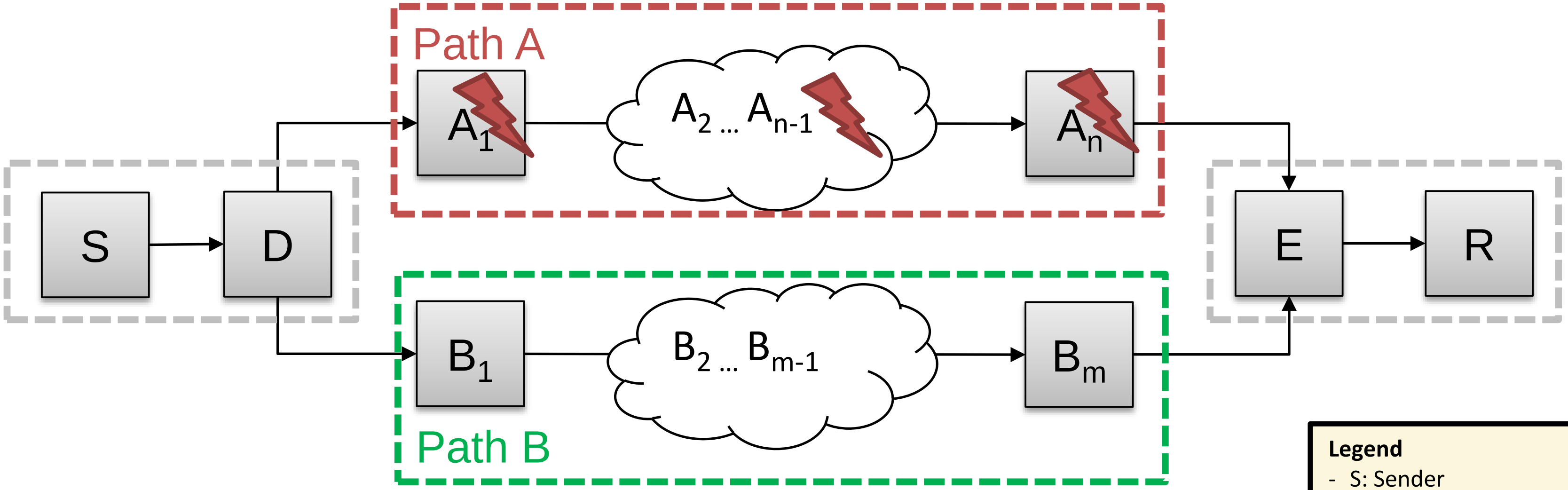
Legend
- S: Sender
- D: Duplication
- A_i, B_j: Bridge on Path A/B
- E: Duplicate elemination
- R: Receiver

Implementation option

- Rollback of CB state in E if FCS check of p_A fails at packet end after FCS-Check
  - p_A is forwarded by E to R
  - CB state is reverted after FCS check of p_A
  - p_B *is* not eliminated but forwarded as first duplicate
- Negative implications:
  1. Increased complexity of bridge implementation
  2. Forwarding p_A plus p_B causes overload at the output of E
     - If R is a bridge and implements policing, it may diagnose E as faulty (false-positive)
     - E itself cannot diagnose path A as faulty by policing: The overload is only visible by channels A&B in combination

→ ***Seems to be a bad idea. Propose to stick with the specification option (previous slide).***

Issue 4: (Any) Message repetition by bridges

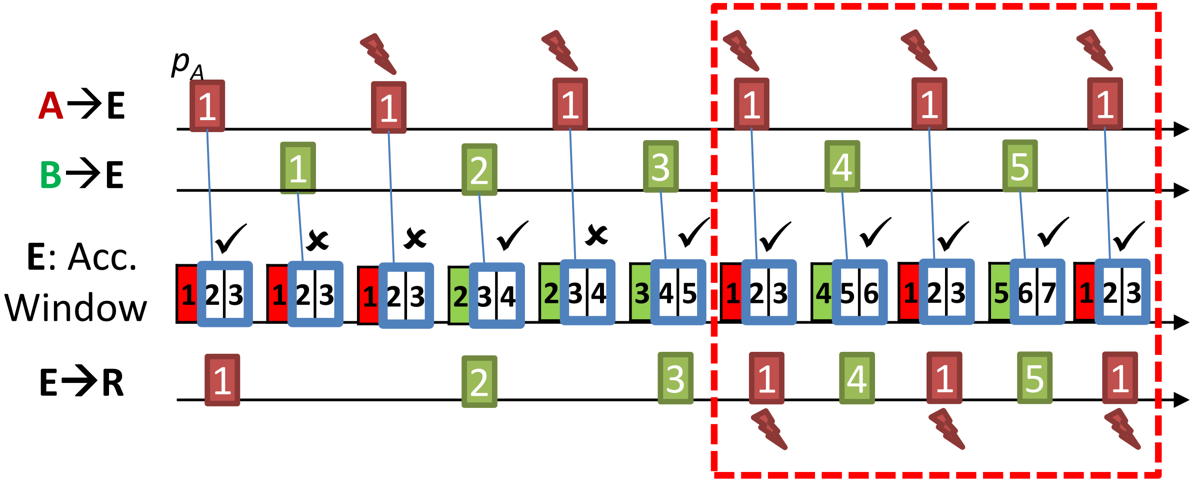


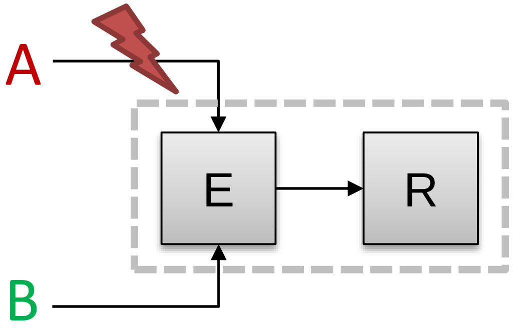
Legend
- S: Sender
- D: Duplication
- A_i, B_j: Bridge on Path A/B
- E: Duplicate elemination
- R: Receiver

Description

- *Message repetition* on path A causes repeated transmission of a packet p_A.
- Failing Components:
  - n*Bridge
  - There is no wire fault that can lead to message repetition (message repetition requires memory; wires don‘t have memory ☺)

Issue 4: (Any) Message repetition by bridges



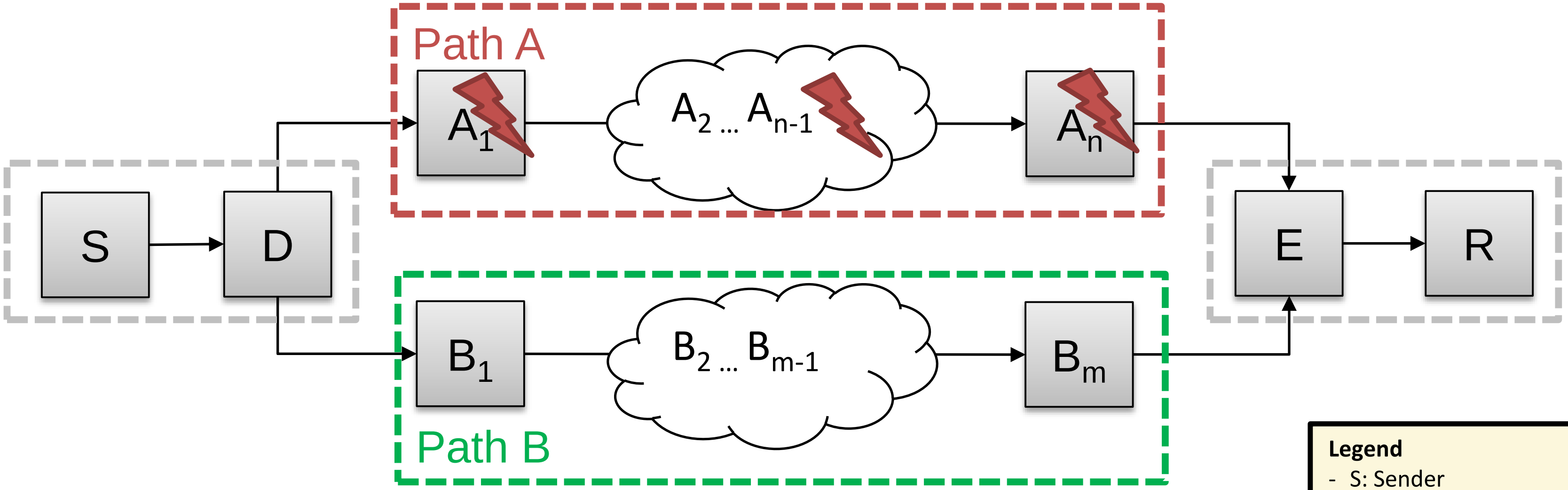


Legend
- S: Sender
- D: Duplication
- A_i, B_j: Bridge on Path A/B
- E: Duplicate elemination
- R: Receiver

Description

- *Message repetition* on path A causes repeated transmission of a packet p_A.
- <u>Consequence:</u>
  - Alternating alignment of the sequence history window to the sequence number of path A and path B (reset at "nearly" every sequence number), i.e. … → p_A → $p_{B,1}$ → p_A → $p_{B,2}$ → p_A → $p_{B,3}$ → …
  - „Nearly" duplicate load sent by E, may cause false positive 802.1Qci diagnosis of E by R. **R may block E entirely.**
- Present countermeasures:
  - **None**

Issue 5: Unaligned Message Delays



Legend

- S: Sender
- D: Duplication
- A_i, B_j: Bridge on Path A/B
- E: Duplicate elemination
- R: Receiver

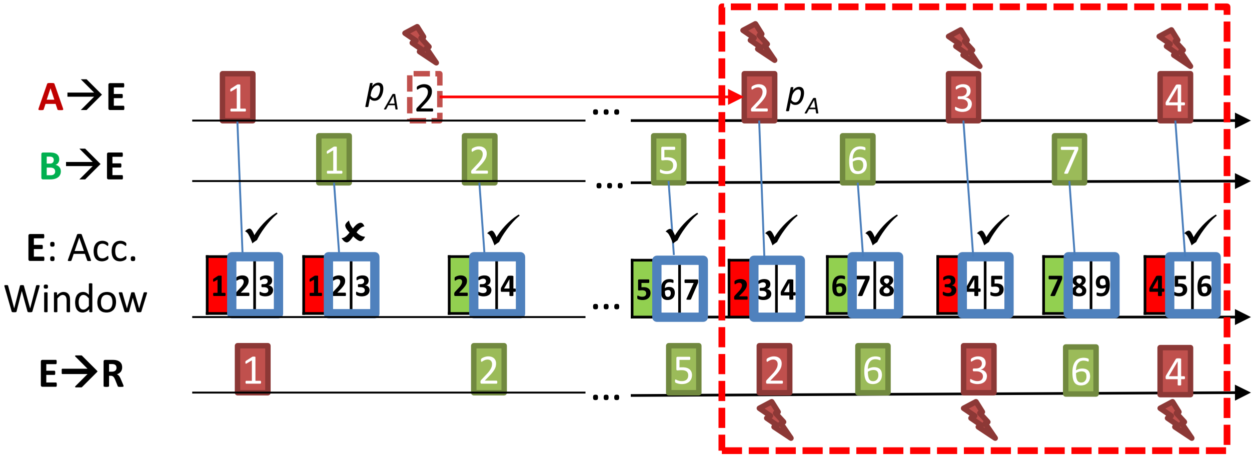
Description

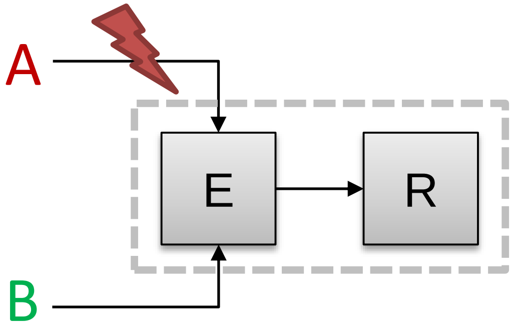
Message delay on path A delay delivery of a packet p_A (and consecutive packets) to E. Delay is ***out of***

*n*seq. num. range + [0; seq. recovery history length]*   *n=1,2,3,…*

- Failing components:
  - n*Bridge
  - Wires are explicitly excluded: Long delays require memory

Issue 5: Unaligned Message Delays



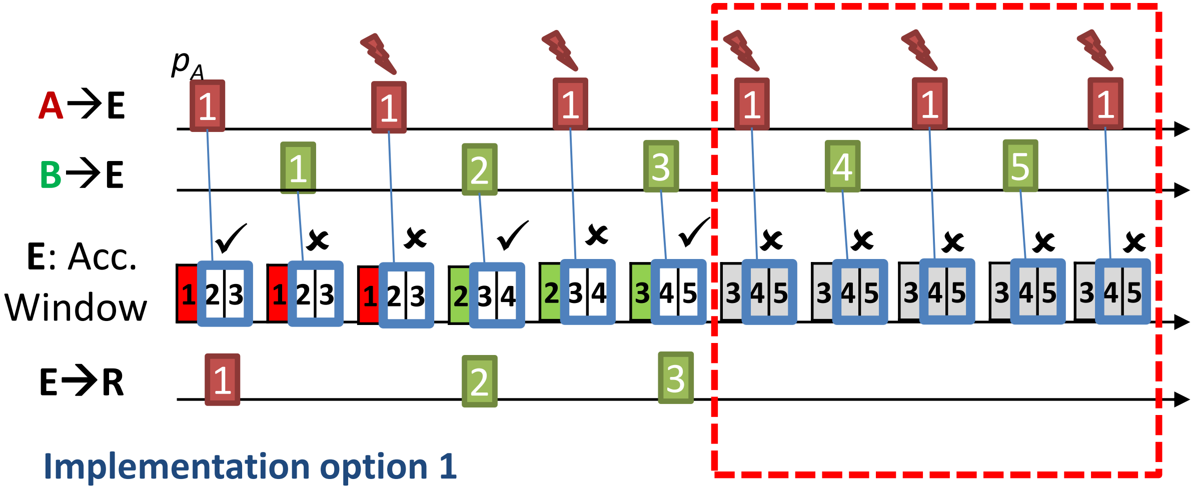


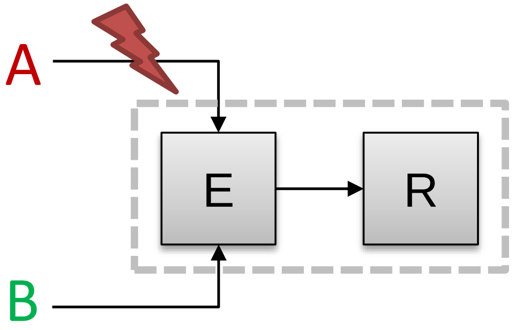
Legend
- S: Sender
- D: Duplication
- A_i, B_j: Bridge on Path A/B
- E: Duplicate elemination
- R: Receiver

Description

- *Message delay* on path A delay delivery of a packet p_A (and consecutive packets) to E. Delay is ***out of***
  *n*seq. num. range + [0; seq. recovery history length]* *n=1,2,3,…*
- Consequence:
  - Alternating alignment of the sequence history window to sequence number of path A and path B (reset at "nearly" every sequence number), i.e.
    … → p_A → $p_{B,1}$ → p_A → $p_{B,2}$ → p_A → $p_{B,3}$ → …
  - „Nearly" duplicate load sent by E, may cause false positive diagnosis of E by R
- Present countermeasures:
  - **None**

Issues 4&5: Potential Countermeasures





Legend
- S: Sender
- D: Duplication
- A_i, B_j: Bridge on Path A/B
- E: Duplicate elemination
- R: Receiver

Implementation option 1

- Optionally disable recovery sequence number reset
- Once *any* sequence number out of the recovery history length is observed, assure that E no longer forwards the stream to R
- Permanently drops all packets of the stream from *both*, path A and B (??)
- Either implemented in 802.1CB, or 802.1CB triggers 802.1Qci

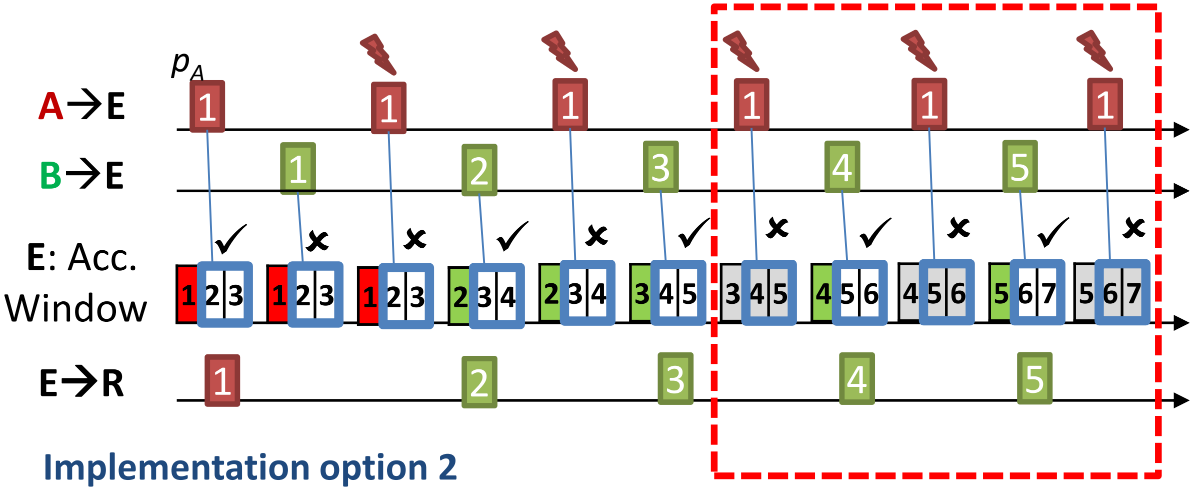
1. Prevents the overload sent to R, i.e. avoids false positive diagnosis by R or congestion, but ..

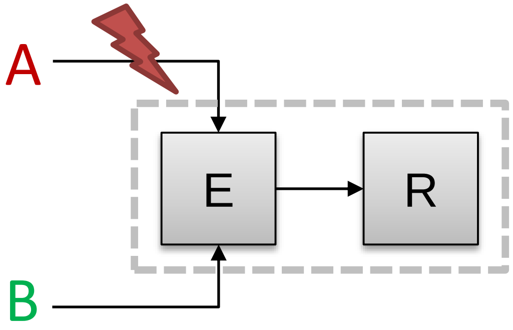
2. Does not improve the reliability of the affected stream itself, since path B packets will also be dropped

→ ***<u>May</u> be sufficient for fail-silent applications***

Note:
Illustration shows the countermeasure for issue 4: message repetition

Issues 4&5: Potential Countermeasures





Legend
- S: Sender
- D: Duplication
- A_i, B_j: Bridge on Path A/B
- E: Duplicate elemination
- R: Receiver

Implementation option 2

- Store recovery sequence number for each path of a stream independently, e.g. 2 offsets for path A and B
- Attention(!):
  - Requires more state, i.e. multiple recovery sequence numbers per stream
  - Sequence history window itself remains as it is (one per stream)

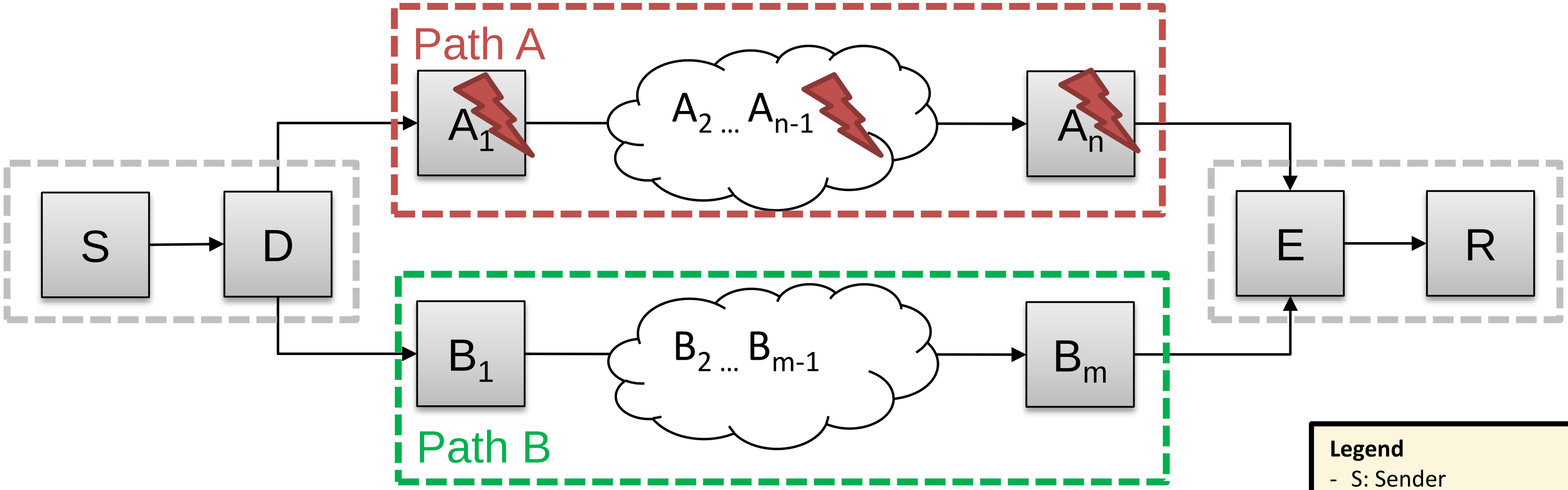
1. Prevents the overload sent to R <u>and</u> …

2. Improves the reliabilty of the affected stream itself (the fault-free path will get through)

→ ***Open Topic: Masked path (A) invisible to listeners, i.e. even fail silent applications can‘t switch off***

Note:
Illustration shows the countermeasure for issue 4: message repetition

Issue 6: Aligned Message Delays



Legend

- S: Sender
- D: Duplication
- A_i, B_j: Bridge on Path A/B
- E: Duplicate elemination
- R: Receiver

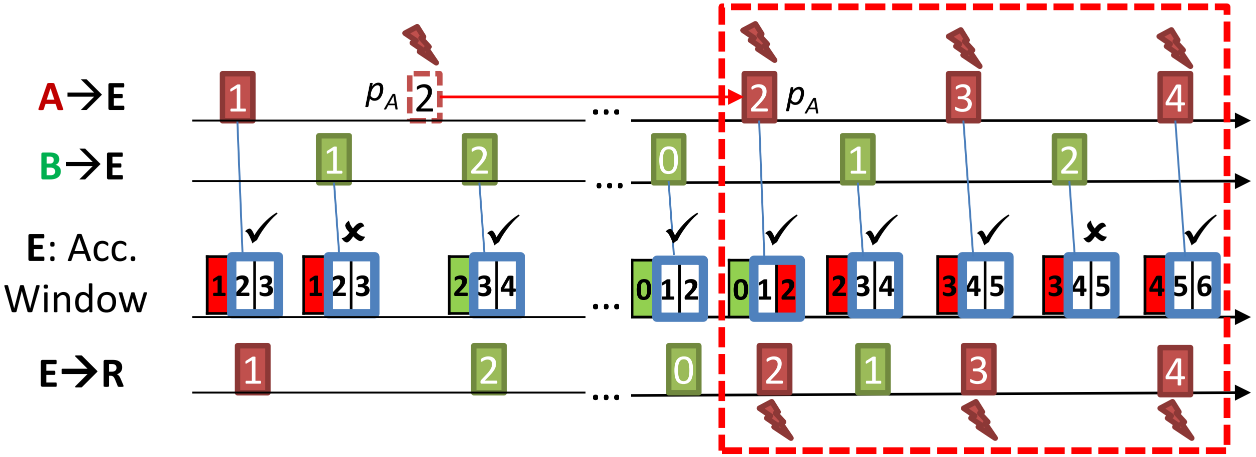
Description

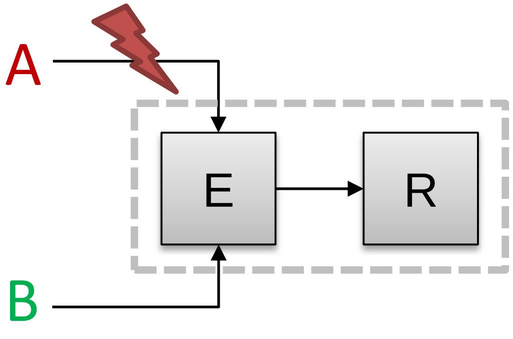
Message delay on path A delay delivery of a packet p_A (and consecutive packets) to E. Delay is ***within***

*n*seq. num. range + [0; seq. recovery history length]* *n=1,2,3,…*

- Failing components:
  - n*Bridge
  - Wires are explicitly excluded: Long delays require memory, wires don‘t have memory

Issue 6: Aligned Message Delays



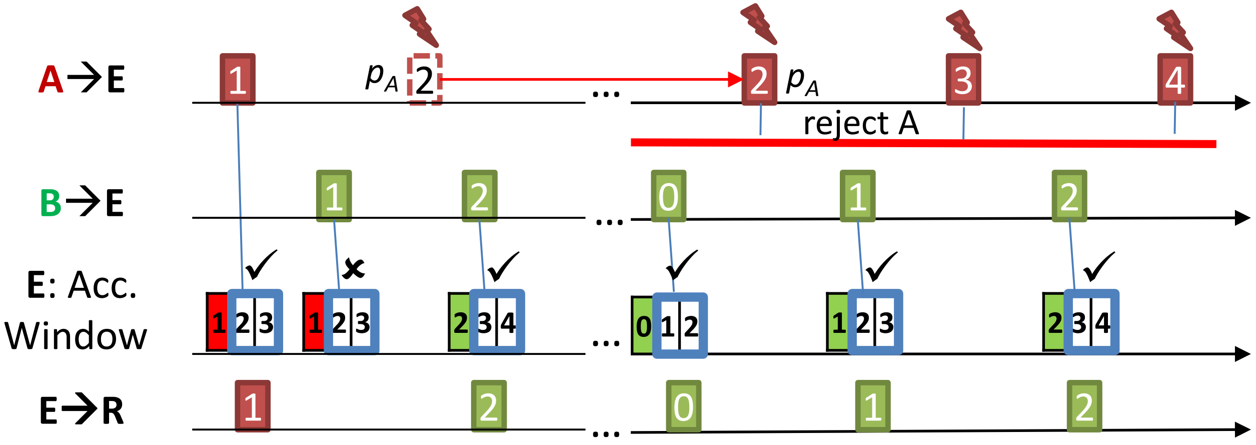


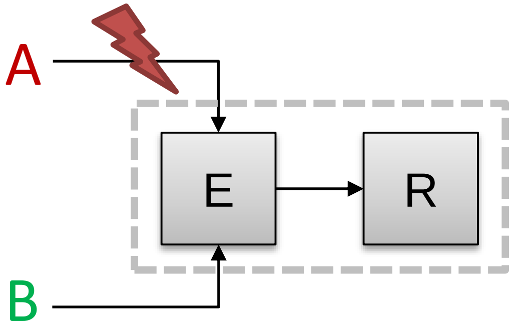
Legend
- S: Sender
- D: Duplication
- A_i, B_j: Bridge on Path A/B
- E: Duplicate elemination
- R: Receiver

Description

- *Message delay* on path A delay delivery of a packet p_A (and consecutive packets) to E. Delay is ***within***
  *n*seq. num. range + [0; seq. recovery history length]* *n=1,2,3,…*
- Consequence:
  - p_A is accepted by E if it arrives within the sequence history window, delayed packets from path A take over sequence number alignment of 802.1CB.
- Present countermeasures:
  - **None**

Issue 6: Possible Countermeasures





Legend

- S: Sender
- D: Duplication
- A_i, B_j: Bridge on Path A/B
- E: Duplicate elemination
- R: Receiver

Implementation option

- This failure is <u>not visible to listeners</u>:
  - No gaps greater than recovery history length visible in the received packet stream

  → This issue is better handled by bridges…
- Implement a relative timeout measured in sequence numbers (i.e. a counter based timeout) between both paths:
  - Once one path alone has progressed the sequence number beyond the recovery history length, discard all packets from the other path
  - Could be done by notifying 802.1Qcj functions once this relative timeout is exceeded
  - Seems feasible, but **<u>requires discussion</u>**

Summary

Addressed Failure modes

- Message corruption
- Unintended Message Repetition
- Message Delay

Proposed countermeasures to be discussed

Issue	Countermeasure	Level
Issue 1: Cut-Through and Corrupt Data	Don't do cut-through	Specification
	Rollback of CB State	Implementation
Issue 2: Cut-Through and Corrupt Sequence Numbers	Don't do cut-through	Specification
	Rollback of CB State	Implementation
Issue 4: (Any) Message repetition by bridges	Optionally disable recovery sequence number reset	Implementation
	Optional per path recovery sequence numbers	
Issue 5: Unaligned Message Delays	Optionally disable recovery sequence number reset	Implementation
	Optional per path recovery sequence numbers	
Issue 6: Aligned Message Delays	Relative Timeout between redundant paths, measured in units of sequence numbers	Implementation

Thank you for your Attention!

Questions, Opinions, Ideas?

Johannes Specht

Dipl.-Inform. (FH)

Dependability of Computing Systems
Institute for Computer Science and
Business Information Systems (ICB)
Faculty of Economics and
Business Administration
University of Duisburg-Essen

Schuetzenbahn 70
Room SH 502
45127 Essen
GERMANY
T +49 (0)201 183-3914
F +49 (0)201 183-4573

Johannes.Specht@uni-due.de
http://dc.uni-due.de