Notional 1FT Voting Architecture with Time-Triggered Ethernet

AES A&S Tag-Up Meeting
7 November 2016

Andrew Loveless (NASA JSC)
andrew.loveless@nasa.gov

Approved for Public Release –
No Export Controlled Data

General Overview

- 1-Byzantine resilient C&DH system (fail-operational).
  - Uses triplex onboard computers (OBCs) executing identical flight software.
  - >1FT relies on sparing and crew intervention (e.g. independent backup).
- Assumes classical reliability requirement of 10^{-9} failures/hour.
- Realizable with currently available COTS technology.*
  - E.g. Can be implemented using a variety of SBCs and real-time OSs.
- Scalable fault tolerance (both in classification and quantity).
  - E.g. Through additional network planes, high-integrity devices, etc.
- Assumes full cross strapping between OBCs, network switches, and end devices/subsystems (e.g. RIUs, IMUs, MBSUs).
  - Minimizes number of 2-fault combinations which can cause system failure.
  - Prioritizes high data availability and architectural flexibility over low SWaP.
- Redundant Time-Triggered Ethernet network used for data exchange and synchronization between computing platforms.
  - Eliminates need for independent Cross-Channel Data Link (CCDL).

* This presentation proposes the use of TTTech's rad-hard space ASIC (available Q3 2017).

Consensus Approach

Different Fault Classifications (there is overlap)

Less severe

More severe

Fault Type	Description	System
Fail-Stop	The node does not produce any output. • E.g. Process halts before "send to all".	Failover/Standby
Crash	The node does not produce any output. • Can remain undetected by good nodes.	N-Modular Redundancy (synchronized majority voting system)
Omission	Follows algorithm, but messages are lost.	
Value	Node produces incorrect computation result.	
Timing	Outputs are delivered too early or too late. • I.e. Node does not meet temporal specifications.	
Symmetric	Peers see the fault manifest in the same way. • E.g. Node send arbitrary data to all or nobody.	
Byzantine	Peers see the fault manifest in different ways. • E.g. Node sends different data to different peers.	Byzantine Agreement

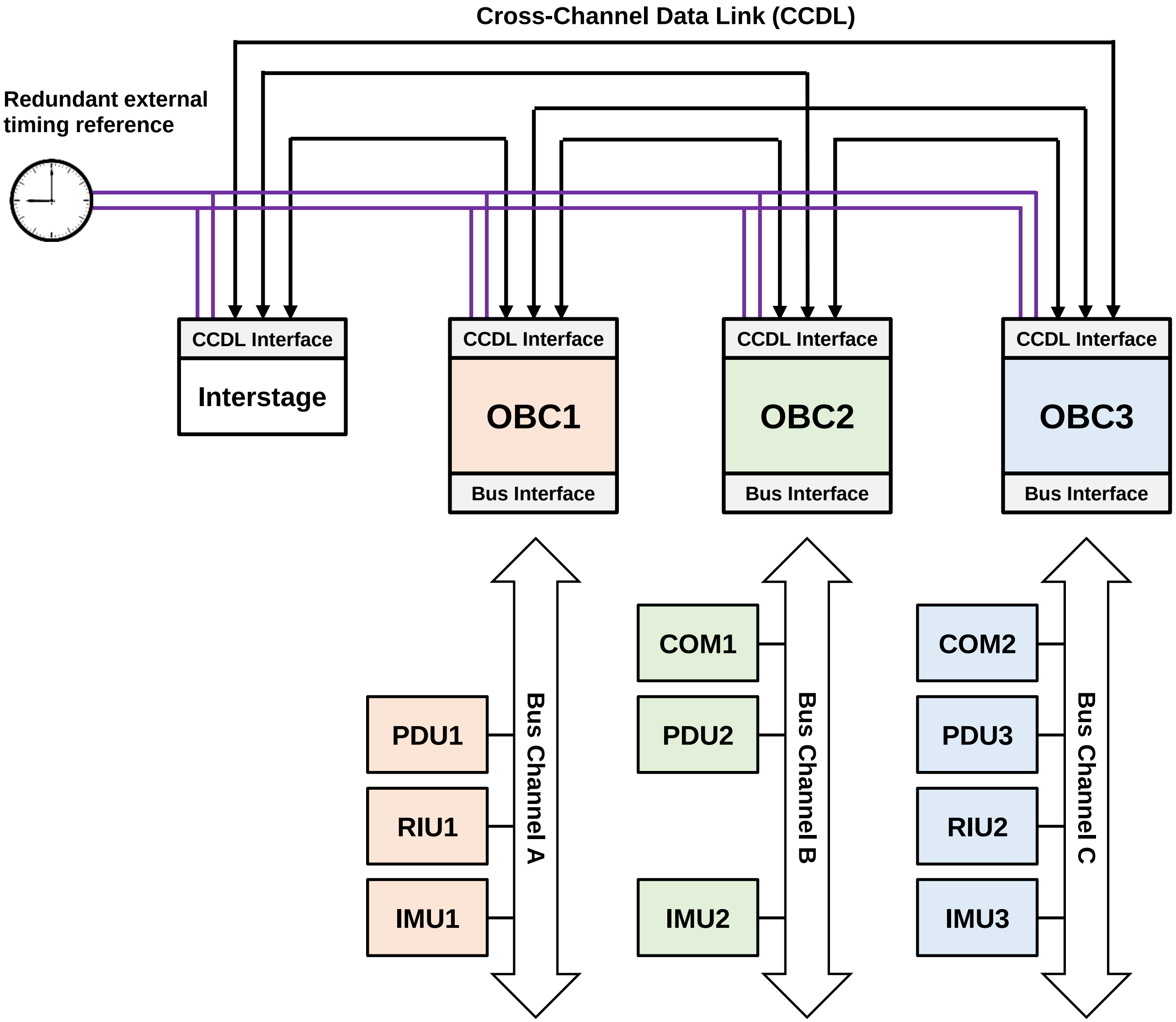
Ensuring Input Data Consistency

- Where does byzantine tolerance matter? Agreeing on input data
  - **Problem:** single source (internal or external) distribution to multiple receivers.
  - In our case, the input seen by each redundant processor must be bitwise identical – i.e. have *interactive consistency*.
  - **Why?** If all processors get the same input, then all non-faulty processors are guaranteed to produce identical output.
    - Can be used to ID faulty processors and resolve commands sent by the OBCs.
- Consensus versus Correctness
  - A faulty input device may provide arbitrary input data to the OBCs.
  - The purpose is to guarantee all OBCs have the same view of the system, and can therefore decide on the same input value.
    - I.e. the IC exchange guarantees consensus, but not that the input is “correct”.
  - If an accurate input value is important, you need redundant input devices.
- Avoiding hardware shortcuts
  - It is tempting to try circumventing the problem through increased connectivity.
    - E.g. Trying to ensure all OBCs read some input data from the same shared wire.
  - However, a faulty device may transmit a marginal signal that may be interpreted as different values by different OBCs.

Rules for Interactive Consistency

- What is an interstage?
  - An interstage is an FCR that participates in the interactive consistency exchange, but does not require consensus.
  - The purpose of an interstage is to provide the necessary functionality to perform byzantine agreement algorithms without requiring all FCRs to be full processors.
- Rules for interactive consistency in 1FT voting systems:
  - Requires ≥ 3(1) + 1 = 4 Fault Containment Regions (FCRs).
  - Each interstage must receive data through ≥1 disjoint paths.
  - Devices requiring consensus get data from ≥ 2(1) + 1 = 3 disjoint paths.
  - Above must be satisfied in (1) + 1 = 2 rounds of data exchange.
  - After data exchange, devices requiring consensus perform an absolute majority vote of received messages.

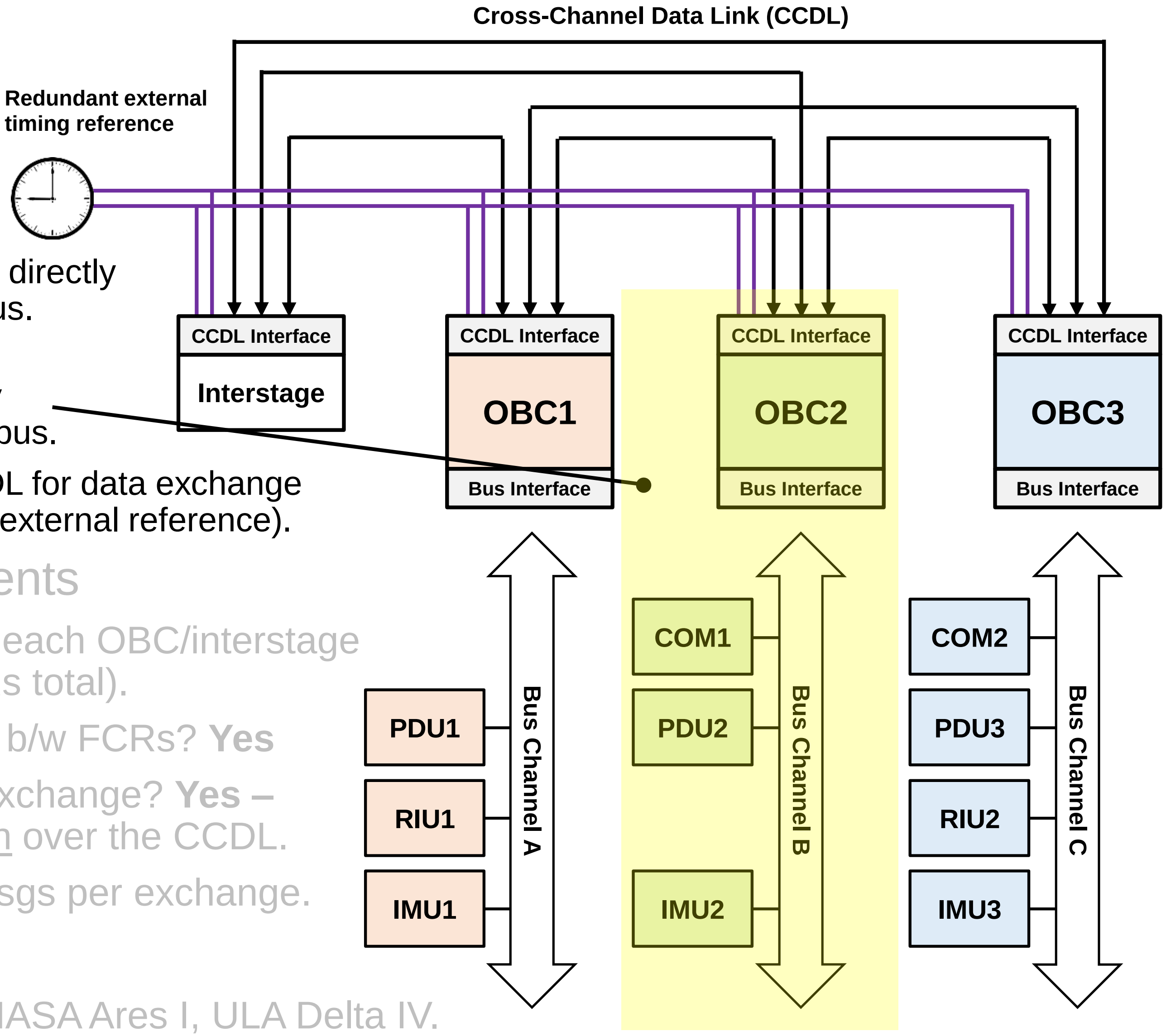
Classical Approach – Channelized Bus



Classical Approach – Channelized Bus

General Overview

- A 1FT design can be realized with either:
  1. 4 full processors/OBCs
  2. 3 OBCs + 1 interstage
- End devices are networked directly to one of the OBCs via a bus.
- Fully channelized design – Each OBC has access only to devices on its own local bus.
- Requires independent CCDL for data exchange and synchronization (or an external reference).



Meeting Requirements

- ≥ 3(1) + 1 FCRs? **Yes -** each OBC/interstage + its CCDL links (4 FCRs total).
- ≥ 2(1) + 1 disjoint paths b/w FCRs? **Yes**
- (1) + 1 rounds of data exchange? **Yes –** performed <u>in succession</u> over the CCDL.
- (4 - 1) + 4(4 - 1) = 15 msgs per exchange.

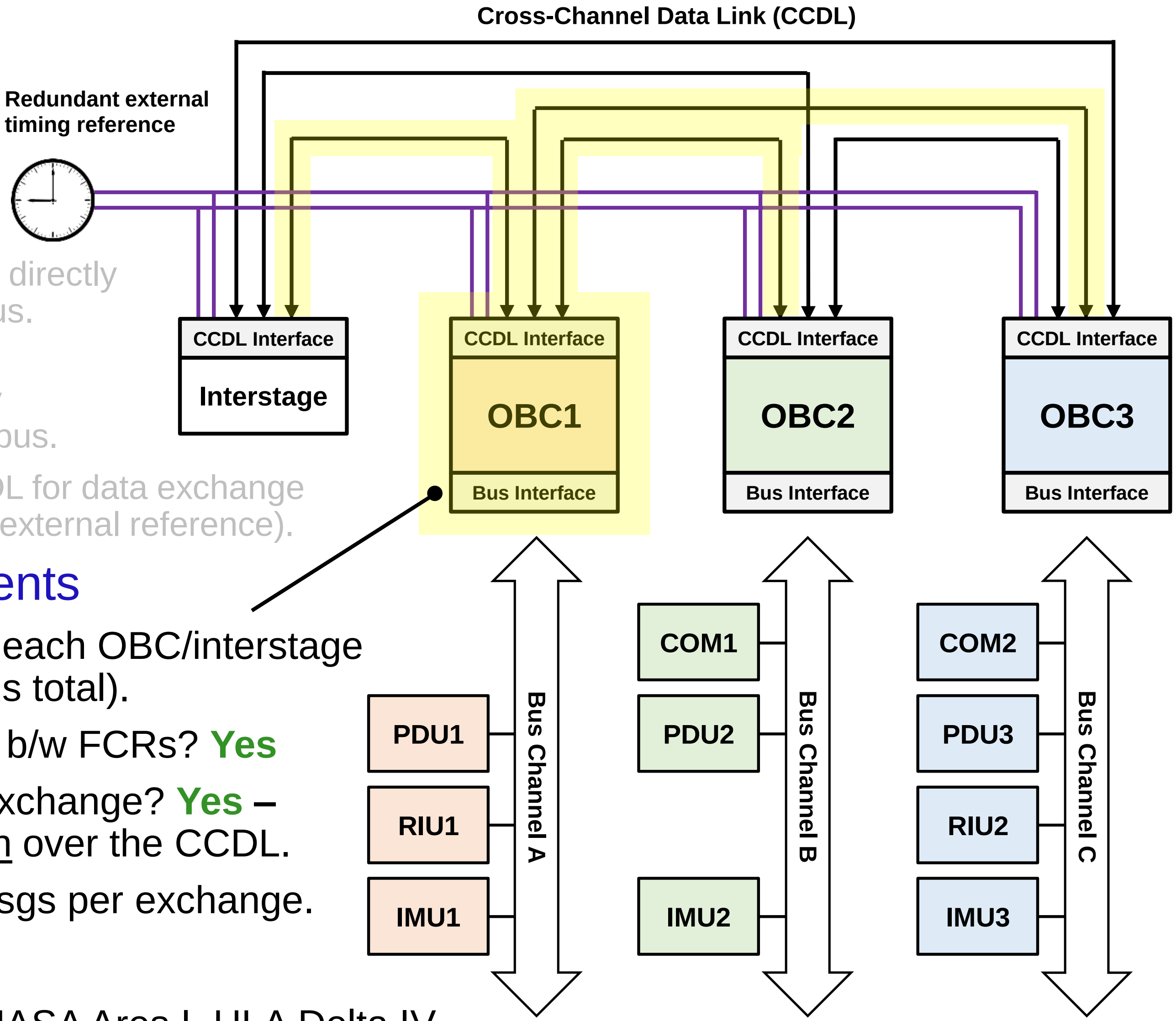
Examples

- NASA X-38, LM X-33, NASA Ares I, ULA Delta IV.

Classical Approach – Channelized Bus

General Overview

- A 1FT design can be realized with either:
  1. 4 full processors/OBCs
  2. 3 OBCs + 1 interstage
- End devices are networked directly to one of the OBCs via a bus.
- Fully channelized design – Each OBC has access only to devices on its own local bus.
- Requires independent CCDL for data exchange and synchronization (or an external reference).



Meeting Requirements

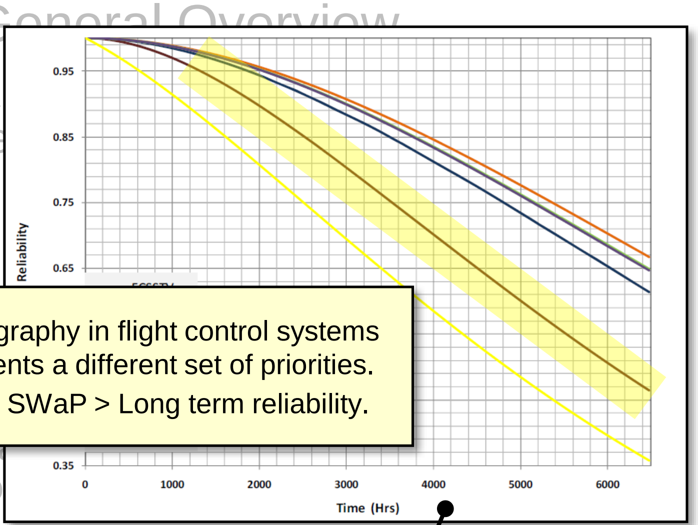
- ≥ 3(1) + 1 FCRs? **Yes** **-** each OBC/interstage + its CCDL links (4 FCRs total).
- ≥ 2(1) + 1 disjoint paths b/w FCRs? **Yes**
- (1) + 1 rounds of data exchange? **Yes** **–** performed <u>in succession</u> over the CCDL.
- (4 - 1) + 4(4 - 1) = 15 msgs per exchange.

Examples

- NASA X-38, LM X-33, NASA Ares I, ULA Delta IV.

Classical Approach – Channelized Bus

General Overview



Cryptography in flight control systems represents a different set of priorities.

- Low SWaP > Long term reliability.

- Requires independent CCDL for data exchange and synchronization (or an external reference).

Detect lying using authentication

- Many launcher applications relax the requirement for 4 FCRs by using the idea of “unforgeable” signed messages.
  - ➢ <u>Insufficient reliability for long mission durations</u>.

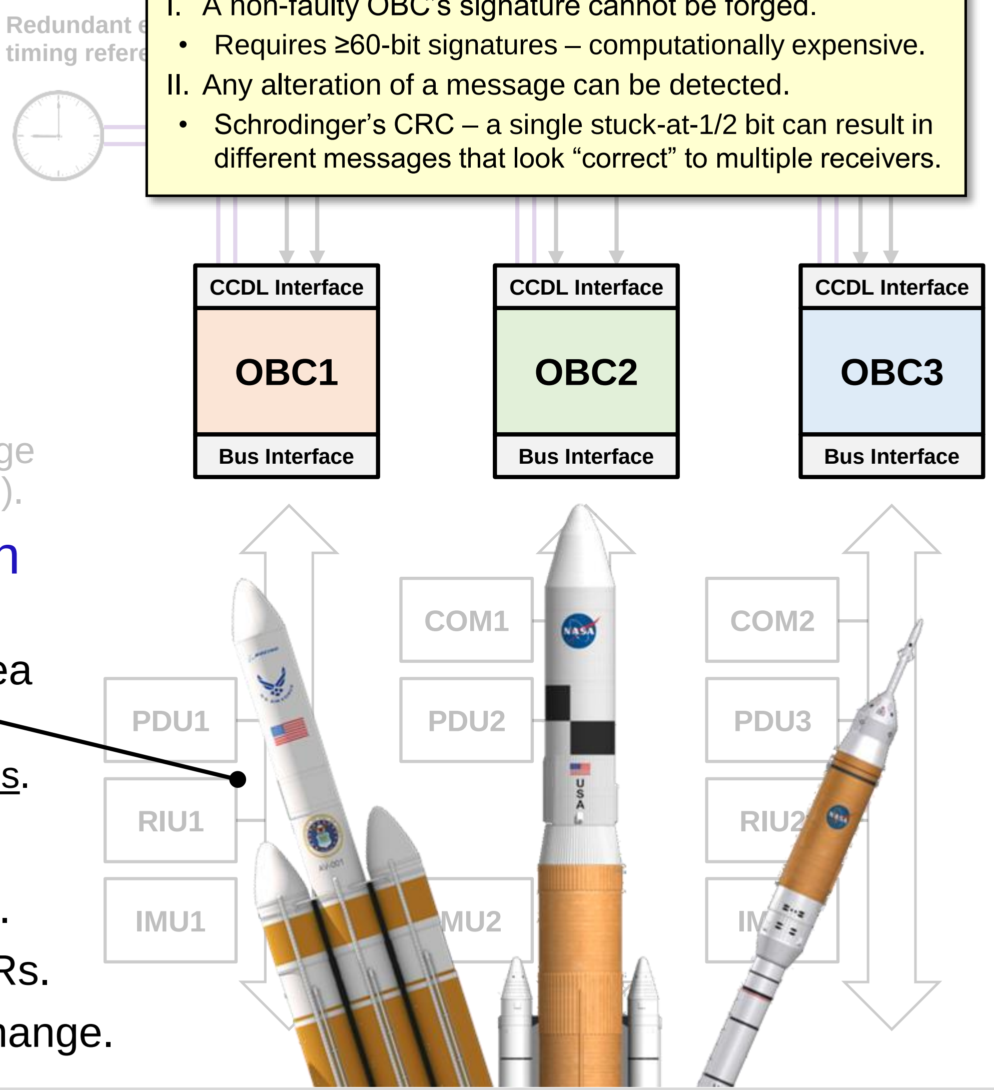
Relaxed Requirements

- ≥ 2(1) + 1 = 3 FCRs - each OBC + links.
- ≥ (1) + 1 = 2 disjoint paths between FCRs.
- (3 - 1) + 3(3 - 1) = 8 messages per exchange.

Certain BFT SM requirements are not fully realizable:

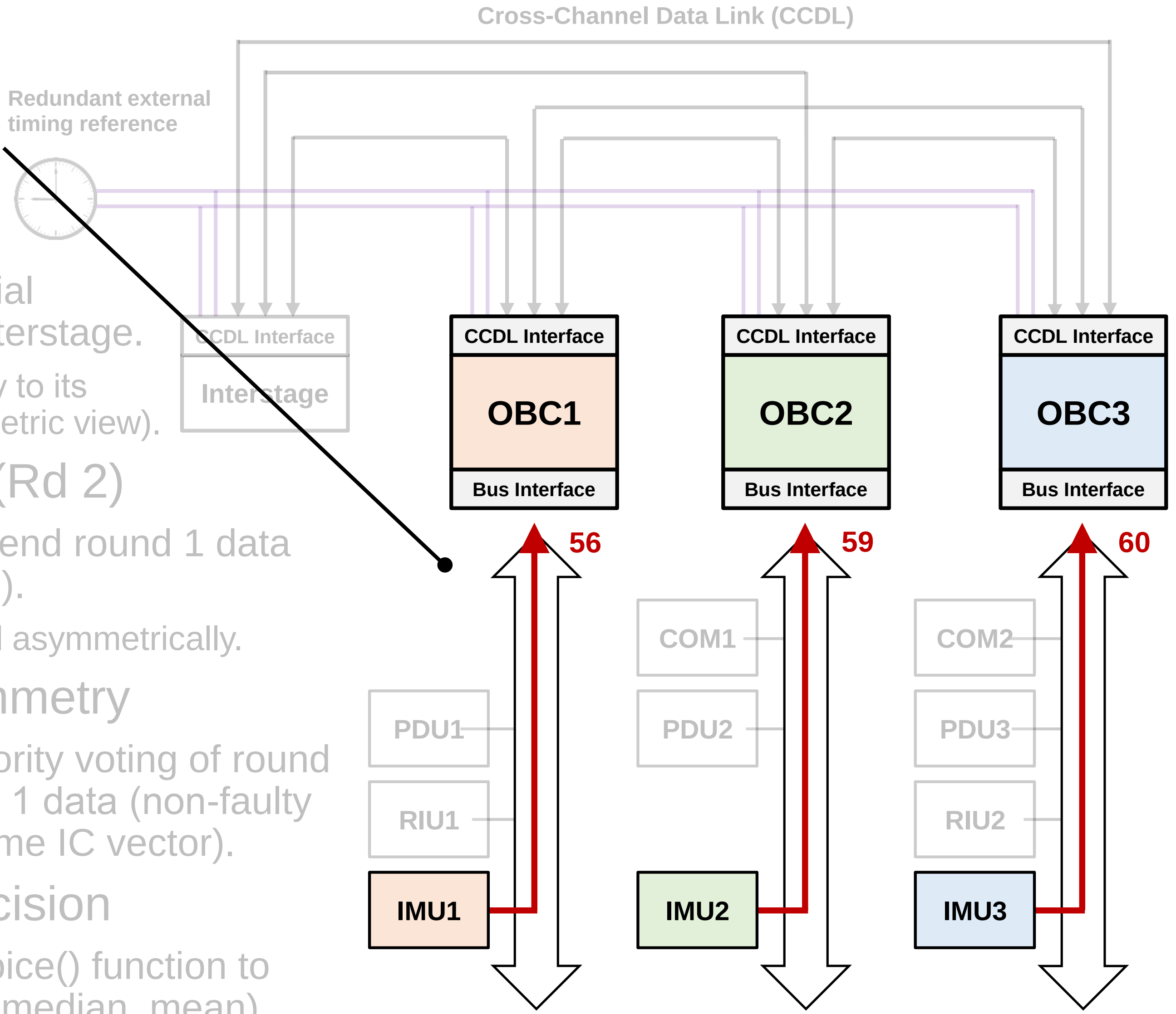
I. A non-faulty OBC’s signature cannot be forged.
- Requires ≥60-bit signatures – computationally expensive.

II. Any alteration of a message can be detected.
- Schrodinger’s CRC – a single stuck-at-1/2 bit can result in different messages that look “correct” to multiple receivers.



Channelized Bus – Reading Data (1)

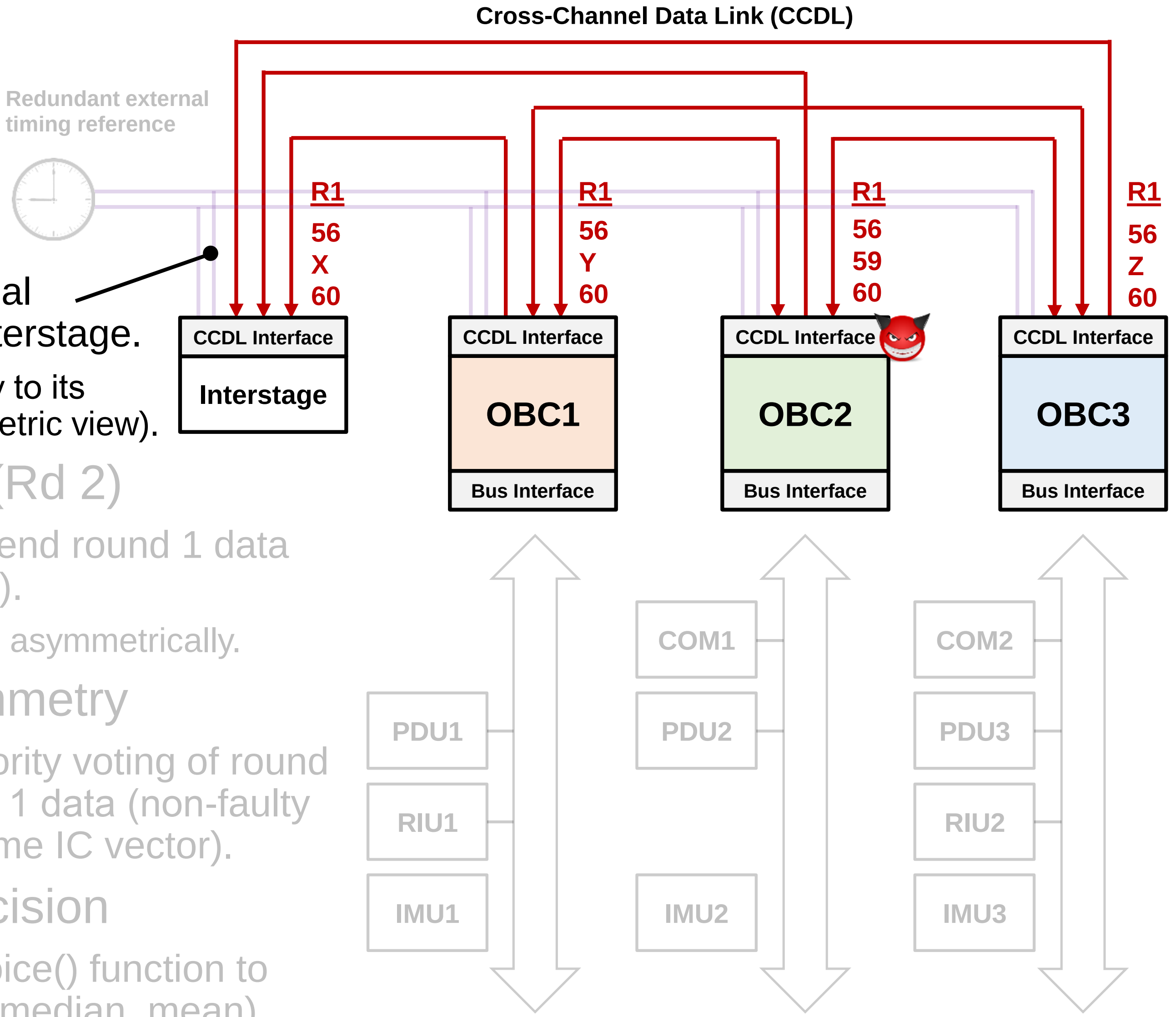
- Step 1: Read data
  - OBCs 1-3 reads data from local input device.
    - No guarantee data agrees.
- Step 2: Exchange
  - OBCs 1-3 send their initial values to OBCs 1-3 + interstage.
    - An OBC may "lie" arbitrarily to its peers (results in an asymmetric view).
- Step 3: Exchange (Rd 2)
  - OBCs 1-3 + interstage send round 1 data to all OBCs 1-3 (round 2).
    - Still, any FCR 1-4 could fail asymmetrically.
- Step 4: Create symmetry
  - OBCs 1-3 performs majority voting of round 2 data to "correct" round 1 data (non-faulty OBCs now share the same IC vector).
- Step 5: Make a decision
  - OBCs 1-3 execute a choice() function to select a final value (e.g. median, mean).



Approved for Public Release –
No Export Controlled Data

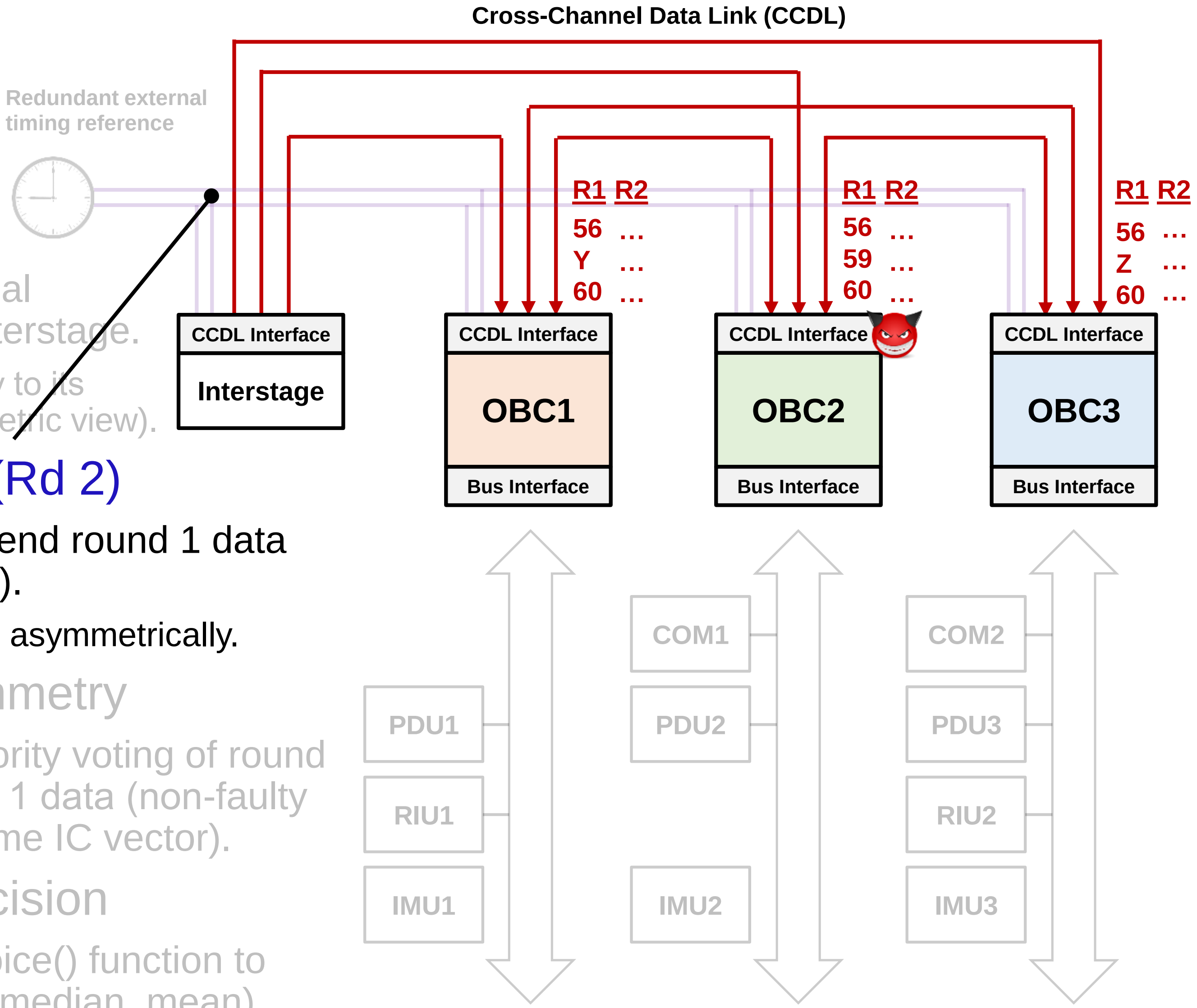
Channelized Bus – Reading Data (2)

- Step 1: Read data
  - OBCs 1-3 reads data from local input device.
    - No guarantee data agrees.
- Step 2: Exchange
  - OBCs 1-3 send their initial values to OBCs 1-3 + interstage.
    - An OBC may “lie” arbitrarily to its peers (results in an asymmetric view).
- Step 3: Exchange (Rd 2)
  - OBCs 1-3 + interstage send round 1 data to all OBCs 1-3 (round 2).
    - Still, any FCR 1-4 could fail asymmetrically.
- Step 4: Create symmetry
  - OBCs 1-3 performs majority voting of round 2 data to “correct” round 1 data (non-faulty OBCs now share the same IC vector).
- Step 5: Make a decision
  - OBCs 1-3 execute a choice() function to select a final value (e.g. median, mean).



Channelized Bus – Reading Data (3)

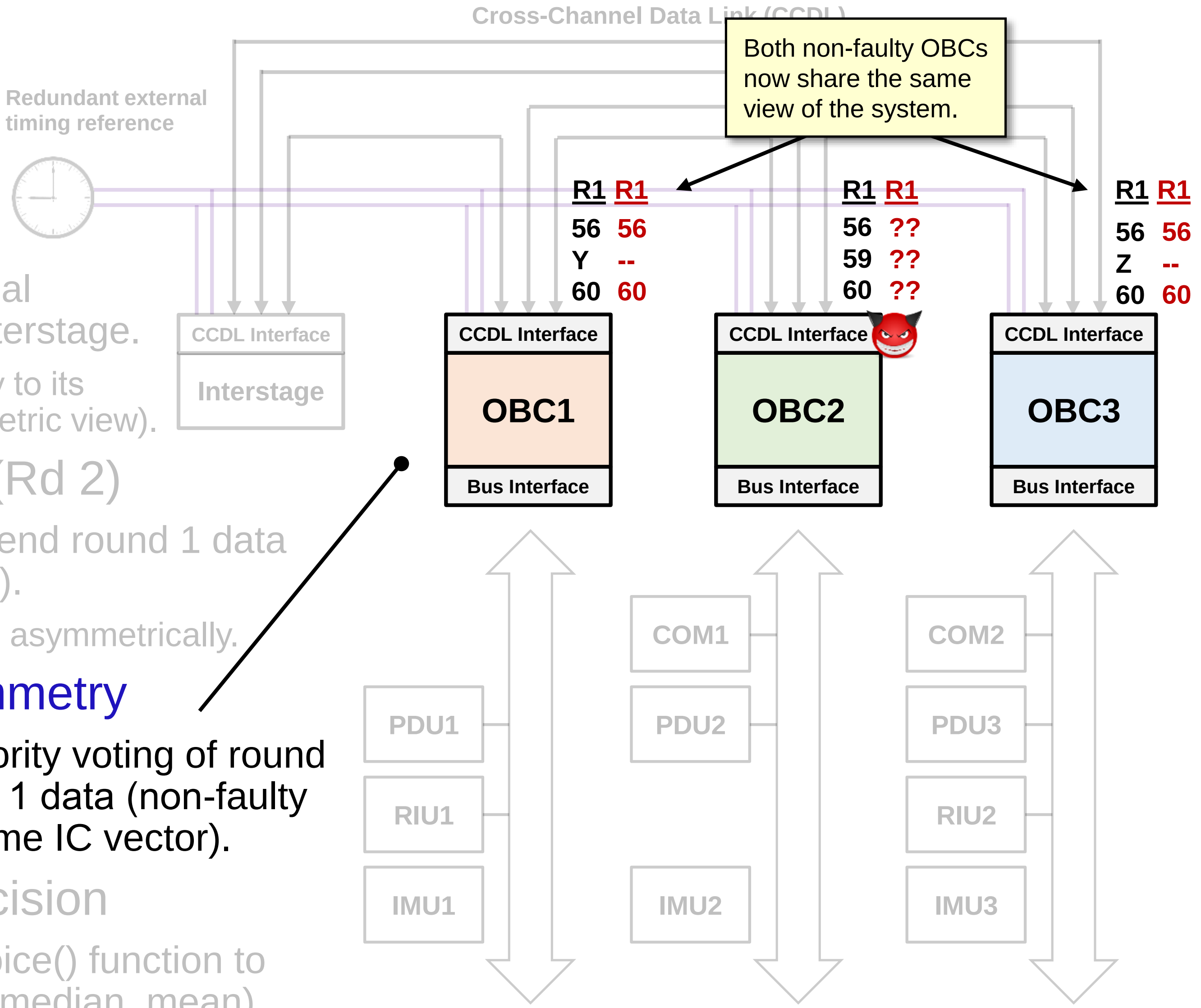
- Step 1: Read data
  - OBCs 1-3 reads data from local input device.
    - No guarantee data agrees.
- Step 2: Exchange
  - OBCs 1-3 send their initial values to OBCs 1-3 + interstage.
    - An OBC may “lie” arbitrarily to its peers (results in an asymmetric view).
- Step 3: Exchange (Rd 2)
  - OBCs 1-3 + interstage send round 1 data to all OBCs 1-3 (round 2).
    - Still, any FCR 1-4 could fail asymmetrically.
- Step 4: Create symmetry
  - OBCs 1-3 performs majority voting of round 2 data to “correct” round 1 data (non-faulty OBCs now share the same IC vector).
- Step 5: Make a decision
  - OBCs 1-3 execute a choice() function to select a final value (e.g. median, mean).



Approved for Public Release –
No Export Controlled Data

Channelized Bus – Reading Data (4)

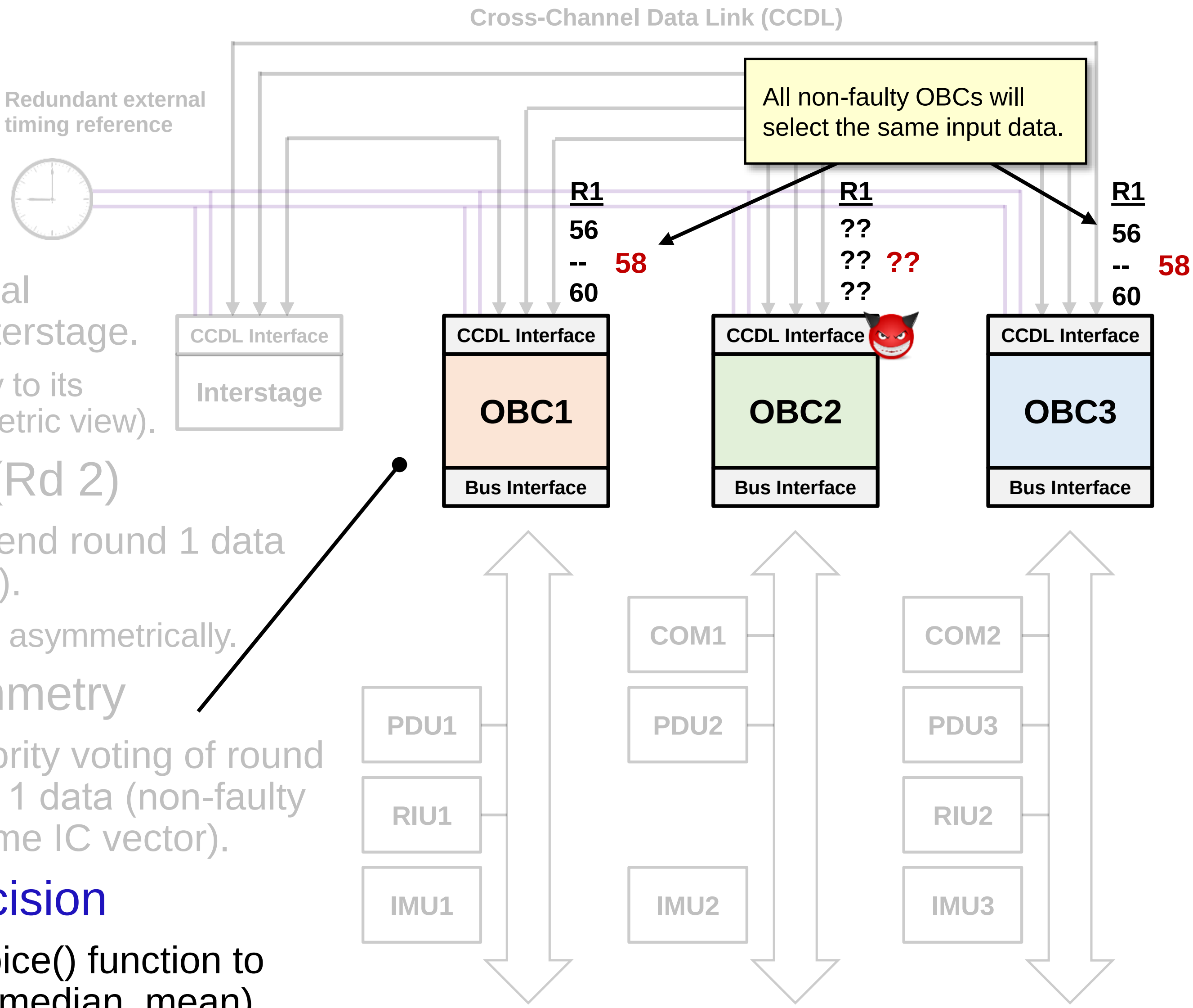
- **Step 1: Read data**
  - OBCs 1-3 reads data from local input device.
    - No guarantee data agrees.
- **Step 2: Exchange**
  - OBCs 1-3 send their initial values to OBCs 1-3 + interstage.
    - An OBC may "lie" arbitrarily to its peers (results in an asymmetric view).
- **Step 3: Exchange (Rd 2)**
  - OBCs 1-3 + interstage send round 1 data to all OBCs 1-3 (round 2).
    - Still, any FCR 1-4 could fail asymmetrically.
- **Step 4: Create symmetry**
  - OBCs 1-3 performs majority voting of round 2 data to "correct" round 1 data (non-faulty OBCs now share the same IC vector).
- **Step 5: Make a decision**
  - OBCs 1-3 execute a choice() function to select a final value (e.g. median, mean).



Both non-faulty OBCs now share the same view of the system.

Channelized Bus – Reading Data (5)

- Step 1: Read data
  - OBCs 1-3 reads data from local input device.
    - No guarantee data agrees.
- Step 2: Exchange
  - OBCs 1-3 send their initial values to OBCs 1-3 + interstage.
    - An OBC may “lie” arbitrarily to its peers (results in an asymmetric view).
- Step 3: Exchange (Rd 2)
  - OBCs 1-3 + interstage send round 1 data to all OBCs 1-3 (round 2).
    - Still, any FCR 1-4 could fail asymmetrically.
- Step 4: Create symmetry
  - OBCs 1-3 performs majority voting of round 2 data to “correct” round 1 data (non-faulty OBCs now share the same IC vector).
- Step 5: Make a decision
  - OBCs 1-3 execute a choice() function to select a final value (e.g. median, mean).



Channelized Bus – Commanding (1)

Step 1: Prepare Command

- After computation, OBCs 1-3 each generate a command.
  - All non-faulty OBCs agree.

Step 2: Exchange

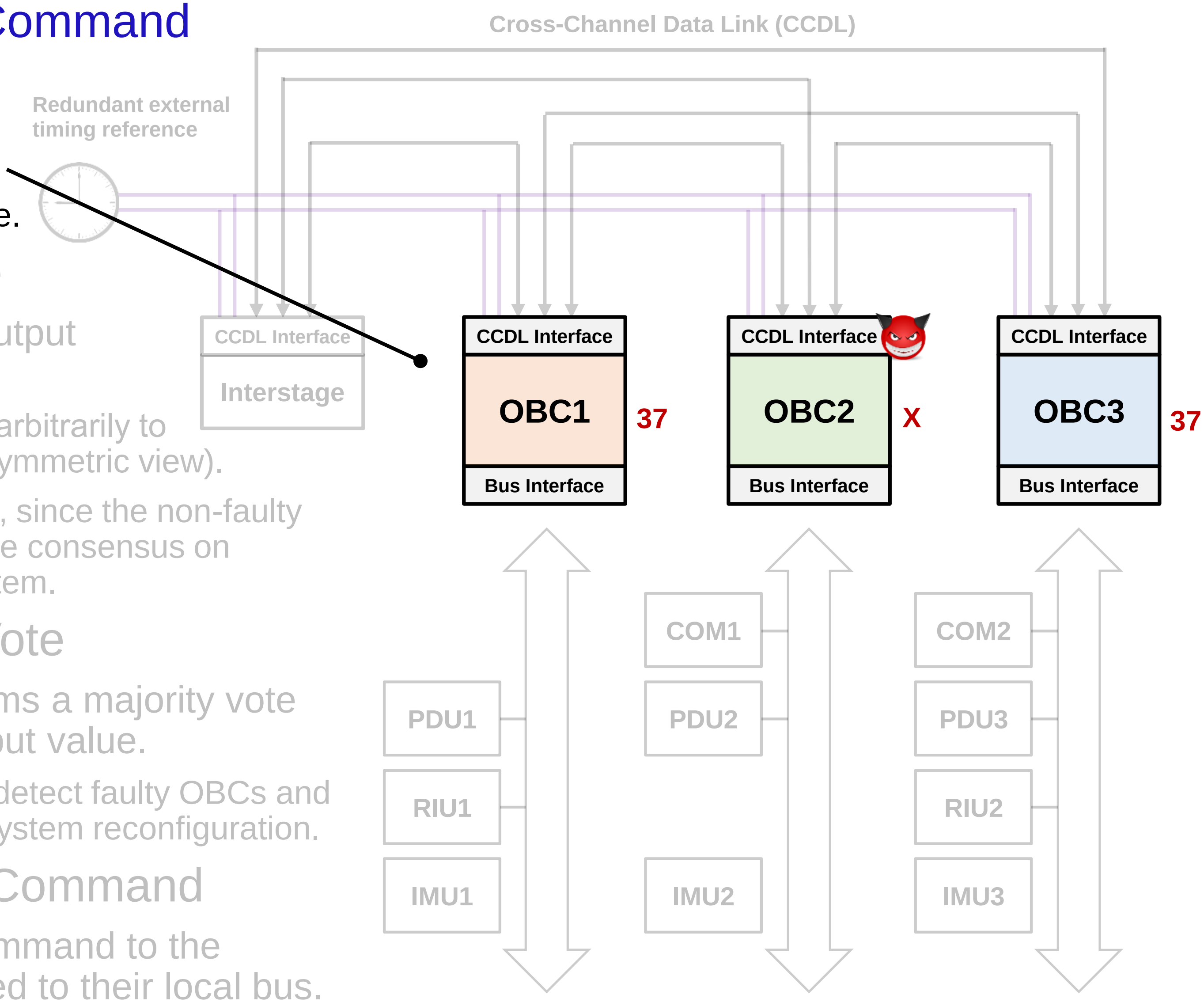
- OBCs 1-3 send their output values to OBCs 1-3.
  - Again, an OBC may “lie” arbitrarily to its peers (results in an asymmetric view).
  - This behavior is tolerated, since the non-faulty OBCs do not need to have consensus on the entire view of the system.

Step 3: Majority Vote

- Each OBCs 1-3 performs a majority vote to correct its initial output value.
  - .Process can be used to detect faulty OBCs and initiate fault recovery or system reconfiguration.

Step 4: Transmit Command

- OBCs 1-3 send the command to the output device connected to their local bus.



Channelized Bus – Commanding (2)

Step 1: Prepare Command

- After computation, OBCs 1-3 each generate a command.
  - All non-faulty OBCs agree.

Step 2: Exchange

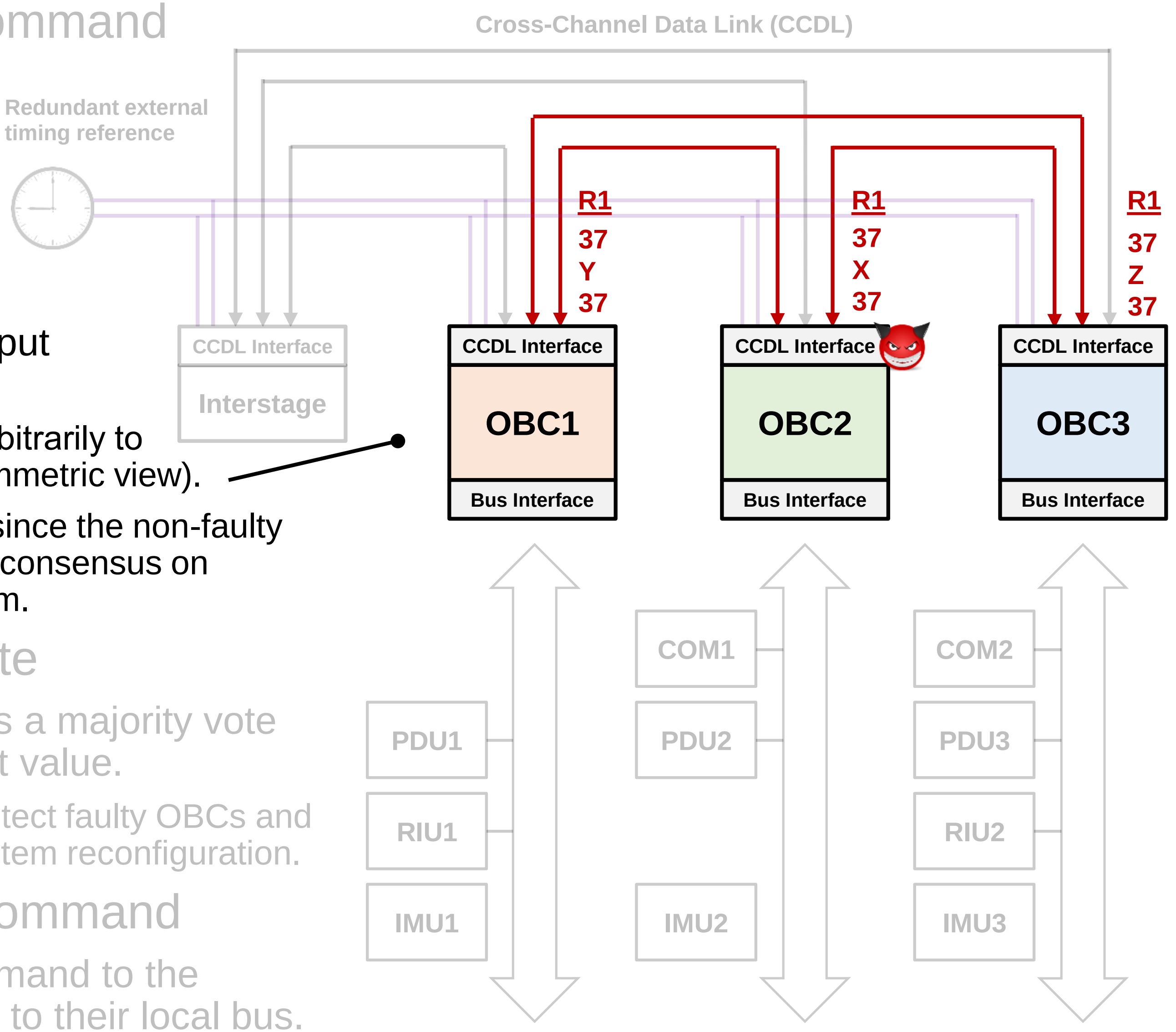
- OBCs 1-3 send their output values to OBCs 1-3.
  - Again, an OBC may "lie" arbitrarily to its peers (results in an asymmetric view).
  - This behavior is tolerated, since the non-faulty OBCs do not need to have consensus on the entire view of the system.

Step 3: Majority Vote

- Each OBCs 1-3 performs a majority vote to correct its initial output value.
  - .Process can be used to detect faulty OBCs and initiate fault recovery or system reconfiguration.

Step 4: Transmit Command

- OBCs 1-3 send the command to the output device connected to their local bus.



Channelized Bus – Commanding (3)

Step 1: Prepare Command

- After computation, OBCs 1-3 each generate a command.
  - All non-faulty OBCs agree.

Step 2: Exchange

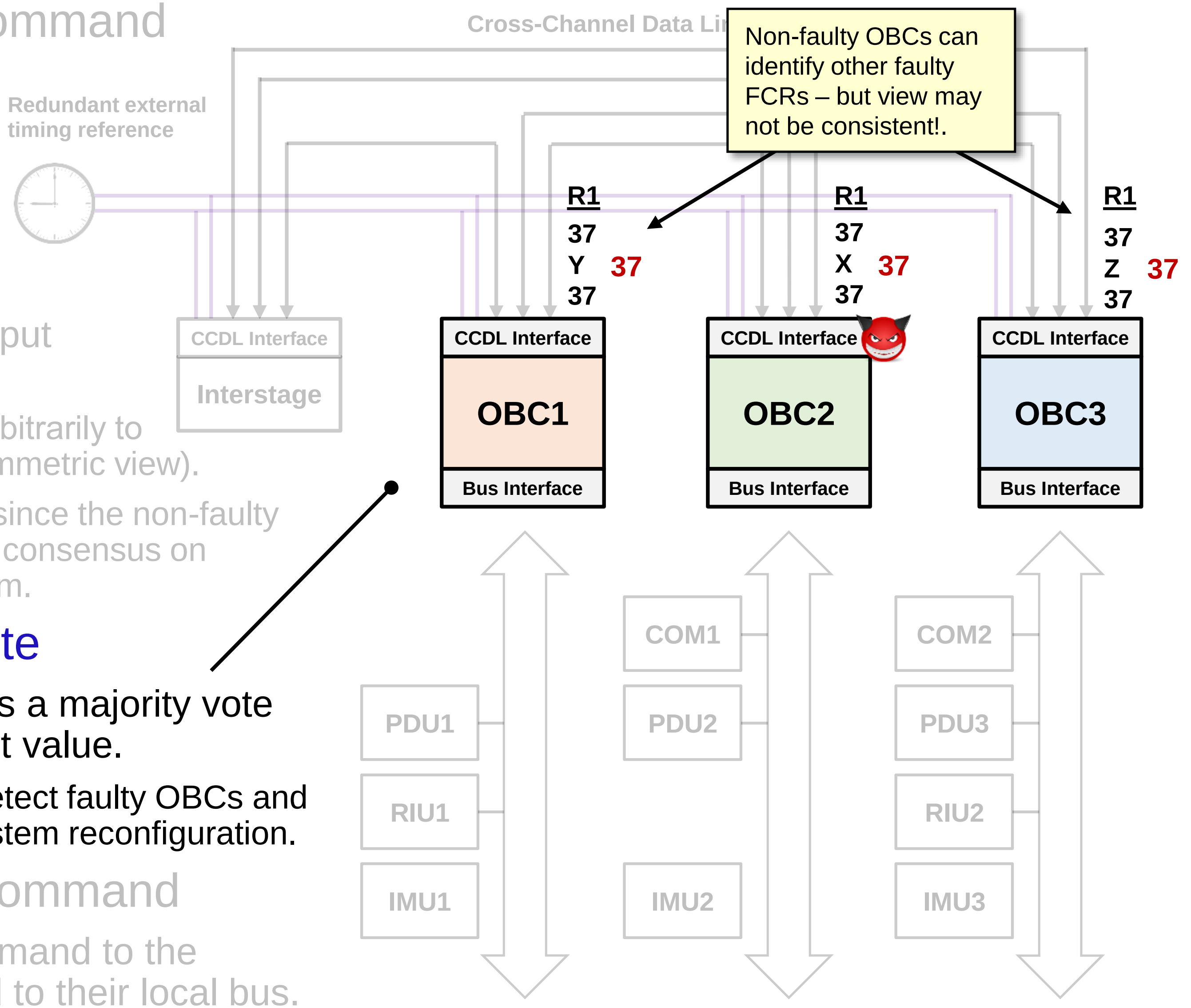
- OBCs 1-3 send their output values to OBCs 1-3.
  - Again, an OBC may "lie" arbitrarily to its peers (results in an asymmetric view).
  - This behavior is tolerated, since the non-faulty OBCs do not need to have consensus on the entire view of the system.

Step 3: Majority Vote

- Each OBCs 1-3 performs a majority vote to correct its initial output value.
  - .Process can be used to detect faulty OBCs and initiate fault recovery or system reconfiguration.

Step 4: Transmit Command

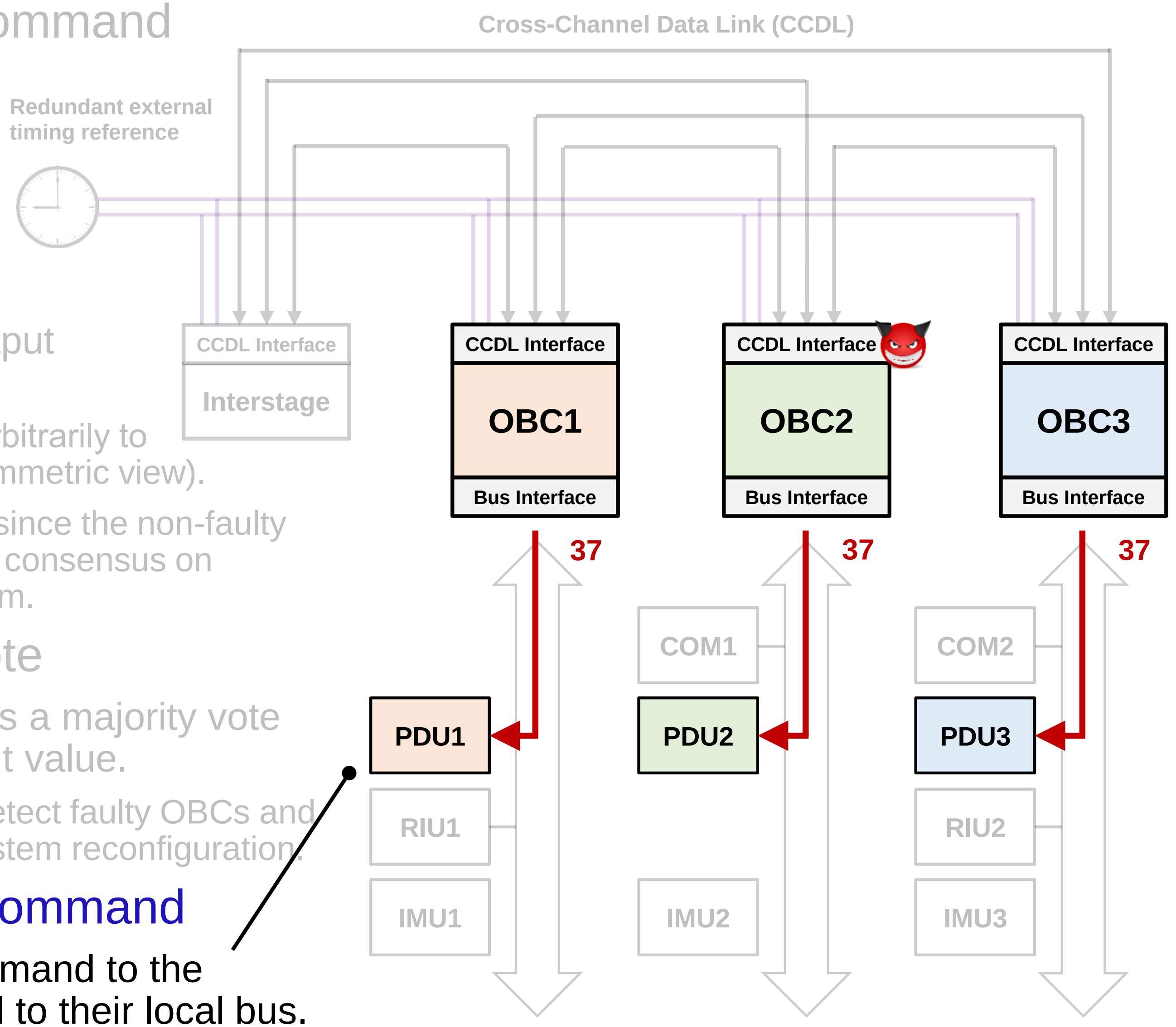
- OBCs 1-3 send the command to the output device connected to their local bus.



Non-faulty OBCs can identify other faulty FCRs – but view may not be consistent!.

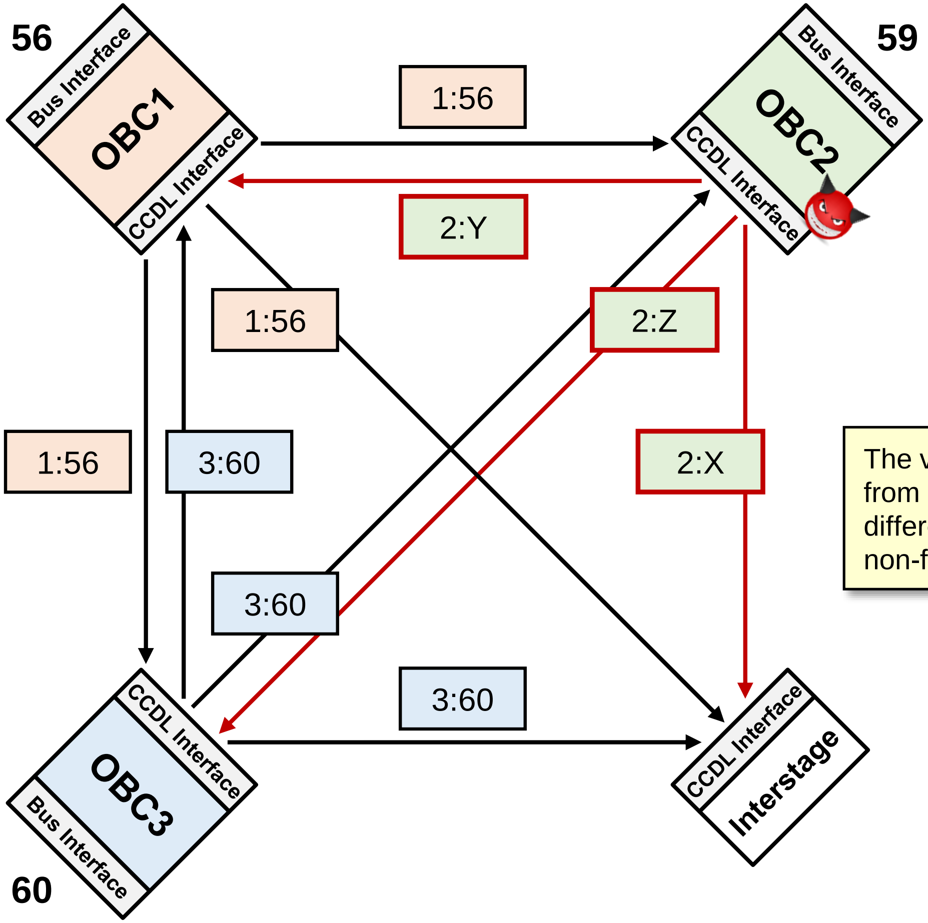
Channelized Bus – Commanding (4)

- **Step 1: Prepare Command**
  - After computation, OBCs 1-3 each generate a command.
    - All non-faulty OBCs agree.
- **Step 2: Exchange**
  - OBCs 1-3 send their output values to OBCs 1-3.
    - Again, an OBC may “lie” arbitrarily to its peers (results in an asymmetric view).
    - This behavior is tolerated, since the non-faulty OBCs do not need to have consensus on the entire view of the system.
- **Step 3: Majority Vote**
  - Each OBCs 1-3 performs a majority vote to correct its initial output value.
    - .Process can be used to detect faulty OBCs and initiate fault recovery or system reconfiguration.
- **Step 4: Transmit Command**
  - OBCs 1-3 send the command to the output device connected to their local bus.



Channelized Bus – Detailed Exchange

Information Exchange – Round 1



The value received from OBC2 is different for each non-faulty FCR.

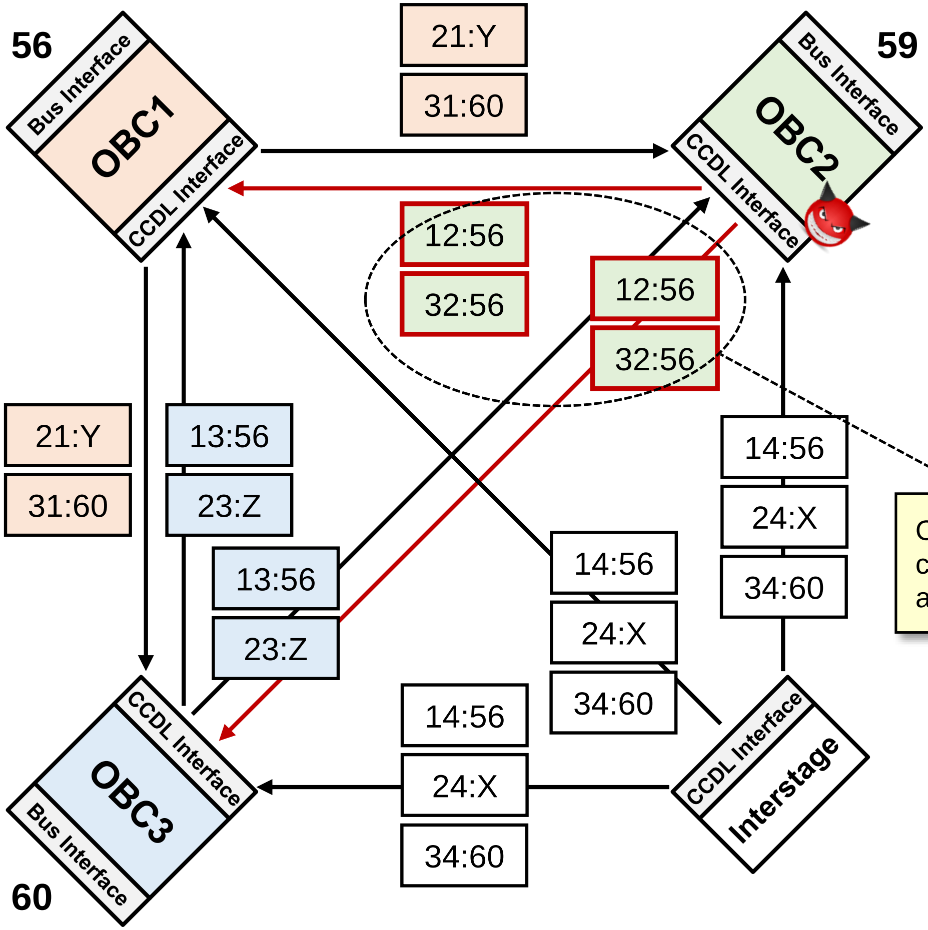
	1	2	3	
IC Vector:	56	Y	60	
OBC1	▨			1
		▨		2
			▨	3
				4

	1	2	3	
IC Vector:	56	Z	60	
OBC3	▨			1
		▨		2
			▨	3
				4

	1	2	3	
IC Vector:	56	X	60	
Interstage	▨	▨	▨	1
	▨	▨	▨	2
	▨	▨	▨	3
	▨	▨	▨	4

Channelized Bus – Detailed Exchange

Information Exchange – Round 2



OBC2 tries to force consensus by agreeing with OBC1.

	1	2	3	
IC Vector:	56	Y	60	
	▨	Y	60	1
OBC1	56	▨	56	2
	56	Z	▨	3
	56	X	60	4

	1	2	3	
IC Vector:	56	Z	60	
	▨	Y	60	1
OBC3	56	▨	56	2
	56	Z	▨	3
	56	X	60	4

	1	2	3	
IC Vector:	56	X	60	
	▨	▨	▨	1
Interstage	▨	▨	▨	2
	▨	▨	▨	3
	▨	▨	▨	4

Channelized Bus – Detailed Exchange

Create Symmetry - Majority Voting

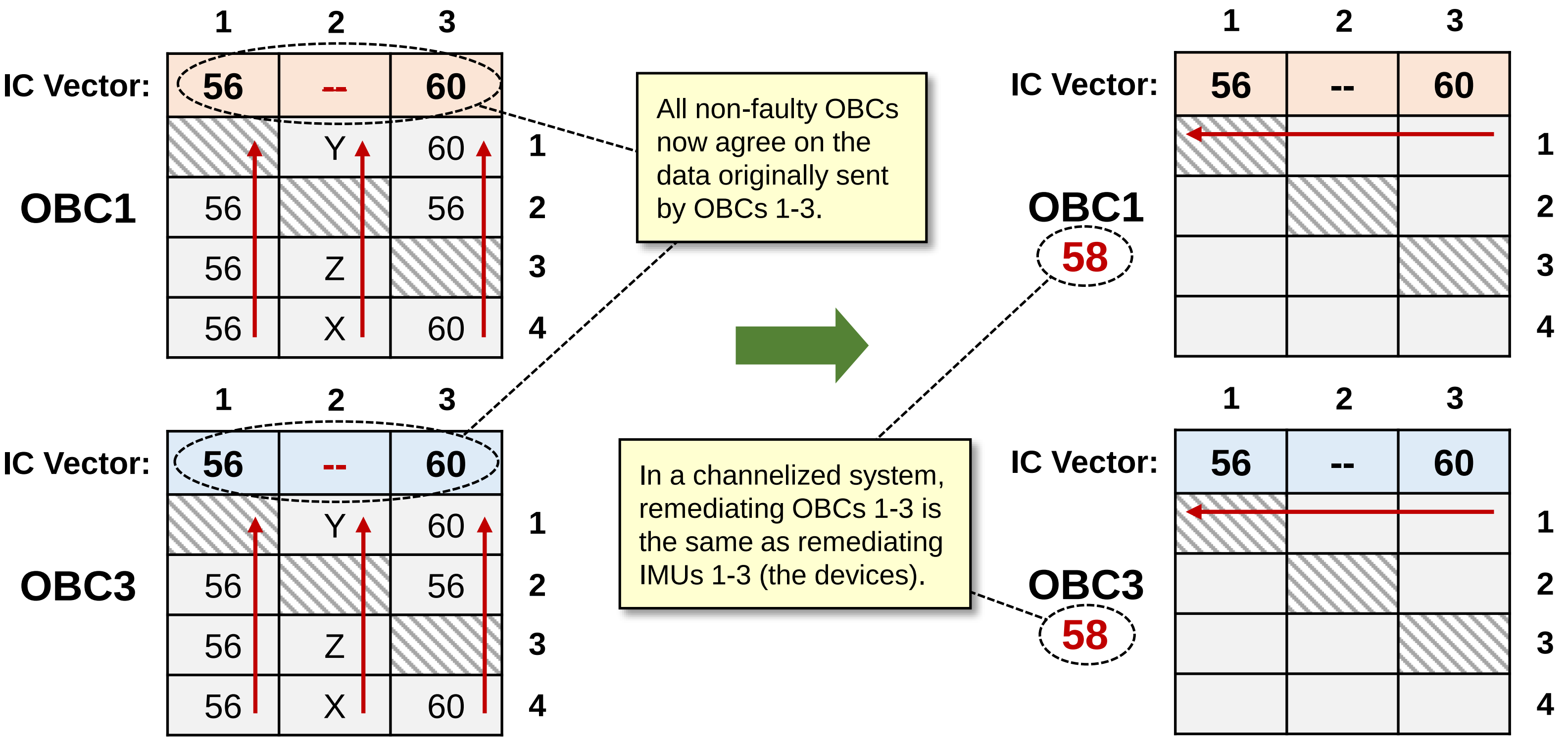
On OBCs 1-3, each element in the interactive consistency (IC) vector is set to the strict majority of its children.

→ I.e. OBCs 1,3 must agree on data from OBC 2.

Making a Decision

On OBCs 1-3, a choice() function is used to determine a final value from those contained in the IC vector.

→ E.g. a mid-value selection.



OBC1

	1	2	3	
IC Vector:	56	--	60	
	▨	Y	60	1
	56	▨	56	2
	56	Z	▨	3
	56	X	60	4

OBC3

	1	2	3	
IC Vector:	56	--	60	
	▨	Y	60	1
	56	▨	56	2
	56	Z	▨	3
	56	X	60	4

All non-faulty OBCs now agree on the data originally sent by OBCs 1-3.

In a channelized system, remediating OBCs 1-3 is the same as remediating IMUs 1-3 (the devices).

OBC1 58

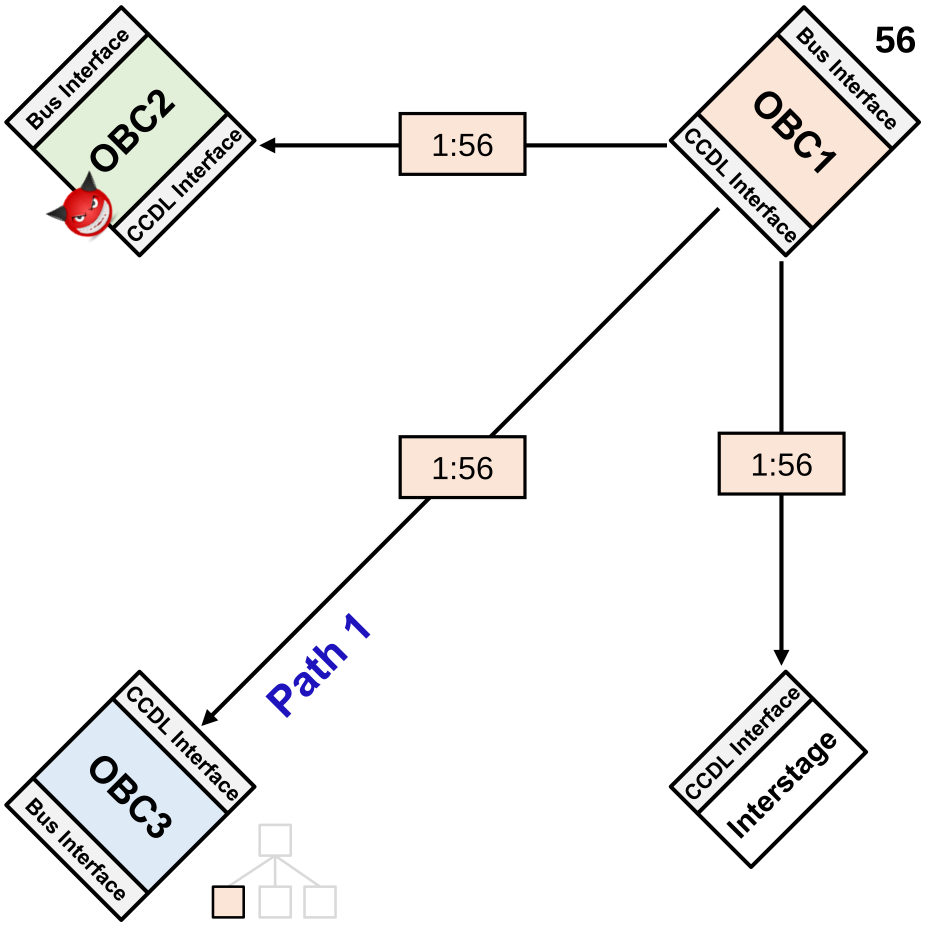
	1	2	3	
IC Vector:	56	--	60	
	▨			1
		▨		2
			▨	3
				4

OBC3 58

	1	2	3	
IC Vector:	56	--	60	
	▨			1
		▨		2
			▨	3
				4

IC Exchange – Alternate Viewpoint (1)

- **“Flattening” the classical two-round exchange**
  - Can be analyzed as messaging over redundant paths (from different FCRs).
  - Makes it easier to see why 4 FCRs and 3 disjoint paths are necessary.



OBC1

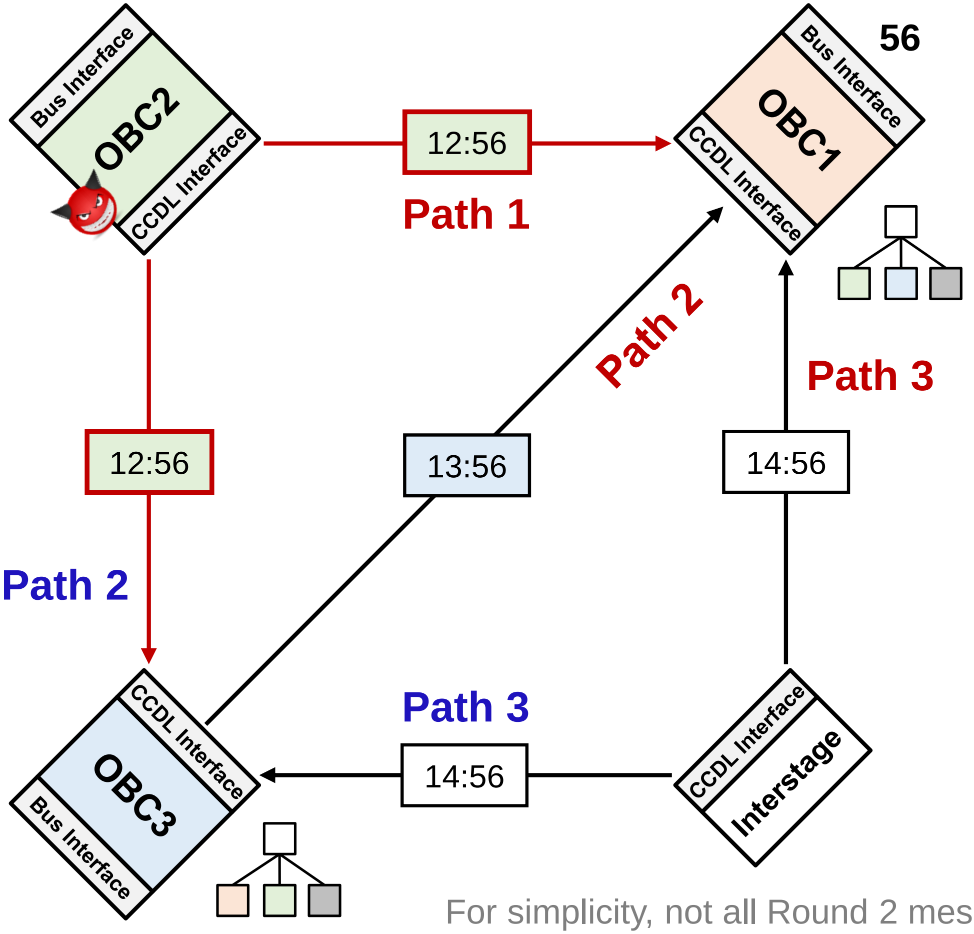
	1	2	3
IC Vector:			
1	▨	▨	▨
2			
3			
4			

OBC3

	1	2	3
IC Vector:			
1	56		
2			
3	▨	▨	▨
4			

IC Exchange – Alternate Viewpoint (2)

- “Flattening” the classical two-round exchange
  - Can be analyzed as messaging over redundant paths (from different FCRs).
  - Makes it easier to see why 4 FCRs and 3 disjoint paths are necessary.



For simplicity, not all Round 2 messages are shown.

	1	2	3	
IC Vector:	**56**			
OBC1	▨	▨	▨	1
	56			2
	56			3
	56			4

	1	2	3	
IC Vector:	**56**			
OBC3	56			1
	56			2
	▨	▨	▨	3
	56			4

Generalizing use of interstages (1)

Example 1:

- Four total FCRs
- Two interstages
- Two devices require consensus

Rules for IC in 1FT voting systems:

- Requires ≥ 3(1) + 1 = 4 FCRs.
- Interstages need data from ≥1 paths.
- Devices requiring consensus need data from ≥ 2(1) + 1 = 3 disjoint paths.
- Two rounds of data exchange.
- Devices requiring consensus perform majority vote over received messages.

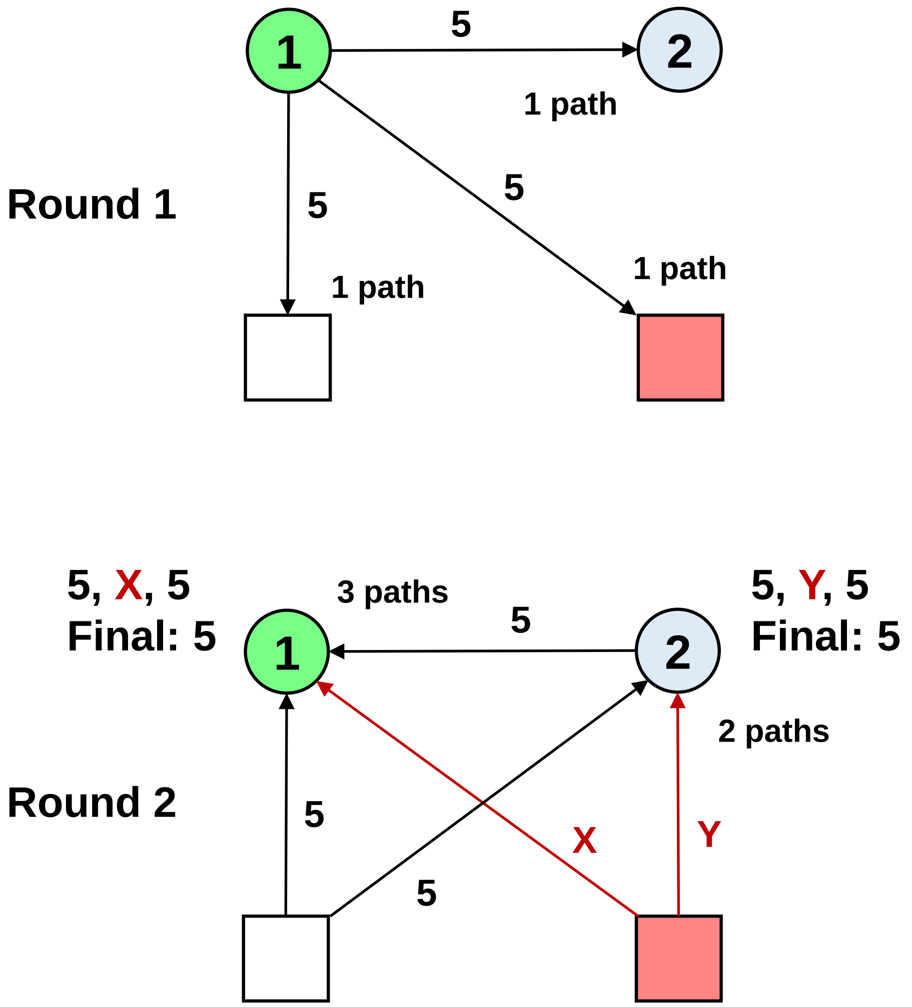
○ Device requiring consensus

☐ Interstage (does not require consensus)

🟩 Designates originating device

🟥 Designates faulty device

Assumption: Any device may fail arbitrarily (omission, symmetric, asymmetric, byzantine).



Generalizing use of interstages (2)

Example 2:

- Five total FCRs
- Three interstages
- Two devices require consensus

Rules for IC in 1FT voting systems:

- Requires ≥ 3(1) + 1 = 4 FCRs.
- Interstages need data from ≥1 paths.
- Devices requiring consensus need data from ≥ 2(1) + 1 = 3 disjoint paths.
- Two rounds of data exchange.
- Devices requiring consensus perform majority vote over received messages.

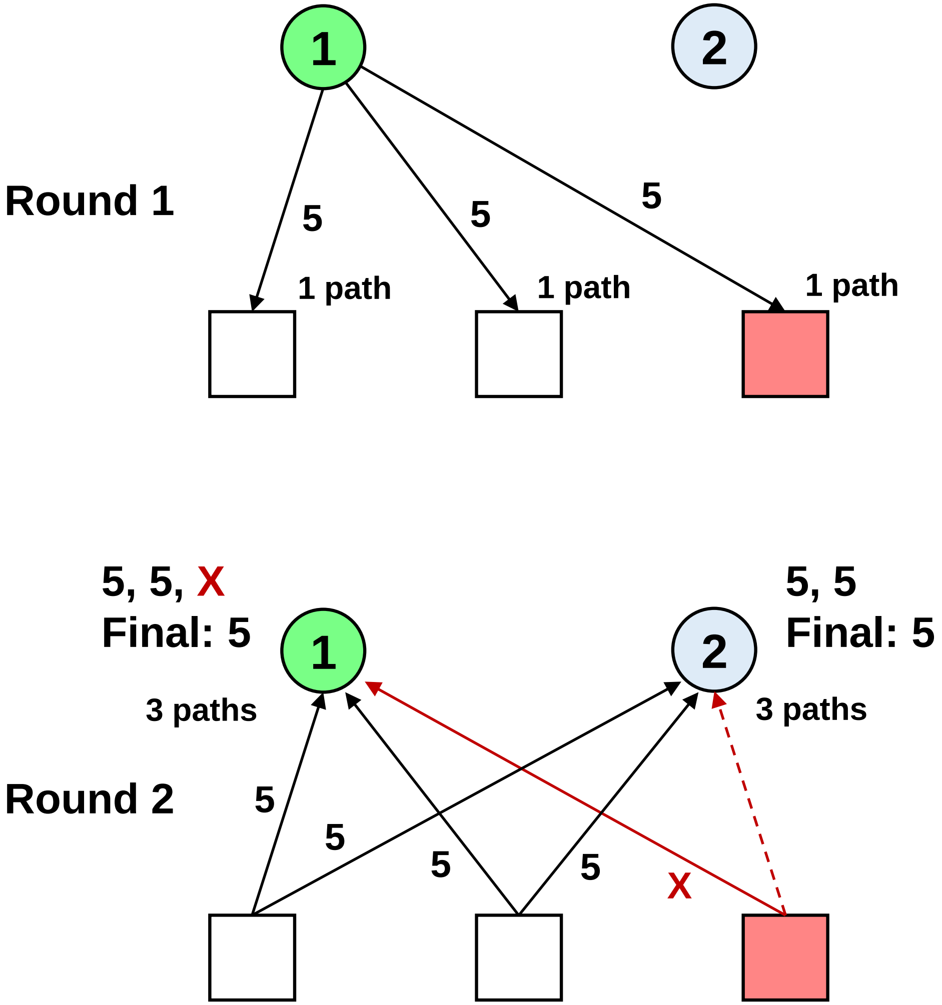
○ Device requiring consensus

☐ Interstage (does not require consensus)

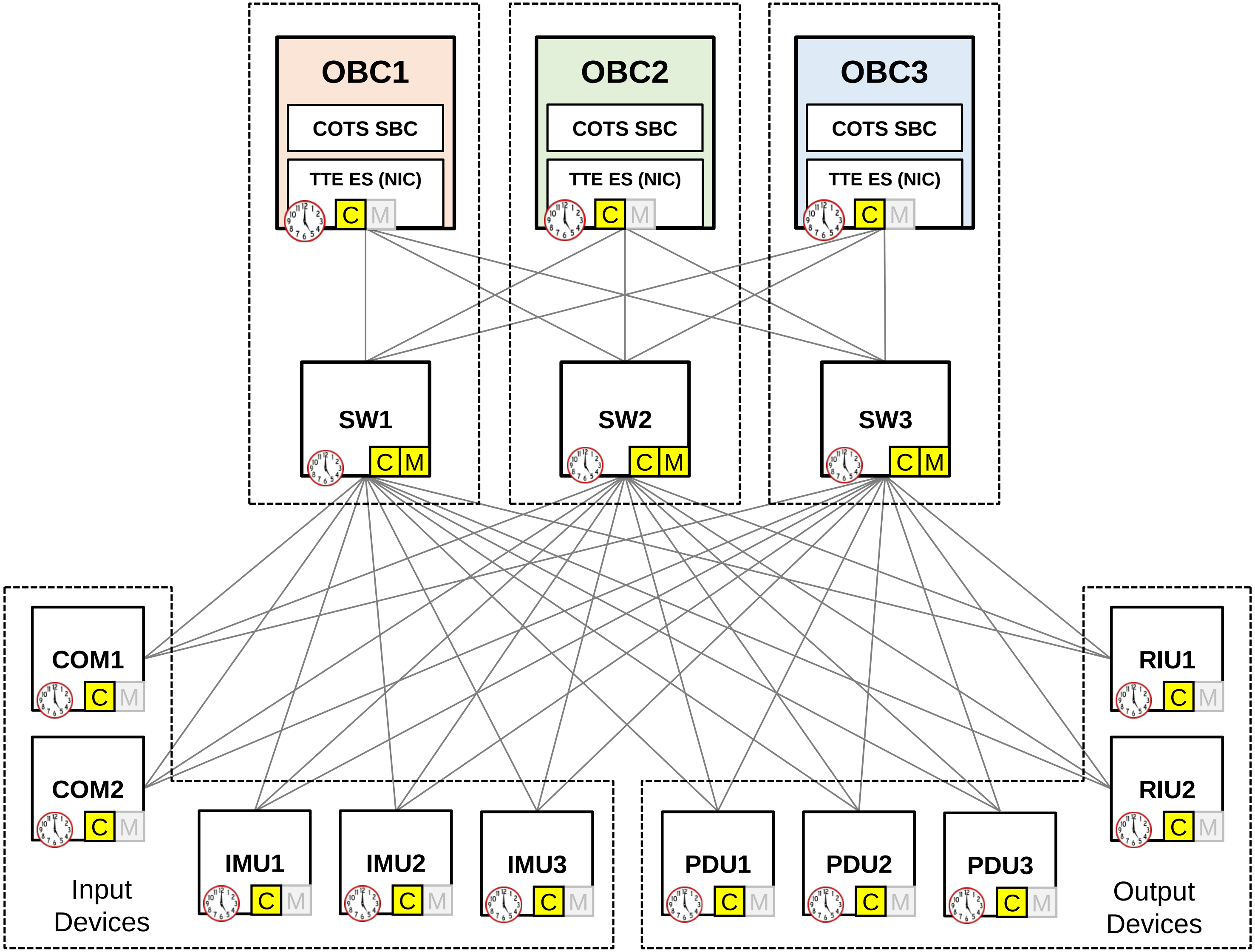
🟩 Designates originating device

🟥 Designates faulty device

Assumption: Any device <u>may fail arbitrarily</u> (omission, symmetric, asymmetric, byzantine).



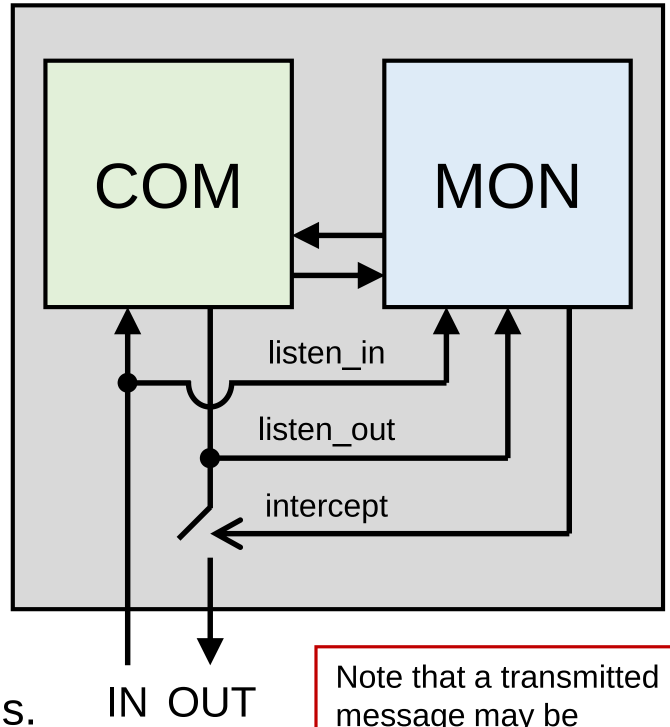
Switched Triplex (Fully Cross-strapped)



High-Integrity Devices in TTEthernet

High-Integrity Design

- Command/Monitor (COM/MON) design aims for error containment within the device.
  - Contains two fault containment regions.
- Input is forwarded to both COM and MON.
- Congruency exchange ensures both COM and MON have identical input data (i.e. IC).
- Both COM and MON process data in parallel.
- Output from COM is forwarded to MON.
- If disagreement, MON terminates the transmission.



Note that a transmitted message <u>may be truncated</u> – the receiver rejects the message.

Device Failure Assumptions

- Standard devices may be subject to *byzantine* failures.
  - Device may send arbitrary messages (of any contents).
  - Device may transmit messages at arbitrary points in time.
  - Device may send different messages through different network planes (channels).
- High-Integrity devices may be subject to *inconsistent omission* failures.
  - Faulty device will not create (nor modify existing to produce) a new valid message.
  - Device may drop or fail to receive an arbitrary number of messages.
  - Device may fail to relay messages asymmetrically – <u>some receivers may not get data</u>.

Rules for Interactive Consistency

What is an interstage?

- An interstage is an FCR that participates in the interactive consistency exchange, but does not require consensus.
- The purpose of an interstage is to provide the necessary functionality to perform byzantine agreement algorithms without requiring all FCRs to be full processors.

Rules for interactive consistency in 1FT voting systems:

- Requires ≥ 3(1) + 1 = 4 Fault Containment Regions (FCRs).
- Each interstage must receive data through ≥1 disjoint paths.
- Devices which require consensus must get data from:
  - I. ≥ 2(1) + 1 = 3 standard-integrity devices, or
  - II. ≥ (1) + 1 = 2 high-integrity devices, or
  - III. A combination of the above
- Above must be satisfied in (1) + 1 = 2 rounds of data exchange.
- After data exchange, devices requiring consensus perform an absolute majority vote of received messages.

Generalizing use of (HI) interstages

Example 3:

- Six total FCRs
- Two HI interstages
- Two devices require consensus

Rules for IC in 1FT voting systems:

- Requires ≥ 3(1) + 1 = 4 FCRs.
- Interstages need data from ≥1 paths.
- Devices requiring consensus need data:
  - I. from ≥ 2(1) + 1 = 3 LI devices
  - II. from from ≥ (1) + 1 = 2 HI devices
  - III. from a combination of the above
- Two rounds of data exchange.
- Majority vote used to reach consensus.

○ Device requiring consensus

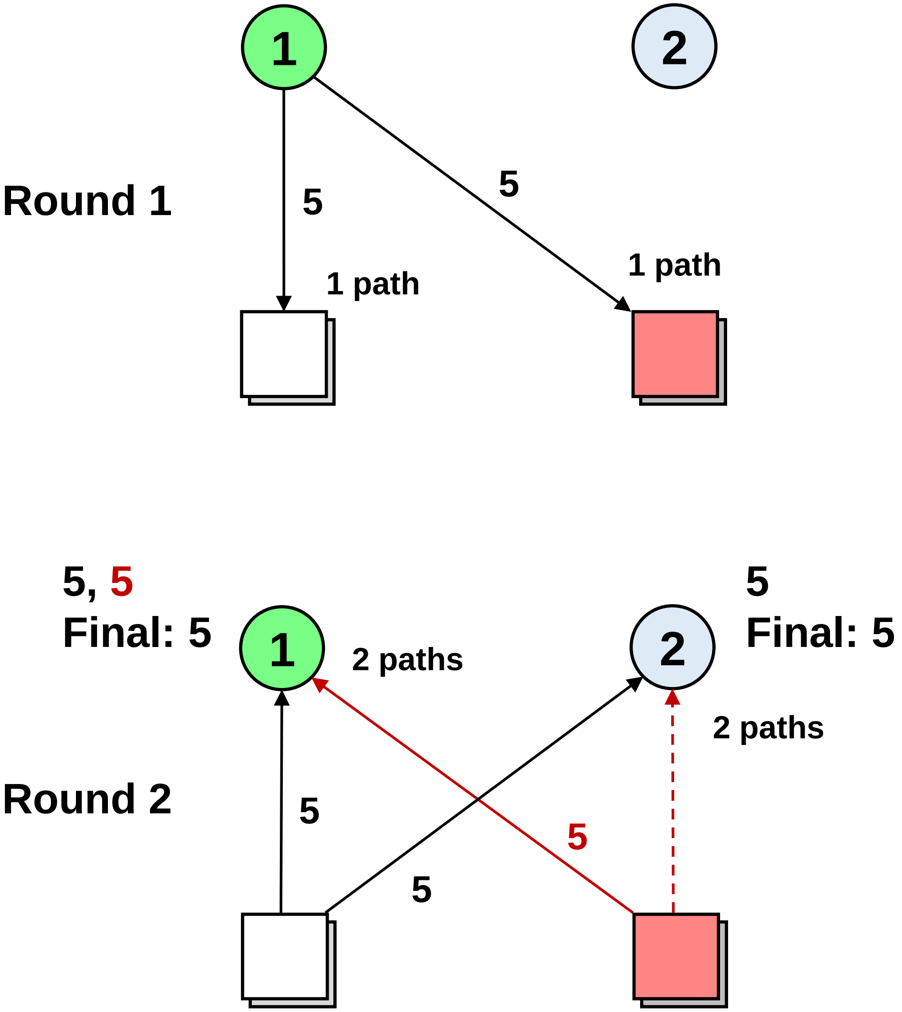
☐ LI Interstage (does not require consensus)

⧉ HI interstage (does not require consensus)

🟩 Designates originating device

🟥 Designates faulty device

Assumption: LI devices may fail arbitrarily, HI devices may fail via inconsistent omission.



Generalizing use of (HI) interstages

Example 4:

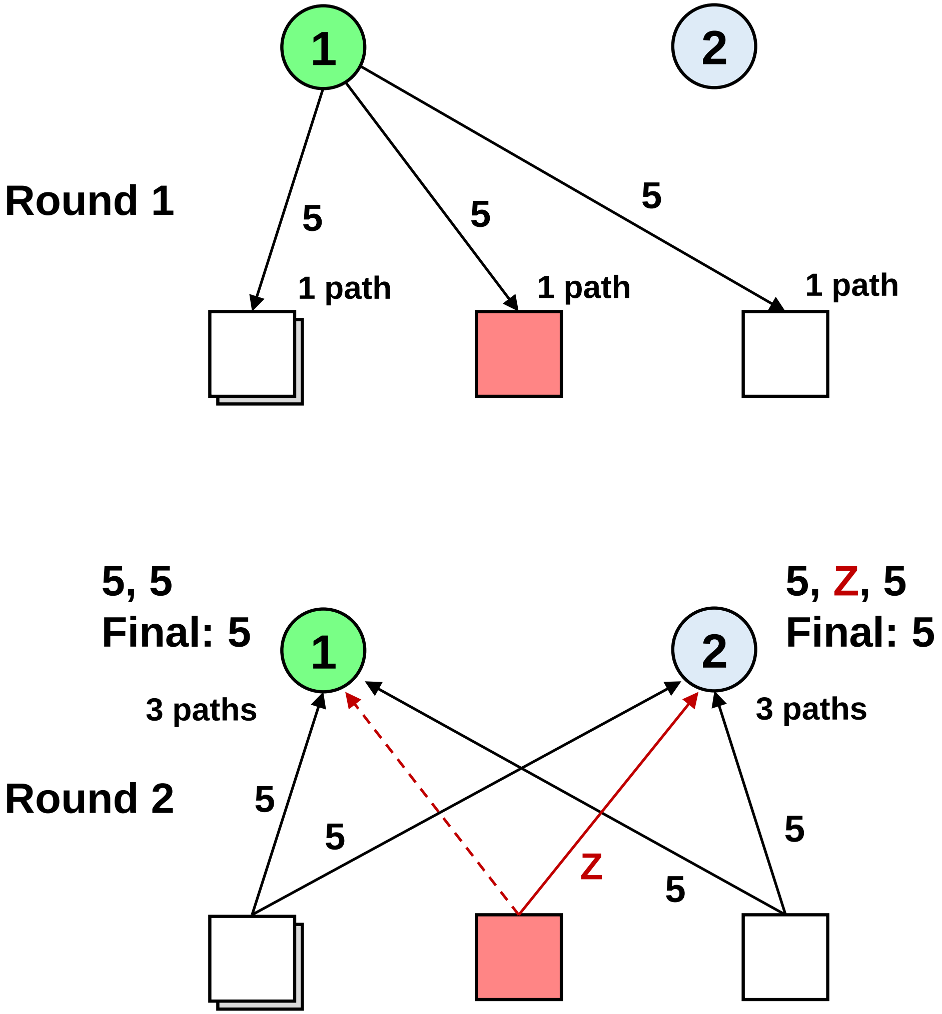
- Six total FCRs
- One HI + two LI interstages
- Two devices require consensus

Rules for IC in 1FT voting systems:

- Requires ≥ 3(1) + 1 = 4 FCRs.
- Interstages need data from ≥1 paths.
- Devices requiring consensus need data:
  1. from ≥ 2(1) + 1 = 3 LI devices
  2. from from ≥ (1) + 1 = 2 HI devices
  3. from a combination of the above
- Two rounds of data exchange.
- Majority vote used to reach consensus.

○ Device requiring consensus
☐ LI Interstage (does not require consensus)
⧉ HI interstage (does not require consensus)
🟩 Designates originating device
🟥 Designates faulty device

Assumption: LI devices may fail arbitrarily, HI devices may fail via inconsistent omission.



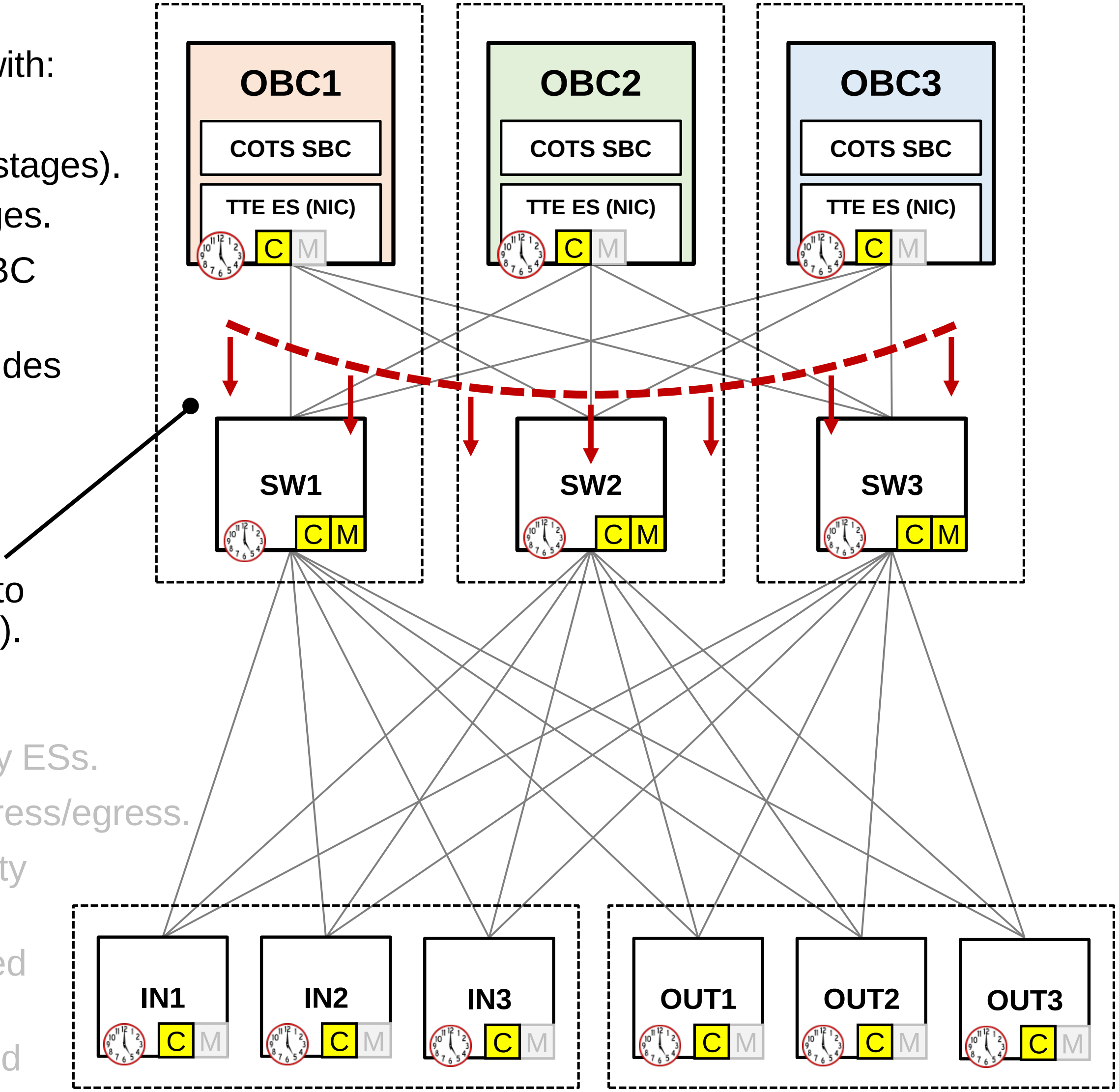
Switched Triplex (Fully Cross-strapped)

General Overview

- Scalable 1FT design can be realized with:
  - 3 full processors/OBCs
  - 2-3 redundant network planes (interstages).
  - Majority voting of redundant messages.
- Fully-cross strapped design – each OBC has access to any networked device.
- Time-Triggered Ethernet network provides data distribution and synchronization between platforms.
  - Does not require separate CCDL or timing/synchronization hardware.
- Triplex OBCs do not directly interface to any end devices (insulated by network).

Device Characteristics

- COM/MON switches, standard integrity ESs.
- Error Containment Unit b/w switch ingress/egress.
- Switches provide 1FT or 2FT availability depending on number of channels.
- COM/MON switches required as trusted Compression Masters (CM) for sync.
- HI switches cannot protect against valid frames containing erroneous data.



Approved for Public Release –
No Export Controlled Data

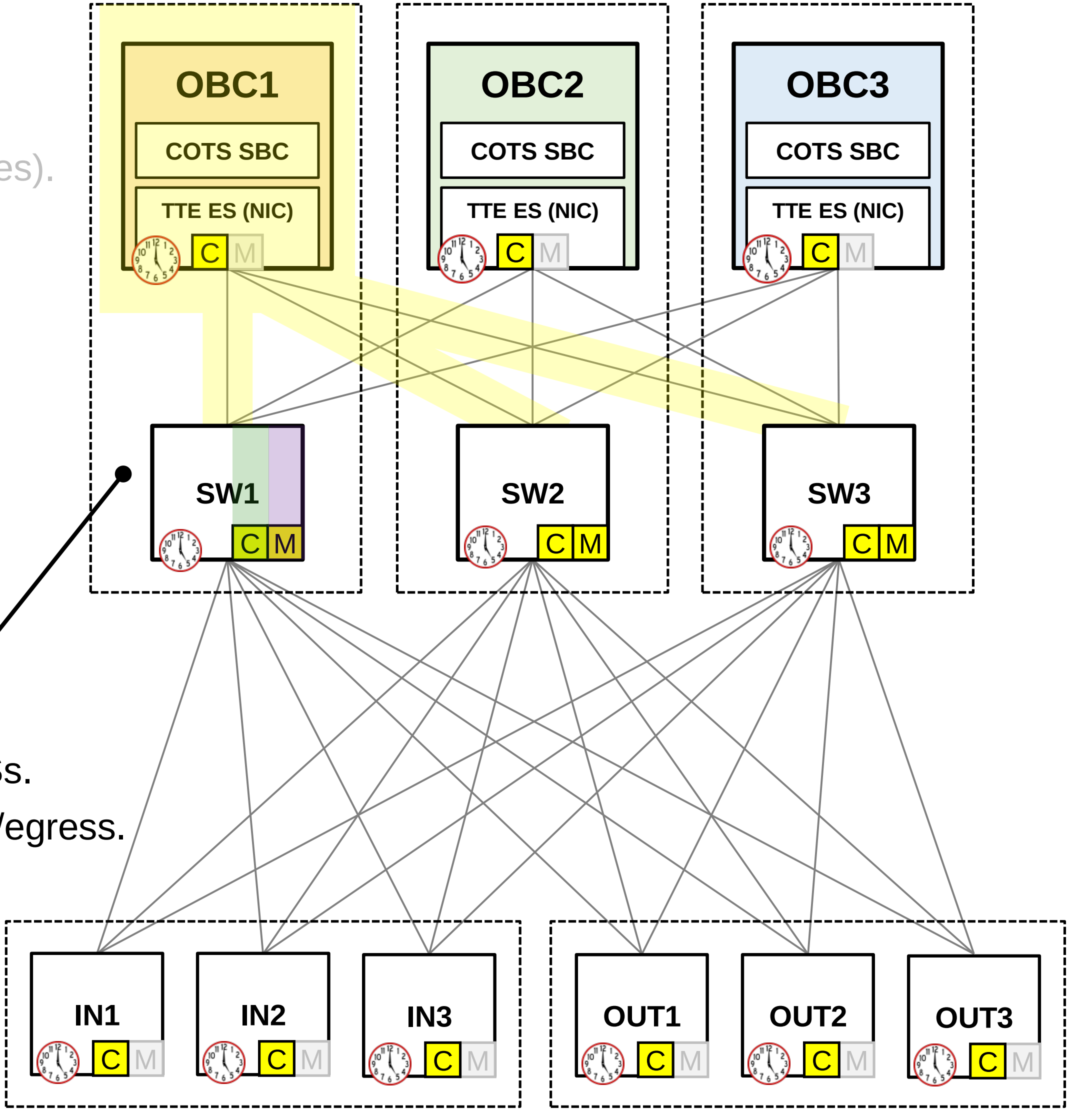
Switched Triplex (Fully Cross-strapped)

General Overview

- Scalable 1FT design can be realized with:
  - 3 full processors/OBCs
  - 2-3 redundant network planes (interstages).
  - Majority voting of redundant messages.
- Fully-cross strapped design – each OBC has access to any networked device.
- Time-Triggered Ethernet network provides data distribution and synchronization between platforms.
  - Does not require separate CCDL or timing/synchronization hardware.
- Triplex OBCs do not directly interface to any end devices (insulated by network).

Device Characteristics

- COM/MON switches, standard integrity ESs.
- Error Containment Unit b/w switch ingress/egress.
- Switches provide 1FT or 2FT availability depending on number of channels.
- COM/MON switches required as trusted Compression Masters (CM) for sync.
- HI switches cannot protect against valid frames containing erroneous data



Approved for Public Release –
No Export Controlled Data

Switched Triplex (Dual-Channel)

Required redundant channels

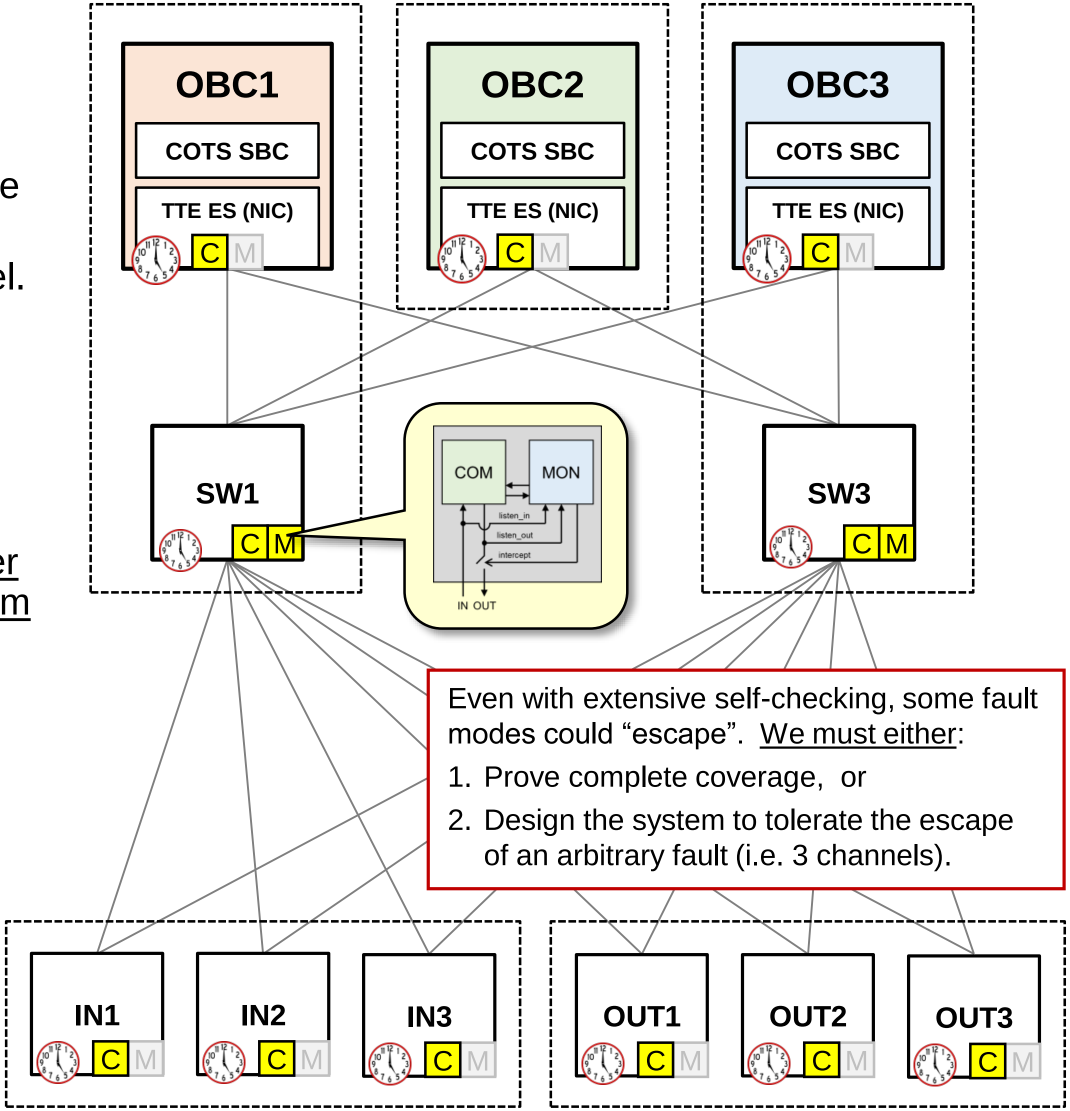
- A 1FT configuration requiring only two network planes is possible only if switches are fully self-checking (fail-silent).
- A restricted failure mode model requires the realization of two independent FCRs.
  - Inconsistent omission is a reduced model.
- Must <u>eliminate common mode elements</u>:
  - E.g. Shared timer, dielectric isolation, physical space, temperature.
- If the switch may fail arbitrarily, then three redundant channels are always required.
- In all cases, <u>3x channels minimizes number of two-fault combinations resulting in system failure over 2x channels.</u>

Current Implementation

- TTTech COM/MON devices share power (with separate power monitor).
- A shared oscillator is used for COM/MON, with a dedicated clock monitor to prevent common mode clock failures.

Fault-propagation from switches <u>theoretically requires dual-correlated simultaneous faults</u>.

→ 1e-6 X 1e-6 = ~1e-12 failures/hour



Even with extensive self-checking, some fault modes could "escape". <u>We must either</u>:

1. Prove complete coverage, or
2. Design the system to tolerate the escape of an arbitrary fault (i.e. 3 channels).

Switched Triplex – Reading Data (1)

Step 1: Exchange (Round 1)

- Each redundant input device (any #) transmits its data to switches 1-3.
  - No guarantee non-faulty devices agree.
  - A failed device may transmit arbitrarily.

Step 2: Exchange (Round 2)

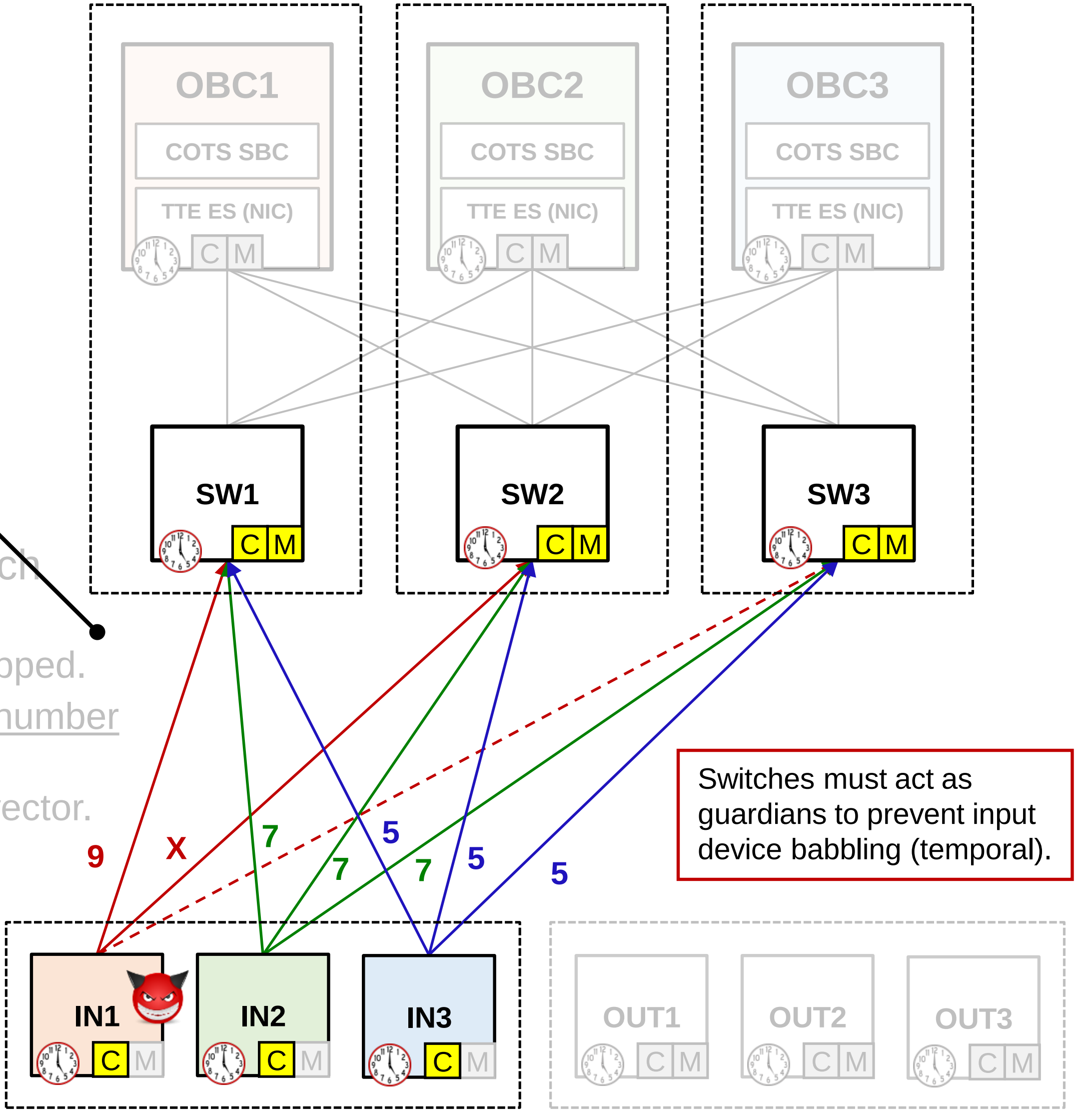
- Switches 1-3 send each redundant input message to all OBCs 1-3.

Step 3: Create symmetry

- OBCs 1-3 performs a majority vote of the message copies received from each redundant network channel.
  - Messages that violate the protocol are dropped.
  - Majority must be determined according to number of messages received (i.e. not static 2/3).
  - Non-faulty OBCs now share the same IC vector.

Step 4: Make a decision

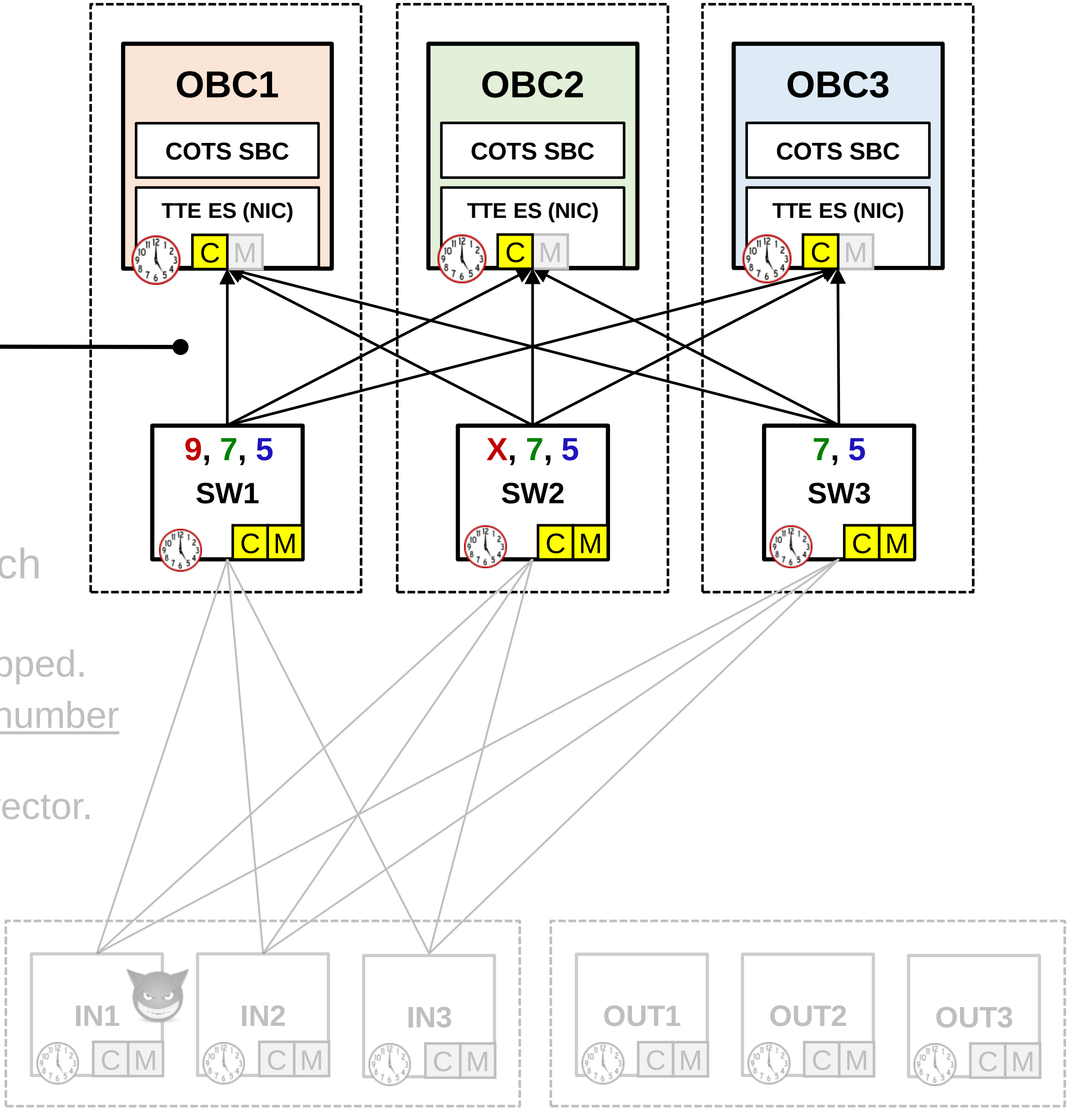
- OBCs 1-3 execute a choice() function to select a final value from the redundant input devices (e.g. median, mean).



Switches must act as guardians to prevent input device babbling (temporal).

Switched Triplex – Reading Data (2)

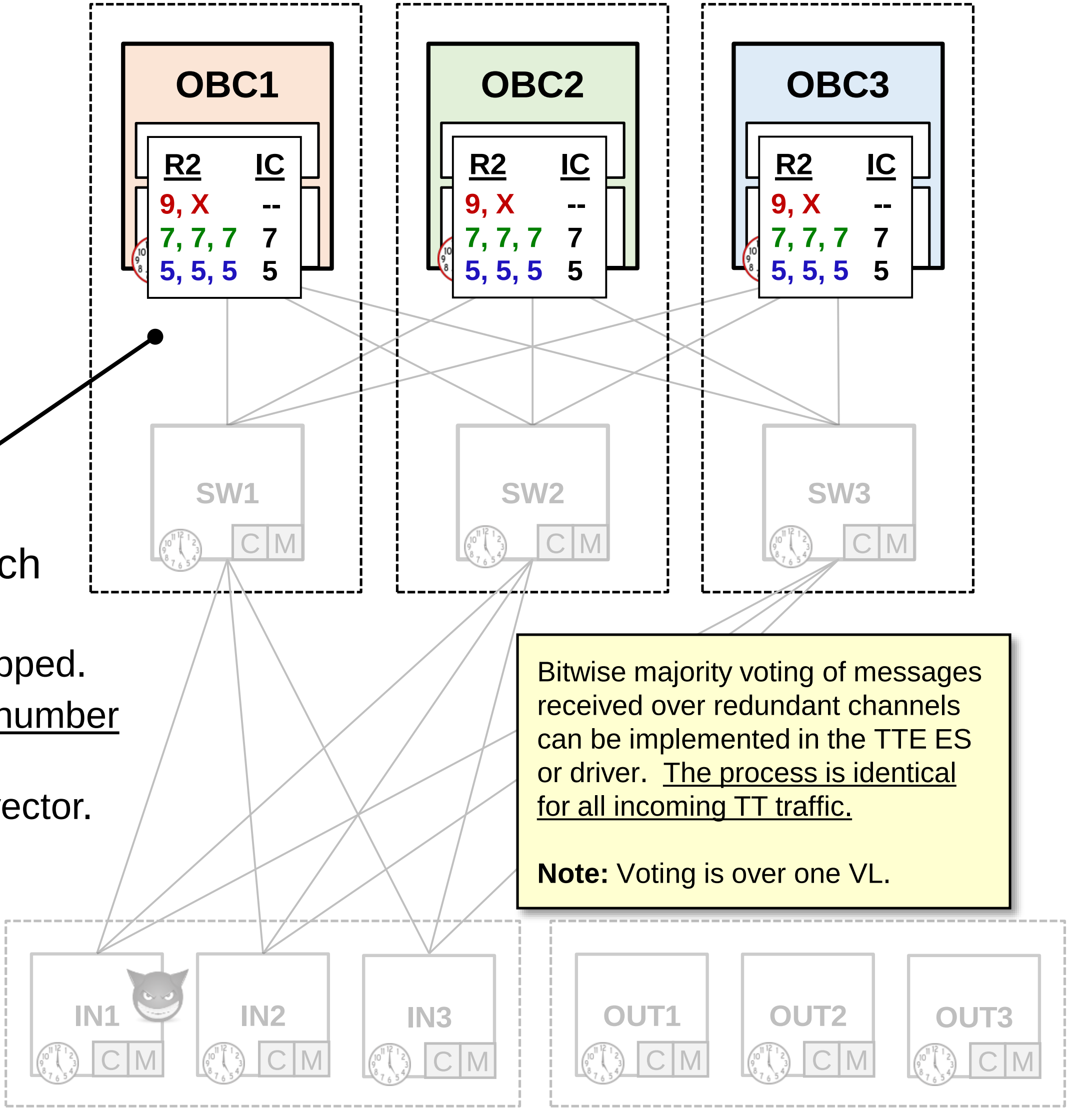
- Step 1: Exchange (Round 1)
  - Each redundant input device (any #) transmits its data to switches 1-3.
    - No guarantee non-faulty devices agree.
    - A failed device may transmit arbitrarily.
- Step 2: Exchange (Round 2)
  - Switches 1-3 send each redundant input message to all OBCs 1-3.
- Step 3: Create symmetry
  - OBCs 1-3 performs a majority vote of the message copies received from each redundant network channel.
    - Messages that violate the protocol are dropped.
    - Majority must be determined according to number of messages received (i.e. not static 2/3).
    - Non-faulty OBCs now share the same IC vector.
- Step 4: Make a decision
  - OBCs 1-3 execute a choice() function to select a final value from the redundant input devices (e.g. median, mean).



Approved for Public Release – No Export Controlled Data

Switched Triplex – Reading Data (3)

- **Step 1: Exchange (Round 1)**
  - Each redundant input device (any #) transmits its data to switches 1-3.
    - No guarantee non-faulty devices agree.
    - A failed device may transmit arbitrarily.
- **Step 2: Exchange (Round 2)**
  - Switches 1-3 send each redundant input message to all OBCs 1-3.
- **Step 3: Create symmetry**
  - OBCs 1-3 performs a majority vote of the message copies received from each redundant network channel.
    - Messages that violate the protocol are dropped.
    - Majority must be determined according to number of messages received (i.e. not static 2/3).
    - Non-faulty OBCs now share the same IC vector.
- **Step 4: Make a decision**
  - OBCs 1-3 execute a choice() function to select a final value from the redundant input devices (e.g. median, mean).



Bitwise majority voting of messages received over redundant channels can be implemented in the TTE ES or driver. The process is identical for all incoming TT traffic.

Note: Voting is over one VL.

Switched Triplex – Reading Data (4)

Step 1: Exchange (Round 1)

- Each redundant input device (any #) transmits its data to switches 1-3.
  - No guarantee non-faulty devices agree.
  - A failed device may transmit arbitrarily.

Step 2: Exchange (Round 2)

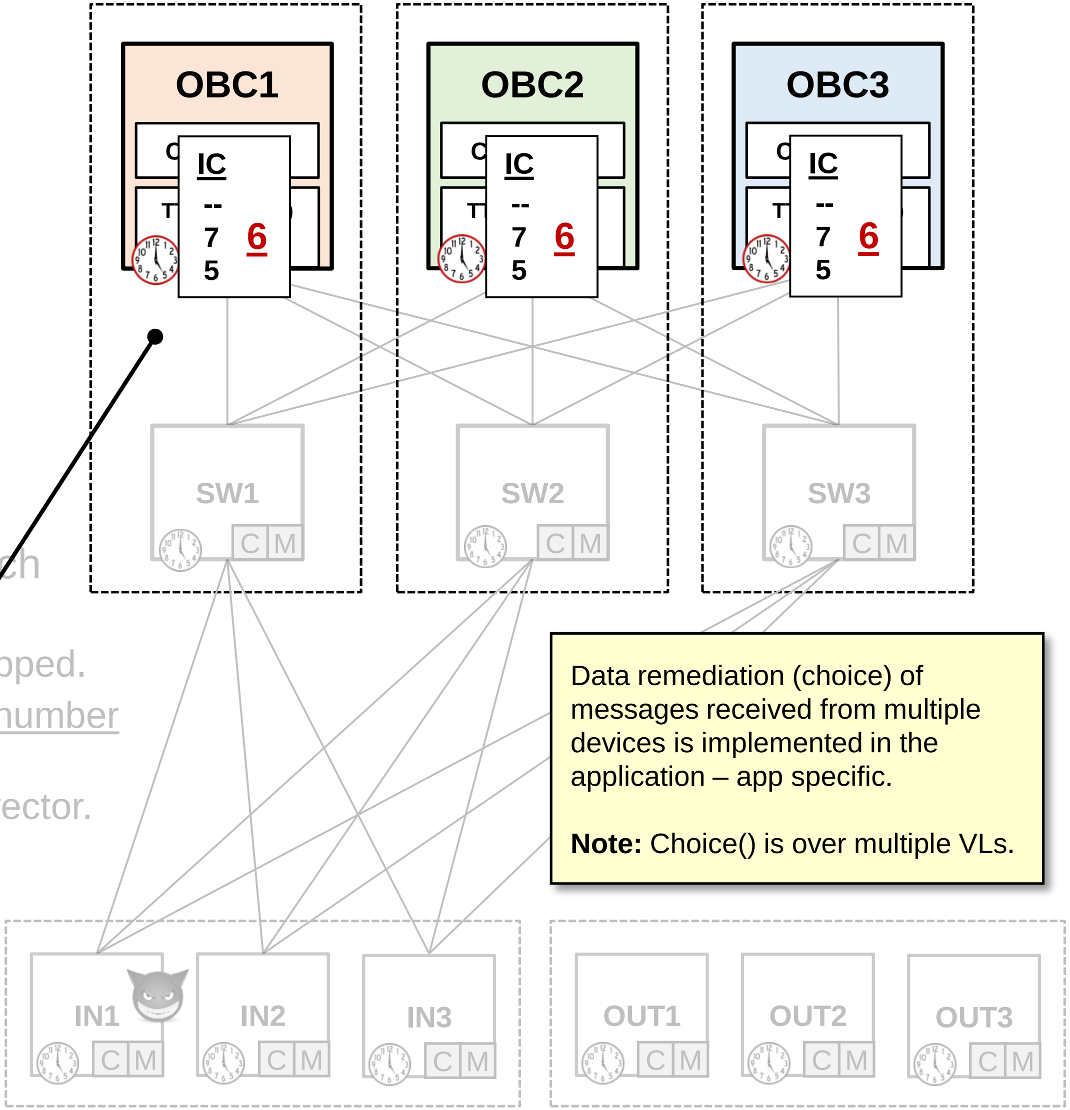
- Switches 1-3 send each redundant input message to all OBCs 1-3.

Step 3: Create symmetry

- OBCs 1-3 performs a majority vote of the message copies received from each redundant network channel.
  - Messages that violate the protocol are dropped.
  - Majority <u>must be determined according to number of messages received</u> (i.e. not static 2/3).
  - Non-faulty OBCs now share the same IC vector.

Step 4: Make a decision

- OBCs 1-3 execute a choice() function to select a final value from the redundant input devices (e.g. median, mean).

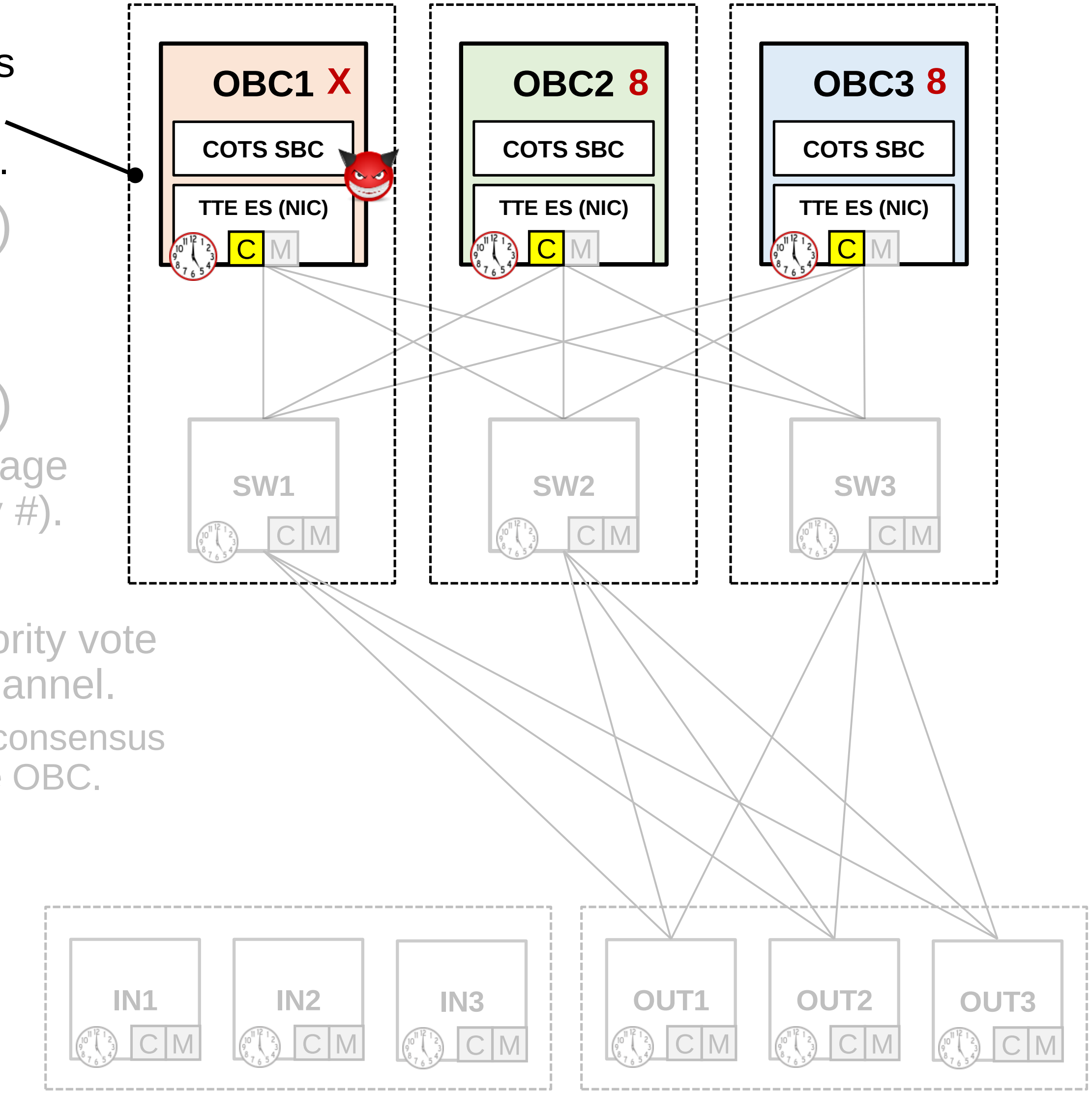


Data remediation (choice) of messages received from multiple devices is implemented in the application – app specific.

Note: Choice() is over multiple VLs.

Switched Triplex – Commanding (1)

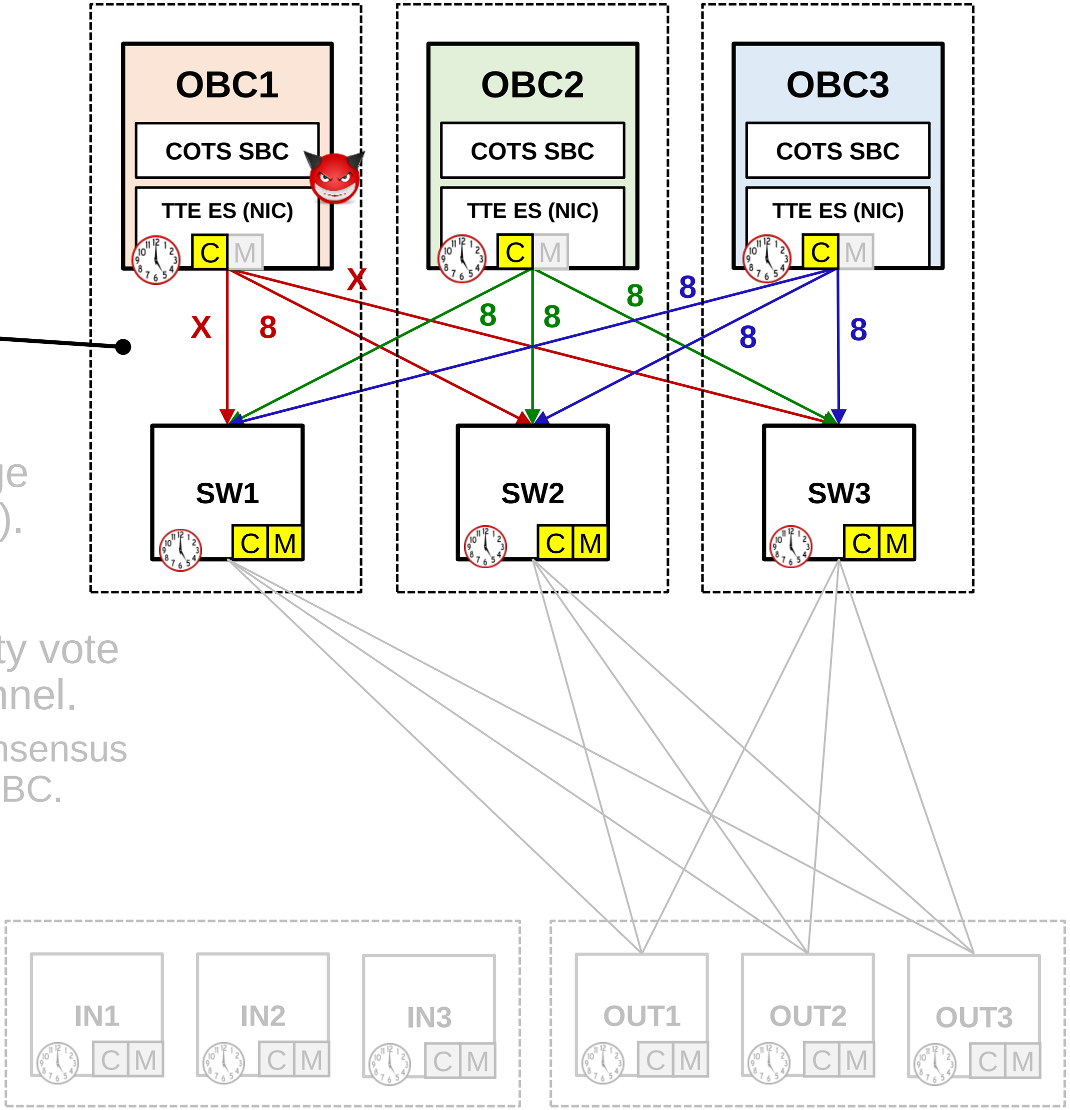
- Step 1: Prepare Command
  - After performing computation, OBCs 1-3 each generate a command.
    - All non-faulty OBCs agree on the output.
- Step 2: Exchange (Round 1)
  - Each OBC 1-3 transmits its output value to all switches 1-3.
- Step 3: Exchange (Round 2)
  - Switches 1-3 send each input message to all redundant output devices (any #).
- Step 4: Create symmetry
  - Each output device performs a majority vote of messages received from each channel.
    - This IC exchange is required to ensure consensus of multiple output devices in case of one OBC.
- Step 5: Make a decision
  - Each output device performs a <u>second majority vote</u> over the commands from each OBC.
    - I.e. the choice() function for output devices is always a bitwise majority.



Approved for Public Release –
No Export Controlled Data

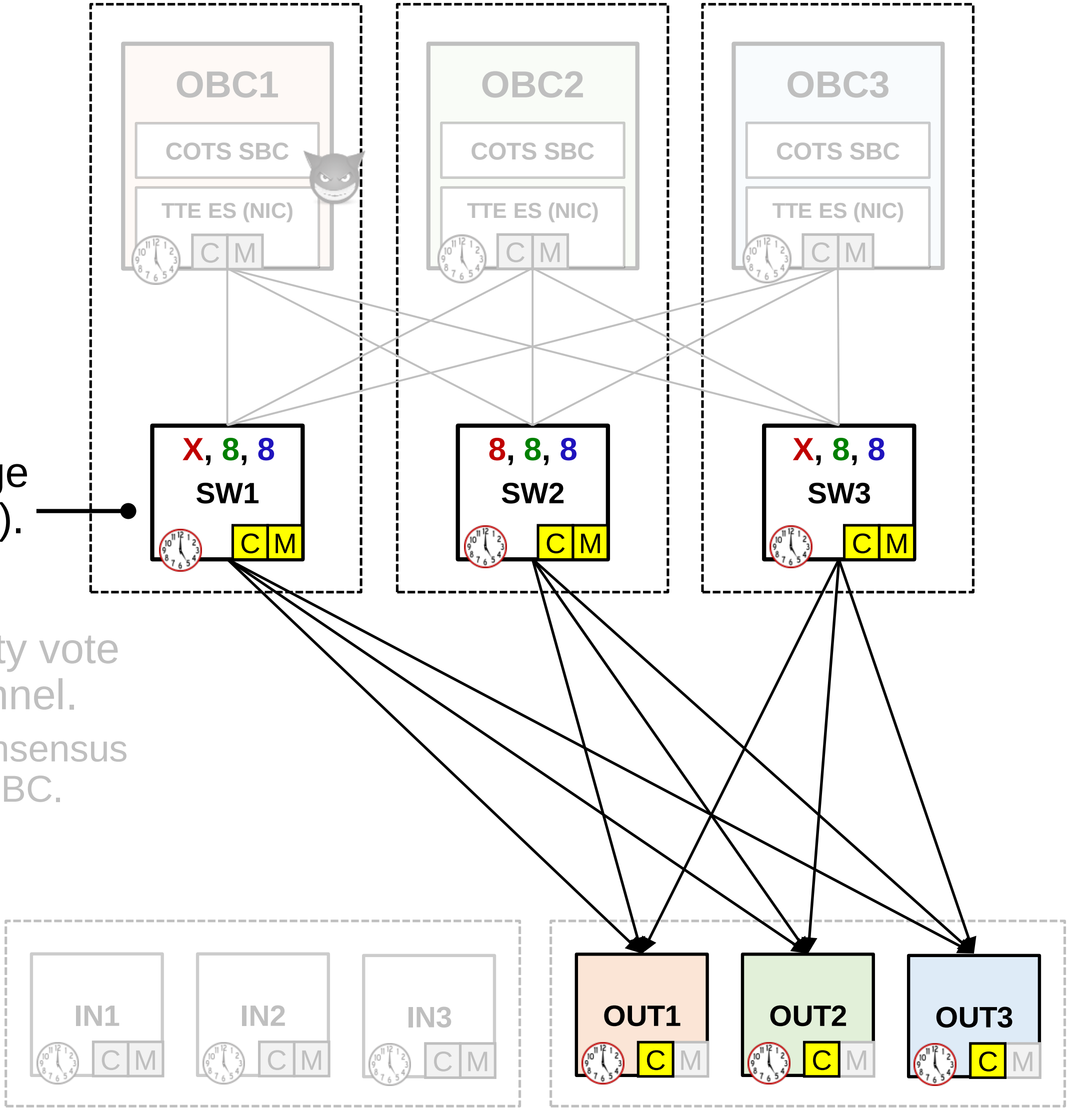
Switched Triplex – Commanding (2)

- Step 1: Prepare Command
  - After performing computation, OBCs 1-3 each generate a command.
    - All non-faulty OBCs agree on the output.
- Step 2: Exchange (Round 1)
  - Each OBC 1-3 transmits its output value to all switches 1-3.
- Step 3: Exchange (Round 2)
  - Switches 1-3 send each input message to all redundant output devices (any #).
- Step 4: Create symmetry
  - Each output device performs a majority vote of messages received from each channel.
    - This IC exchange is required to ensure consensus of multiple output devices in case of one OBC.
- Step 5: Make a decision
  - Each output device performs a second majority vote over the commands from each OBC.
    - I.e. the choice() function for output devices is always a bitwise majority.



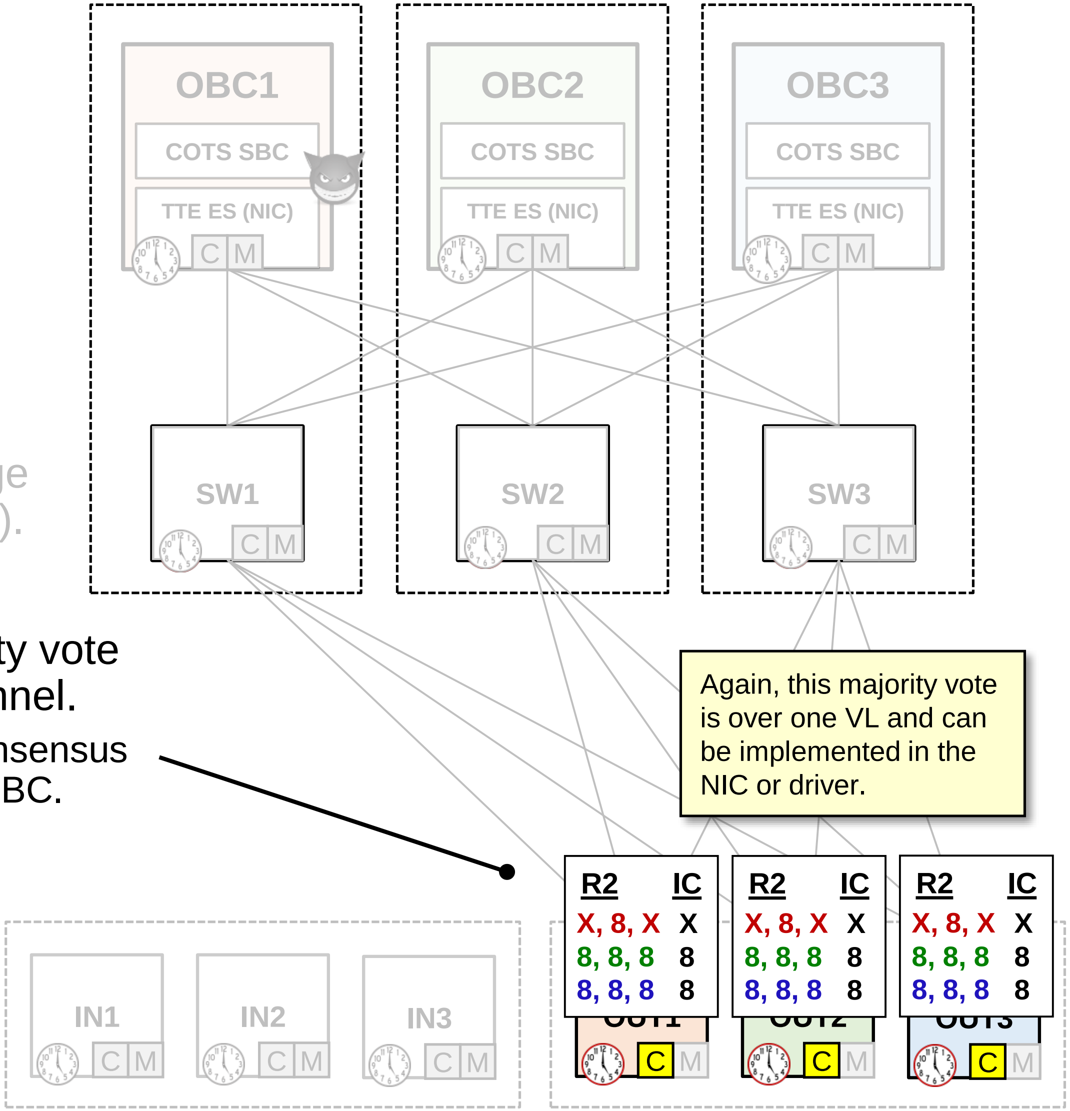
Switched Triplex – Commanding (3)

- Step 1: Prepare Command
  - After performing computation, OBCs 1-3 each generate a command.
    - All non-faulty OBCs agree on the output.
- Step 2: Exchange (Round 1)
  - Each OBC 1-3 transmits its output value to all switches 1-3.
- Step 3: Exchange (Round 2)
  - Switches 1-3 send each input message to all redundant output devices (any #).
- Step 4: Create symmetry
  - Each output device performs a majority vote of messages received from each channel.
    - This IC exchange is required to ensure consensus of multiple output devices in case of one OBC.
- Step 5: Make a decision
  - Each output device performs a second majority vote over the commands from each OBC.
    - I.e. the choice() function for output devices is always a bitwise majority.



Switched Triplex – Commanding (4)

- Step 1: Prepare Command
  - After performing computation, OBCs 1-3 each generate a command.
    - All non-faulty OBCs agree on the output.
- Step 2: Exchange (Round 1)
  - Each OBC 1-3 transmits its output value to all switches 1-3.
- Step 3: Exchange (Round 2)
  - Switches 1-3 send each input message to all redundant output devices (any #).
- Step 4: Create symmetry
  - Each output device performs a majority vote of messages received from each channel.
    - This IC exchange is required to ensure consensus of multiple output devices in case of one OBC.
- Step 5: Make a decision
  - Each output device performs a second majority vote over the commands from each OBC.
    - I.e. the choice() function for output devices is always a bitwise majority.



Again, this majority vote is over one VL and can be implemented in the NIC or driver.

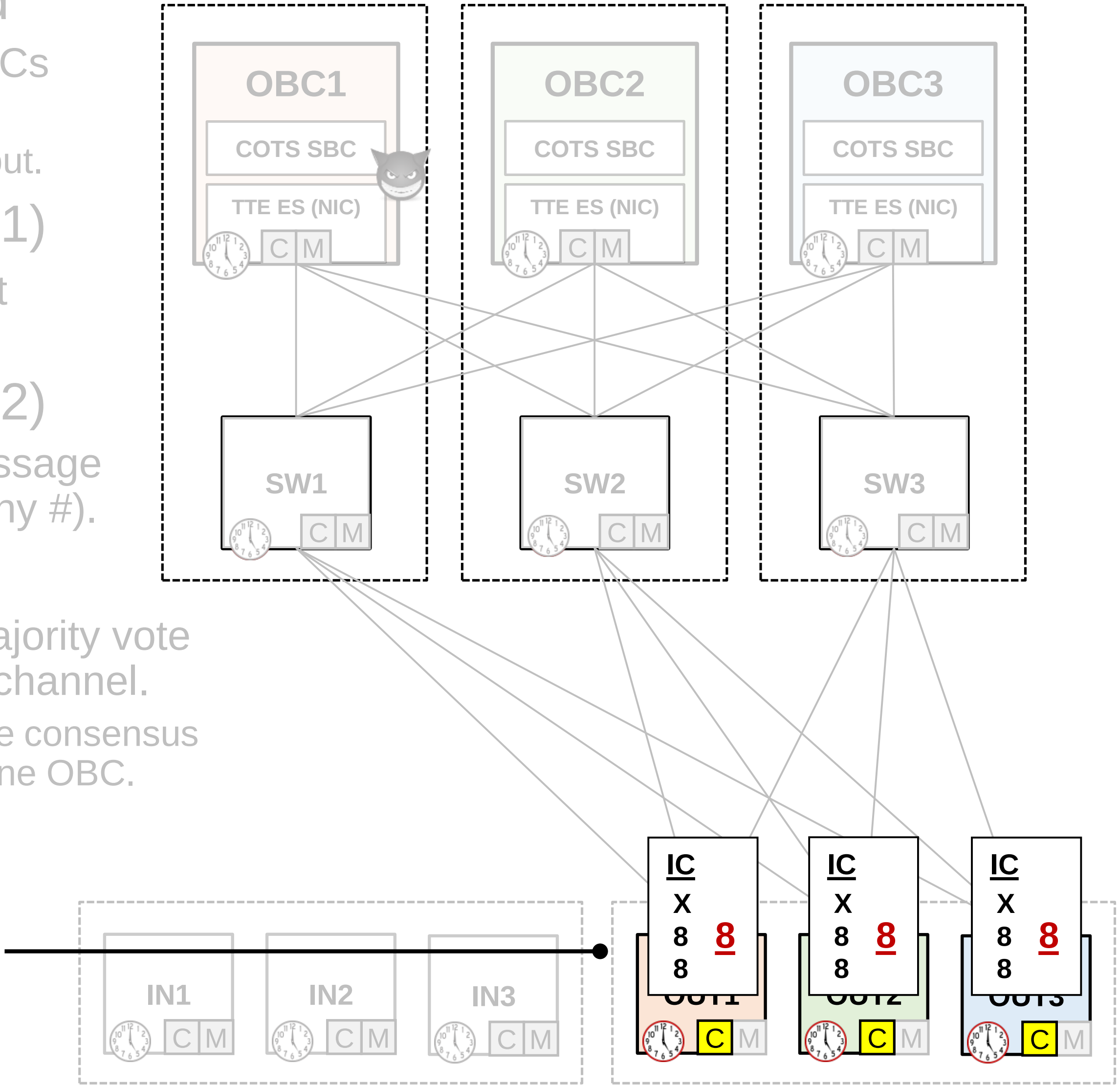
R2	IC
X, 8, X	X
8, 8, 8	8
8, 8, 8	8

R2	IC
X, 8, X	X
8, 8, 8	8
8, 8, 8	8

R2	IC
X, 8, X	X
8, 8, 8	8
8, 8, 8	8

Switched Triplex – Commanding (5)

- Step 1: Prepare Command
  - After performing computation, OBCs 1-3 each generate a command.
    - All non-faulty OBCs agree on the output.
- Step 2: Exchange (Round 1)
  - Each OBC 1-3 transmits its output value to all switches 1-3.
- Step 3: Exchange (Round 2)
  - Switches 1-3 send each input message to all redundant output devices (any #).
- Step 4: Create symmetry
  - Each output device performs a majority vote of messages received from each channel.
    - This IC exchange is required to ensure consensus of multiple output devices in case of one OBC.
- Step 5: Make a decision
  - Each output device performs a <u>second majority vote</u> over the commands from each OBC.
    - I.e. the choice() function for output devices is always a bitwise majority.



Switched Triplex – Monitoring (1)

Step 1: Prepare Command

- After performing computation, OBCs 1-3 each generate a command.
  - All non-faulty OBCs agree on the output.

Step 2: Exchange (Round 1)

↓ Happening Simultaneously

Step 3: Exchange (Round 2)

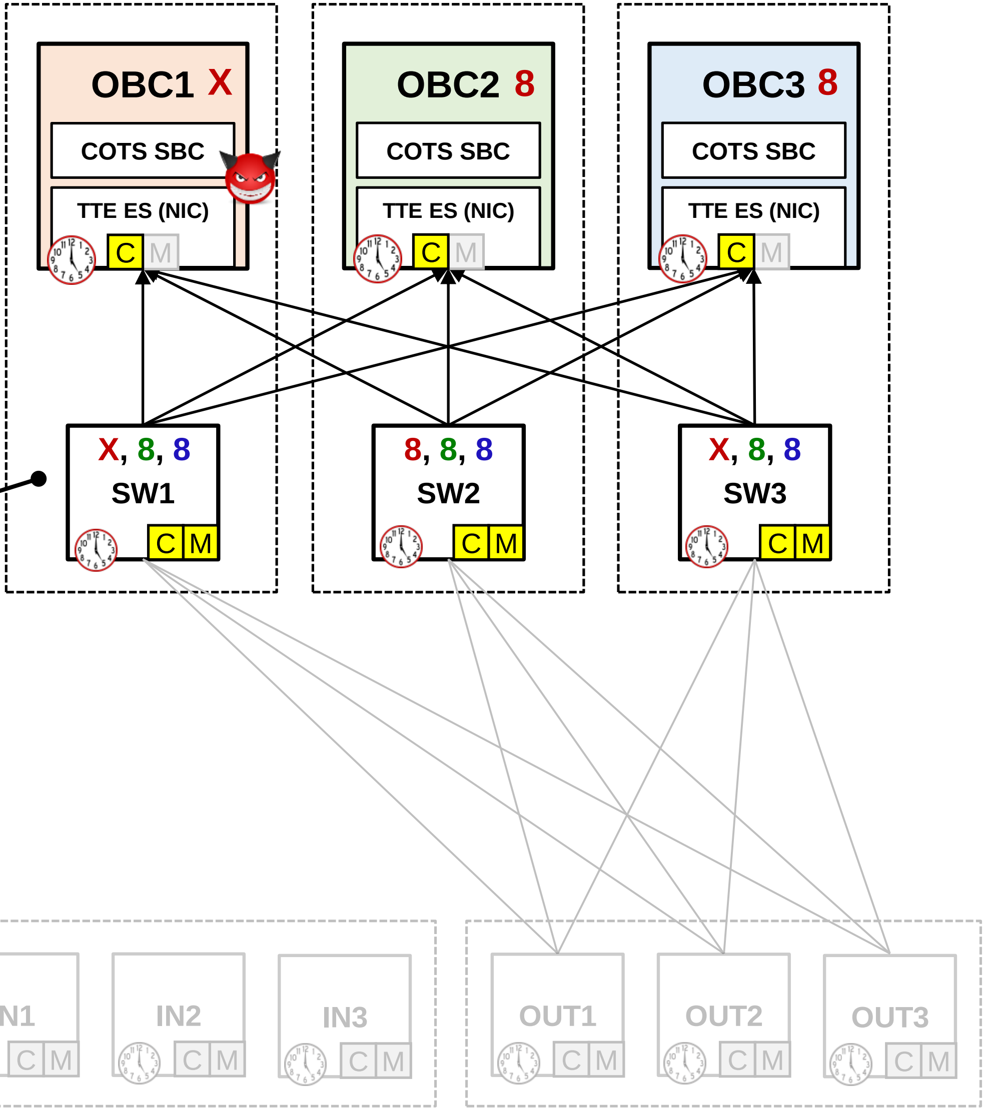
- Switches 1-3 send each input message "reflected" back to each OBC 1-3.
  - **Why?** Allows CFS app to monitor OBCs for the purpose of fault detection and reconfiguration.

Step 4: Create symmetry

- Each OBC 1-3 performs a majority vote of messages received from each channel.

Step 5: Identify faulty OBC

- OBCs 1-3 perform a majority vote over the commands from each OBC.
  - Identical to action performed by OUT 1-3.
  - Can be used to identify OBCs that do not agree with the majority (for FDIR).

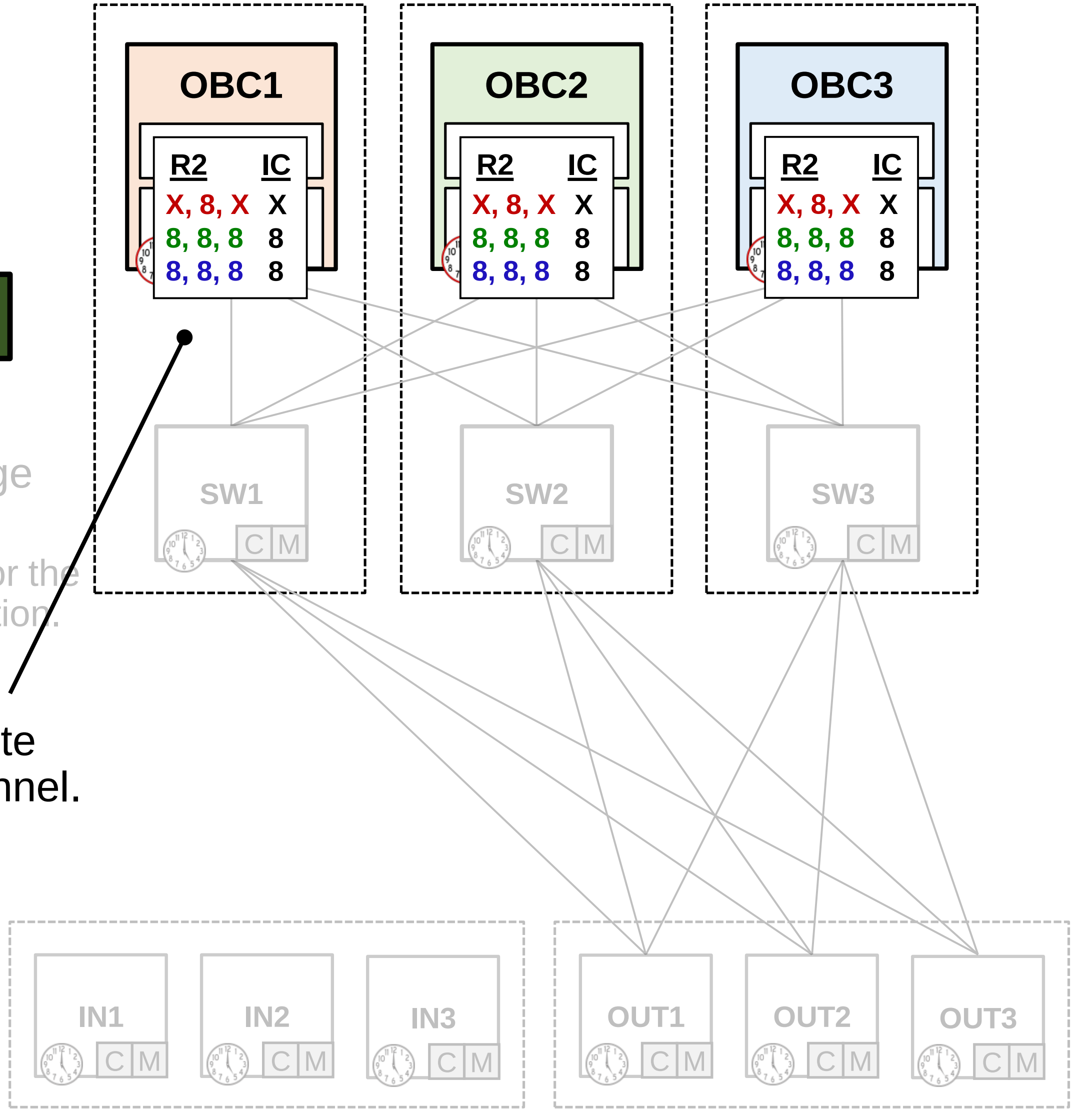


Switched Triplex – Monitoring (2)

- Step 1: Prepare Command
  - After performing computation, OBCs 1-3 each generate a command.
    - All non-faulty OBCs agree on the output.
- Step 2: Exchange (Round 1)

↓ Happening Simultaneously

- Step 3: Exchange (Round 2)
  - Switches 1-3 send each input message “reflected” back to each OBC 1-3.
    - **Why?** Allows CFS app to monitor OBCs for the purpose of fault detection and reconfiguration.
- Step 4: Create symmetry
  - Each OBC 1-3 performs a majority vote of messages received from each channel.
- Step 5: Identify faulty OBC
  - OBCs 1-3 perform a majority vote over the commands from each OBC.
    - Identical to action performed by OUT 1-3.
    - Can be used to identify OBCs that do not agree with the majority (for FDIR).

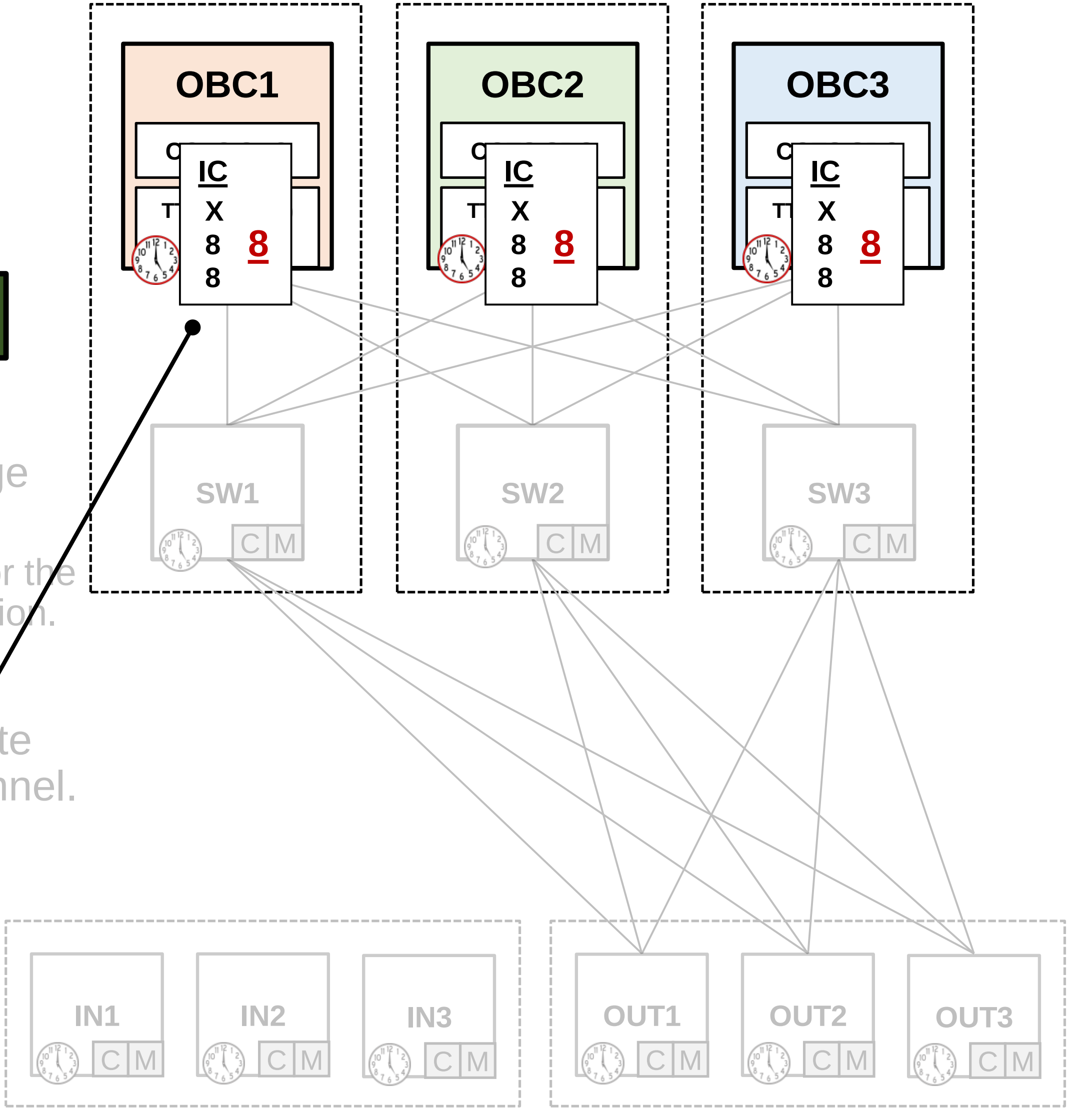


Switched Triplex – Monitoring (3)

- Step 1: Prepare Command
  - After performing computation, OBCs 1-3 each generate a command.
    - All non-faulty OBCs agree on the output.
- Step 2: Exchange (Round 1)

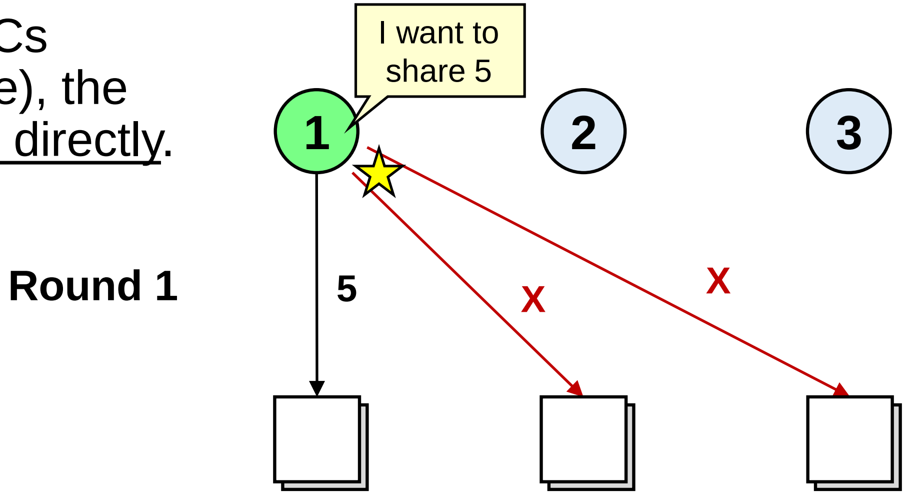
↓ Happening Simultaneously

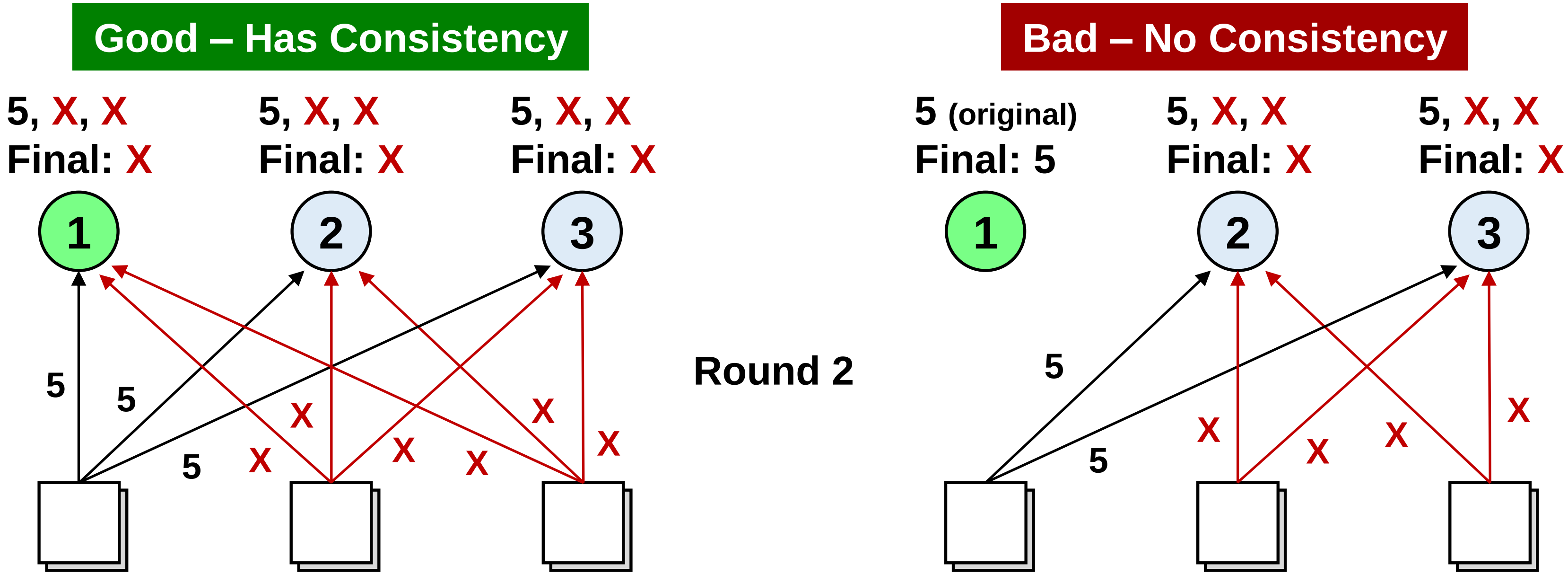
- Step 3: Exchange (Round 2)
  - Switches 1-3 send each input message "reflected" back to each OBC 1-3.
    - **Why?** Allows CFS app to monitor OBCs for the purpose of fault detection and reconfiguration.
- Step 4: Create symmetry
  - Each OBC 1-3 performs a majority vote of messages received from each channel.
- Step 5: Identify faulty OBC
  - OBCs 1-3 perform a majority vote over the commands from each OBC.
    - Identical to action performed by OUT 1-3.
    - Can be used to identify OBCs that do not agree with the majority (for FDIR).



Side Note – Sharing between OBCs

- When sharing a value between OBCs (e.g. output monitoring, shared state), the original sender cannot use its value directly.
- Instead, it performs a majority vote of the values reflected back from the switches (i.e. IC).
- This ensures consensus in case of an arbitrary transmission error.





Network-Level IC Advantages

Network-Level IC = no host blocking

- Consensus between multiple receivers can be achieved transparent to the flight software (no impact on CFS).
- If you read a value, you already know it is the voted answer from a two round exchange – consistent across all receivers (1FT).
- Eliminates classical “acceptance window” for exchanges.
- No need for “read, send, wait … read, send, etc.”
- Minimizes use of host resources (especially if in NIC).

RIUs and Distributed Intelligence

The Role of the Remote Interface Unit (RIU)

- The RIU acts as a gateway between the TTE network, analog devices, and legacy buses (e.g. MIL-STD-1553, ARINC 429).
- Moves signal conditioning closer to sensor/effectors, reducing noise and wiring mass.
- Functions it may implement include A/D conversion, network formatting, range checking, scaling, linearization, and threshold/filter services specific to each subsystem.
- Uses configuration files to map local buffer data to TTE dataports.

RIUs and Distributed Intelligence

Approach 1:

- One RIU
- One sensor

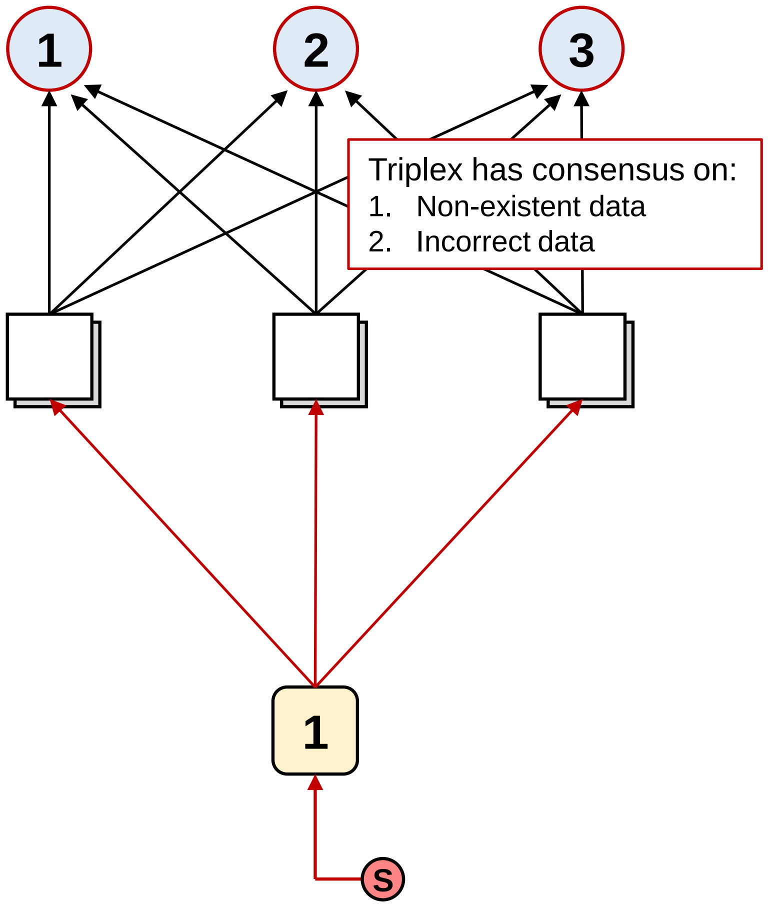
Problems?

- Sensor data sent to RIU may be wrong.

The Fix:

- Add redundant sensors and have RIU remediate between them.

Subsystem cannot function



Triplex has consensus on:

1. Non-existent data
2. Incorrect data

○ Onboard Flight Computer
🟨 Remote Interface Unit (RIU)
● Sensor or Actuator
❐ TTE network switch (COM/MON)
🟥 Designates faulty device

RIUs and Distributed Intelligence

Approach 2:

- One RIU
- Remediation b/w multiple sensors

Problems?

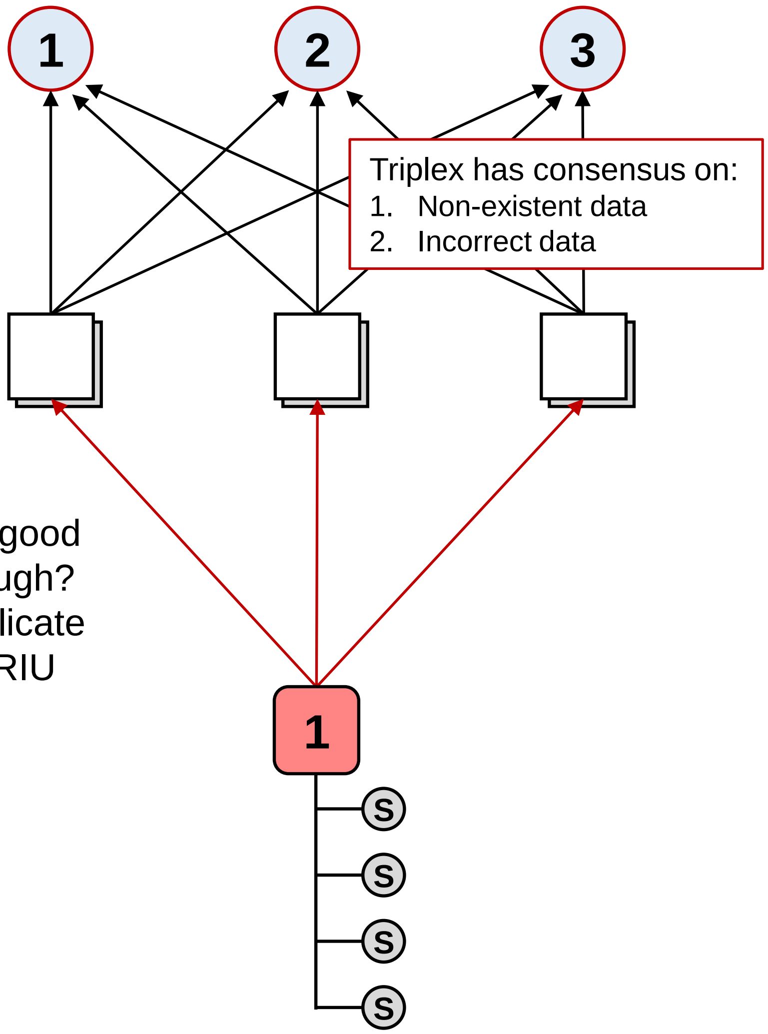
- RIU could fail internally, resulting in:
  1. No-transmission
  2. Symmetric faulty transmission

The Fix:

- Increase resilience of the RIU:
  1. TMR of processor elements (e.g. Maxwell SCS750 used on ESA Gaia satellite).
  2. True dual-core lock-step processor (i.e. fully isolated self-checking).
     - COTS products like ARM Cortex-R4/R5 not available in rad-tolerant variants.

Not good enough? Replicate the RIU

Subsystem <u>cannot function</u>



Triplex has consensus on:
1. Non-existent data
2. Incorrect data

○ Onboard Flight Computer
🟨 Remote Interface Unit (RIU)
○ Sensor or Actuator
❐ TTE network switch (COM/MON)
🟥 Designates faulty device

RIUs and Distributed Intelligence

Approach 3:

- One RIU with HI processor
- Remediation b/w multiple sensors

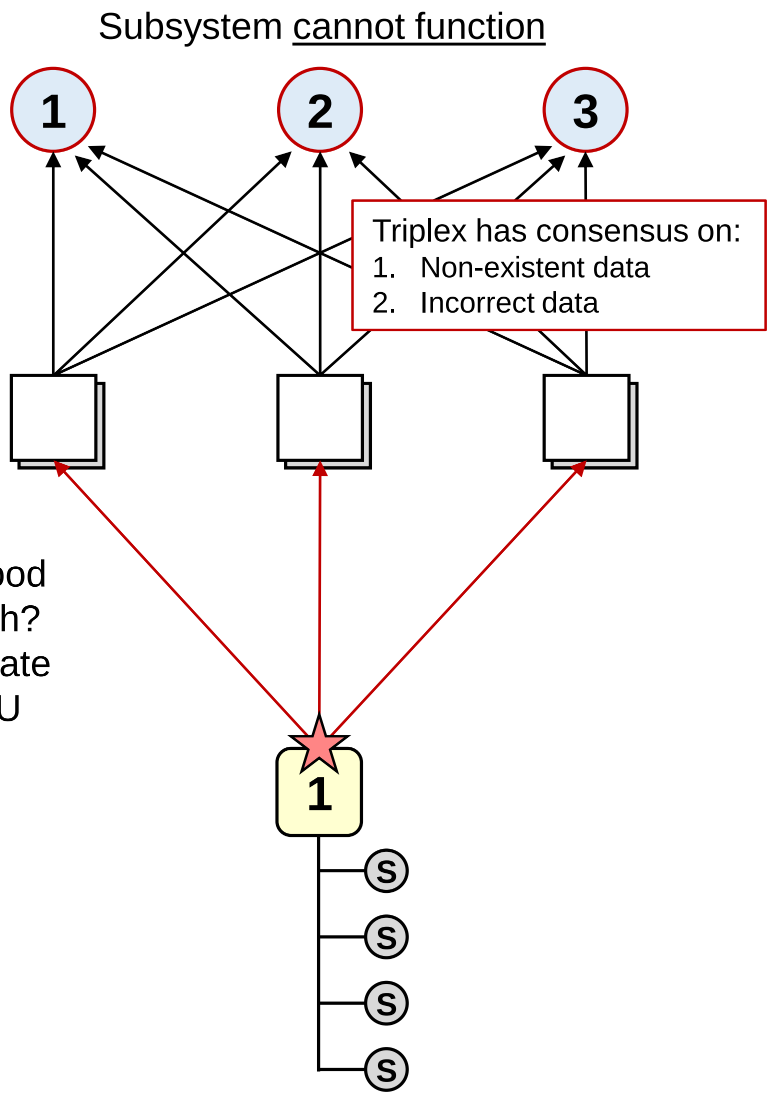
Problems?

- TTE ES could fail arbitrarily, resulting in:
  1. No-transmission
  2. Symmetric faulty transmission
  3. Byzantine transmission

The Fix:

- Increase resilience of the end system:
  1. TMR in the TTE Chip-IP MAC layer.
  2. Use a COM/MON HI end system.
     - Not available in TTTech Space ASIC.

Not good enough? Replicate the RIU



Subsystem <u>cannot function</u>

Triplex has consensus on:
1. Non-existent data
2. Incorrect data

○ Onboard Flight Computer
□ Remote Interface Unit (RIU)
○ Sensor or Actuator
❐ TTE network switch (COM/MON)
🟥 Designates faulty device

RIUs and Distributed Intelligence

Approach 4:

- Multiple RIUs
- Each reads redundant sensors

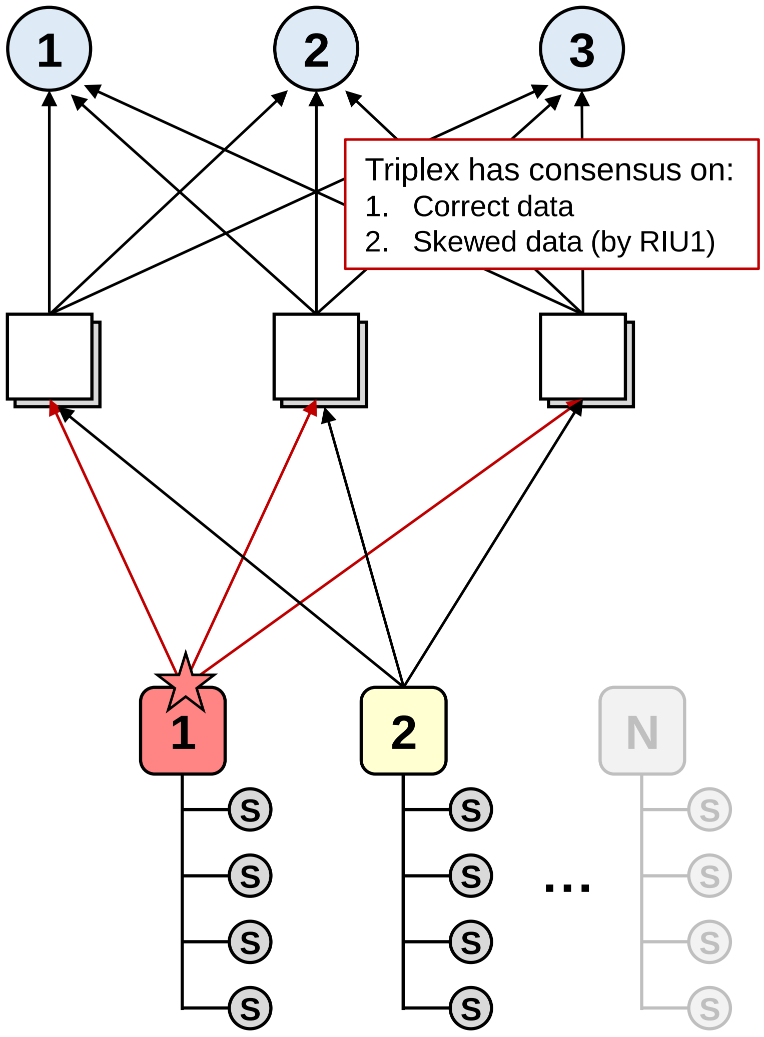
Problems?

- None. Any arbitrary failure of an RIU is tolerated by the Triplex computers:
  - Choice() function is application specific.

Caveats:

- Each RIU performs only minimal local processing (e.g. message packing).
- No consensus is required between RIUs before transmitting data.
  - Since OBCs make decisions, OBCs require the consistency.

Subsystem <u>able to function</u>



Triplex has consensus on:
1. Correct data
2. Skewed data (by RIU1)

○ Onboard Flight Computer
☐ Remote Interface Unit (RIU)
○ Sensor or Actuator
❐ TTE network switch (COM/MON)
🟥 Designates faulty device

RIUs and Distributed Intelligence

Approach 5:

- Multiple RIUs
- Each reads redundant sensors
- RIUs require consensus

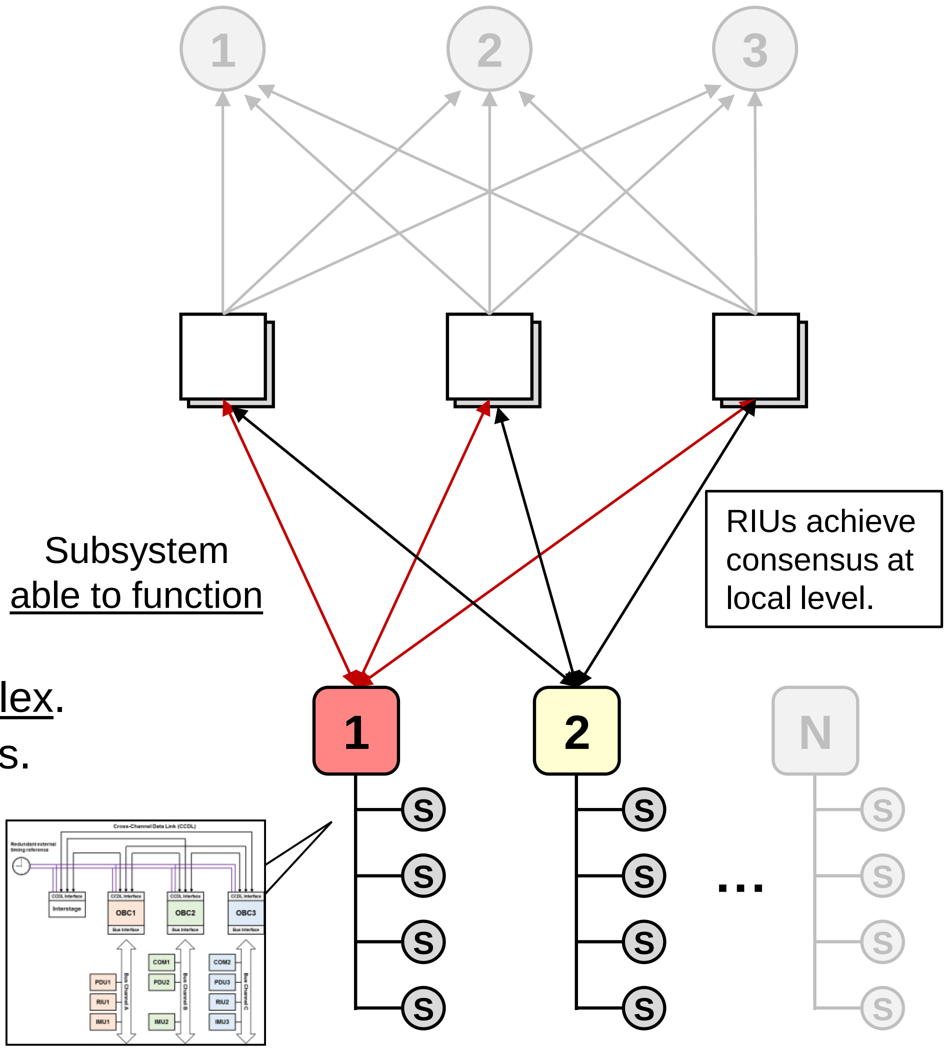
Description

- If consensus between RIUs is necessary without interacting with the OBCs, then IC can be performed between RIUs.
  - Uses redundant network channels to provide the necessary FCRs.
  - Process is similar to classical channelized bus voting approach.

Caveats:

- Can make architecture much more complex.
- 1FT bus commanding may require 3 RIUs.

○ Onboard Flight Computer
🟨 Remote Interface Unit (RIU)
● Sensor or Actuator
❐ TTE network switch (COM/MON)
🟥 Designates faulty device



Subsystem able to function

RIUs achieve consensus at local level.

Notional Onboard Traffic Flow

Best-Effort (IEEE 802.3)

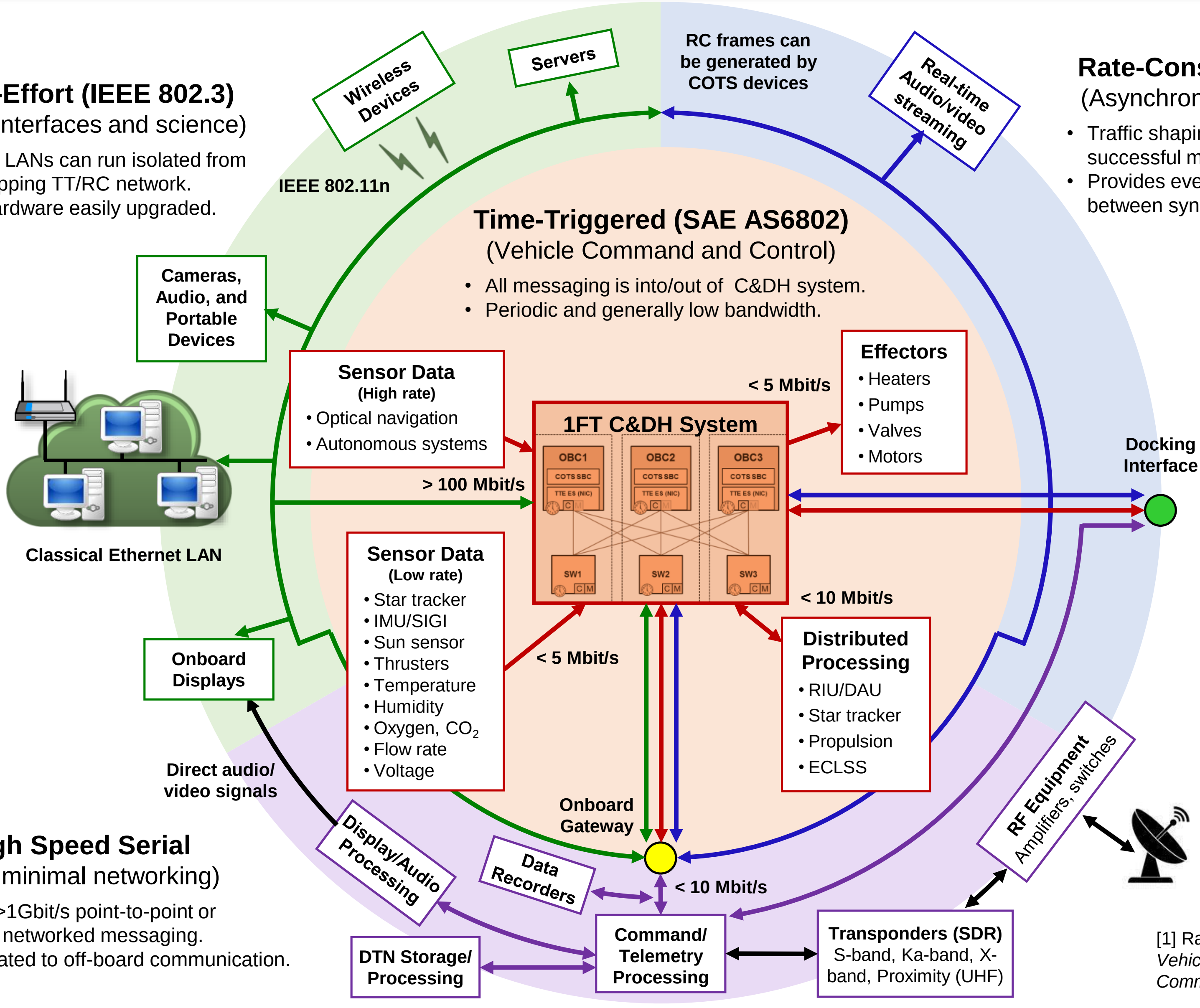
(Crew interfaces and science)

- Classical LANs can run isolated from or overlapping TT/RC network.
- COTS hardware easily upgraded.

Rate-Constrained (A664-p7)

(Asynchronous critical systems)

- Traffic shaping and policing ensures successful message delivery.
- Provides event-driven communication between synchronization domains.



Time-Triggered (SAE AS6802)

(Vehicle Command and Control)

- All messaging is into/out of C&DH system.
- Periodic and generally low bandwidth.

Wireless Devices

Servers

RC frames can be generated by COTS devices

Real-time Audio/video streaming

IEEE 802.11n

Cameras, Audio, and Portable Devices

Classical Ethernet LAN

Sensor Data

(High rate)

- Optical navigation
- Autonomous systems

> 100 Mbit/s

< 5 Mbit/s

1FT C&DH System

Effectors

- Heaters
- Pumps
- Valves
- Motors

Docking Interface

Sensor Data

(Low rate)

- Star tracker
- IMU/SIGI
- Sun sensor
- Thrusters
- Temperature
- Humidity
- Oxygen, CO_2
- Flow rate
- Voltage

< 5 Mbit/s

< 10 Mbit/s

Distributed Processing

- RIU/DAU
- Star tracker
- Propulsion
- ECLSS

Onboard Displays

Direct audio/ video signals

Onboard Gateway

< 10 Mbit/s

Display/Audio Processing

Data Recorders

RF Equipment Amplifiers, switches

DTN Storage/ Processing

Command/ Telemetry Processing

Transponders (SDR) S-band, Ka-band, X-band, Proximity (UHF)

High Speed Serial

(P2P, minimal networking)

- Provides >1Gbit/s point-to-point or (possibly) networked messaging.
- Mostly related to off-board communication.

[1] Rakow, Glenn *Spacecraft Crew-Vehicle Avionics Networks and Communication Flow*

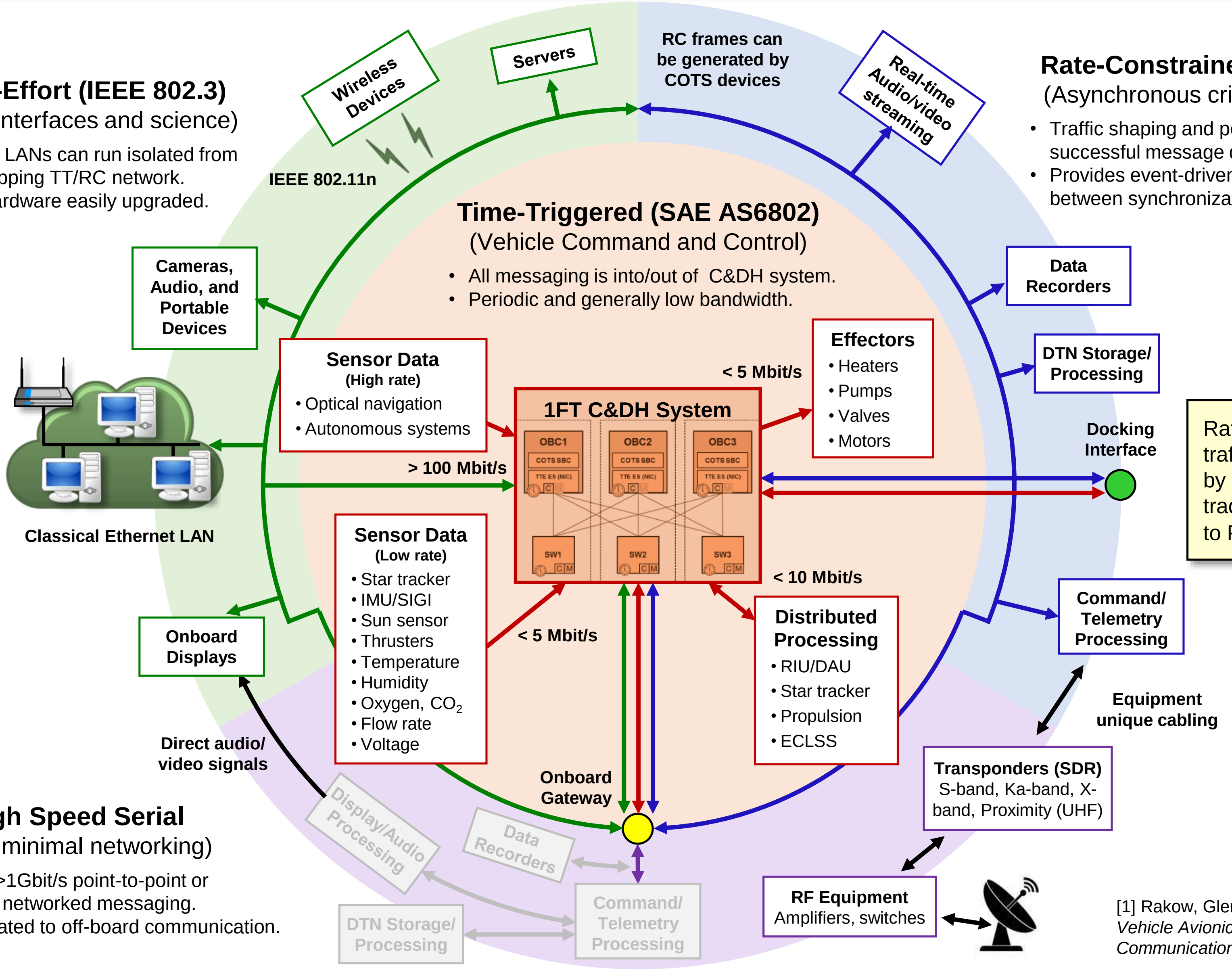
Notional Onboard Traffic Flow

Best-Effort (IEEE 802.3)
(Crew interfaces and science)

- Classical LANs can run isolated from or overlapping TT/RC network.
- COTS hardware easily upgraded.

Rate-Constrained (A664-p7)
(Asynchronous critical systems)

- Traffic shaping and policing ensures successful message delivery.
- Provides event-driven communication between synchronization domains.



Time-Triggered (SAE AS6802)
(Vehicle Command and Control)

- All messaging is into/out of C&DH system.
- Periodic and generally low bandwidth.

Rate-constrained traffic can be used by subsystems traditionally limited to P2P comm.

High Speed Serial
(P2P, minimal networking)

- Provides >1Gbit/s point-to-point or (possibly) networked messaging.
- Mostly related to off-board communication.

[1] Rakow, Glenn *Spacecraft Crew-Vehicle Avionics Networks and Communication Flow*

Notional Onboard Traffic Flow

Best-Effort (IEEE 802.3)

(Crew interfaces and science)

- Classical LANs can run isolated from or overlapping TT/RC network.
- COTS hardware easily upgraded.

Rate-Constrained (A664-p7)

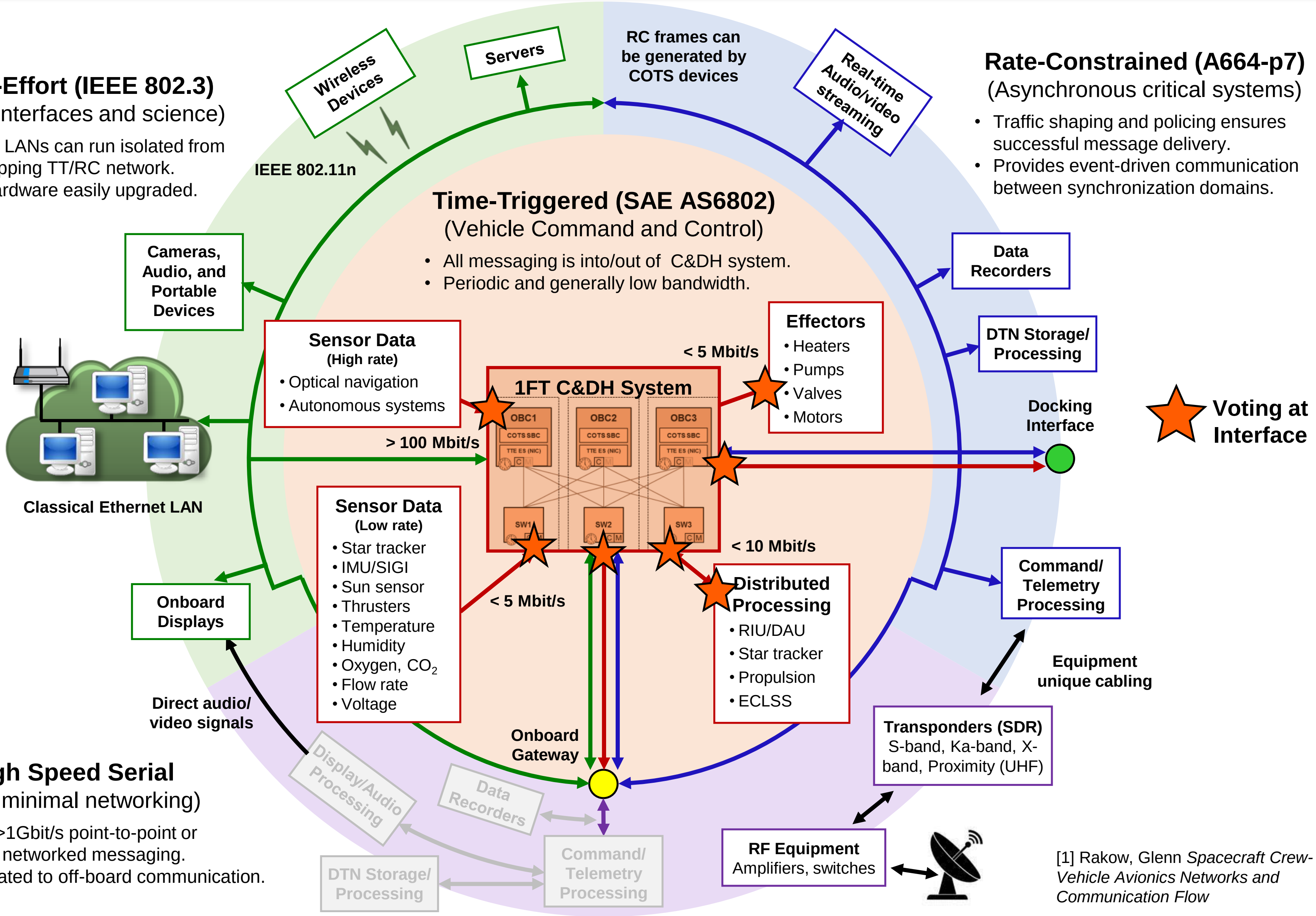
(Asynchronous critical systems)

- Traffic shaping and policing ensures successful message delivery.
- Provides event-driven communication between synchronization domains.

Time-Triggered (SAE AS6802)

(Vehicle Command and Control)

- All messaging is into/out of C&DH system.
- Periodic and generally low bandwidth.



Wireless Devices

Servers

RC frames can be generated by COTS devices

Real-time Audio/video streaming

IEEE 802.11n

Cameras, Audio, and Portable Devices

Classical Ethernet LAN

Sensor Data
(High rate)
- Optical navigation
- Autonomous systems

> 100 Mbit/s

< 5 Mbit/s

Effectors
- Heaters
- Pumps
- Valves
- Motors

1FT C&DH System

Data Recorders

DTN Storage/ Processing

Docking Interface

Voting at Interface

Sensor Data
(Low rate)
- Star tracker
- IMU/SIGI
- Sun sensor
- Thrusters
- Temperature
- Humidity
- Oxygen, CO_2
- Flow rate
- Voltage

< 5 Mbit/s

< 10 Mbit/s

Distributed Processing
- RIU/DAU
- Star tracker
- Propulsion
- ECLSS

Command/ Telemetry Processing

Equipment unique cabling

Onboard Displays

Direct audio/ video signals

Onboard Gateway

Display/Audio Processing

Data Recorders

Command/ Telemetry Processing

DTN Storage/ Processing

Transponders (SDR)
S-band, Ka-band, X-band, Proximity (UHF)

RF Equipment
Amplifiers, switches

High Speed Serial

(P2P, minimal networking)

- Provides >1Gbit/s point-to-point or (possibly) networked messaging.
- Mostly related to off-board communication.

[1] Rakow, Glenn *Spacecraft Crew-Vehicle Avionics Networks and Communication Flow*

Questions?