A Proposed Byzantine Fault-Tolerant Voting Architecture Using Time-Triggered Ethernet

Andrew Loveless, NASA Johnson Space Center
Christian Fidi, Stefan Wernitznigg, TTTech

SAE 2017 AeroTech Congress & Exhibition
Fort Worth, TX
26 – 28 September 2017

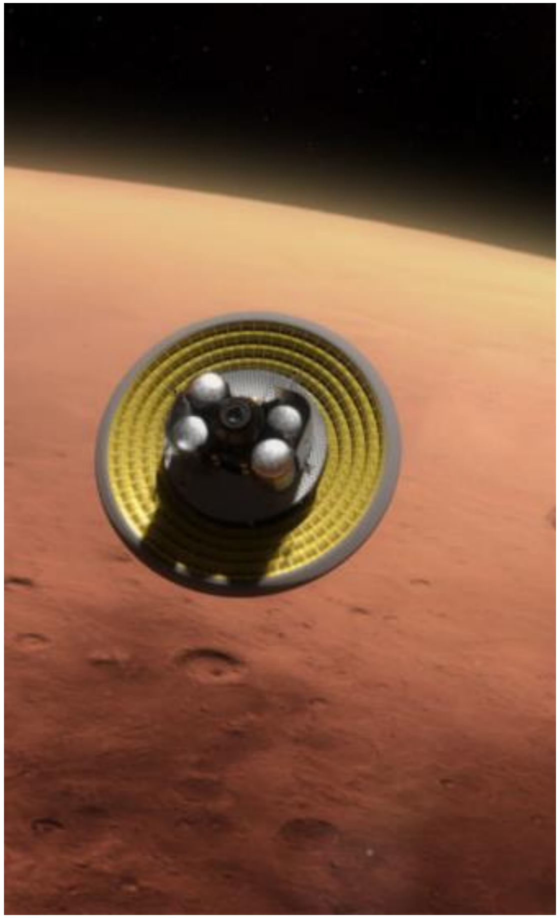
COTS in Manned Spacecraft

- **COTS technologies are attractive for use in human-rated spacecraft.**
  - Reduces development and upgrade costs.
  - Lowers the need for new design work.
  - Eliminates reliance on individual suppliers.
  - Leverages larger knowledge base.
  - Minimizes schedule risk.
- **Problem? Hard to meet the high reliability and fault tolerance requirements.**
  - E.g. 10^{-9} failures/hour in ultra-dependable systems.
  - E.g. Crit-1, “fly-through” fault tolerance.
  - Studies for Orion showed purely COTS designs would result in poor reliability and undue expense.

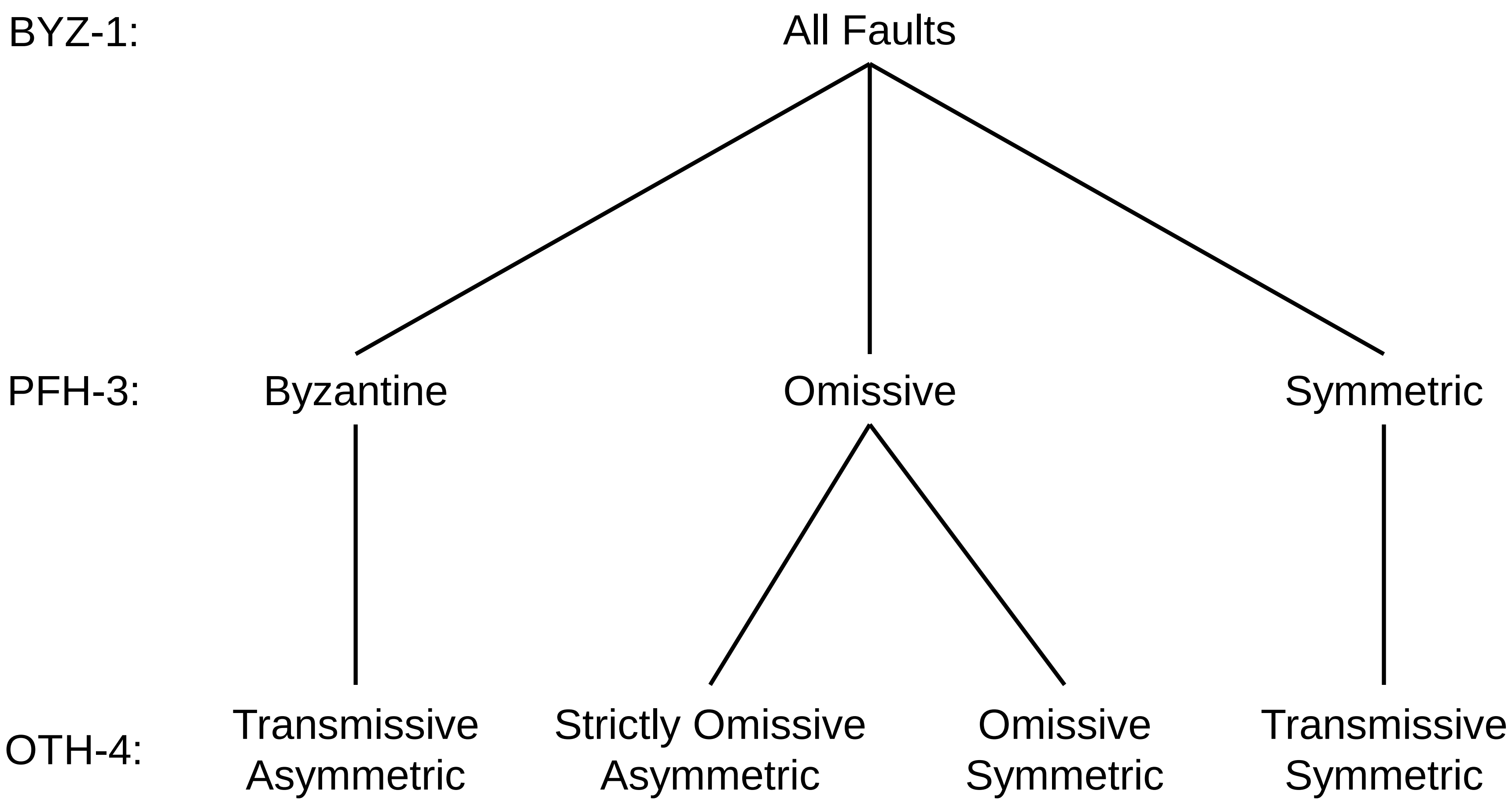
Often custom proprietary solutions are needed.

COTS in Manned Spacecraft (cont.)

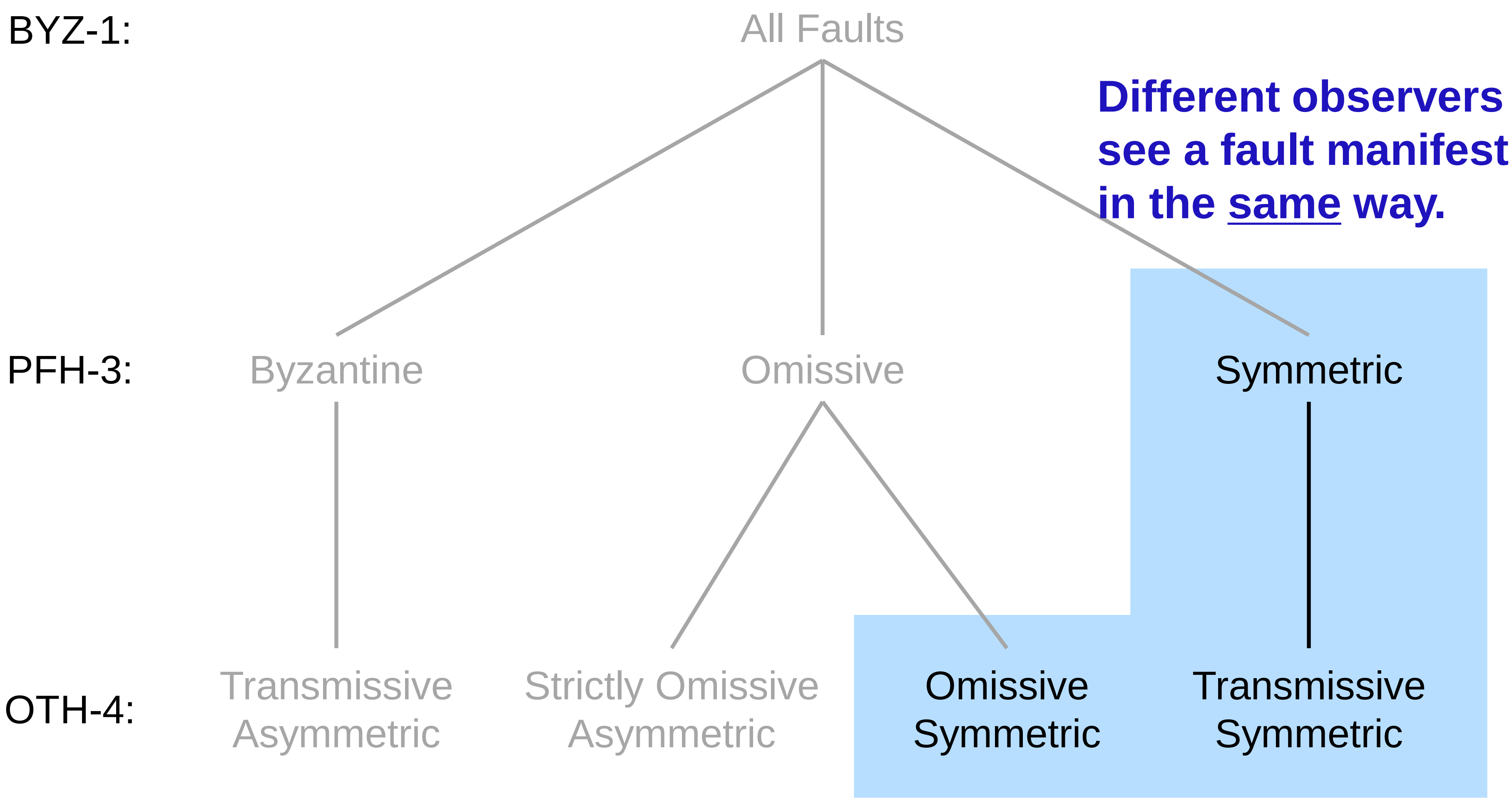
- **But the inclusion of COTS technologies is becoming more feasible.**
  - Greater availability of rad-tolerant components.
  - TMR (Maxwell SCS750), lock-step (ARM R5).
  - Ability to realize fault-containment regions.
  - Growing number of suppliers.
- **NASA's strategy for future spacecraft has heavily prioritized using COTS parts.**
  - Includes launchers, landers, etc.
- **Multiple projects have explored realizing safety-critical systems using COTS.**
  - Scalable Processor-Independent Design for Extended Reliability (SPIDER).
  - Heavy Lift Vehicle (HLV) Architecture Study.
  - Evolvable Mars Campaign (lander).



Fault Classifications

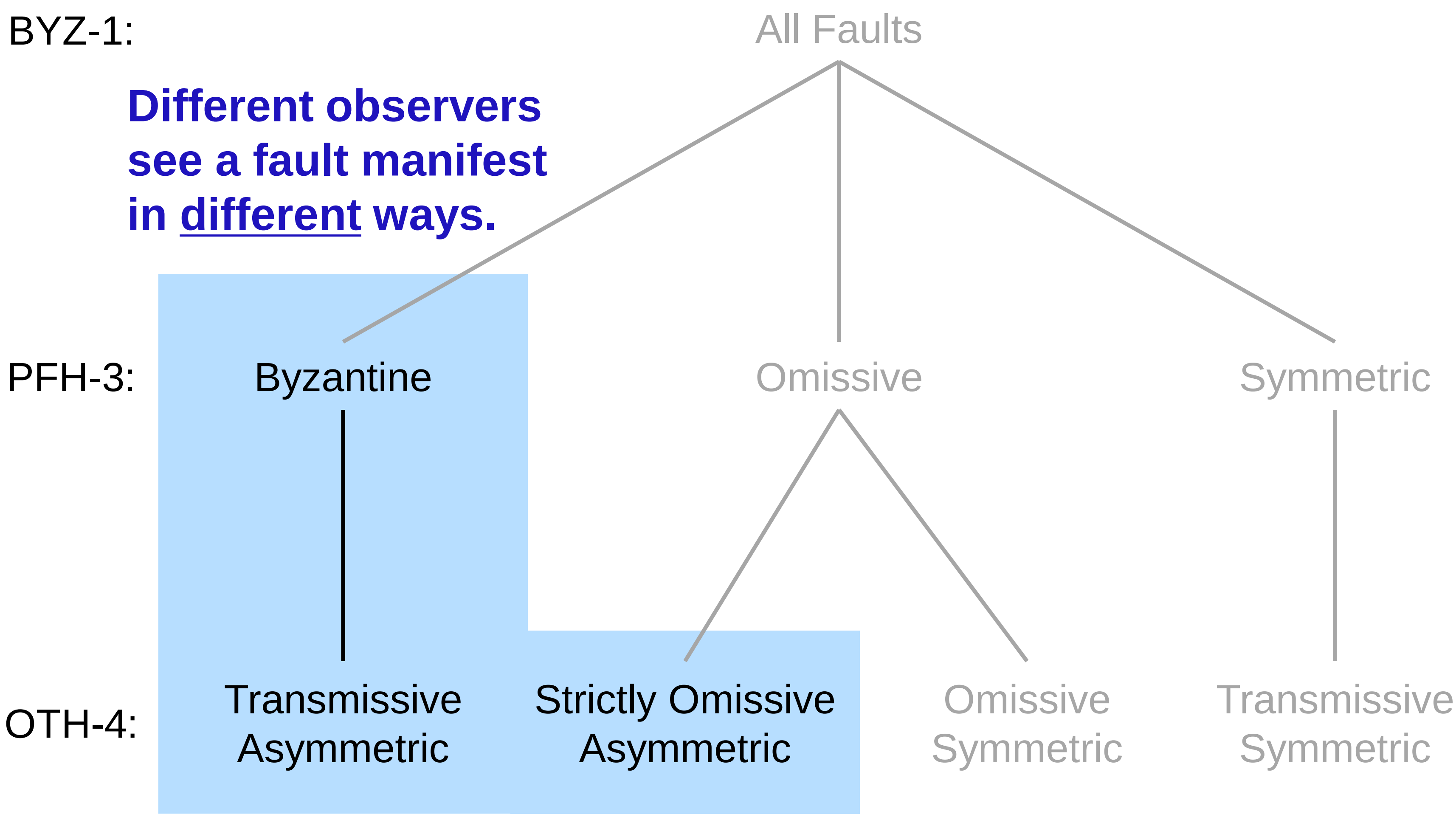


Fault Classifications (cont.)



Fault Classifications (cont.)

Different observers see a fault manifest in <u>different</u> ways.



Fault Classifications (cont.)

- **Manned spacecraft must tolerate Byzantine faults.**
  - Especially for dynamic mission phases with short time to effect.
  - Higher number of "all-or-none" events (e.g. deploy parachutes).
  - Failure could result in loss of life.

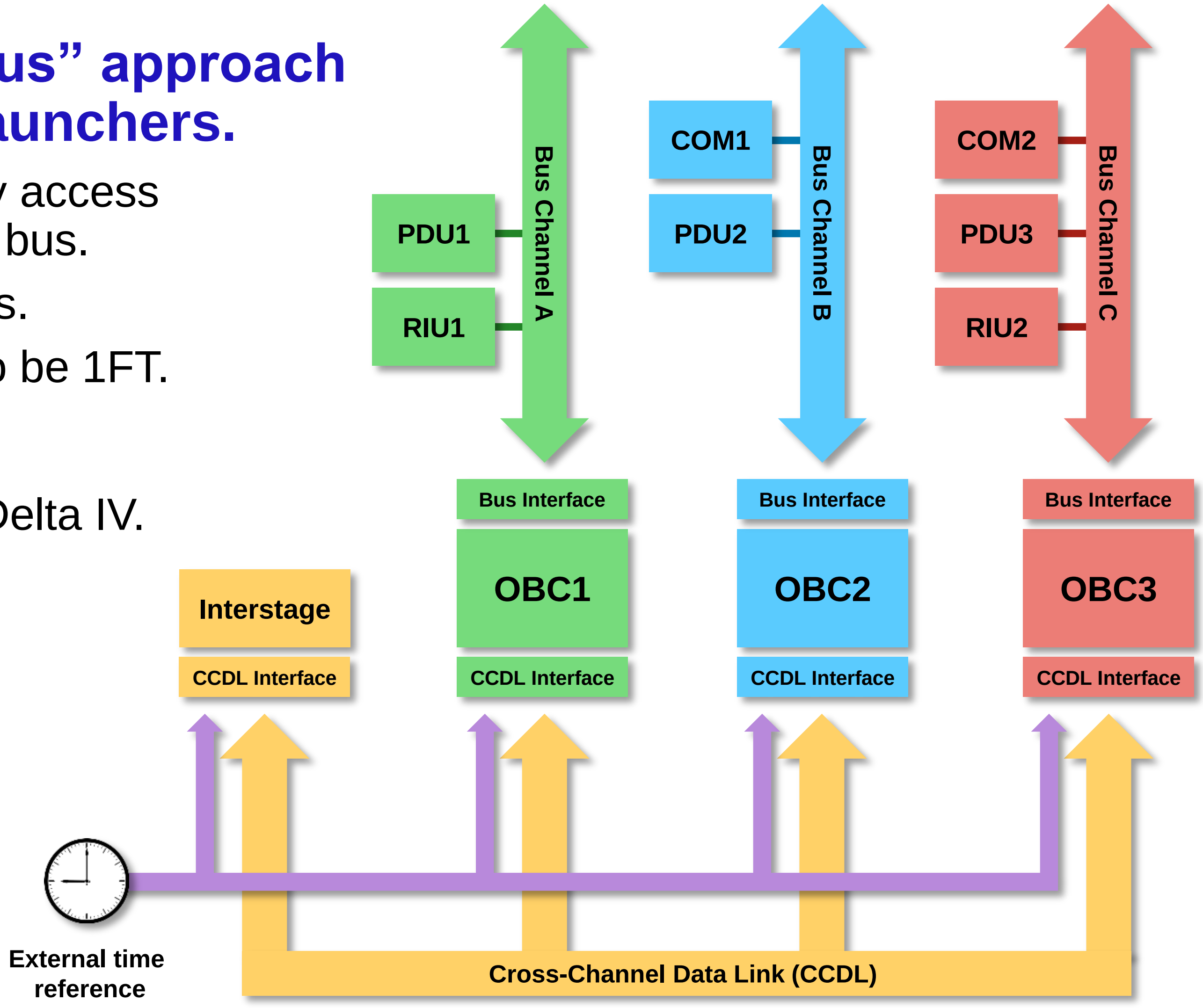
- **Byzantine faults are often not considered in satellites.**
  - Possibility is considered low enough to not warrant additional complexity.
  - Impacts of faults are less severe (e.g. not taking a picture).

Byzantine Faults

- **Byzantine faults can disrupt consensus among redundant processors.**
  - E.g. on internal state information.
  - E.g. on sensor data.
  - E.g. on diagnosis of system faults.
- **Occur at rates much > 10^{-9} failures/hour.**
  - Slightly-off-specification (SOS) hardware.
  - Stuck transmitter – different receivers can interpret a marginal signal differently.
  - Time base corruption – messages received slightly too early or too late.
- **Several architectural approaches for Byzantine-resilient systems.**
  - Hierarchical – e.g. SAFEbus, Orion VMCs.
  - Full exchange – e.g. Draper FTMP, SPIDER.

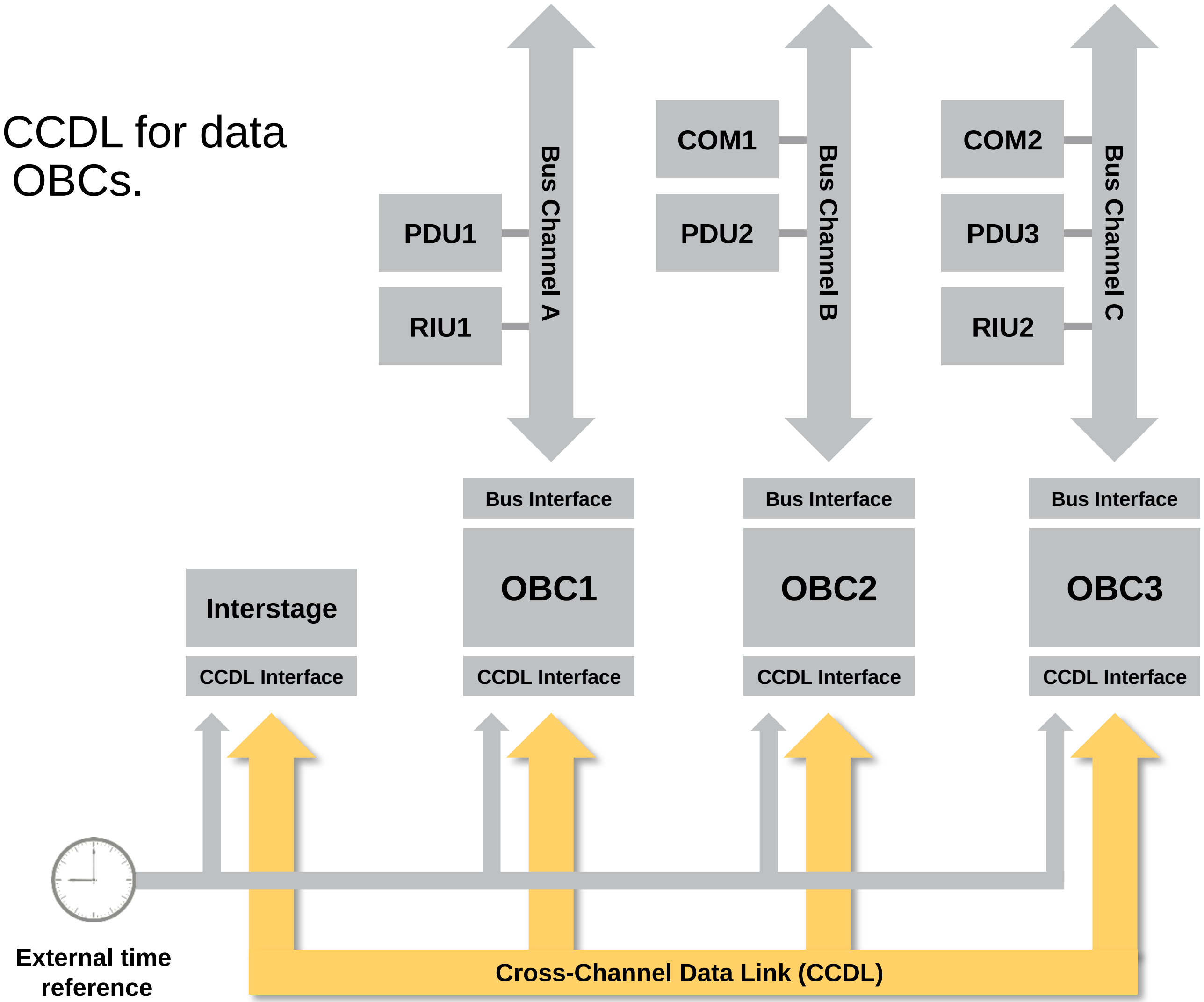
A Typical Approach

- **"Channelized bus" approach is common in launchers.**
  - Each OBC can only access devices on its local bus.
  - Uses full exchanges.
  - Usually designed to be 1FT.
- **Examples:**
  - X-38 CRV, Ares I, Delta IV.
- **Shortcomings?**



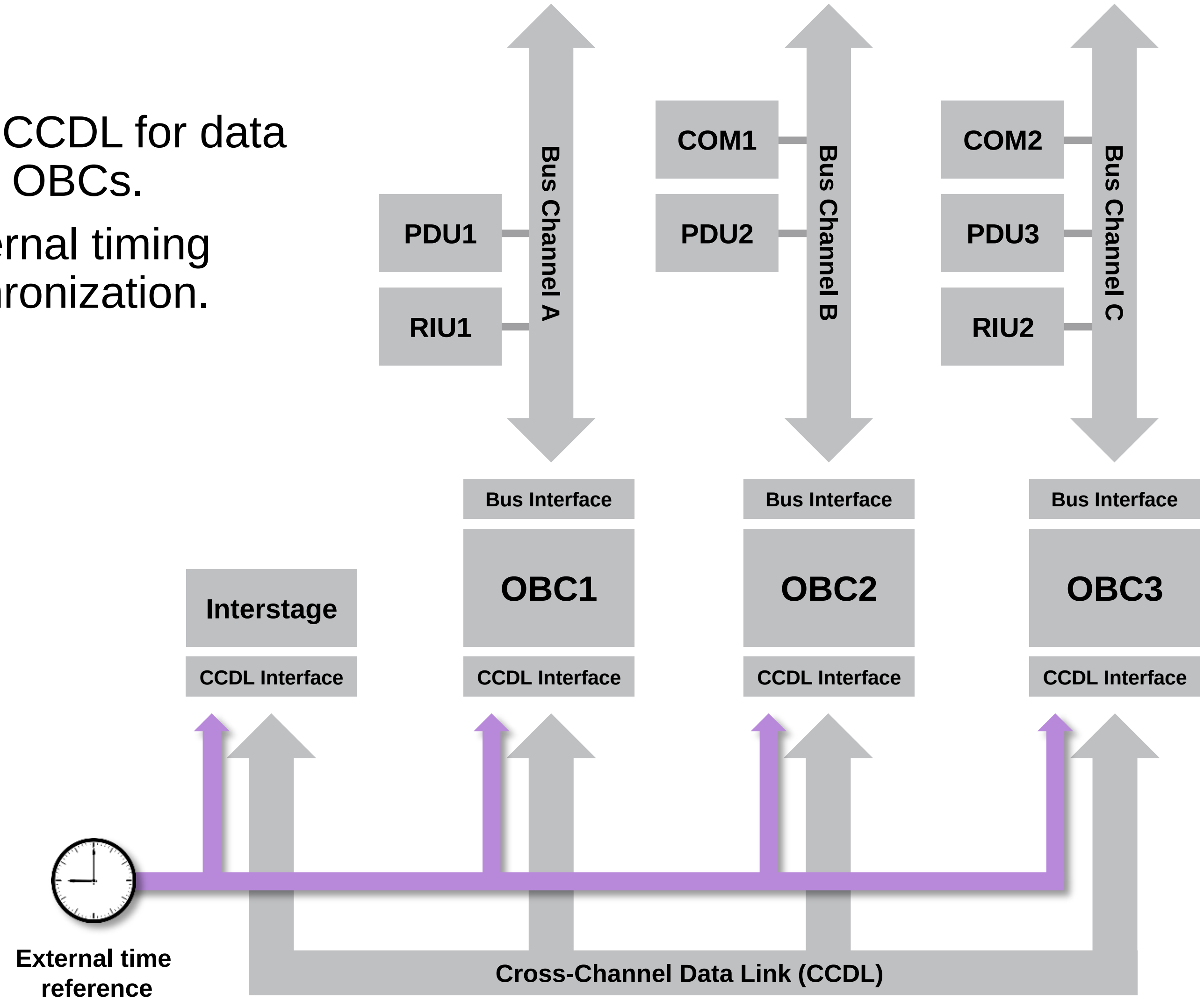
A Typical Approach (cont.)

- **Shortcomings?**
  1. Requires separate CCDL for data exchange between OBCs.



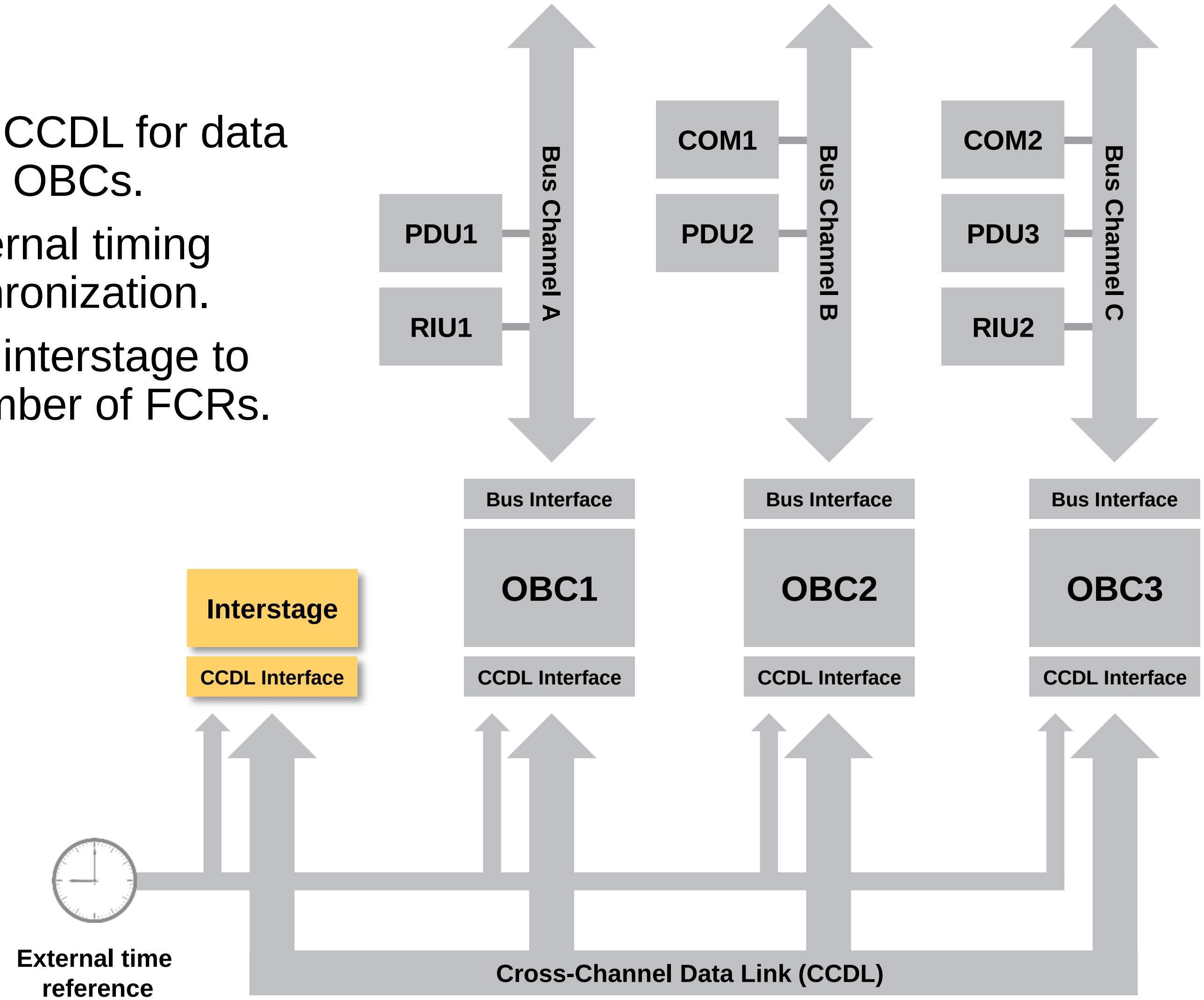
A Typical Approach (cont.)

- **Shortcomings?**
  1. Requires separate CCDL for data exchange between OBCs.
  2. Often requires external timing hardware for synchronization.



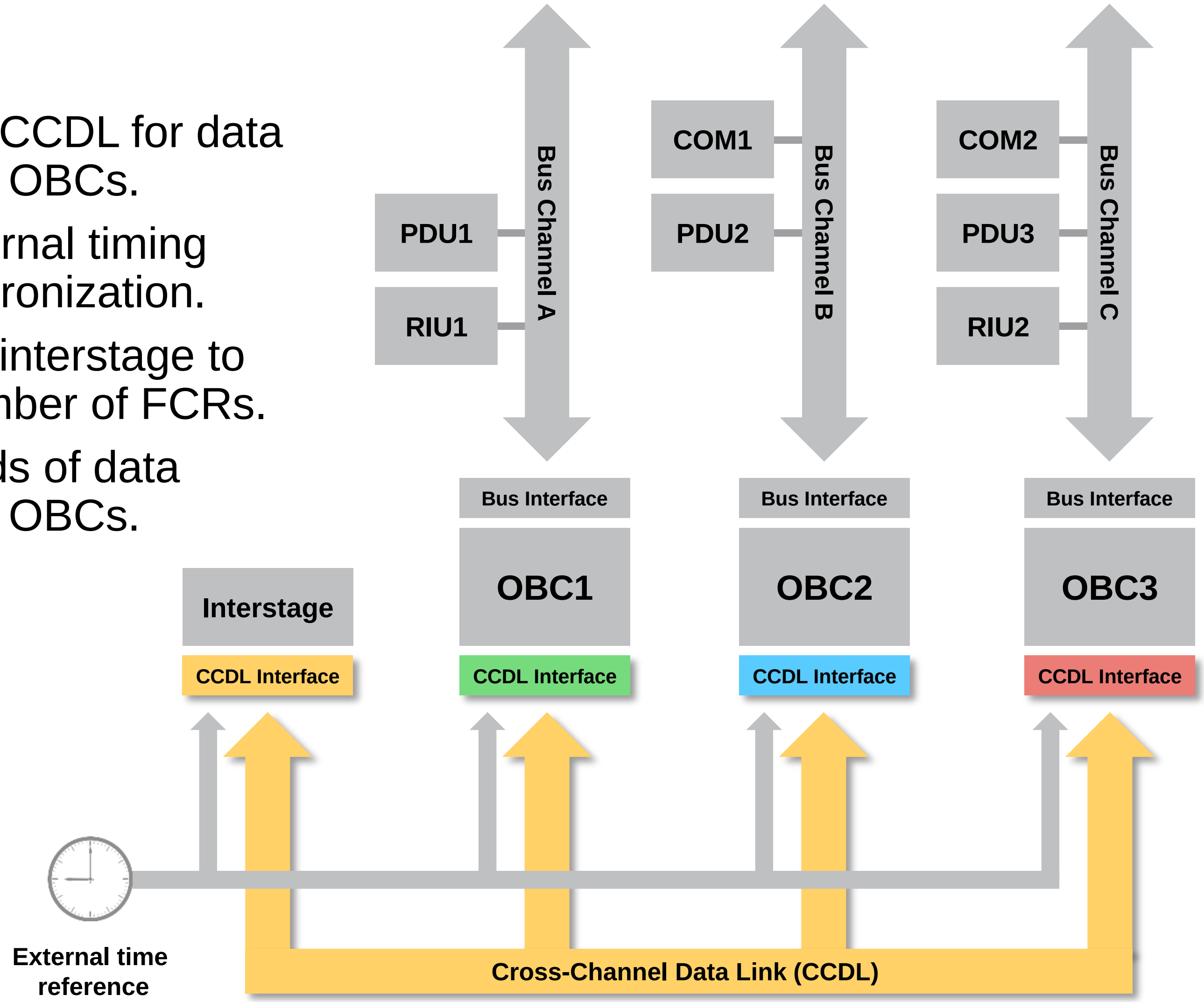
A Typical Approach (cont.)

- **Shortcomings?**
  1. Requires separate CCDL for data exchange between OBCs.
  2. Often requires external timing hardware for synchronization.
  3. Requires separate interstage to meet minimum number of FCRs.



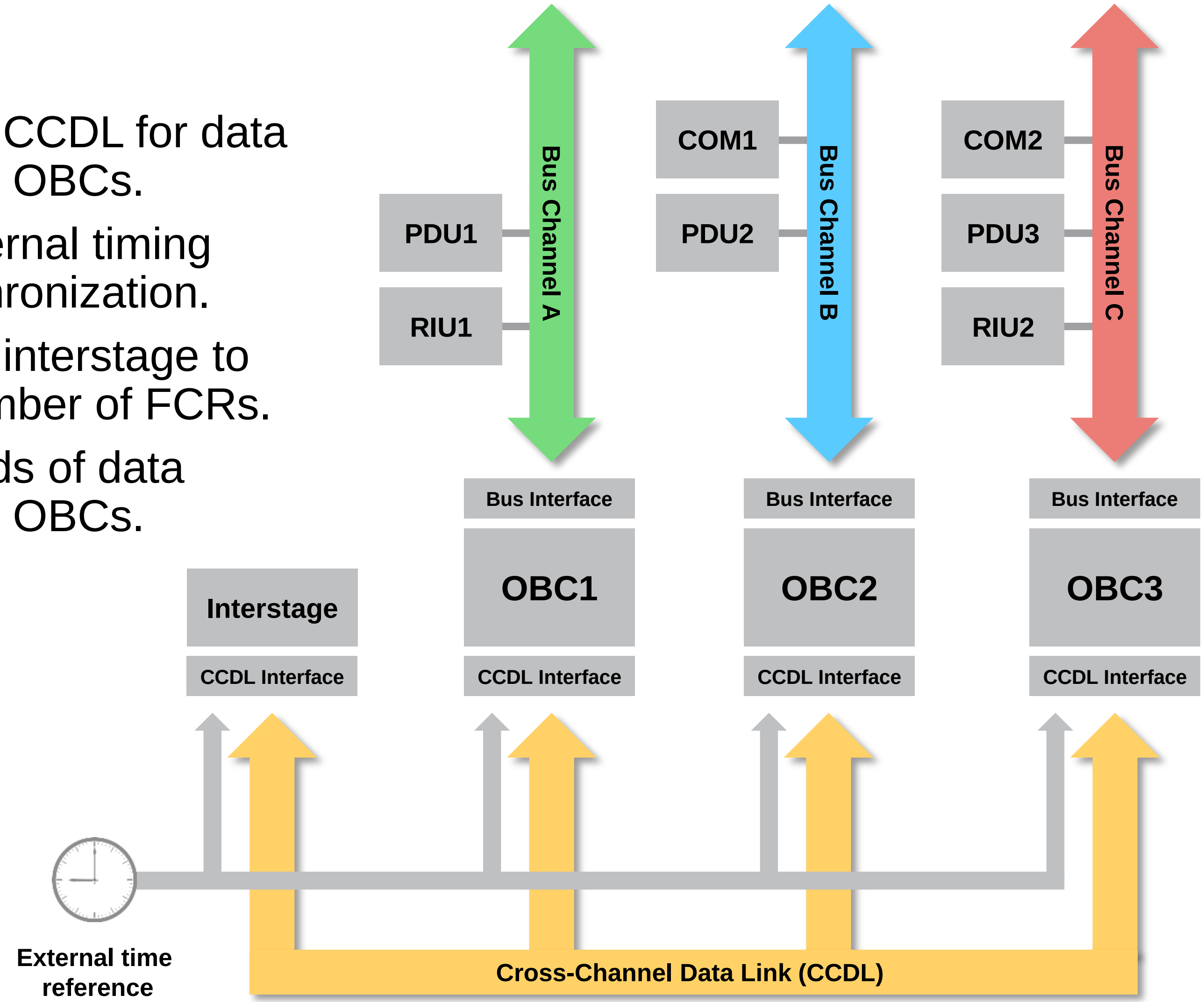
A Typical Approach (cont.)

- **Shortcomings?**
  1. Requires separate CCDL for data exchange between OBCs.
  2. Often requires external timing hardware for synchronization.
  3. Requires separate interstage to meet minimum number of FCRs.
  4. Requires two rounds of data exchange between OBCs.



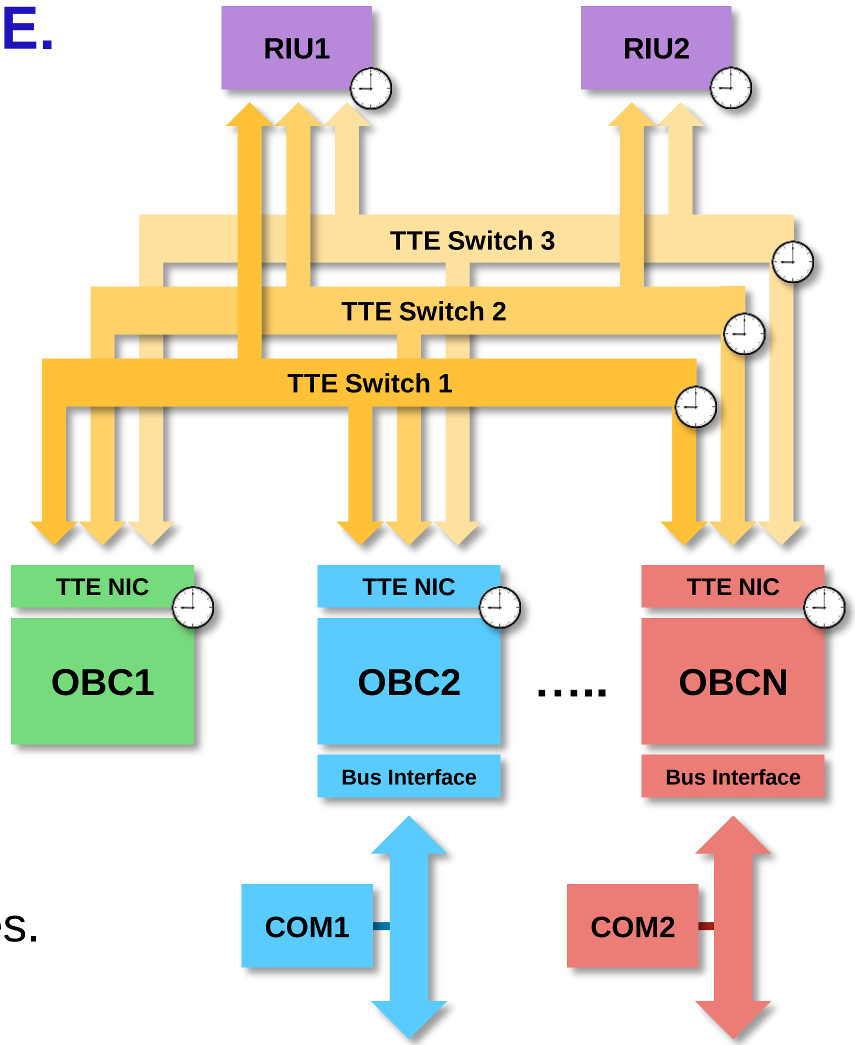
A Typical Approach (cont.)

- **Shortcomings?**
  1. Requires separate CCDL for data exchange between OBCs.
  2. Often requires external timing hardware for synchronization.
  3. Requires separate interstage to meet minimum number of FCRs.
  4. Requires two rounds of data exchange between OBCs.
  5. Bandwidth limited.



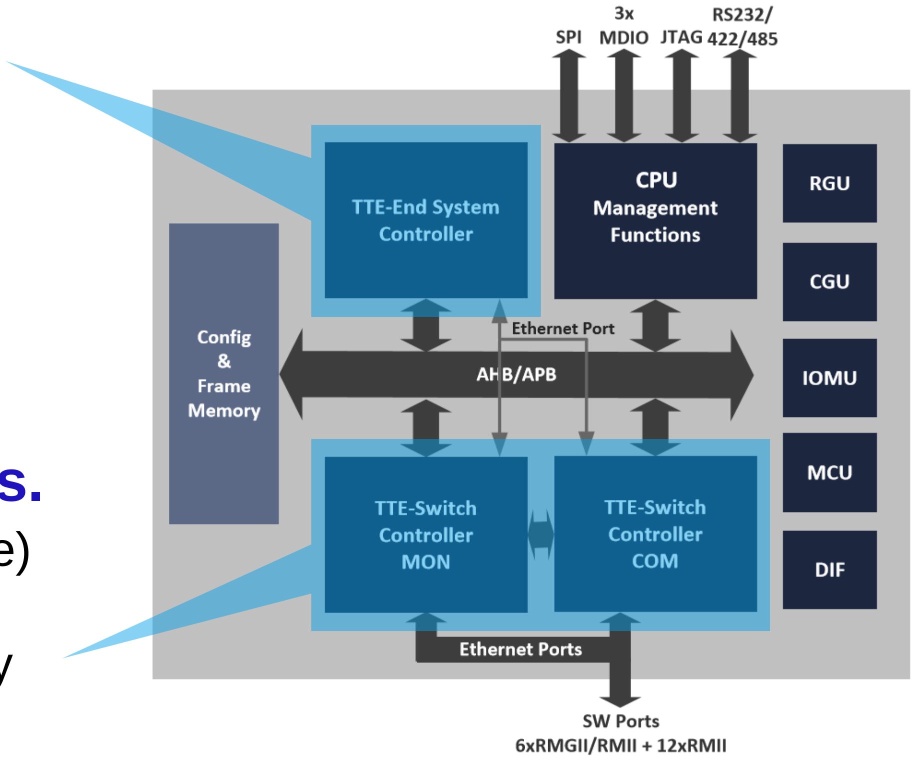
An Approach Using TTE

- **1FT "switched voter" using TTE.**
  - Requires only 3 full processors.
  - Requires 2-3 redundant switches.
  - Devices can connect to OBCs directly or via TTE network.
  - *Assumes minimum number of SMs and CMs are present for sync.*
- **TTE network used for data distribution and sync.**
  - Eliminates need for separate CCDL.
  - Eliminates need for timing hardware.
  - Bandwidth up to 1 Gbit/s.
- **Switches act as interstages.**
  - Messages reflected to/from the switches.
  - Eliminates need for fourth processor.



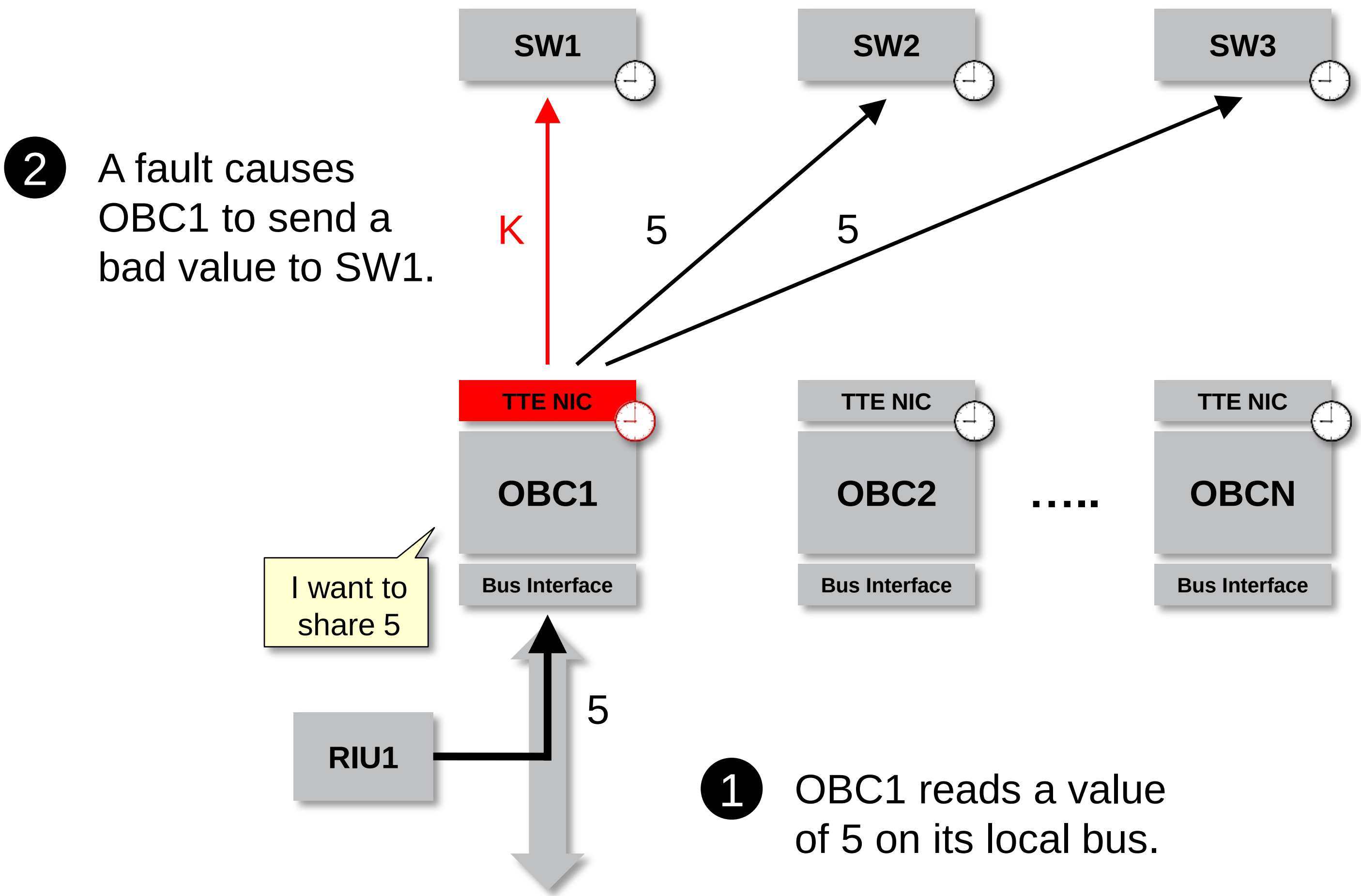
Failure Assumptions

- **End systems may be subject to Byzantine failures.**
  - May send arbitrary messages.
  - May transmit at any point in time.
  - May send different messages to different switches.
- **Switches are restricted to inconsistent omission failures.**
  - May not create (nor modify to produce) a new "valid" message.
  - May drop or fail to receive an arbitrary number of messages.
  - May relay messages asymmetrically – some receivers may not get data.
  - Acts as a "trusted sender".

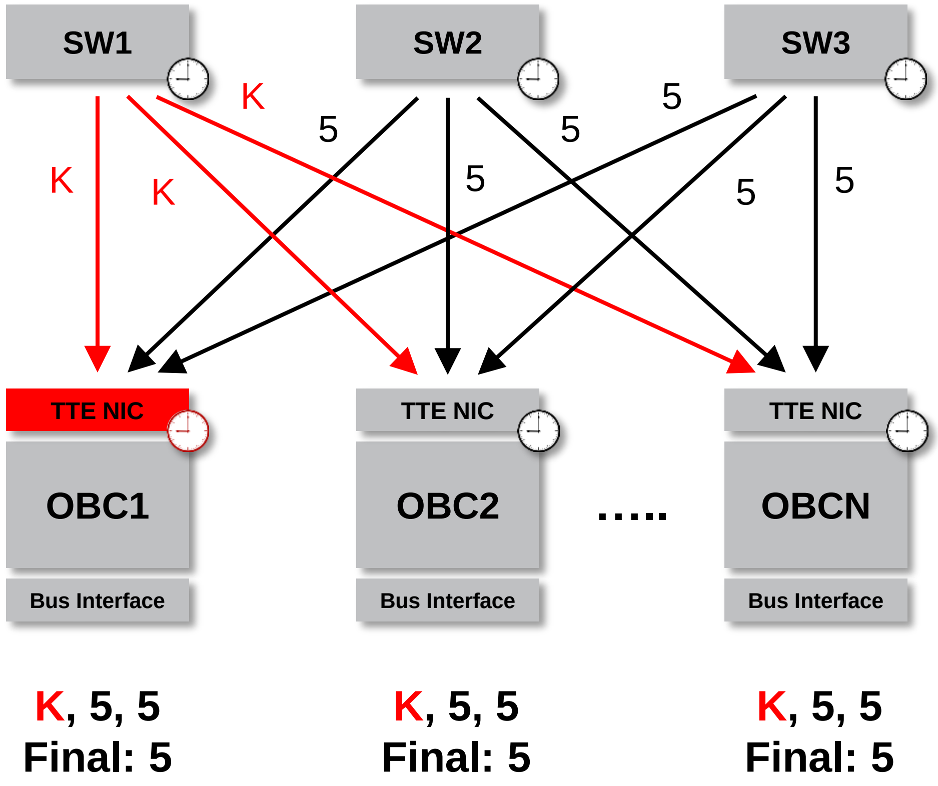


Fault propagation from switches theoretically requires dual-correlated simultaneous faults.
→ $10^{-6} \times 10^{-6} = \sim 10^{-12}$ failures/hour

Agreement on Local Data



Agreement on Local Data (cont.)



3 Each switch relays the data to all OBCs.

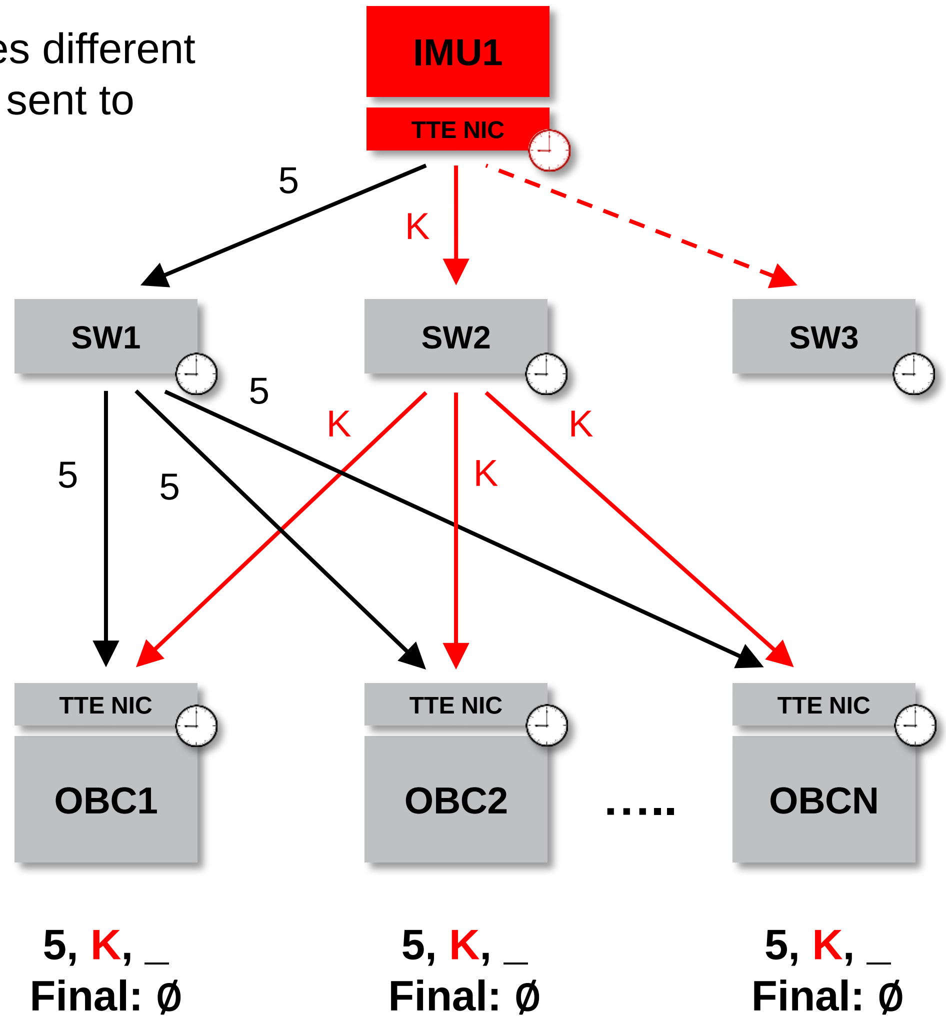
4 Each OBC votes the values sent from the switches.

Absent data <u>is not</u> included in the vote.

! Vote could be implemented in TTE NIC or in software on the OBCs.

Agreement on External Data

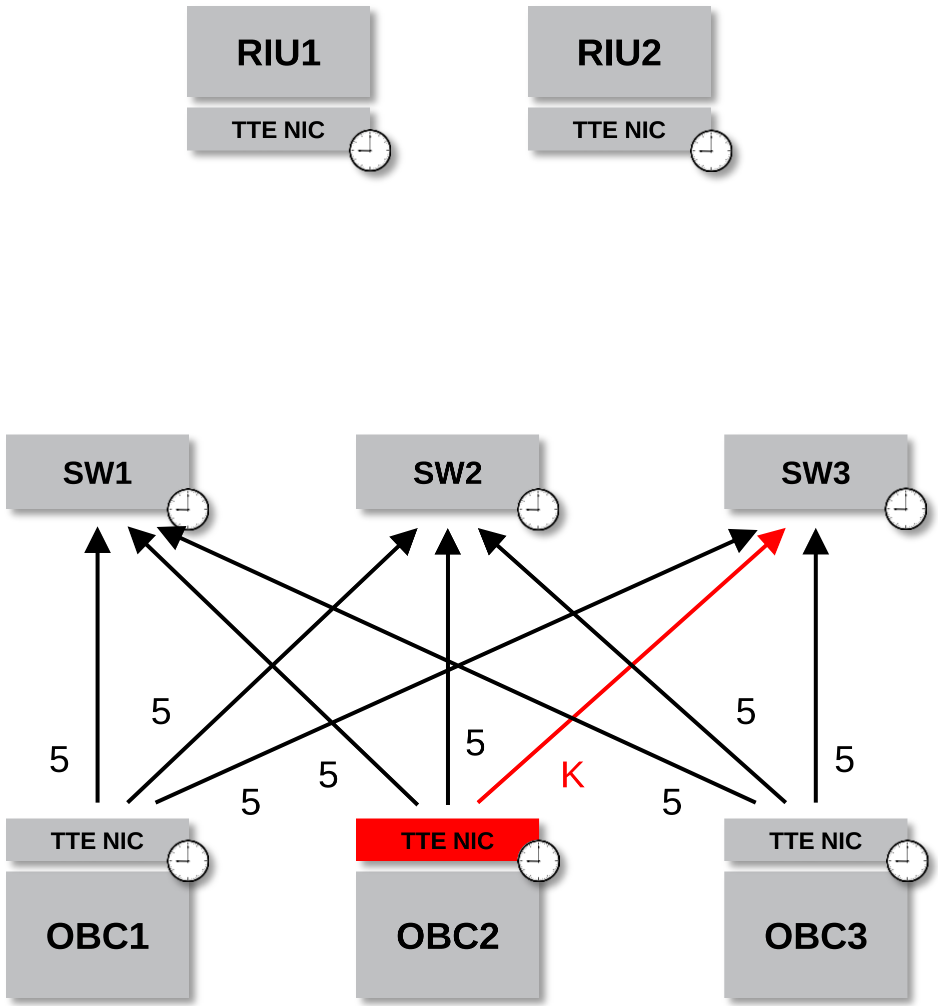
1. A fault causes different values to be sent to each switch.



2. Each switch relays the data to all OBCs.

3. All OBCs agree that no majority is found.

Commanding



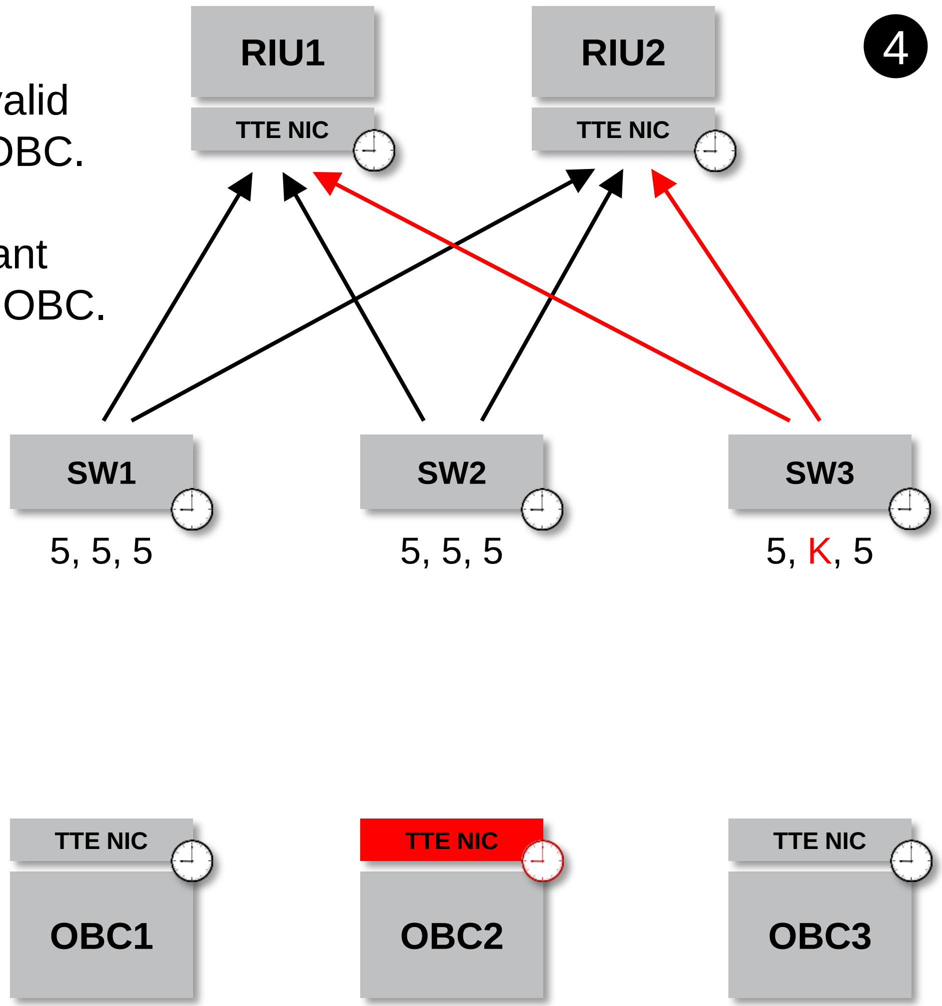
1 A fault causes OBC2 to send a bad value to SW3.

Commanding (cont.)

3 Each RIU either:
1. Accepts the first valid value from each OBC.
or
2. Votes the redundant values from each OBC.

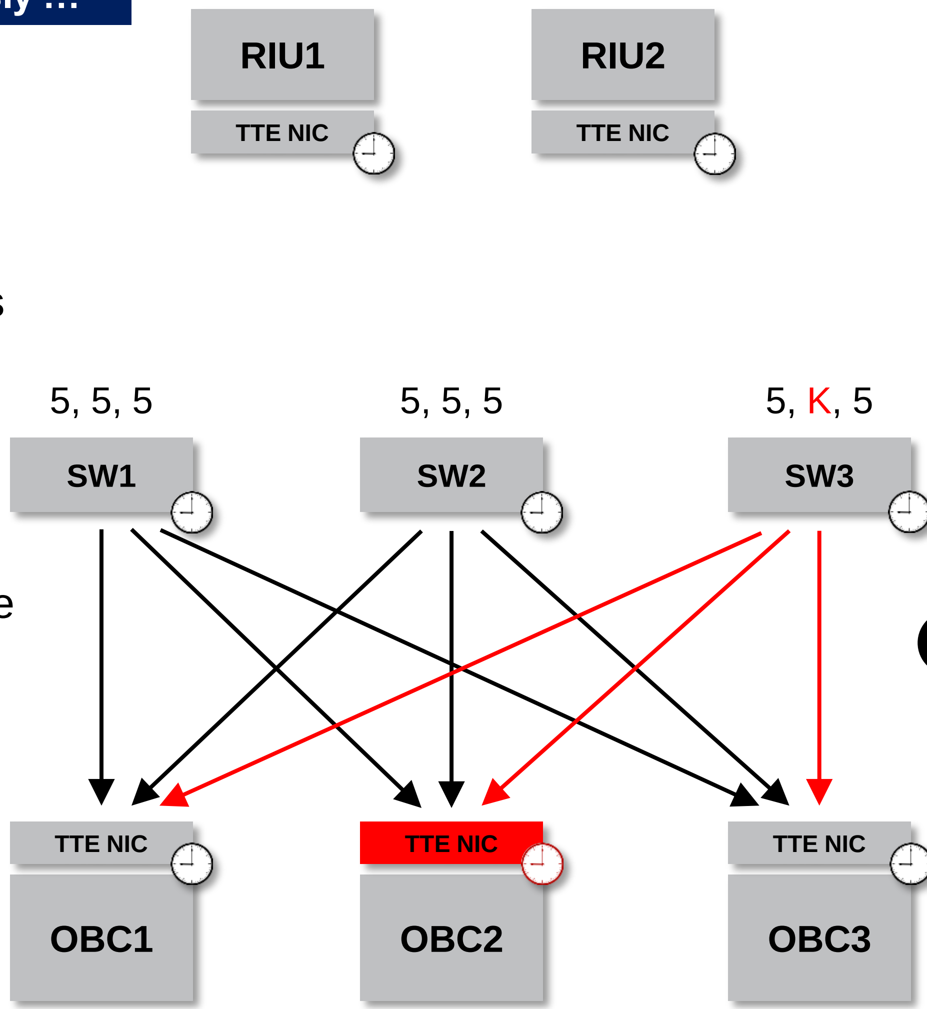
4 Each RIU votes the values accepted in Step 3.
Absent data is included in the vote.

2 Each switch relays the data from each OBC to all RIUs.



Commanding (cont.)

Happening Simultaneously …



5 Each switch reflects the original data back to all OBCs.

6 Each OBC votes the redundant values from each OBC.
Absent data <u>is not</u> included in the vote.

7 Each OBC votes the results from Step 6 to diagnose faulty OBCs.
Absent data <u>is</u> included in the vote.

Questions?