EVOLUTIONS OF SPACEWIRE PROTOCOLS FOR DETERMINISTIC DATA DELIVERY

Krzysztof Romanowski, **Piotr Tyczka** – ITTI, Poznań, Poland
David Jameux – ESTEC, European Space Agency, Noordwijk, The Netherlands

9th International SpaceWire and SpaceFibre Conference
Pisa, 17-19 October 2022

OUTLINE

- Background
- Time coordination
- Intermediate protocol
- FDIR mechanisms
- Simulation
- Conclusion

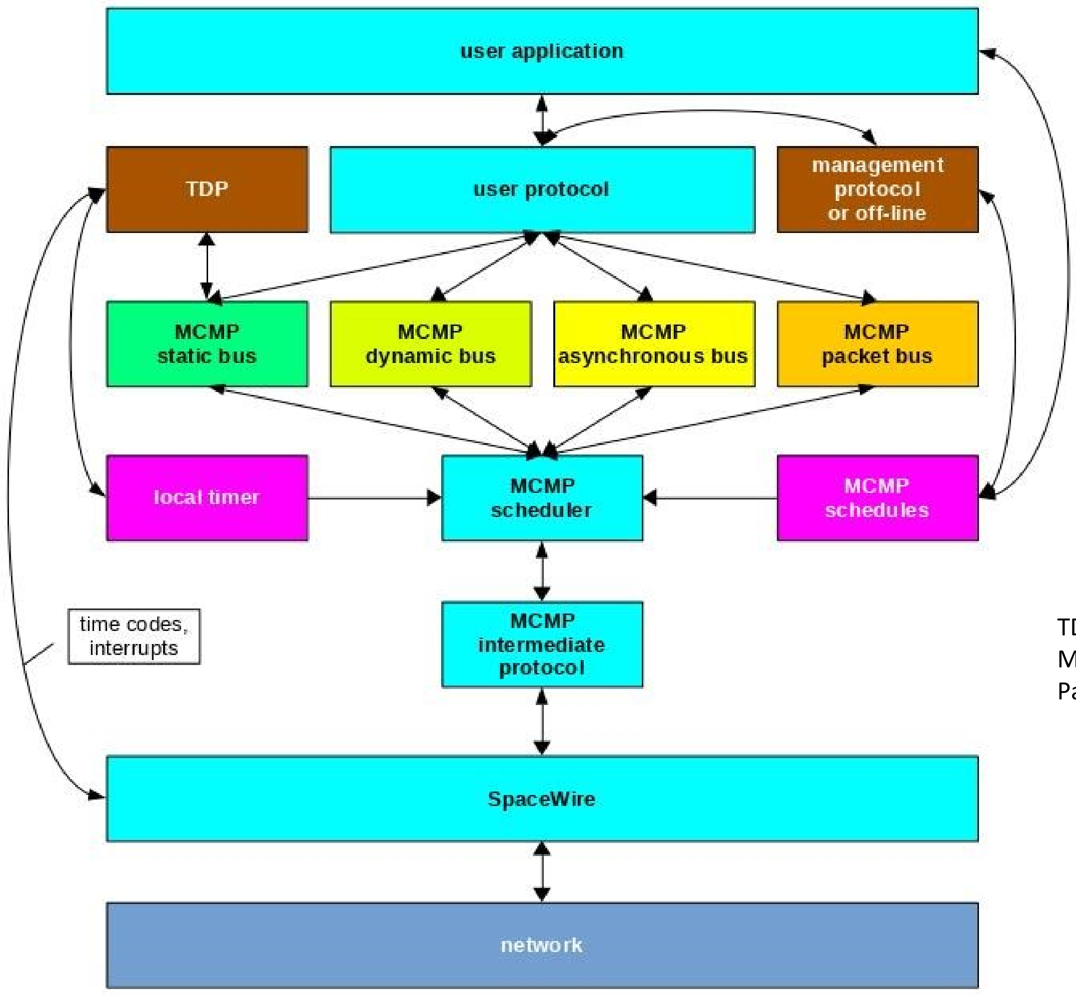
BACKGROUND

- Why deterministic data delivery on SpaceWire?
  - Single network for payload data as well as command and control
- Related protocols developed or proposed in the past:
  - SpaceWire-RT / SpaceWire-T (University of Dundee, drafts 2008, 2009)
  - SpaceWire-D (University of Dundee, STAR-Dundee, 2010-2017)
  - STP-ISS (SUAI, since 2014)
- SpaceWire-D was adopted as starting point in the SpaceDet project
  - Uses time division multiplexing of network resources and scheduling
  - Runs over existing SpW networks with no modifications to existing routing switches
  - Uses RMAP to provide basic communication mechanism of transactions reading/writing from/to memory of a remote target node
  - Splits time into time slots, defined/controlled by SpW time codes or local timer with SpW time codes for synchronization
  - Provides four principal services – buses: static (fully deterministic), dynamic, asynchronous, packet.

ALTERNATIVE DESIGN SOLUTIONS IN *SPACEDET*

- *SpaceDet* project proposed MCMP (Mixed Criticality Message Passing protocol), based on SpaceWire-D, with alternative design solutions
- Time slots:
  - Instead of relying on time codes (which limits their number to 64 per epoch), use local timers synchronized by the SpaceWire Time Distribution Protocol (Aeroflex Gaisler, 2014)
    - Benefits: flexibility in the structure of the epoch: large numbers of time slots possible; boundaries can be defined with high precision; delay and jitter can be mitigated to the level of single bit transmission periods (Sakthivel et al., 2014).
- Intermediate protocol (between the layer of the virtual buses and SpaceWire) proposed: MRAP (MCMP Register Access Protocol):
  - Instead of using plain RMAP, use a dedicated protocol (own protocol ID) based on RMAP, with modifications to the header
    - Benefits: differentiation from RMAP – leaves RMAP code and address space unaffected; optimization of packet header usage resulting in flexible time slot structure and packet sizes.
- FDIR – additional mechanisms:
  - SpW interrupts, guard intervals (slot margins)
    - Benefits: improve the fault isolation capabilities.

MCMP PROTOCOL STACK



TDP = Time Distribution Protocol
MCMP = Mixed Criticality Message Passing protocol

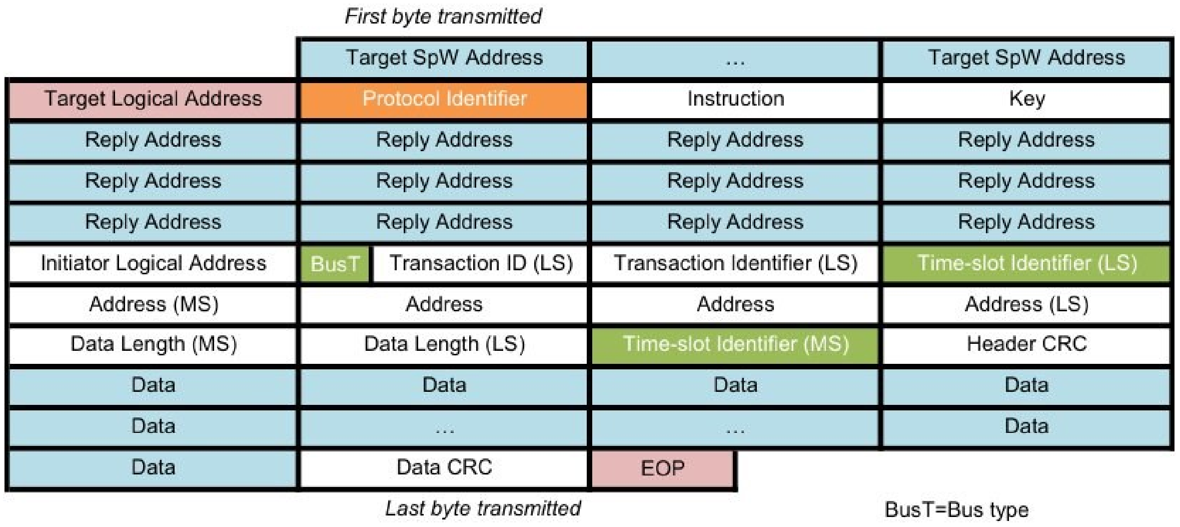
TIME COORDINATION PRINCIPLES

- Local timers: mandatory at master nodes (initiators of transmissions on the bus), optional at slave nodes (targets of transmissions on the bus). (Note: the same node can be master on one bus and slave on another.)
- Master nodes responsible for maintaining the time slot sequence and following the schedules.
- Slave nodes responsible for replying within known bounded time.
- Beginning and length of the epoch and the time slot sequence: same at all master nodes. Time slots defined by time related to epoch beginning, rather than time codes.
- Local time synchronization: via SpW-TDP (TDSP).
- Master node cannot initiate transactions unless its local timer is synchronized.
- At least the first time slot in an epoch is dedicated to SpW-TDP transactions.
- Schedule tables at different master nodes may be different provided they do not lead to resource demand conflicts.

INTERMEDIATE PROTOCOL

- Two transaction models:
  - Primary: *acknowledged*: commands and replies.
  - Secondary: *unacknowledged*: one-way communication (i.e. actually not a transaction). No replies needed, or replies provided in separate communication.
- Intermediate protocol, MRAP (MCMP Register Access Protocol):
  - User data encapsulated in a format based on RMAP but with dedicated protocol ID and customized header fields.
  - Customization based on the dedicated status of the header format: “memory” range does not need to refer to memory, data length can be limited, etc., which can leave as much as 24 bits for MCMP-related information.
- Optional: alternative raw protocol for special use cases:
  - No encapsulation: user data unit transmitted as is in on appropriate bus, immediately following the target address fields. User application is responsible for invoking the appropriate virtual bus service, but after sending by the master node, there is no possibility of verifying whether the packet is travelling or arrives in the proper bus.

PACKET FORMAT EXAMPLES



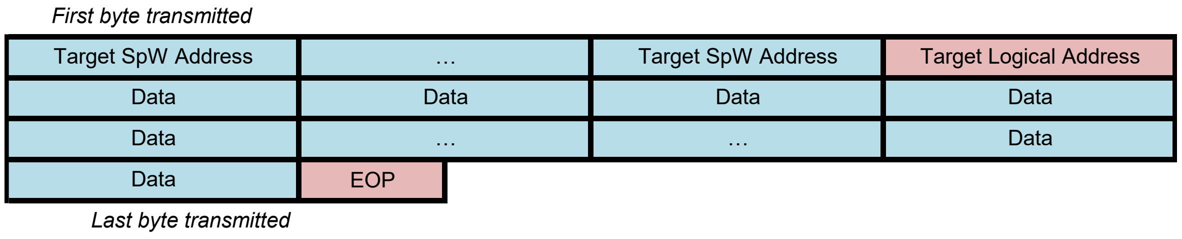
First byte transmitted

	Target SpW Address	…	Target SpW Address
Target Logical Address	Protocol Identifier	Instruction	Key
Reply Address	Reply Address	Reply Address	Reply Address
Reply Address	Reply Address	Reply Address	Reply Address
Reply Address	Reply Address	Reply Address	Reply Address
Initiator Logical Address	BusT Transaction ID (LS)	Transaction Identifier (LS)	Time-slot Identifier (LS)
Address (MS)	Address	Address	Address (LS)
Data Length (MS)	Data Length (LS)	Time-slot Identifier (MS)	Header CRC
Data	Data	Data	Data
Data	…	…	Data
Data	Data CRC	EOP	

Last byte transmitted

BusT=Bus type

MRAP write command header. ‘Protocol Identifier’: dedicated. ‘Bus Type’: 2 bits (selects S/D/A/P). ‘Time-slot Identifier’: 16 bits.



First byte transmitted

Target SpW Address	…	Target SpW Address	Target Logical Address
Data	Data	Data	Data
Data	…	…	Data
Data	EOP		

Last byte transmitted

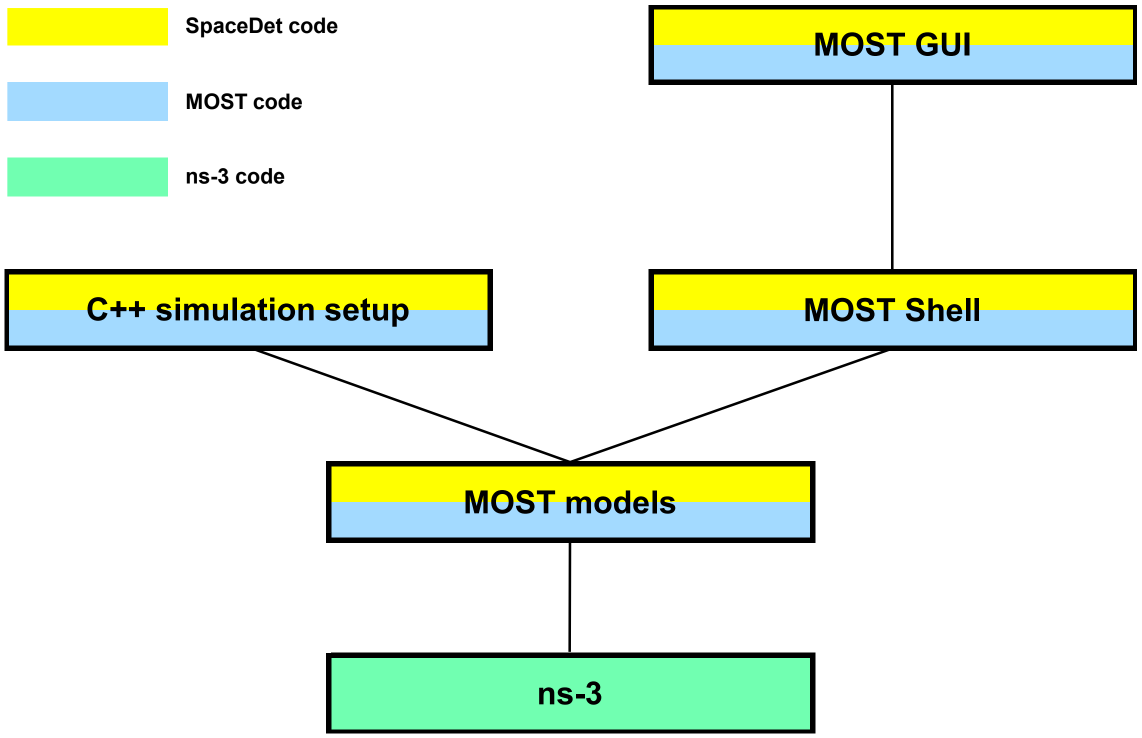
Optional: unencapsulated raw packet

FDIR MECHANISMS

- SpaceWire distributed interrupts
  - Intended to make receiving switches drop packets
  - Triggered by (master) nodes when slot boundary violation detected (transmission in progress over slot boundary)
  - May be sent proactively by a master node (e.g. OBC) at end of every slot
  - Requirement: no multislots or the same multislot organization at all nodes
- Guard (silent) intervals (slot margins)
  - At the beginning and the end of a slot
- This is in addition to mechanisms already known in SpaceWire-D:
  - time-code error detection (early, late, missing),
  - SpaceWire errors (link failures, EEP reception),
  - error status signalled in RMAP (MRAP) replies,
  - late or missing RMAP (MRAP) replies.

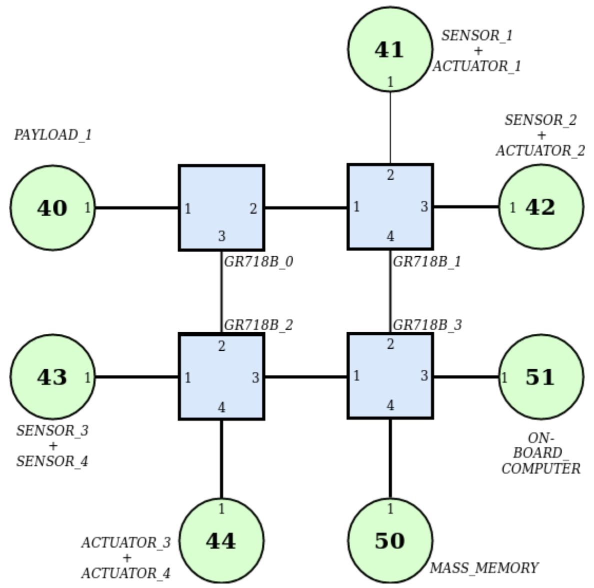
SIMULATION: PLATFORM

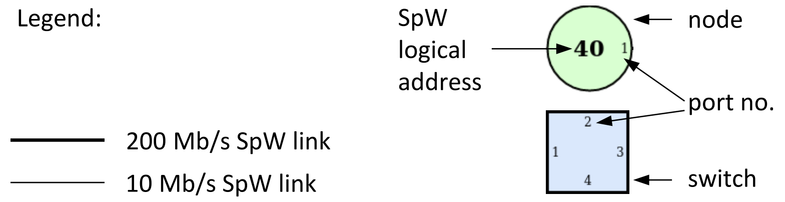
- Based on *MOST-X* and *ns-3*
- Additions to SpW codec, RMAP, and SpW-D support
  - distributed interrupt handling and related packet truncation capability
  - guard intervals (slot margins)
  - MCMP/MRAP protocols



SIMULATION: SPACEWIRE NETWORK MODEL

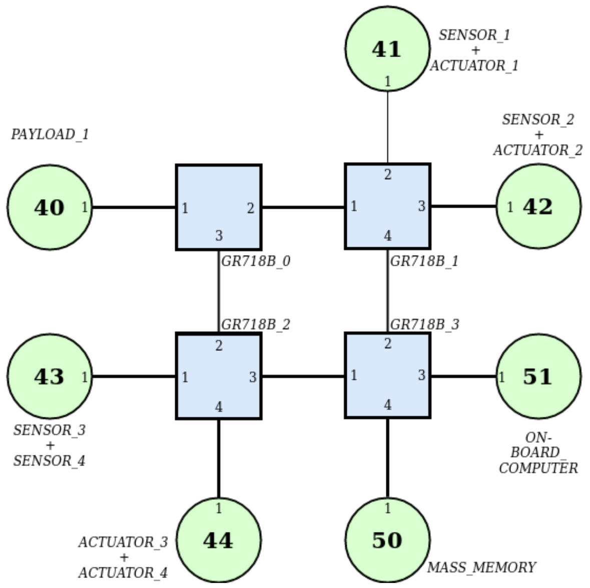
- 7 nodes (1 port used per node)
  - 5 "peripheral" nodes
    - hosting 5 sensors and 4 actuators
  - 2 "central" nodes
    - Mass memory (MM)
    - On-board computer (OBC)
- 4 switches (max. 4 ports used per switch)
  - Marked "GR718B", since they can model a feature of Cobham Gaisler GR718B of truncating packets on reception of distributed interrupts (they do NOT model other GR718B-specific features)





SIMULATION: SPACEWIRE TRAFFIC

- 9 data streams
  - 4 peripheral → central
    - 1 x sensor → MM: using *write* commands
    - 3 x sensor → OBC: using *write* commands
  - 5 central → peripheral
    - 1 x OBC → sensor: using *read* commands
    - 4 x OBC → actuator: using *write* commands
- Nearest SAVOIR OCS Communication Classes:
  - 1 – low frequency, small/medium data size, non-time critical (1 stream)
  - 2b – medium frequency, medium data size, time critical, bounded latency (3 streams)
  - 4 – low frequency, big data size, non-time critical, (1 stream)
  - 5b – high frequency, medium data size, time critical, bounded latency (4 streams)



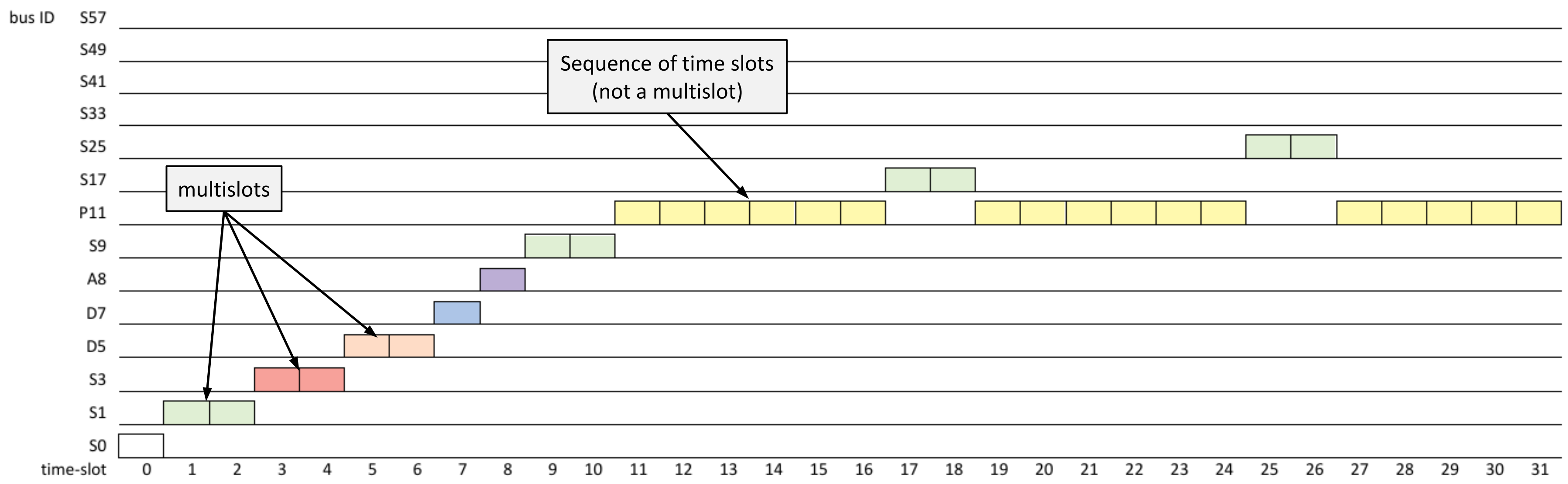
DATA STREAMS FOR SIMULATION

Data stream no.	Peripheral node	Direct link data signalling rate	Central node OBC= On-board Computer MM= Mass Memory	Initiator P= Peripheral node C= Central node	Data size (excluding headers):		Frequency	Estimated bandwidth required (excluding headers)	SAVOIR traffic class	Bus S=Static D=Dynamic A=Async. P=Packet	Priority on the async. bus (if relevant)	Minimum time allocation per epoch for:	
					sent	received						data stream	bus or bus series
	Symbol [SpW log.addr.]	Mbit/s	OBC/MM [SpW log.addr.]	P/C	bytes	bytes	Hz	bit/s	1-7	S/D/A/P		time-slots	time-slots
1	**Payload sensor 1** Pl1 [40]	200	MM [50]	P	10 M	0	1	100 M	4	P		4546	4546
2	**Sensor 1** Se1 [41]	10	OBC [51]	C	0	3	8	240	2b	A	high	8	8
3	**Actuator 1** Ac1 [41]	10	OBC [51]	C	3	0	0.125	3.75	1	A	low	1	
4	**Sensor 2** Se2 [42]	200	OBC [51]	P	4 k	0	8	320 k	2b	D^{a}(2)		16	16
5	**Actuator 2** Ac2 [42]	200	OBC [51]	C	700	0	8	56 k	2b	D^{b}		8	8
6	**Sensor 3** Se3 [43]	200	OBC [51]	P	72	0	1 k	720 k	5b	S^{a}		1000	2000
7	**Sensor 4** Se4 [43]	200	OBC [51]	P	4 k	0	1 k	40 M	5b	S^{a}(2)		2000	
8	**Actuator 3** Ac3 [44]	200	OBC [51]	C	4 k	0	100	4 M	5b	S^{b}(2)		200	200
9	**Actuator 4** Ac4 [44]	200	OBC [51]	C	8	0	100	8 k	5b	S^{b}		100	
												TOTAL:	6778

SIMULATION: SLOTS AND BUSES

- Time slots:
  - epoch length: 1 second
  - 8000 time slots of 125 µs (6778 used)
  - some slots are multislots composed of 2 time slots
  - 2 guard times of 2 µs each at the beginning and end of the slot
  - 1 target response delay margin of 8 µs
  - 1 time slot can accept ca. 2220 bytes (excluding headers) at 200 Mb/s (111 at 10 Mb/s)
- Virtual buses:
  - multiple static buses (for 4 streams) – 2 *series* of buses: $\mathbf{S^a}$, $\mathbf{S^b}$
    - each bus is allocated a single slot (precisely, a multislot of 2 time slots)
    - there are multiple buses in a bus series – for use by the same data stream
    - 2 streams are assigned to a single series of static buses when both streams have the same initiator node and the same target node
    - an extra static bus for time synchronization (no multislots)
  - 2 dynamic buses (for 2 streams): $\mathbf{D^a}$ (multislots of 2 time slots), $\mathbf{D^b}$ (no multislots)
  - 1 asynchronous bus (for 2 streams): **A** (no multislots)
  - 1 packet bus (for 1 stream): **P** (no multislots; data sent in segments of 2220 bytes)

SIMULATION: SLOT TIMELINE



Initial time slots (0-31) out of the 8000 that make up the epoch

Bus ID names are composed of bus-type letter and the number of first time-slot allocated.
Bus ID S0 is dedicated for time synchronization.

SIMULATION: KEY EVENT STATISTICS

- Key event counters recorded for each node:
  - **cmdSent** (for an initiator node): number of commands sent
  - **replyRcv** (for an initiator node): number of replies received without an error
  - **errorReplyRcv** (for an initiator node): number of replies received with errors
  - **cmdRcv** (for a target node): number of commands received without an error
  - **errorRcv** (for a target node): number of commands received with errors
  - **replySent** (for a target node): number of replies sent
  - **earlyTimecode**: number of time slots too short – the equivalent of early time-code reception (outside the tolerance time window)
  - **lateTimecode**: number of time slots too long – the equivalent of late time-code reception (outside the tolerance time window)
  - **interrupt**: number of distributed interrupts sent (optional: by the OBC – at slot boundaries)
- Nominal operation:
  - cmdSent(initiator,stream) = cmdRcv(target,stream) = replySent(target,stream) = replyRcv(initiator,stream)
  - errorRcv(target) = errorReplyRcv(initiator) = 0
  - earlyTimecode = lateTimecode = 0

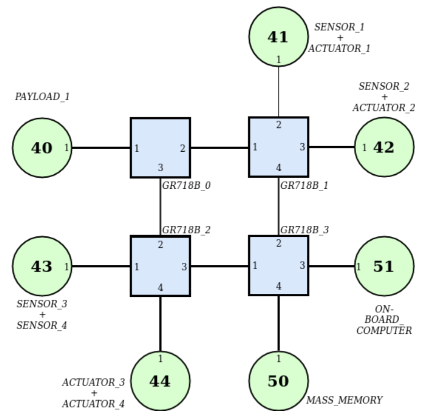
SIMULATION: TEST EXAMPLE – SINGLE ERROR

nodeID	cmdSent	replyRcv	errorReplyRcv	cmdRcv	errorRcv	replySent	earlyTimecode	lateTimecode	interrupt
40	9092	9092	0	0	0	0	0	0	0
41	0	0	0	17	0	17	0	0	0
42	16	16	0	16	0	16	0	0	0
43	4000	4000	0	0	0	0	0	0	0
44	0	0	0	400	0	400	0	0	0
MM	0	0	0	9092	0	9092	0	0	0
OBC	433	433	0	4016	0	4016	0	0	0

- Nominal operation for reference
  - Simulated time: 2 epochs (2 seconds, 16,000 time slots)

nodeID	cmdSent	replyRcv	errorReplyRcv	cmdRcv	errorRcv	replySent	earlyTimecode	lateTimecode	interrupt
40	9092	9092	0	0	0	0	0	0	0
41	0	0	0	17	0	17	0	0	0
42	16	16	0	16	0	16	0	0	0
43	4000	3999	1	0	0	0	0	0	0
44	0	0	0	400	0	400	0	0	0
MM	0	0	0	9092	0	9092	0	0	0
OBC	433	433	0	4015	1	4016	0	0	0

- Link failure: connection of node 43 taken down momentarily during transmission, resulting in SpW interface reset
  - One transaction lost

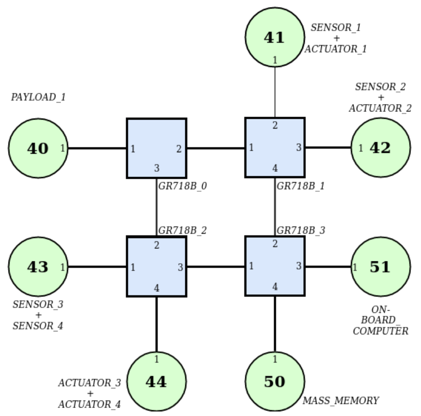


SIMULATION: TESTS EXAMPLE – SYSTEMATIC FAULTS

nodeID	cmdSent	replyRcv	errorReplyRcv	cmdRcv	errorRcv	replySent	earlyTimecode	lateTimecode	interrupt
40	9092	9092	0	0	0	0	0	0	0
41	0	0	0	17	0	17	0	0	0
42	16	16	0	16	0	16	0	0	0
43	4000	4000	0	0	0	0	0	0	0
44	0	0	0	400	0	400	0	0	0
MM	0	0	0	9092	0	9092	0	0	0
OBC	433	433	0	4016	0	4016	0	0	13784

nodeID	cmdSent	replyRcv	errorReplyRcv	cmdRcv	errorRcv	replySent	earlyTimecode	lateTimecode	interrupt
40	9092	9092	0	0	0	0	0	0	0
41	0	0	0	17	0	17	0	0	0
42	16	16	0	16	0	16	0	0	0
43	4000	4000	0	0	0	0	0	0	0
44	0	0	0	400	0	400	0	0	0
MM	0	0	0	9092	0	9092	0	0	0
OBC	433	417	16	4016	0	4016	0	0	13784

- Nominal operation for reference
  - Simulated time: 2 epochs (2 seconds, 16,000 time slots)
  - OBC sends interrupts at the end of every slot (with multislots, not at every time slot)



- Late replies (108 µs instead of expected 8 µs) by node 42 (in its target role), resulting in those reply packets to be truncated by the switch.
  - Node 42 sends all replies
  - OBC receives error indications
  - Other data streams unaffected

CONCLUSION

- A protocol proposed for deterministic data delivery based on SpaceWire-D concepts: MCMP (Mixed Criticality Message Passing protocol) / MRAP (MCMP Register Access Protocol)
- Implemented by extending the MOST-X simulator
- Tested for basic use cases
- Although the protocol relies mainly on the functionality of nodes, support in switches can add significantly to fault detection and isolation (e.g. interrupts)

THANK YOU FOR YOUR ATTENTION

Krzysztof.Romanowski@itti.com.pl