# Improvements to the Distributed Interrupts Scheme

***18th SpaceWire Working Group, 24th April 2012***
***Noordwijk, Netherlands***

## SpaceWire 1.0

**Current Status**:

- Networking technology for building on-board communications in S/C, used for the interconnection of:
  - Mass-memory
  - OBC
  - Telemetry
  - …
- Designed by ESA and widely used on many ESA, NASA, JAXA, RKA space missions
- The standard specifies point-to-point full duplex links, with flow control mechanism and provides minimal latency characters (time-codes) for timing information

**The Problem:**

- Current SpW standard does not provide a mechanism for propagation of critical events (e.g. alarms)
- With the current standard if an a Node shall notify another node for the occurrence of a critical event a SpW packet shall be sent but:
  - Links between Nodes/Switches may be temporarily blocked
  - The event may be delivered with unacceptable delay

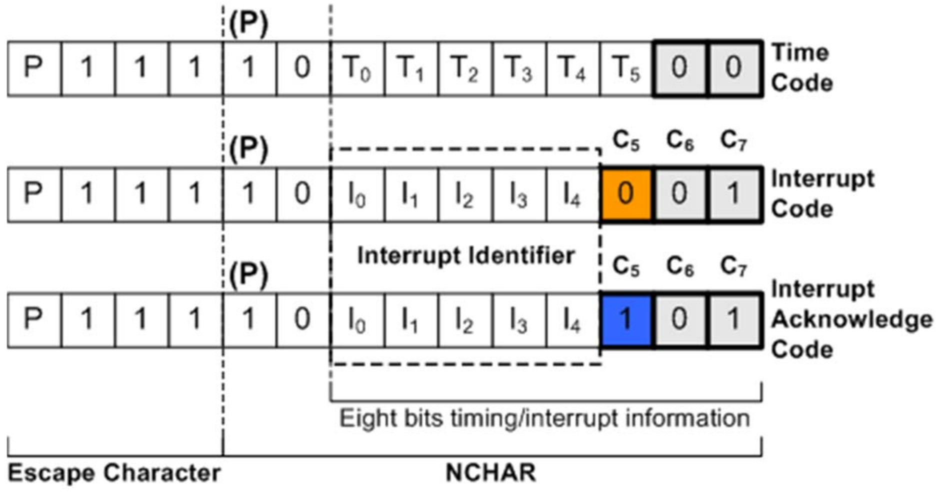
**The University of St Petersburg (SUAI) proposed Solution**:

- SUAI has proposed a solution for distribution of  minimized latency interrupts and interrupt acknowledgements over SpW based on unassigned time-code characters

## SUAI Proposal: Time-Codes & Interrupts

**<u>The Time Code Characters:</u>**

- Time Code is a minimal latency character broadcasted throughout the entire SpW network
- Six bits of time information are held in the least significant six bits of the Time-Code (T0-T5)
- Two bits (T6, T7), assigned to “00”, contain control flags Time-Code
- The rest three T6, T7 combinations are reserved for future use


(P)
P 1 1 1 1 0 $T_0$ $T_1$ $T_2$ $T_3$ $T_4$ $T_5$ 0 0
Time Code
$C_5$ $C_6$ $C_7$
P 1 1 1 1 0 $I_0$ $I_1$ $I_2$ $I_3$ $I_4$ 0 0 1
Interrupt Code
Interrupt Identifier
P 1 1 1 1 0 $I_0$ $I_1$ $I_2$ $I_3$ $I_4$ 1 0 1
Interrupt Acknowledge Code
Eight bits timing/interrupt information
Escape Character
NCHAR

**<u>The SUAI Proposal:</u>**

- Define **Interrupt Codes (INTR)**, which is a signal representing a request to handle an event of high priority
- Define **Interrupt Acknowledge Codes (INTA)** which acknowledge Interrupt Code acceptance for processing by a handler
- Use the time-codes propagation mechanism to distribute interrupts/acknowledgements
- This ensures minimal propagation latency and propagation through blocked links
- Use one of the reserved T6, T7 combinations to define interrupts/acknowledgements
- Use on bit ($C_5$) to distinguish between Interrupt Code and Interrupt Acknowledge Code
- Use a five bits **interrupt identifier** ($I_0$-$I_4$) to define 32 Interrupt Codes and 32 Interrupt Acknowledgement Codes

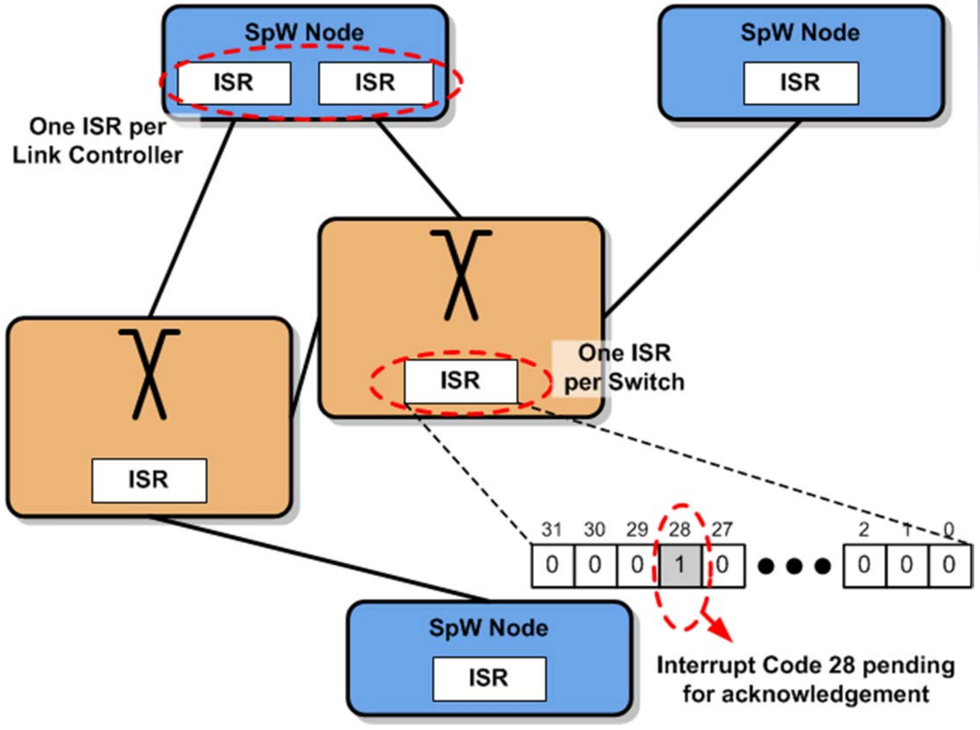
## SUAI Proposal: Interrupt Status Register

**The Interrupt Status Register**

- Each <u>Link Controller of a Node</u> and <u>each Switch</u> contains one 32-bit Interrupt Status Register (ISR)
- Each ISR bit corresponds to one of 32 possible interrupt identifiers
- An ISR bit is set to ‘1’ upon the transmission or reception of an Interrupt Code with the corresponding Interrupt Source Identifier
- An ISR bit is cleared to ‘0’ upon the transmission or reception of an Interrupt Acknowledge Code with the corresponding Interrupt Source Identifier

**Interrupt Status Register Functionality**

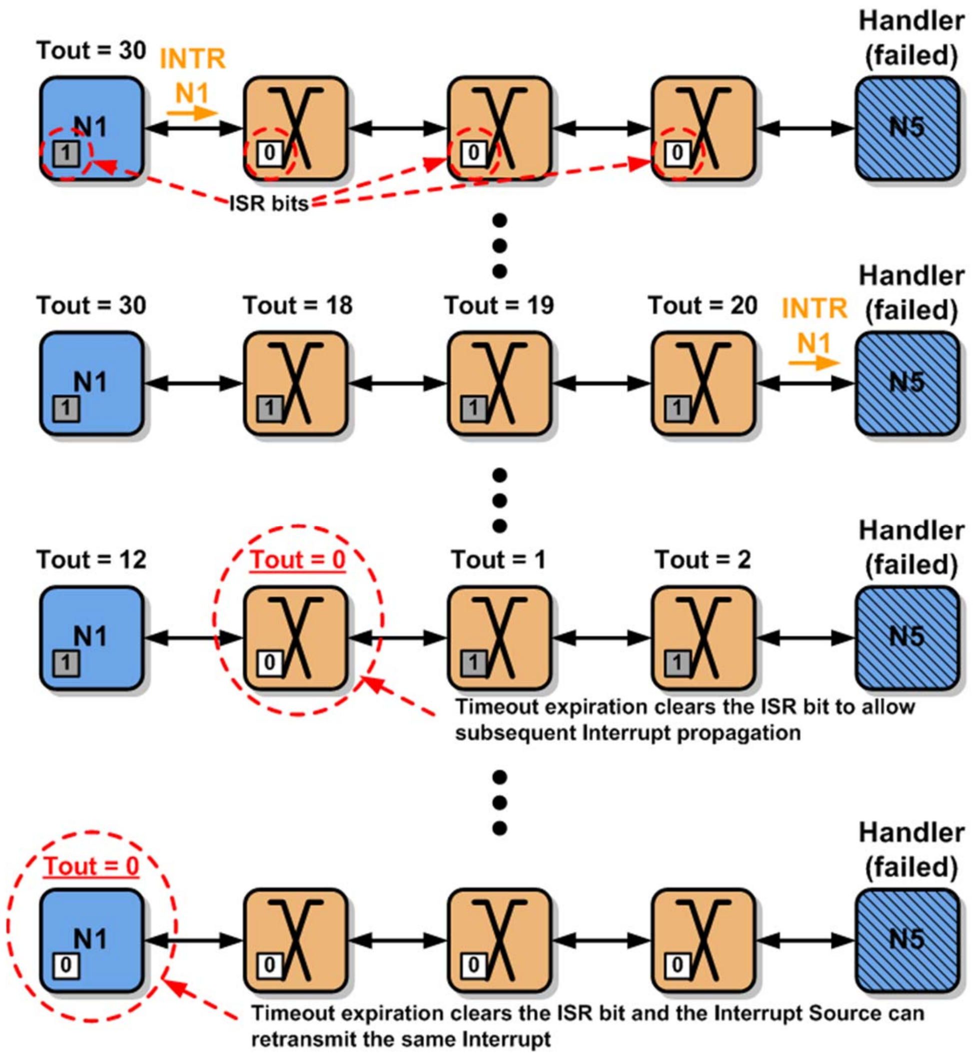
- Stores information about Interrupt Codes pending for Acknowledgement
- Initiates a time-out for each Interrupt Code broadcasted in the network
- Prevents repeated transmission of the same Interrupt Code or same Interrupt Acknowledgement in circular networks



## SUAI Proposal: ISR and Timeouts

**ISR Timeouts**:

- An ISR bit set to ‘1’ in a Switch does not allow a received Interrupt to be broadcasted
- An ISR bit set to ‘1’ in an Interrupt Handler Node does not allow the respective Interrupt Code to be transmitted
- After the transmission of an Interrupt the respective ISR bits in Switches and Nodes will never be cleared if:
  - The INTR does not reach the Interrupt Handler
  - The Interrupt Handler does not respond
  - The INTA gets lost
- For this reason the SUAI proposal specifies:
  - A timeout is started upon transmission/reception of an Interrupt Code at the Nodes
  - A timeout is started upon reception of an Interrupt Code in Switches
  - Expiration of a timeout clears the respective ISR bit



Tout = 30
INTR N1
Handler (failed)
N1
1
0
0
0
N5
ISR bits
Tout = 30
Tout = 18
Tout = 19
Tout = 20
INTR N1
Handler (failed)
Tout = 12
Tout = 0
Tout = 1
Tout = 2
Timeout expiration clears the ISR bit to allow subsequent Interrupt propagation
Tout = 0
Timeout expiration clears the ISR bit and the Interrupt Source can retransmit the same Interrupt

## Proposal: Operation in Circular Networks

- An Interrupt Code is transmitted by the Source and reaches the Handler in two hops
- The Interrupt Handler “immediately” sends back the Acknowledge and the Interrupt Service Routine **executes for the first time**
- The Acknowledge reaches the farthest router where it is ignored since the respective ISR bit is not yet set
- The Interrupt Code reaches the last router and is propagated to the Handler again

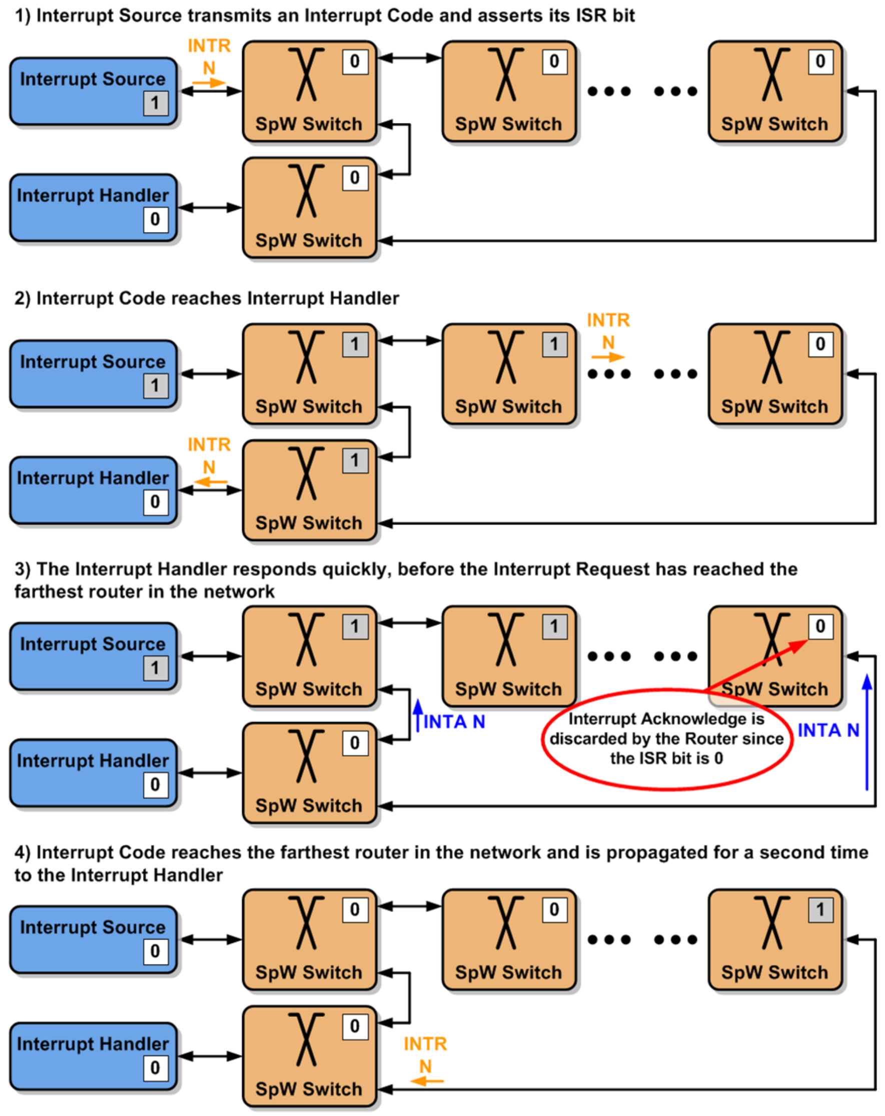
⇨ **The Interrupt Service Routine is executed again**

⇨ **Same case if the Source has redundant links**

⇨ **May be fatal if the Handler controls an actuator**

⇨ **Similar case if a second Interrupt Code is sent while the Acknowledge is still being propagated**

- **Proposed additions:**
  - Specify a minimum for the Interrupt Handler time response related to the network diameter (SUAI)
  - Specify a minimum time for handling the same interrupt request



1) Interrupt Source transmits an Interrupt Code and asserts its ISR bit
Interrupt Source 1
INTR N
SpW Switch 0
SpW Switch 0
SpW Switch 0
Interrupt Handler 0
SpW Switch 0
2) Interrupt Code reaches Interrupt Handler
Interrupt Source 1
SpW Switch 1
SpW Switch 1
INTR N
SpW Switch 0
Interrupt Handler 0
INTR N
SpW Switch 1
3) The Interrupt Handler responds quickly, before the Interrupt Request has reached the farthest router in the network
Interrupt Source 1
SpW Switch 1
SpW Switch 1
SpW Switch 0
INTA N
Interrupt Acknowledge is discarded by the Router since the ISR bit is 0
INTA N
Interrupt Handler 0
SpW Switch 0
4) Interrupt Code reaches the farthest router in the network and is propagated for a second time to the Interrupt Handler
Interrupt Source 0
SpW Switch 0
SpW Switch 0
SpW Switch 1
Interrupt Handler 0
SpW Switch 0
INTR N

## Proposal: Reset on Disconnect

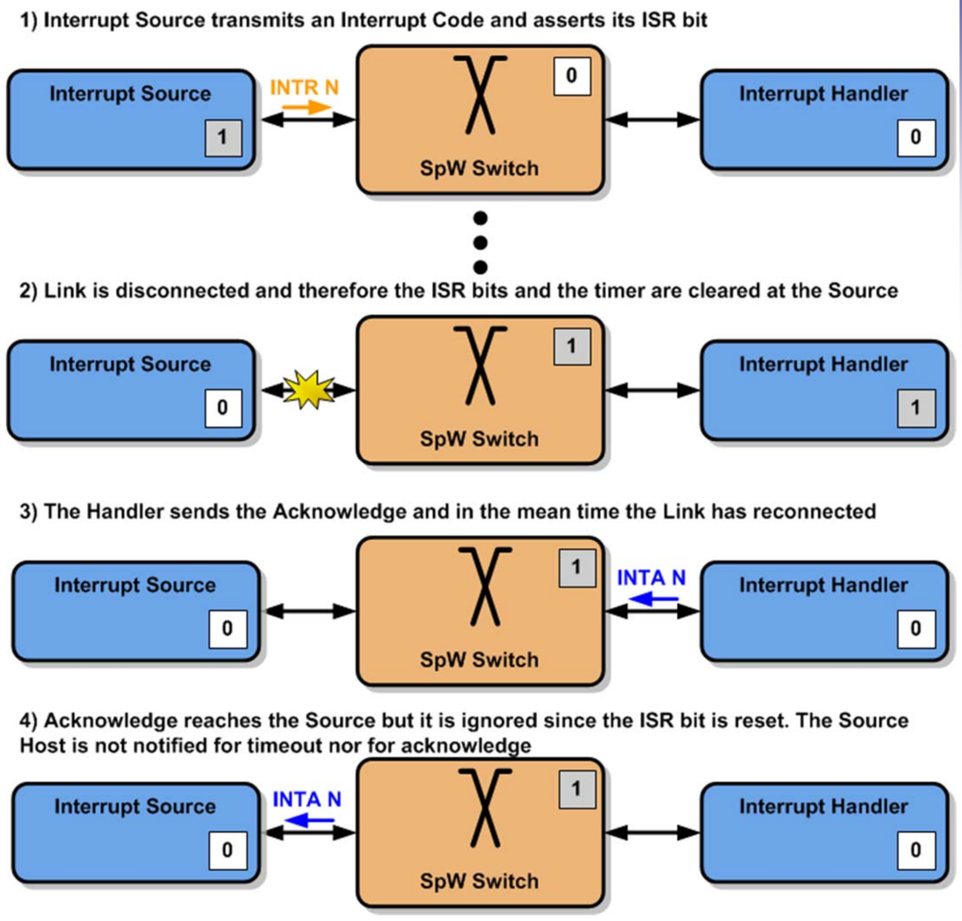
- **8.13.1 w: After a disconnect-reconnect the ISR bits shall be set to zero**
- **8.13.3 g: After a disconnect-reconnect the ISR timers shall be set to zero**

- The Interrupt Source transmits an Interrupt Code
- Interrupt Source's link disconnects and reconnects before receiving the Acknowledge
- The ISR and the ISR timers are reset due to the disconnect
- The Interrupt Handler sends the Interrupt Acknowledge, which reaches the Source but it is ignored since the ISR bit is cleared

⇨ **The Source Host does not receive the Acknowledge indication nor is it notified for timeout. Infinite timeout at the host!**

⇨ **Similar case if the Acknowledge is sent while the Source link is in disconnect**

- **Proposed modification:**
  - ISR bits and timers shall not be affected by a disconnect



1) Interrupt Source transmits an Interrupt Code and asserts its ISR bit
Interrupt Source
1
INTR N
0
SpW Switch
Interrupt Handler
0
2) Link is disconnected and therefore the ISR bits and the timer are cleared at the Source
Interrupt Source
0
1
SpW Switch
Interrupt Handler
1
3) The Handler sends the Acknowledge and in the mean time the Link has reconnected
Interrupt Source
0
1
SpW Switch
INTA N
Interrupt Handler
0
4) Acknowledge reaches the Source but it is ignored since the ISR bit is reset. The Source Host is not notified for timeout nor for acknowledge
Interrupt Source
0
INTA N
1
SpW Switch
Interrupt Handler
0

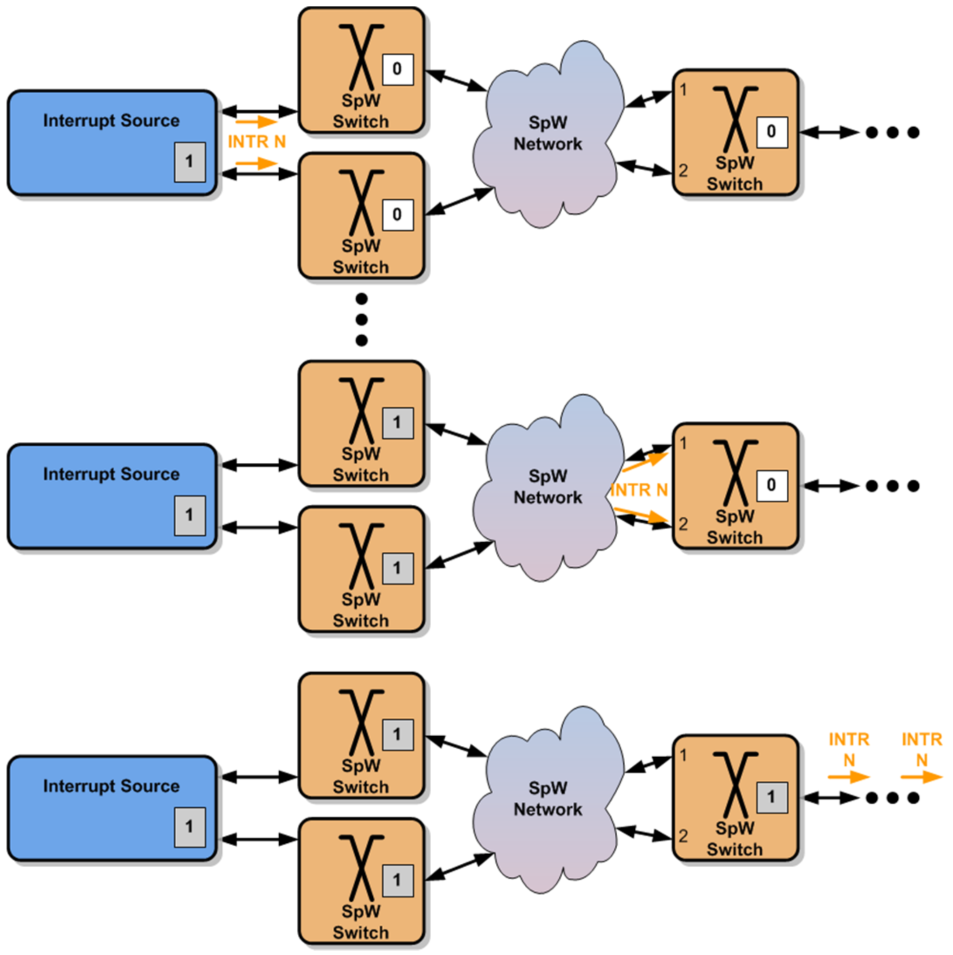
## Proposal: ISR Handling

- A node with redundant links transmits an Interrupt Code
- The Interrupt Code arrives through two paths at a single router
- Upon Interrupt's arrival at port 1 the ISR is checked and found to be 0
- Upon Interrupt's arrival at port 2 the ISR is checked and found to be 0
- An Interrupt Code is queued due to the Interrupt received through port 1 and the ISR bit is set
- An Interrupt Code is queued due to the Interrupt received through port 2 and the ISR bit is set again

⇨ **Two Interrupt Codes are propagated through the link to the next hop**

⇨ **May be logged as an error and initiate FDIR**

- **<u>Proposed clarification:</u>**
  - The routers shall check and set/reset ISR bits through **ATOMIC** operations



## Proposal: Babbling idiot

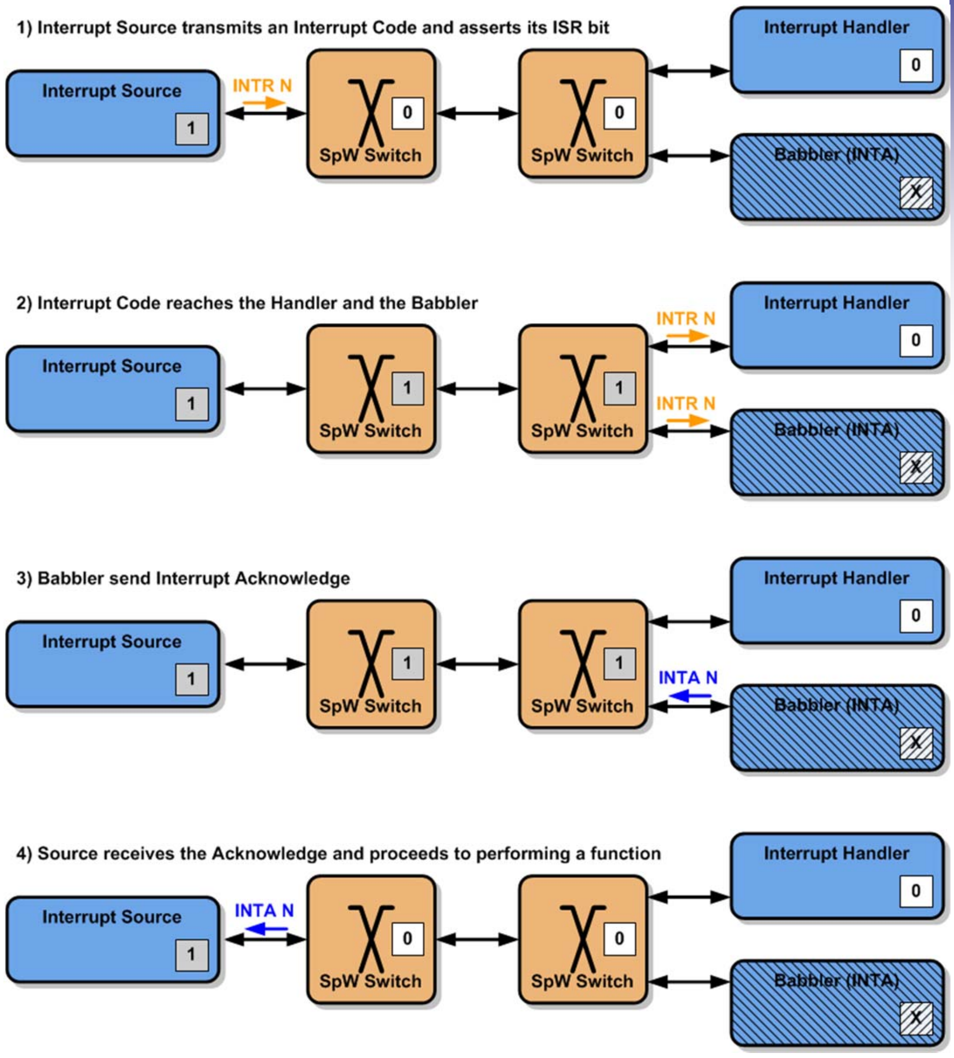
- A faulty node in the network responds to all (or certain) Interrupt Code(s)
- An Interrupt Code is sent by the Interrupt Source
- The Interrupt Code is delivered to both the Interrupt Handler and the faulty node
- The faulty node responds (erroneously) with an Interrupt Acknowledge

⇨ **The Acknowledge is delivered to the Source and "grants" the execution of further actions**

⇨ **Interrupt is erroneously acknowledged if Handler is disconnected or off**

⇨ **Similar case if a Babbler sends Interrupt Codes**

- **Proposed solution:**
  - Protection can be provided by **edge** routers
  - Assign allowed/expected Interrupt Code(s) and Acknowledge(s) per edge Router port
  - Provide error information in routers for FDIR (remote FDIR through PnP)?



## Summary of Functional modifications & additions to the SUAI proposal

- If the link is not in the RUN state upon host's request for Interrupt Code transmission, the Link Controller shall notify the host and ignore the request
- At the Interrupt Source, a timer shall start upon transmission of the Interrupt Code through the SpW link (imposes extensions to the SpW CODEC)
- There is one Interrupt Mask register per node which defines for which Interrupt Identifiers the node is the Interrupt Handler (INFORMATIVE)
- Specify a minimum time for the Interrupt Handler response related to the network diameter (SUAI) – the largest diameter may correspond to a circular connection
- Specify a minimum time for the transmission of an Interrupt Code after reception of an Interrupt Acknowledge with the same Interrupt Identifier
- Implement Cold Redundancy at both the Source in circular networks where possible (INFORMATIVE)
- ISR bits and timers shall not be affected by a disconnect
- The routers shall check and set/reset ISR bits through **ATOMIC** operations
- Protection can be provided by **edge** routers
- Provide error information in routers for FDIR (e.g. in order to support remote node deactivation through PnP)?

## SUAI Proposal Timing Issues: Timeouts Asynchronicity (1/2)

- Interrupts are to be used for control loops
- Desired control loop frequency 1 KHz (SpW WG 15)

⇨ **Interrupts propagation and time-outs shall have sub-milliseconds (e.g. microseconds) resolution**

- Time-Codes propagation time approx. 600 ns (SpW-D draft B)

⇨ **Time from Interrupt transmission to its broadcast by the next Switch approx. 1 us**

- A Node (N1) transmits an Interrupt Code and sets the time-out to a specific value (e.g. Interrupt Handling Time + 15 us)
- At the same time another node (N2) transmits another interrupt (INTR N2)
- Let's assume that N2 is broadcasted before N1 by the first switch
- On the path to the Interrupt Handler more interrupt codes are added

⇨ **Timeouts cannot be guaranteed to be consistent throughout the entire network**

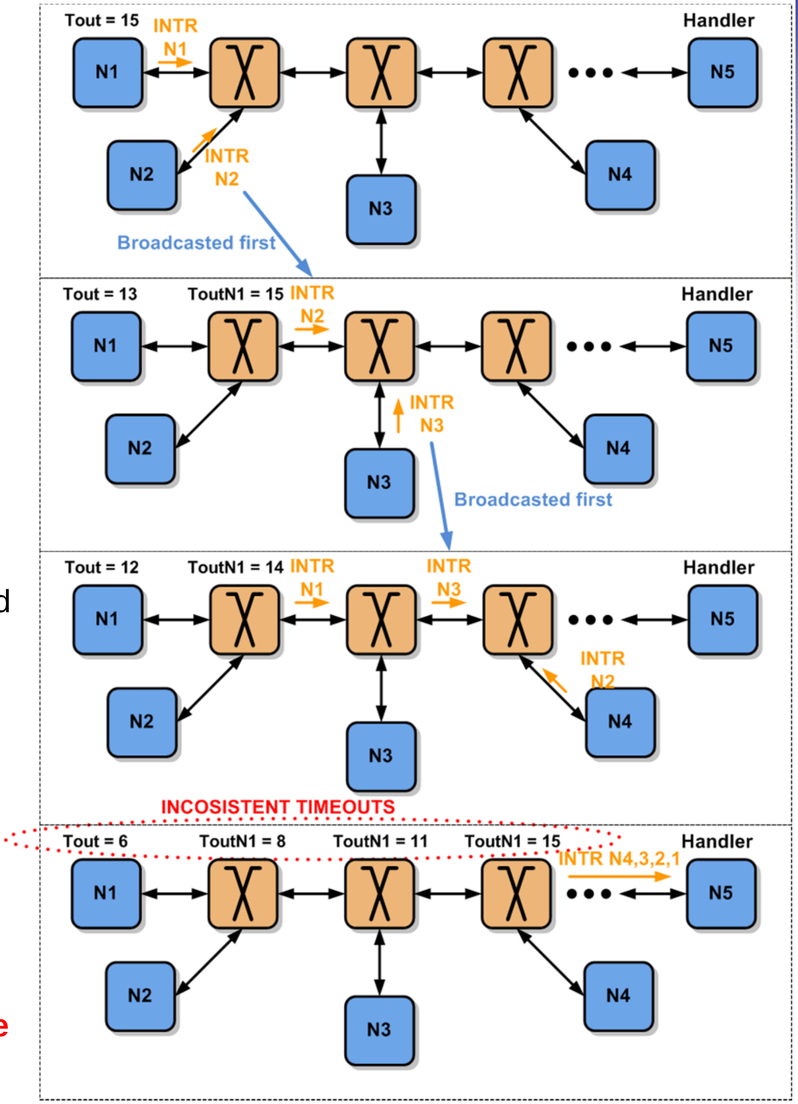
⇨ **The situation gets worse:**

  ⇨ **If more interrupts appear in the path between the source and the handler**

  ⇨ **Taking into account future evolutions (e.g. redundant Time-Codes)**

  ⇨ **With Half Duplex SpW**

  ⇨ **Taking into account simultaneous Interrupt Acknowledge Codes**



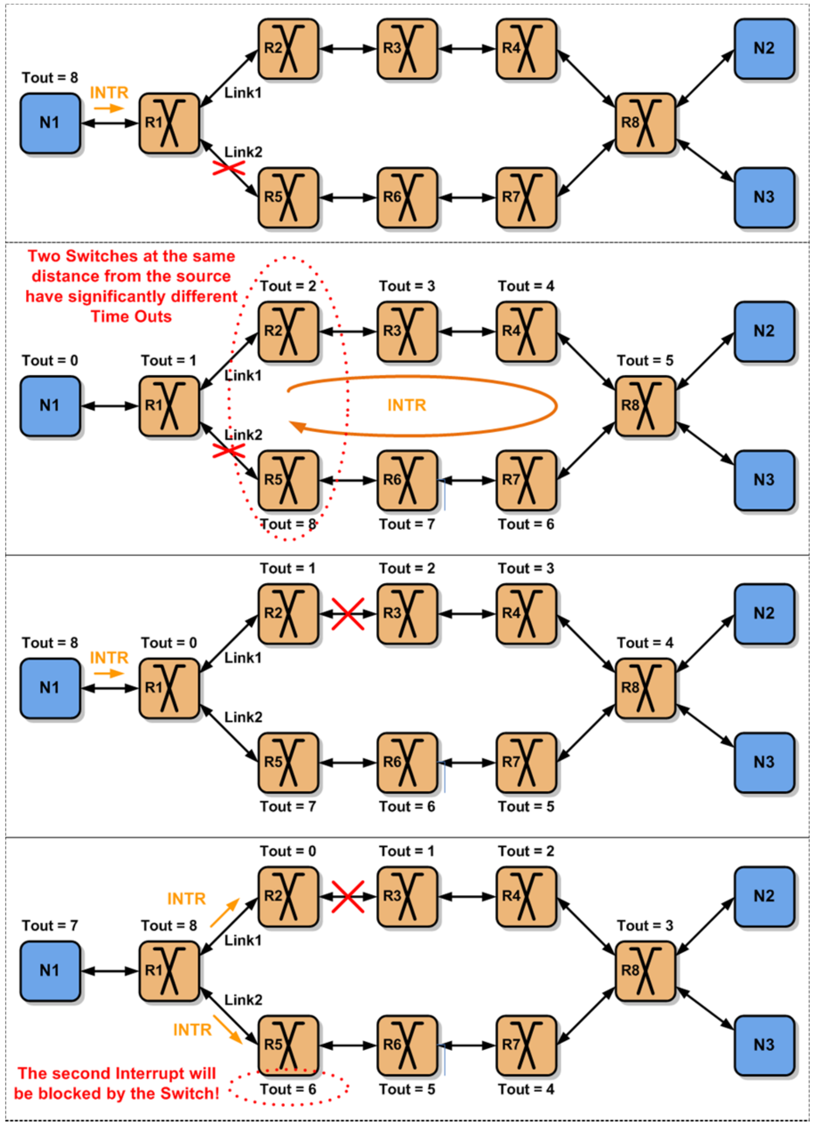
## SUAI Proposal Timing Issues: Timeouts Asynchronicity (2/2)

- Different worst case delay taking into account loops shall be taken into account for the calculation of interrupt time-outs in Nodes and Switches
- In the following, **improbable** but possible, scenario:
  - Node 1 transmits an Interrupt Code in a network with redundant paths
  - A Link is reconnecting upon Interrupt transmission
  - The Interrupt Code reaches R5 after travelling through the entire network
  - Two Switches having the same distance from the source have significantly different time-outs
  - Re-transmitted interrupt will be blocked at R5 and if another link is reconnecting, the interrupt will never reach all nodes although a path is available

⇨ **The longest path, taking into account loops, shall be taken into account for the calculation of time-outs, but**

⇨ **Adopting a common value for all interrupts in all switches <u>decreases maximum obtained frequency</u> (e.g. for interrupts sent from N2 to N3)**

⇨ **Alternatively each device shall implement different Time Out for each Identifier. <u>Expensive</u>**

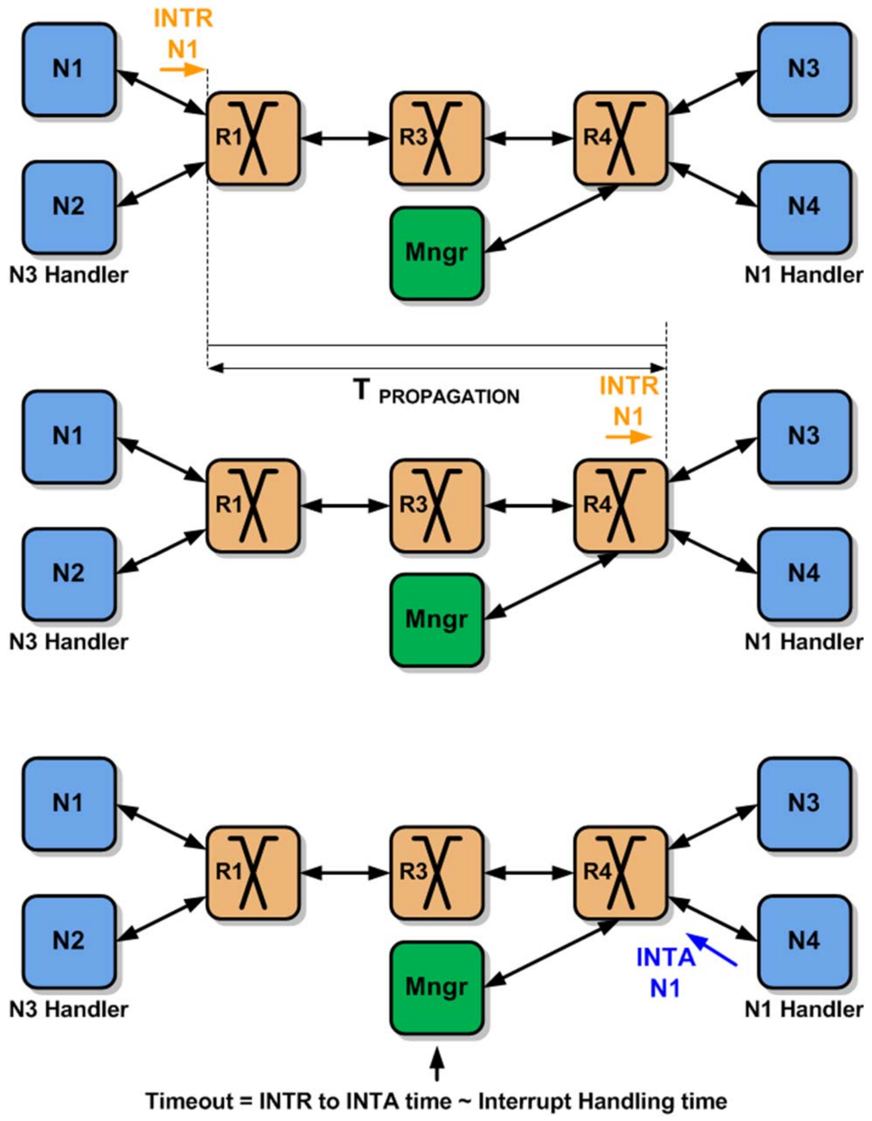


## SUAI Proposal Timing Issues: Management Node Timeouts (1/2)

**SUAI 8.13.3.f: Some nodes in SpW networks (e.g. network management nodes) may have the rights to send Interrupt-Acknowledge-Codes while not being the sources for correspondent Interrupt-Codes**

- The Management node is close to the Handler of a specific interrupt
- Node N1 transmits INTR N1
- INTR N1 arrives at the management node after T $P_{ROPAGATION}$ and activates a Time Out timer
- Node N4 responds after "Interrupt Handling Time" (IHT)

⇨ **The Time Out at the manager shall be programmed to approximately IHT for max efficiency**



## SUAI Proposal Timing Issues: Management Node Timeouts (2/2)

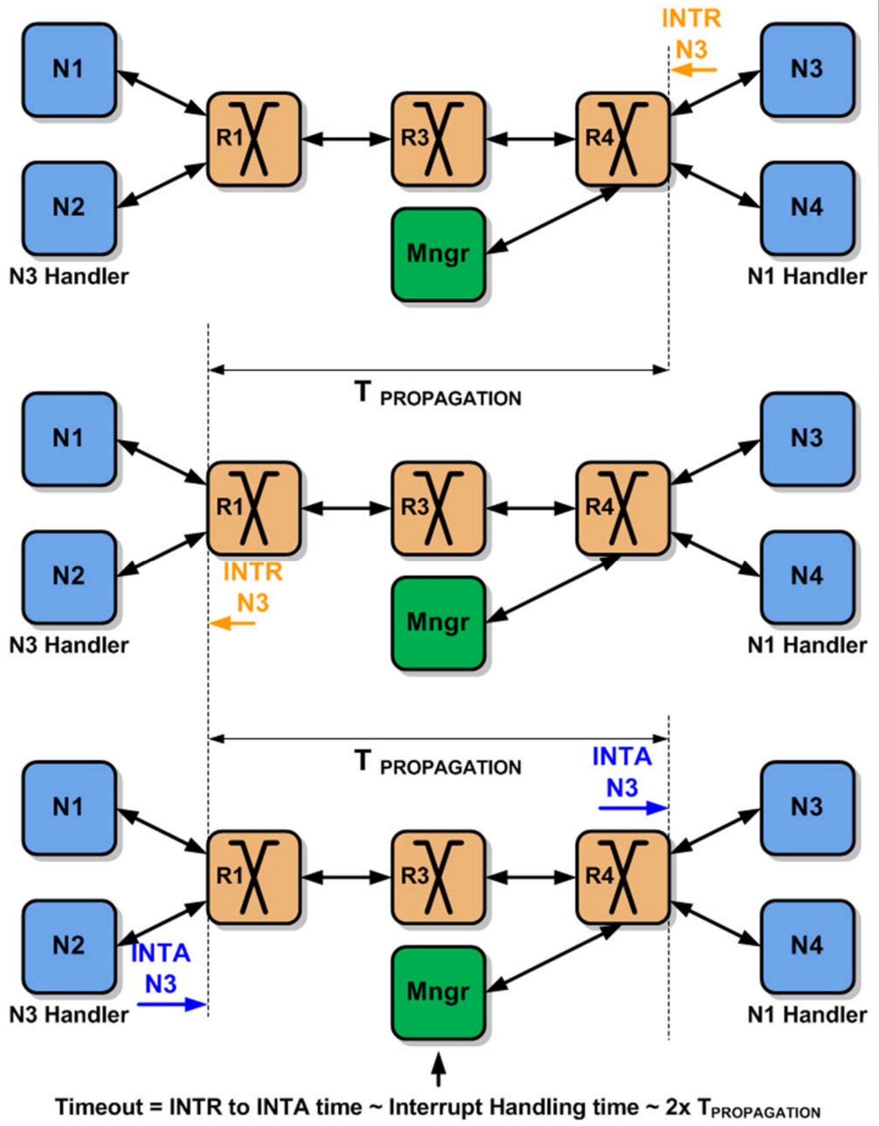
- The Management node is away from the Handler of a specific interrupt
- Node N3 transmits INTR N3
- INTR N1 arrives "immediately" at the management node and activates a Time Out timer and after $T_{PROPAGATION}$ to Node N3
- Node N2 responds after IHT (Interrupt Handling Time)
- The Acknowledge will be received at the Management Node after $T_{PROPAGATION}$

⇨ **The management node time-out shall be different than the timeout of the previous case**

  ⇨ **The Manager shall retain different time-out for each interrupt identifier (taking into account Interrupt Response times) or,**

  ⇨ **The manager shall be in the "middle" of the network (for uniform Interrupt Response times)**

⇨ **The manager may timeout and send INTA after the source has sent a subsequent Interrupt which may clear ISRs in intermediate switches and cause subsequent INTA loss**



## SUAI Proposal Implementation Issues

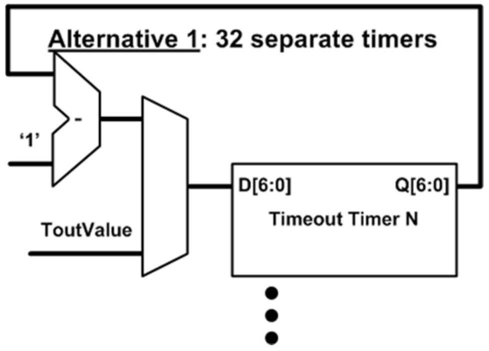
- Each Switch has a 32-bits ISR register
- Each ISR bit has an associated Timeout Timer set upon the reception of an INTR

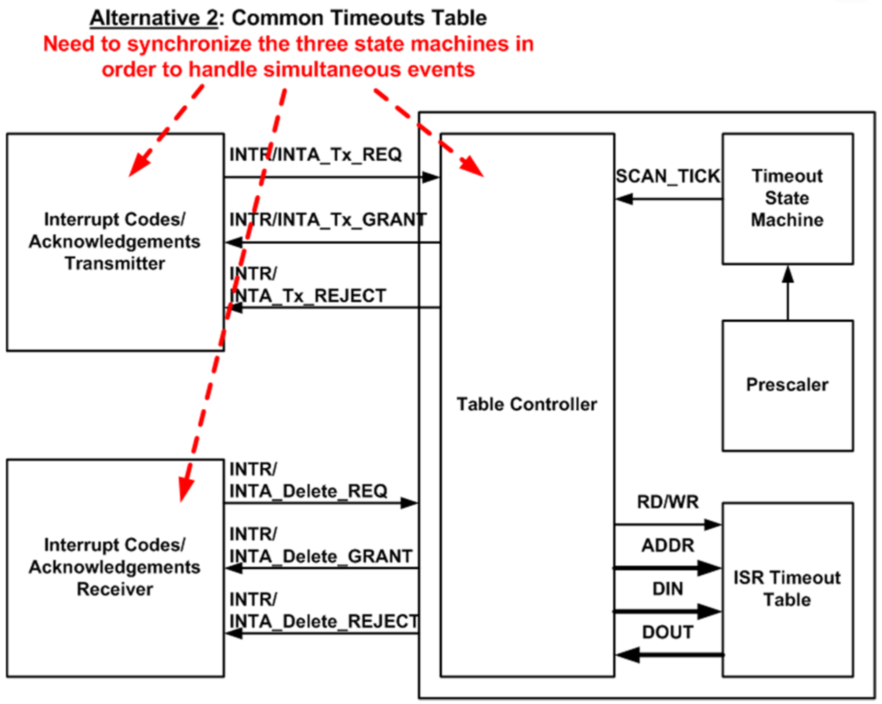
☺ **Advantages:**

- No problem after an INTA loss

☹ **Drawbacks:**

- Interrupts frequency limited by the maximum Interrupt Response time plus two times the worst case "signaling codes" propagation in the network
- Difficult to determine the timeout values for maximum performance
- Complex implementation requiring a lot of resources, e.g. for 1 ms max. timeout with 10 us resolution
- **Alternative 1**: Separate timeout for each Interrupt
  - Logic for timeout start/clear is simple, but
  - 7 bits per timer are needed for single timer per Interrupt Identifier which makes a total **224 FFs** for timeouts
- **Alternative 2**: Common timeout memory
  - Table (DPRAM) Controller logic gets complex to handle simultaneous requests and condition changes (e.g. Link Disconnected after Tx has granted access to the Timeout Table)
  - Need to synchronize the Tx, Rx and Table Controller state machines





## Alternatives to the SUAI proposal

- Timeouts require a lot of HW resources
- Determining the switches timeout values in complex networks is complex
- Timeouts in switches limit the maximum Interrupts frequency
- Separate timeout values for different Interrupt Identifiers are required to satisfy different requirements, but,
- separate timeout increases the required HW resources even more
- Alternative proposal for Distributed Interrupts:
  - Remove the timeouts at the switches
  - Remove the Interrupt Acknowledge Codes
  - Remove both

## Distributed Interrupts without Timeouts

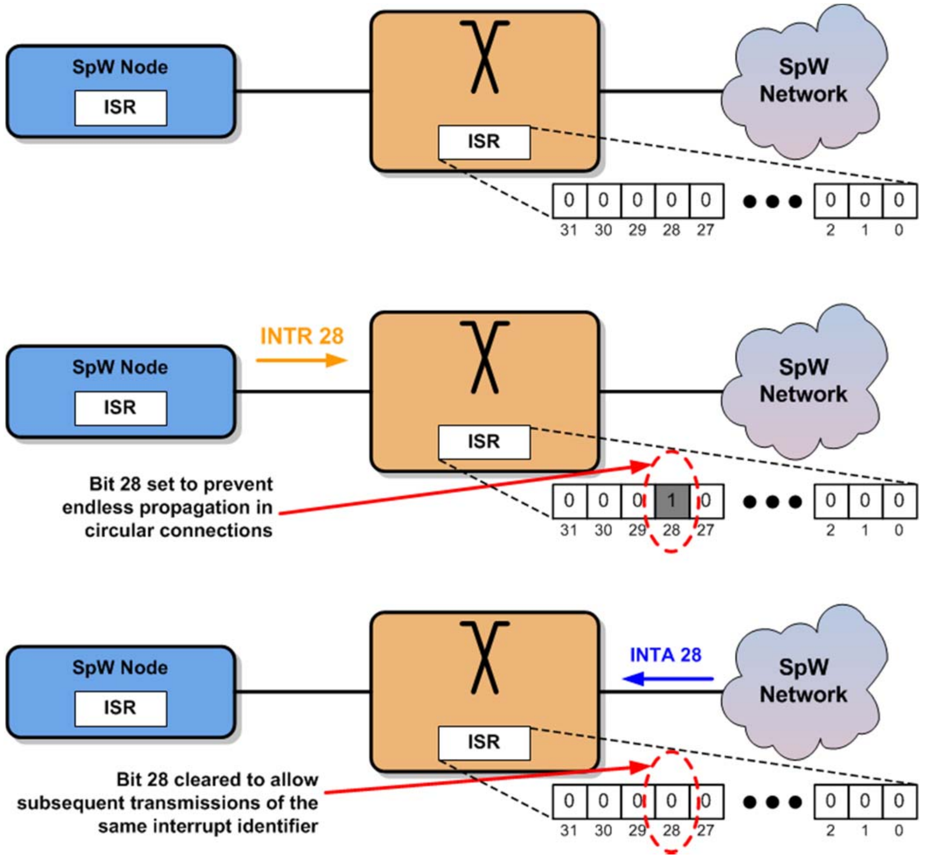
- Each Switch has a 32-bits ISR register
- Each bit is set upon the reception of an Interrupt Code with the respective Interrupt Identifier
- Each bit is cleared upon the reception of an Interrupt Acknowledge Code with the respective Interrupt Identifier
- ISR prevents endless propagation of the same Interrupt in networks with loops

☺ **Advantages:**

- Simple to implement
- Requires few resources due to the absence of the timeouts block at the switch
- Does not require configuration (for timeouts)

☹ **Drawbacks:**

- If an INTA gets lost, the respective ISR bit in the switch(es) remains ‘1’ thus not allowing subsequent transmission Interrupt Codes with the respective Interrupt Identifier
- INTR repetition frequency is limited by the maximum Interrupt Response time plus two times the worst case Interrupt/Acknowledgement propagation time


SpW Node
ISR
ISR
SpW Network
0 0 0 0 0 ● ● ● 0 0 0
31 30 29 28 27 2 1 0
INTR 28
Bit 28 set to prevent endless propagation in circular connections
0 0 0 1 0 ● ● ● 0 0 0
INTA 28
Bit 28 cleared to allow subsequent transmissions of the same interrupt identifier
0 0 0 0 0 ● ● ● 0 0 0

## Distributed Interrupts without Acknowledgements

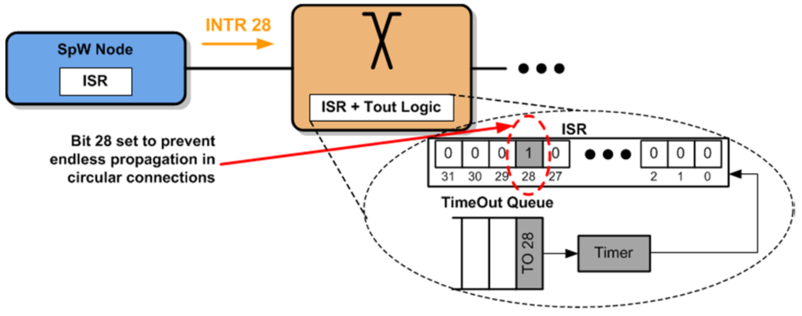
- Each Switch has a 32-bits ISR register used to prevent endless transmission of Interrupt Codes in networks with loops
- Each ISR bit has an associated Timeout Timer set upon the reception of an INTR
- There are no Interrupt Acknowledge Codes
- ISR bits are cleared by ISR Timeouts **ONLY**

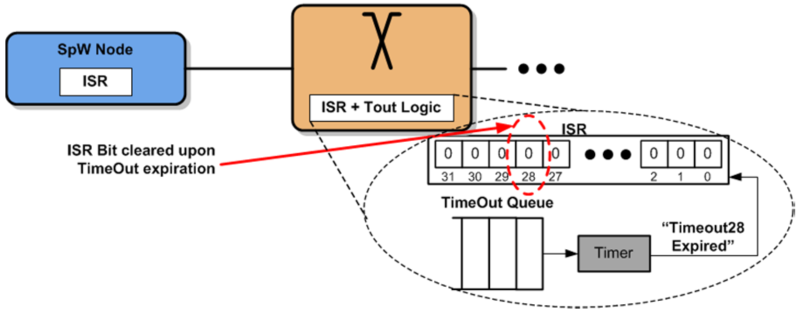
☺ **Advantages:**

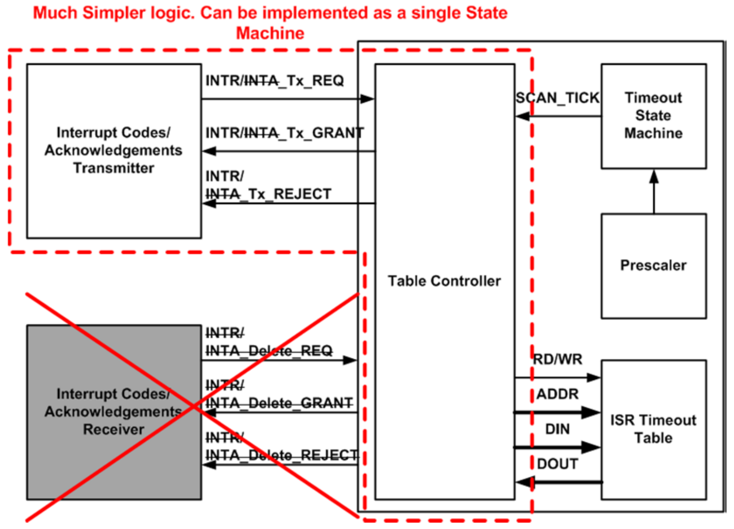
- Supports up to 64 different Interrupts
- Requires simpler logic than the SUAI proposal since there are no simultaneous events
- Interrupt repetition frequency doubles

☹ **Drawbacks:**

- Requires more memory resources than the SUAI proposal (the same per interrupt though)
- If an INTA gets lost, the respective ISR bit in the switch(es) remains '1' thus not allowing subsequent transmission Interrupt Codes with the respective Interrupt Identifier
- INTR repetition frequency is limited by the worst case "signaling codes" propagation time in the network



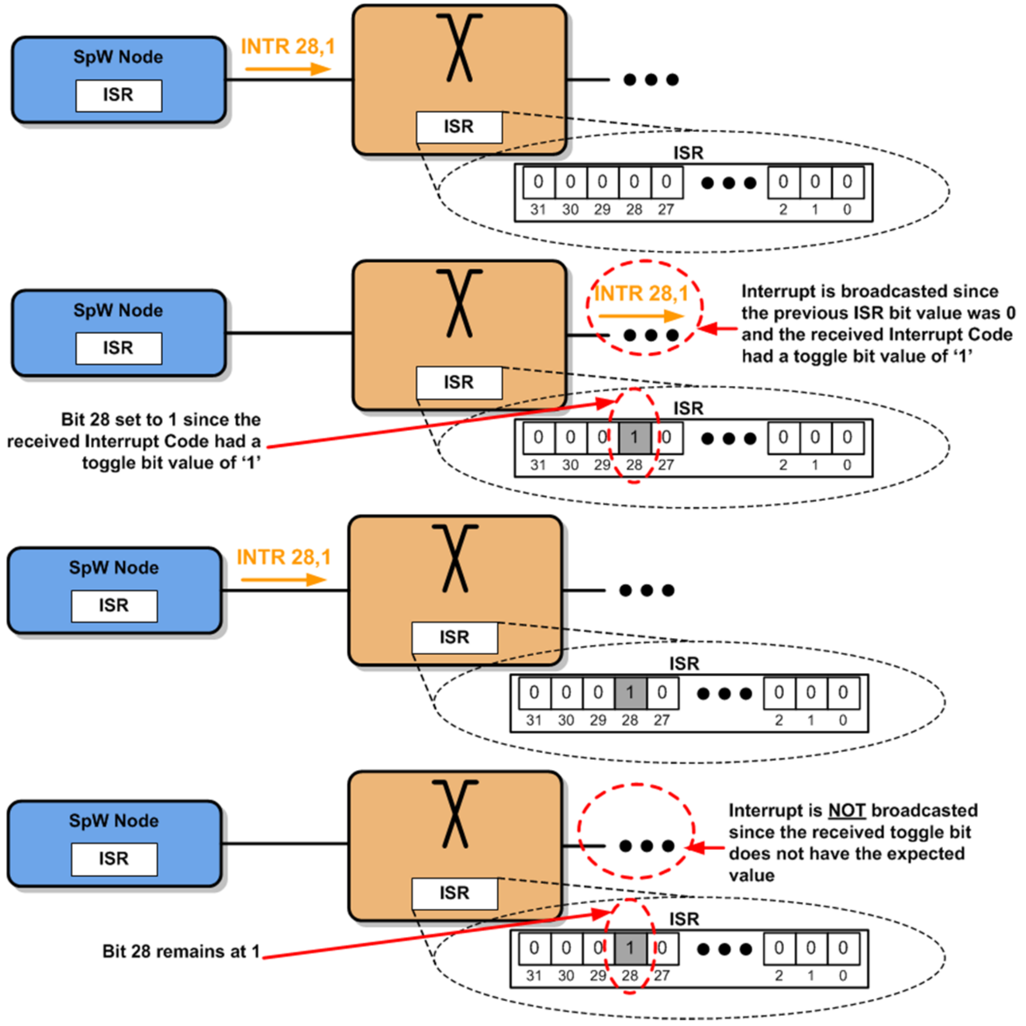




Much Simpler logic. Can be implemented as a single State Machine
Interrupt Codes/ Acknowledgements Transmitter
INTR/~~INTA~~_Tx_REQ
INTR/~~INTA~~_Tx_GRANT
INTR/ ~~INTA~~_Tx_REJECT
Table Controller
SCAN_TICK
Timeout State Machine
Prescaler
Interrupt Codes/ Acknowledgements Receiver
~~INTR/ INTA_Delete_REQ~~
~~INTR/ INTA_Delete_GRANT~~
~~INTR/ INTA_Delete_REJECT~~
RD/WR
ADDR
DIN
DOUT
ISR Timeout Table

## Distributed Interrupts without Acknowledgements and Timeouts (1/3)

- Each Interrupt Code consists of a 5-bits Interrupt Identifier and a toggle bit
- Each Switch has a 32-bits ISR register which stores the value of the last received Toggle Bit
- Each Node Controller has a 32-bits ISR register which stores the value of the last received/transmitted toggle bit
- Upon reception of an Interrupt Code, the Node Controller/Switch checks the received Toggle Bit and compares it with the value of its Toggle Bit and:
    - If they have the same value the Interrupt Code is discarded
    - If they have different values and Interrupt Code is accepted/broadcasted


SpW Node
ISR
INTR 28,1
ISR
ISR
0 0 0 0 0 ••• 0 0 0
31 30 29 28 27 2 1 0
INTR 28,1
Interrupt is broadcasted since the previous ISR bit value was 0 and the received Interrupt Code had a toggle bit value of '1'
Bit 28 set to 1 since the received Interrupt Code had a toggle bit value of '1'
0 0 0 1 0 ••• 0 0 0
INTR 28,1
Interrupt is NOT broadcasted since the received toggle bit does not have the expected value
Bit 28 remains at 1

# Distributed Interrupts without Acknowledgements and Timeouts (2/3)

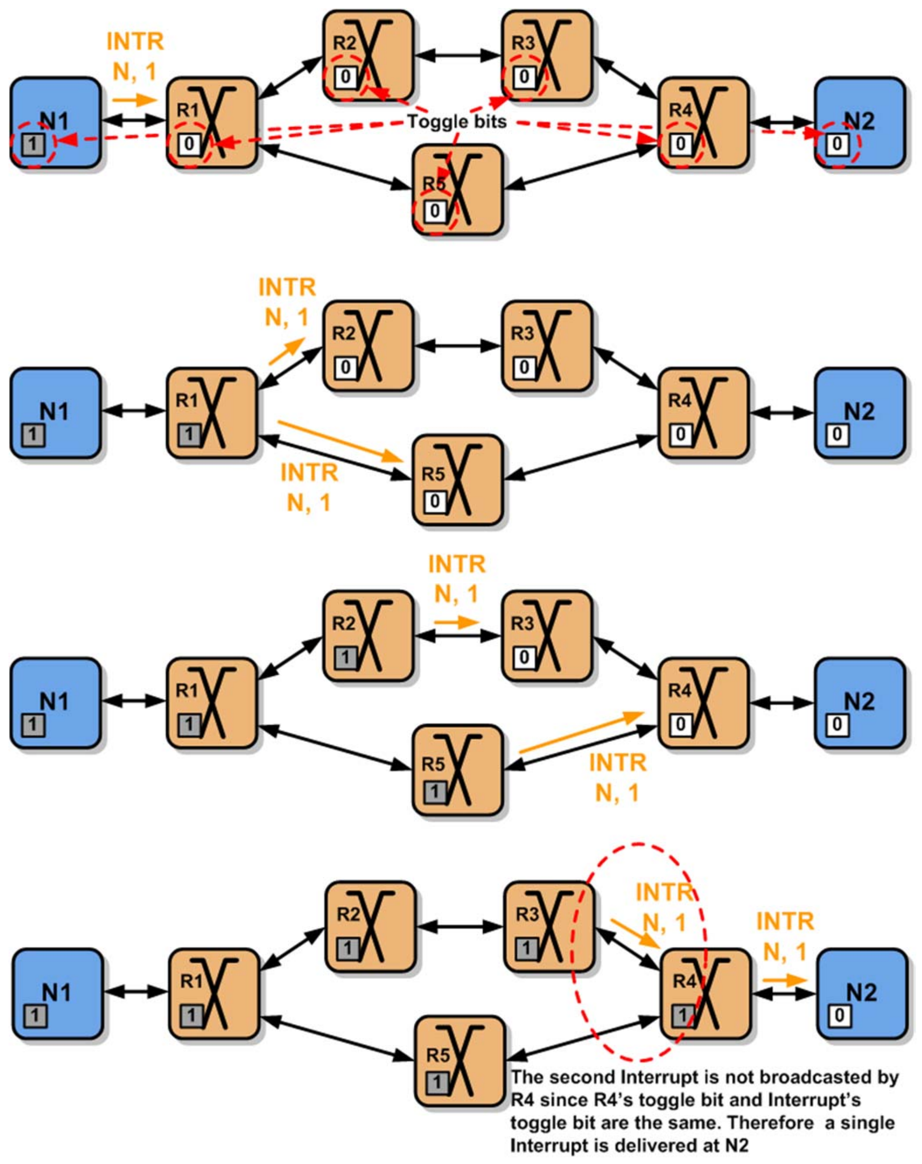
- Successive Interrupt Codes are transmitted with alternate Toggle Bit value
- The Toggle Bit is used in the switches in order to prevent endless propagation of the Interrupt Code in networks with circular connections

☺ **Advantages:**

- Does not require timeouts
- Very simple logic required
- Timeouts handled at the nodes, thus supporting different value per Interrupt Identifier

☹ **Drawbacks:**

- Loss of an Interrupt results in loss of the subsequent Interrupt with the same identifier
- Interrupts may be lost after a switch power-up/reset, which may scale to multiple Interrupts loss in case multiple switches have reset

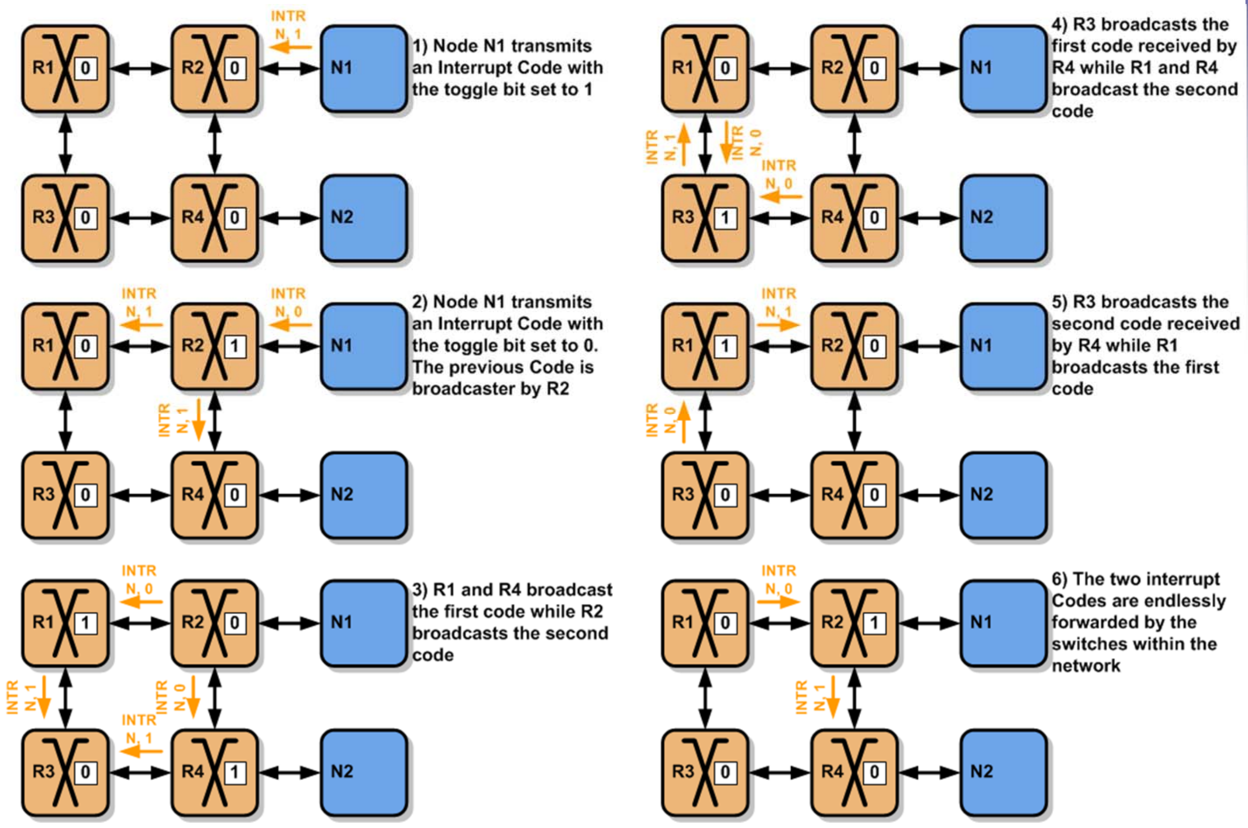


The second Interrupt is not broadcasted by R4 since R4's toggle bit and Interrupt's toggle bit are the same. Therefore a single Interrupt is delivered at N2

## Distributed Interrupts without Acknowledgements and Timeouts (3/3)

😐 **For further study:**

- This is a very promising candidate, mainly because of its simplicity, for Interrupts that do not use Interrupt Acknowledgements but its robustness under certain failures needs further analysis
- The appearance of a Babbling Node in the network may result in endless Interrupt Code propagation in circular networks
- For example, in the example shown here, all network Nodes will be endlessly receiving successive Interrupts and the Interrupts will also be delivered to the originating node
- This may be fixed by imposing a short (one or a few characters) and fixed-length delay between interrupts of the same number (needs further studying).



1) Node N1 transmits an Interrupt Code with the toggle bit set to 1

2) Node N1 transmits an Interrupt Code with the toggle bit set to 0. The previous Code is broadcaster by R2

3) R1 and R4 broadcast the first code while R2 broadcasts the second code

4) R3 broadcasts the first code received by R4 while R1 and R4 broadcast the second code

5) R3 broadcasts the second code received by R4 while R1 broadcasts the first code

6) The two interrupt Codes are endlessly forwarded by the switches within the network

## Multi-casted Interrupts

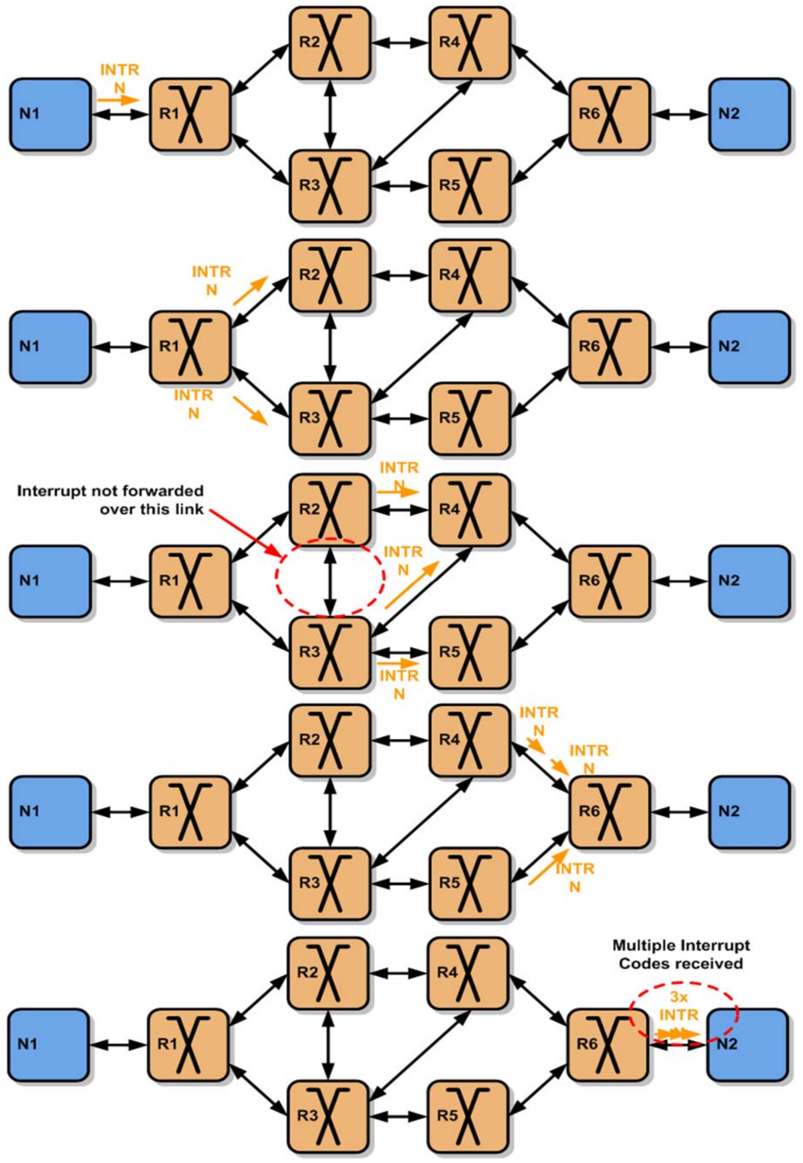
- Each switch maintains a table determining to which port(s) each interrupt can be forwarded

☺ **Advantages:**

- Unlimited Interrupts frequency
- Does not require ISR nor timeouts
- Simple logic required
- Timeouts handled at the nodes, thus supporting different value per Interrupt Identifier
- Works with and without Acknowledgements
- Supports up to 64 Interrupt Code if Acknowledgements are not used

☹ **Drawbacks:**

- Arrival of multiple interrupt copies at the Handler/Target nodes – need to define minimum interrupt response time (if INTA is used) equal to the maximum Interrupt Codes arrival skew
- Multi-cast table shall be initialized upon switch power-up/reset
- Misconfiguration can cause endless propagation of an Interrupt Code if Babbling Idiot protection is not employed at the switches
- Multi-cast table requires EDAC/Memory scrubbing
- **Not compatible with SpW 1.0 standard**



N1
R1
R2
R3
R4
R5
R6
N2
INTR N
Interrupt not forwarded over this link
Multiple Interrupt Codes received
3x INTR

## Comparison of alternative solutions for Distributed Interrupts

| | SUAI | SUAI without timeouts | SUAI without INTAs | Interrupts Without INTAs and Timeouts | Multi-casted Interrupts |
|---|---|---|---|---|---|
| Number of Interrupts | 32 | 32 | 64 | 32 | 32(64) |
| Repetition frequency | 1/(2xTprop) | 1/(2xTprop) | 1/Tprop | 1/Tprop | No limit, topology independent |
| Individual Timeouts Support | No | Yes | No | Yes | Yes |
| Robustness | Good | Bad | Good | Good | Good |
| Logic Complexity | High | Low | Medium | Low | Low |
| Required Memory Resources | 32xTimeoutWidth, if different timeout per identifier is not required | None | 64xTimeoutWidth, if different timeout per identifier is not required | None | 32(64)x NoOfPorts |
| EDAC, Memory scrubbing required | No if different timeout per identifier is not required | No | No | No | Yes |
| Configuration required | Yes | No | No | No | Yes |
| SpW 1.0 compliance | Yes | Yes | Yes | Yes | No |