SpaceWire-RT case study

Thomas Ferrandiz

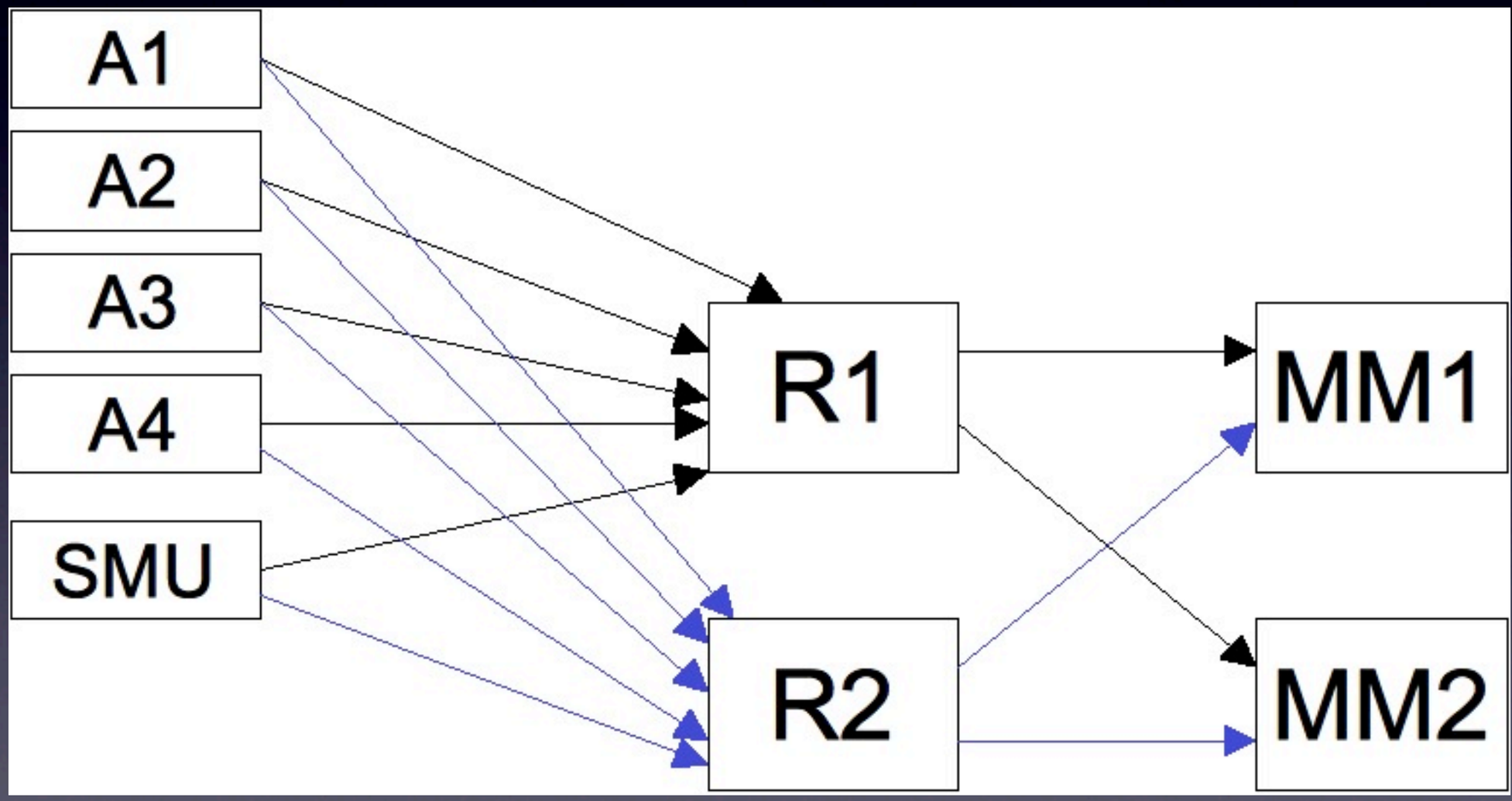
Institut Supérieur de l'Aéronautique et de l'Espace

Introduction

- Goal: use SpaceWire for both command/ control and payload traffic
- Problem: Will critical messages be delivered on-time ?
- Individuals delays are very hard to determine because of Wormhole Routing and recursive blocking

➡ compute a deterministic upper-bound on the worst-case delay of every flow

Case Study (1)



Case Study (2)

- Each A_i sends 4 KB packets periodically to one of the MM module
- The SMU sends 100 B packets periodically to one of the MM module
- Each SpaceWire interface works independently of its redundant interface
- A_1 and A_2 do not emit all the time (A1 emits 44 % of a 16 s cycle)

Basic SpaceWire

- Delay for a SMU packet:
  - caused by the router
    - SMU packets gets blocked behind A_i packets
    - Worst-case: 4 4-KB packets may go through before the SMU packet
- If we use priority on logical adresses:
  - Worst-case: 1 4-KB packets may go through before the SMU packet

SpaceWire-RT: Best Effort

- Virtual channels: 5 sources, 2 destinations, 2 routes to each destination ⇒ 20 channels
- VC + segmentation in 256 B chunks
  ⇒ delay divided by 16
- Channel priorities are not interesting here
- Flow control: does not slow transmissions if reception buffers are big enough

SpaceWire-RT: Assured

- Acknowledgments may be useful to detect the failure of a route
- Retries not useful here
- On each source: redundant interfaces are independent ⇒ no automatic switching
- ACK used to warn the apps that a route is down so that they can switch

➡ Delays in the same order as for BE

SpaceWire-RT: Reserved

- 36 slots necessary to transmit all the traffic
- we can give the SMU almost 1/2 the slots
- worst-case delay = 3 * length of a slot
- flow control: no slowdown because a phase is reserved in each slot
- Apps have no way to determine if a route is valid or not (maybe Flow Control ?)

Summing up

	SpaceWire standard	Best Effort	Assured	Reserved	Guaranteed
Worst-case delay for SMU packets					
Switching between redundant routes					

- Assured/Guaranteed useful only because of the ACK
- Synchronous mode gives worse results than asynchronous mode on a simple network

Conclusion

- Assured/Guaranteed classes do not reduce worst-case delays
- Automatic redundancy best done at the application level
- ACK are useful to warn an application that a route is down

➡ Suggestion: merge Assured/Guaranteed with BE/Reserved by adding ACK as as optional feature