Satellite Networks: Spacewire

Jérôme Ermont
jerome.ermont@n7.fr

INPT ENSEEIHT

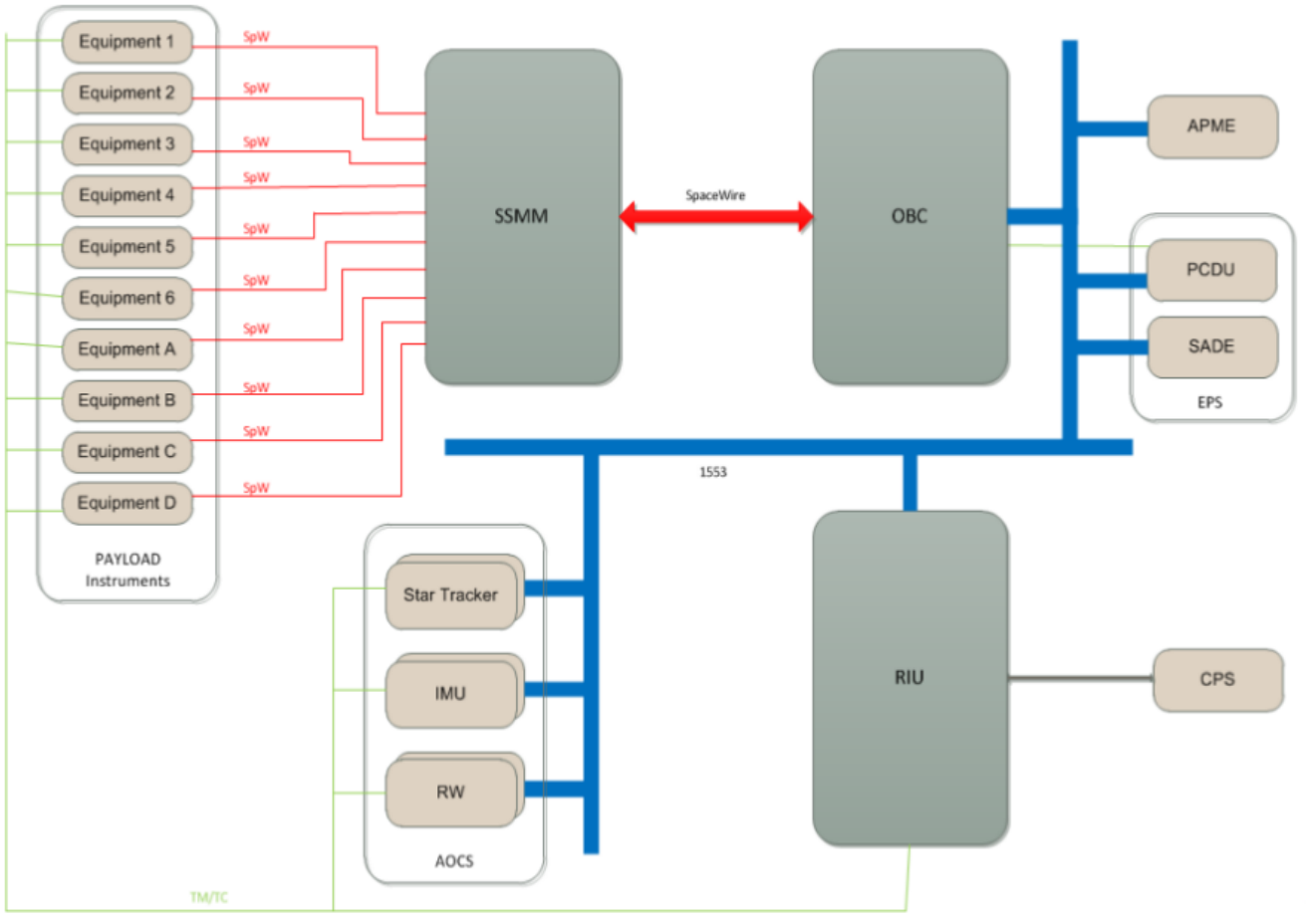
3SN SEMBIIOT

2020-2021

Plan

Agenda

Nowadays typical scientific spacecraft architecture



(Source: O. Notebeart, Airbus Defence and Space)

On-board communications recall

Vehicule guidance and control

- Data related to vehicule guidance, navigation, orbital control, altitude,
- Orientation of antennas and instruments,
- **Low data rate** but **high determinism**.

Payload data transfer

Data measured by the various instruments have to be transferred to mass memory.

- Data are high resolution images, radar data, etc. . .
- Requires large throughput from instruments to mass memory.
- Data flow is continuous, generally.

Communication constraints recall

Contraints on data transfer

- Compression and storage of instrument data is mandatory.
- Satellite functions may require a common clock: synchronization required.
- Security and robustness: ciphering, command authentification, no computing errors.

Resistance to radiations

- Space environment is harsh: electronic age faster with space particules blasts, messages get corrupted (random error).
- New material, resistant to cosmic radiations, are required (e.g. SOI)
- Complex system and expensive!

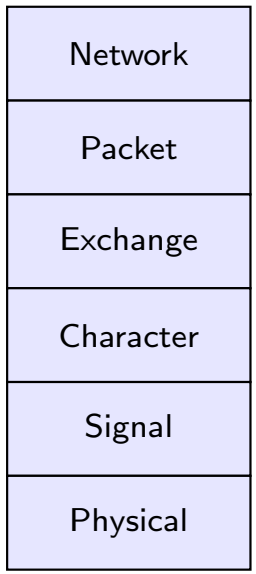
Agenda

Spacewire

- Embedded network for satellite developped by ESA and Dundee University
  - Inspired by IEEE 1355
- High speed point to point serial links: 200 Mbps
- Low energy consumption
- Adapted to space constraints

6 layers of the Spacewire network architecture

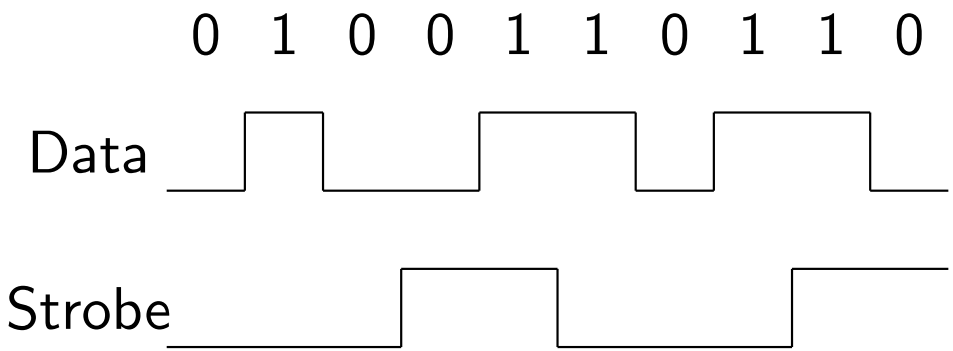
- correspond to the 3 low layers of OSI model
- each layer uses the functionnalities of the under layer



Spacewire

Physical and signal layers

- LVDS (Low Voltage Differential Signalling) $\Rightarrow$ low energy consumption
- Data-Strobe Encoding
- Full duplex $\rightarrow$ 4 twisted pairs



Spacewire

Character layer

- Transmission Data Unit
- 2 types:
  - Data: 10 bits
    - basic element of the packets
  - Control: 4 bits
    - to control the transmission on the network
    - FCT : Flow Control Token / ESC : Escape
    - EOP : End of Packet / EEP : Error End Of Packet

Flow control

- For each link
- Receptor sends grant character to the sender
- One grant message for each 8 bytes (FCT)
- 56 bytes max. can be granted
  - Input ports of the equipments: 64 bytes buffers
- FCT characters are sent before data characters

Packet format and address

- No constraint on the data length
- Logical addressing: each port has a network address, 224 possible addresses

Wormhole Routing

Network elements connected through routers

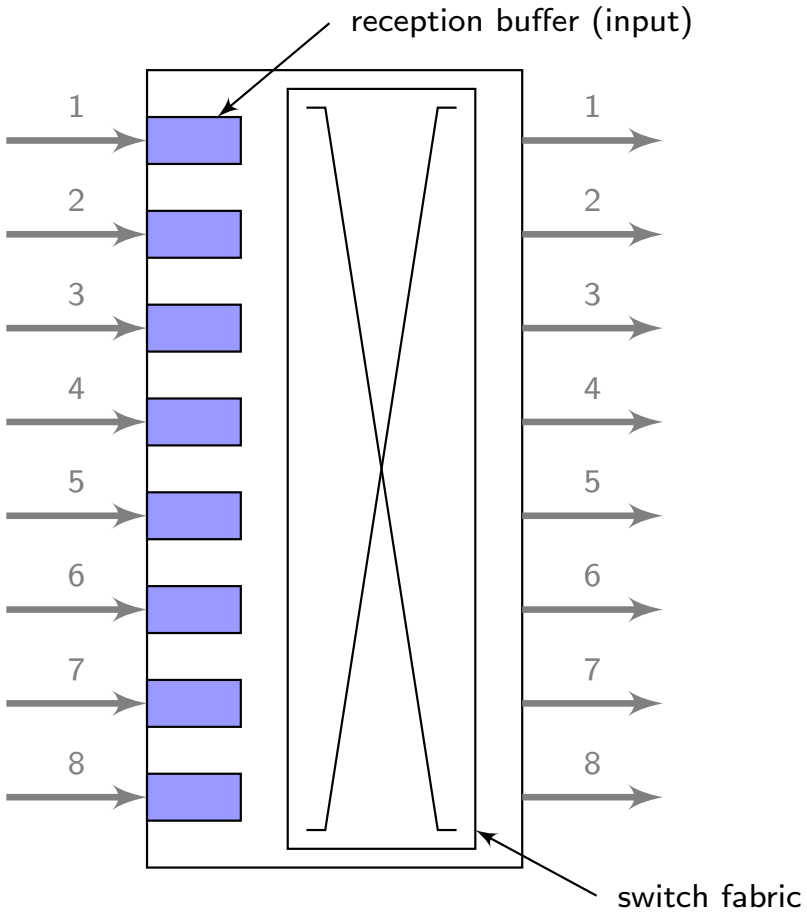
Goal

To increase the data transmission without adding more point to point links

Constraints

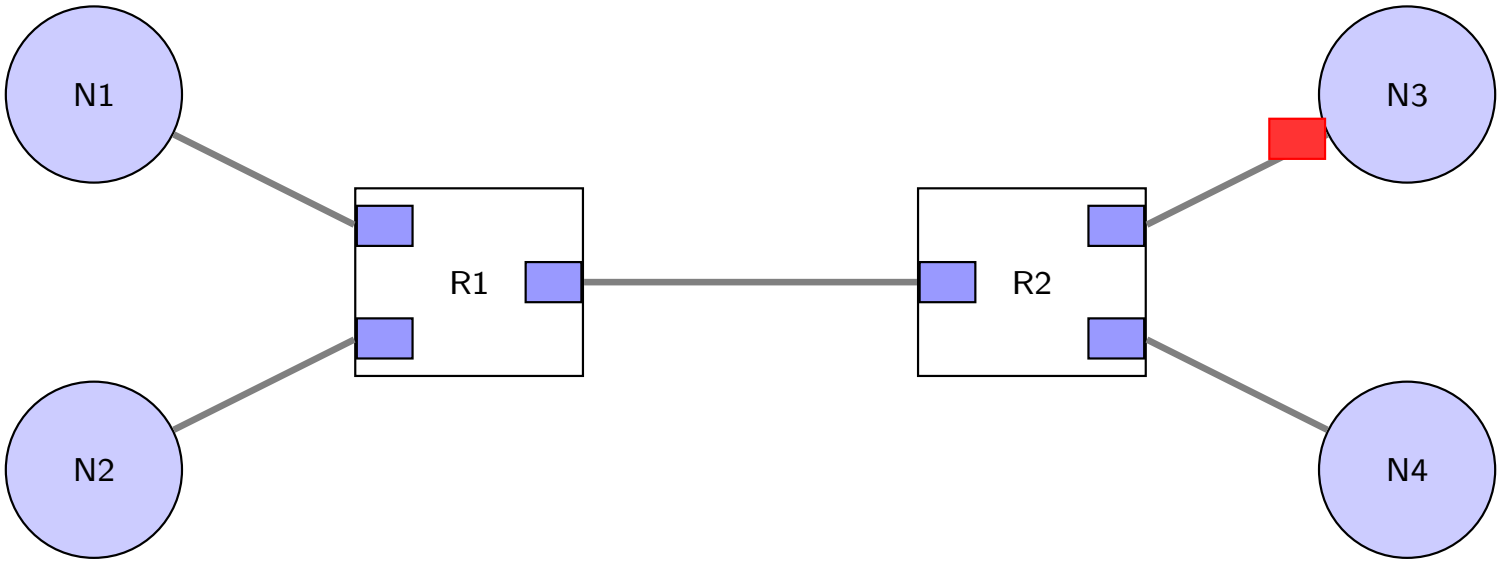
- Do not use lot of memory because of the cost of protected memory
- Compliant to the point to point link

SpaceWire Router



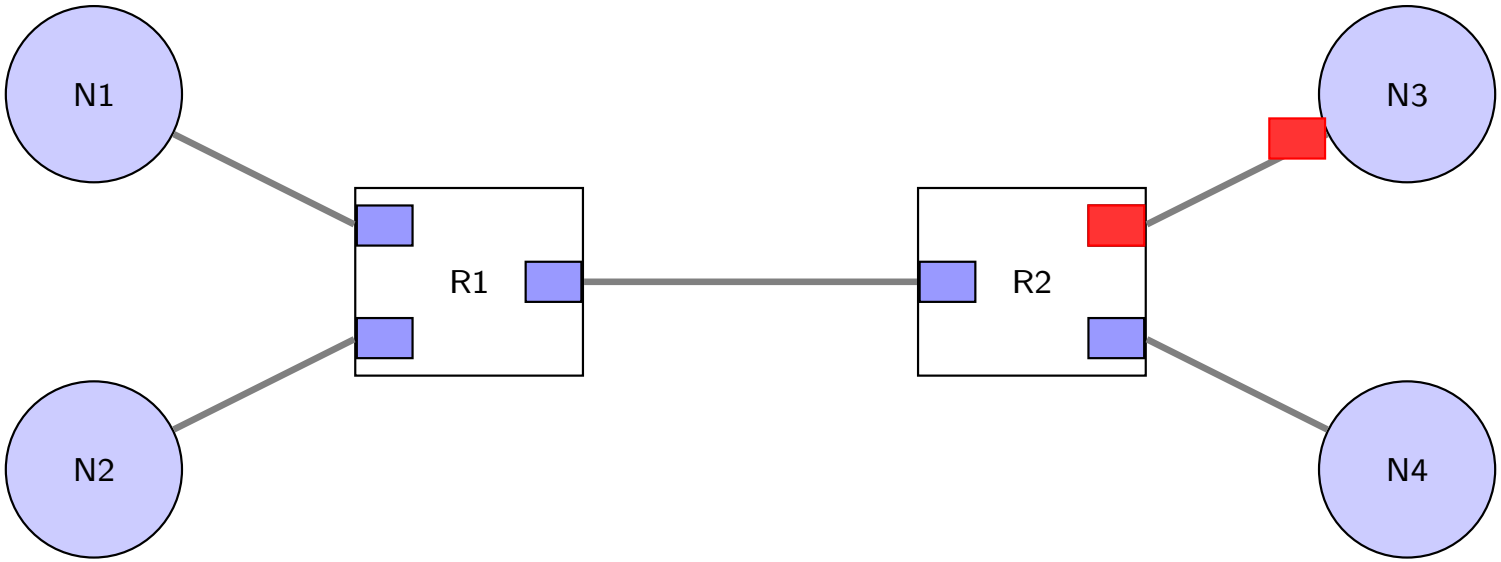
- Reception buffers: 64 buffers

Wormhole Routing behaviour

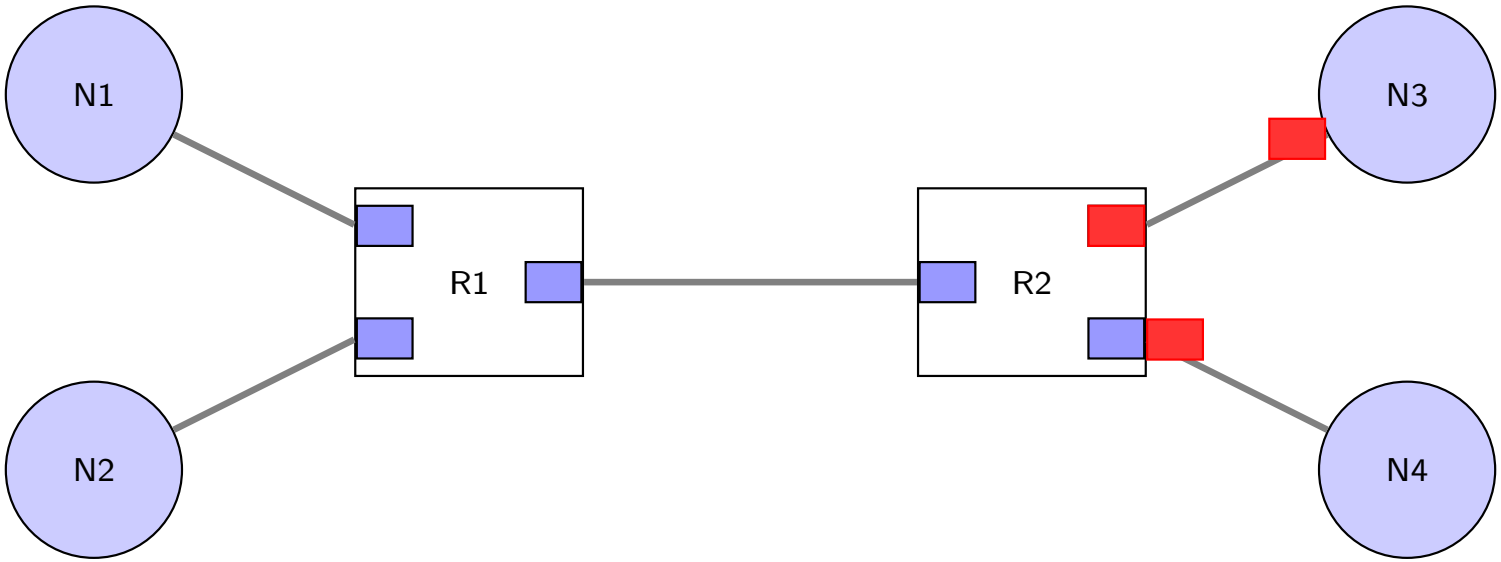


N3 sends a packet to N4

Wormhole Routing behaviour

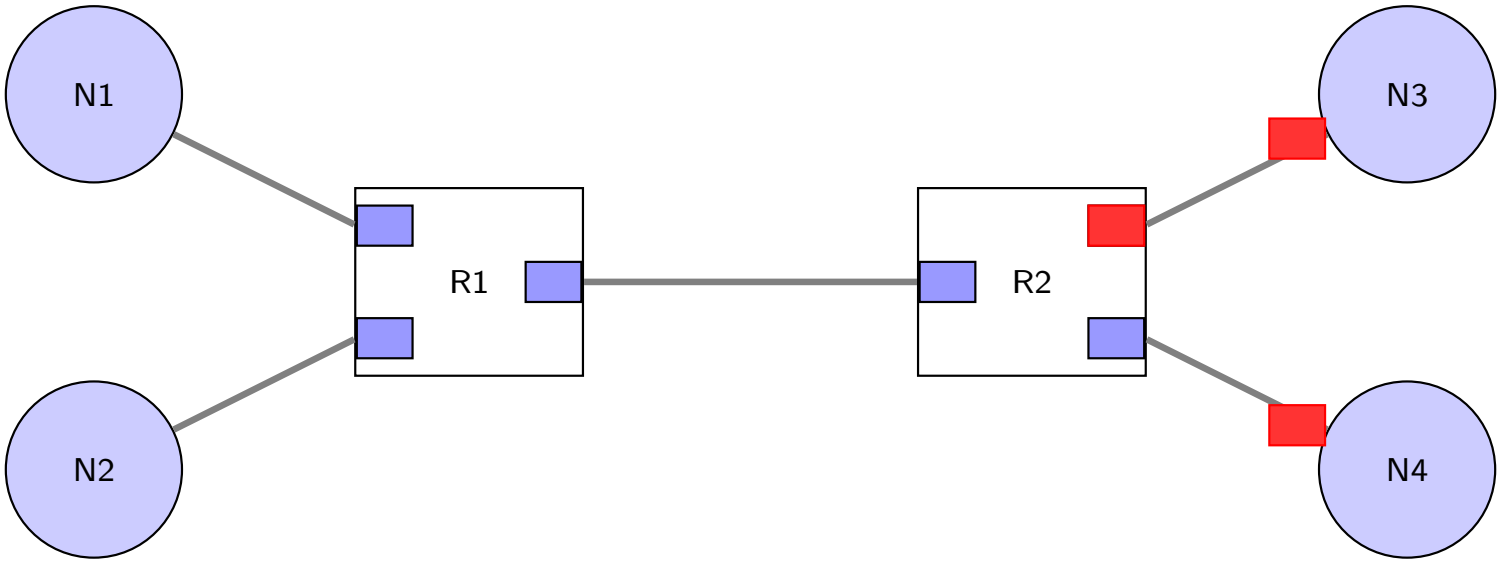


Wormhole Routing behaviour



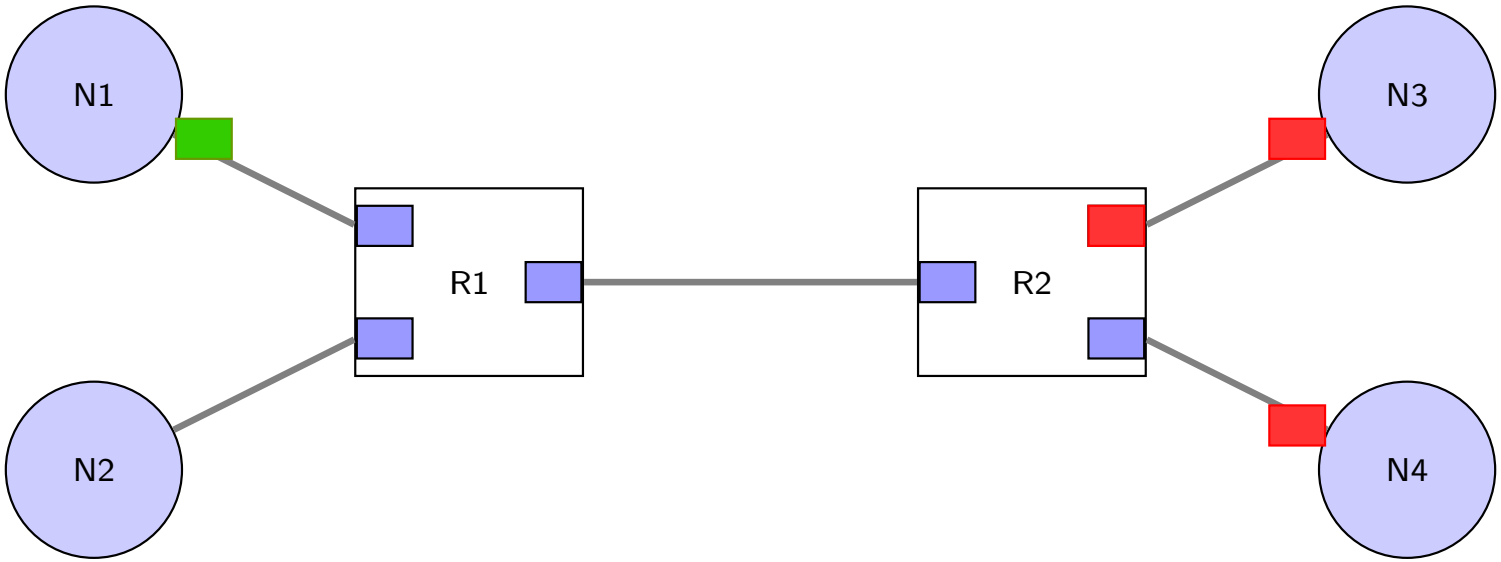
The output of R2 is reserved to the packet of N3

Wormhole Routing behaviour



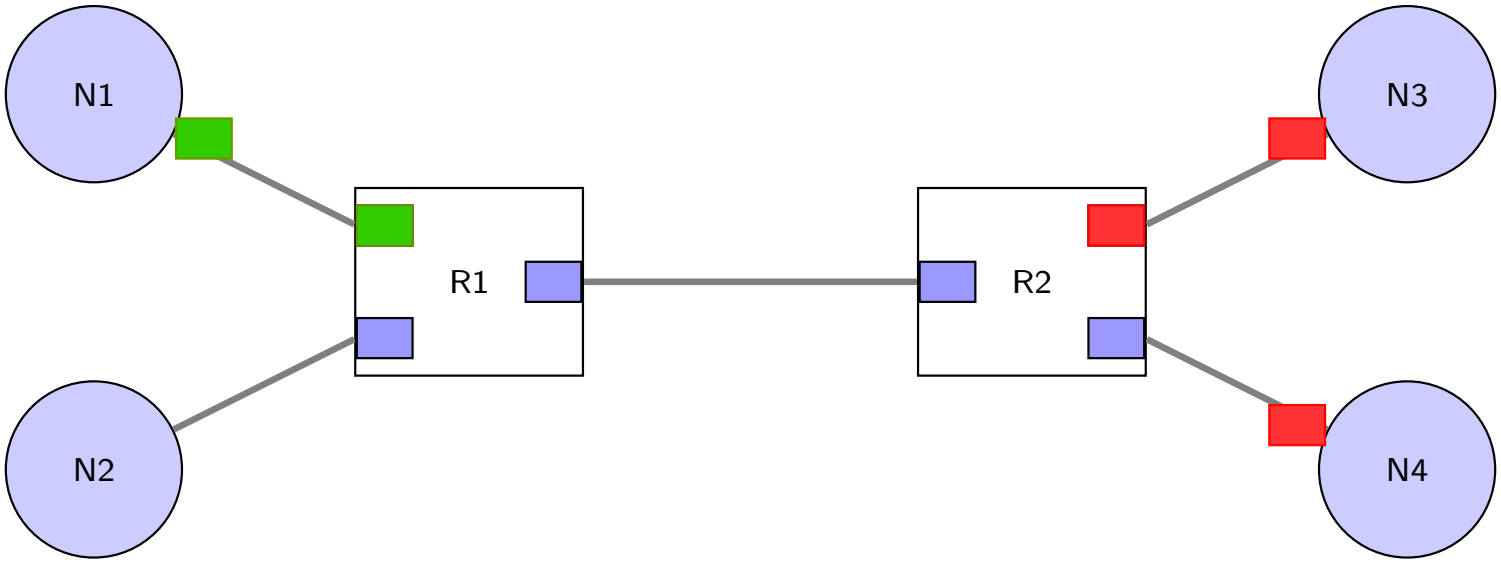
Les données du paquet de N3 est transmit au fur et à mesure par R2

Wormhole Routing behaviour

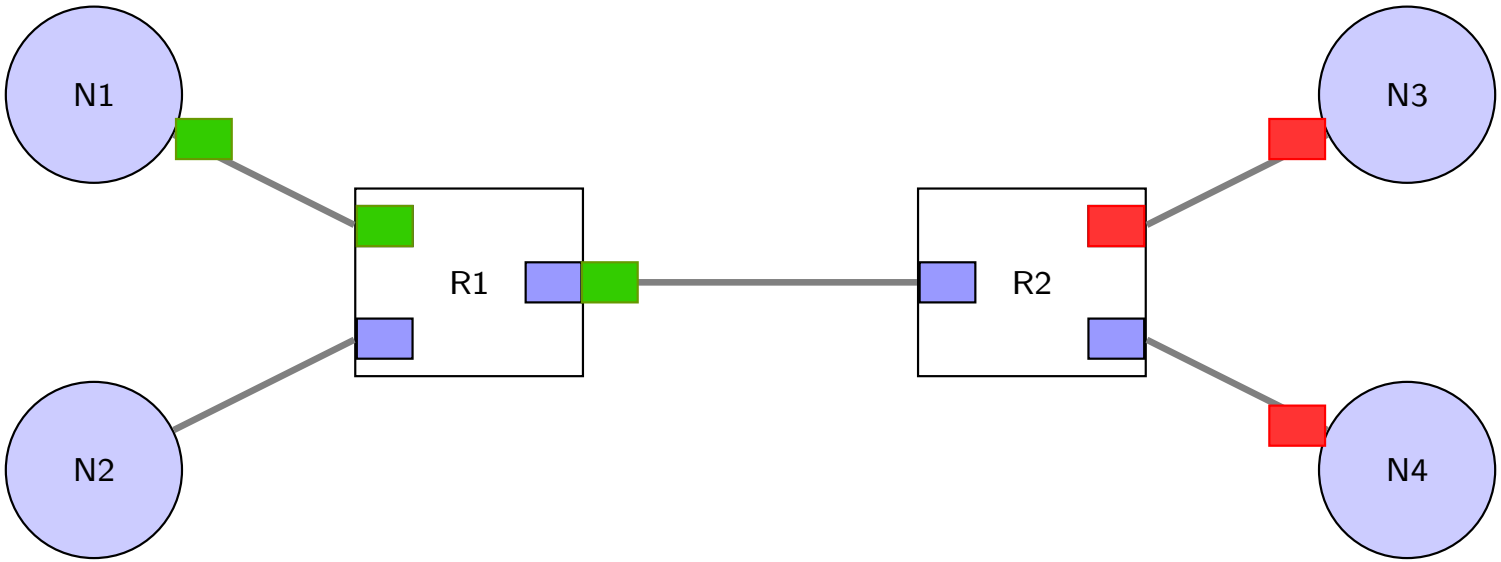


N1 sends a packet to N4

Wormhole Routing behaviour



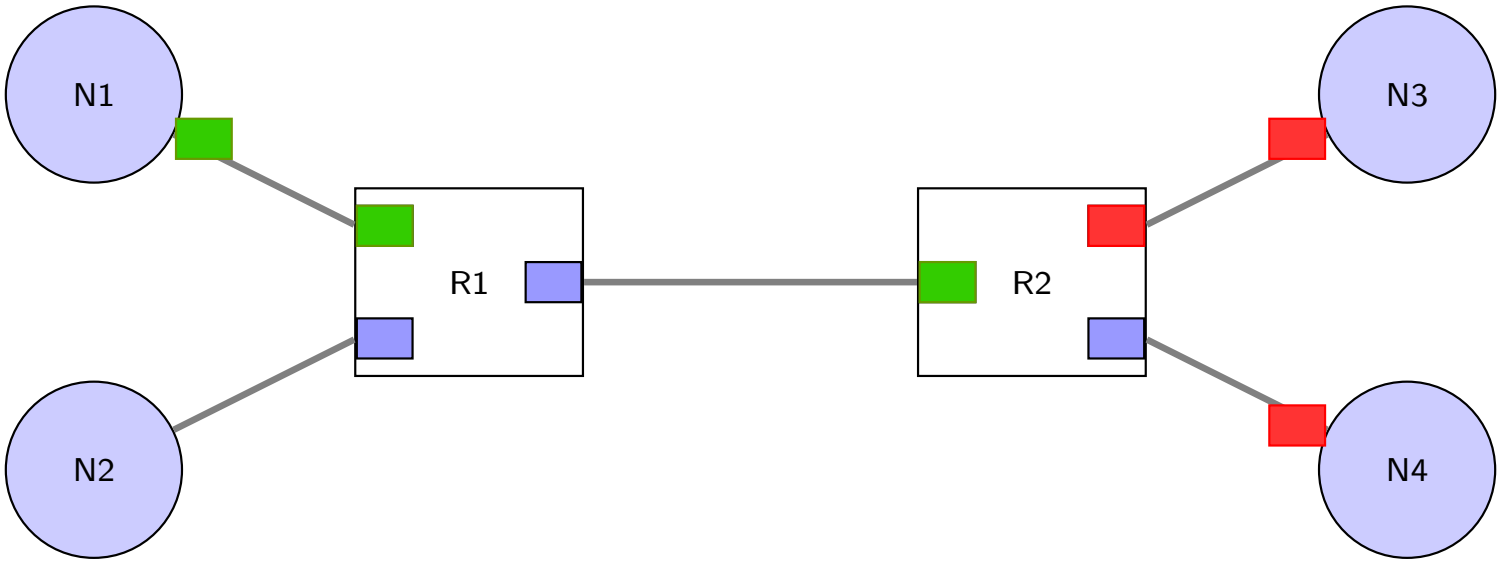
Wormhole Routing behaviour



The corresponding reception buffer of R2 is free $\Rightarrow$ the packet from N1 can be stored in R2

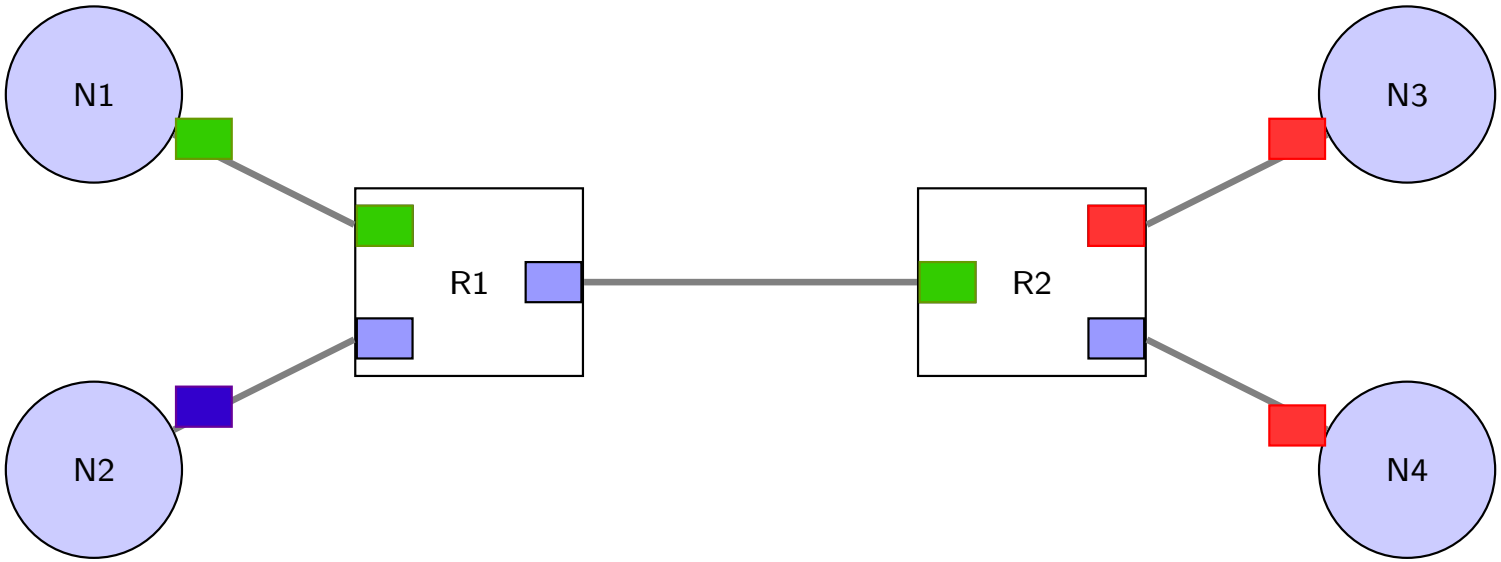
The output port of R1 is reserved for the packet of N1

Wormhole Routing behaviour



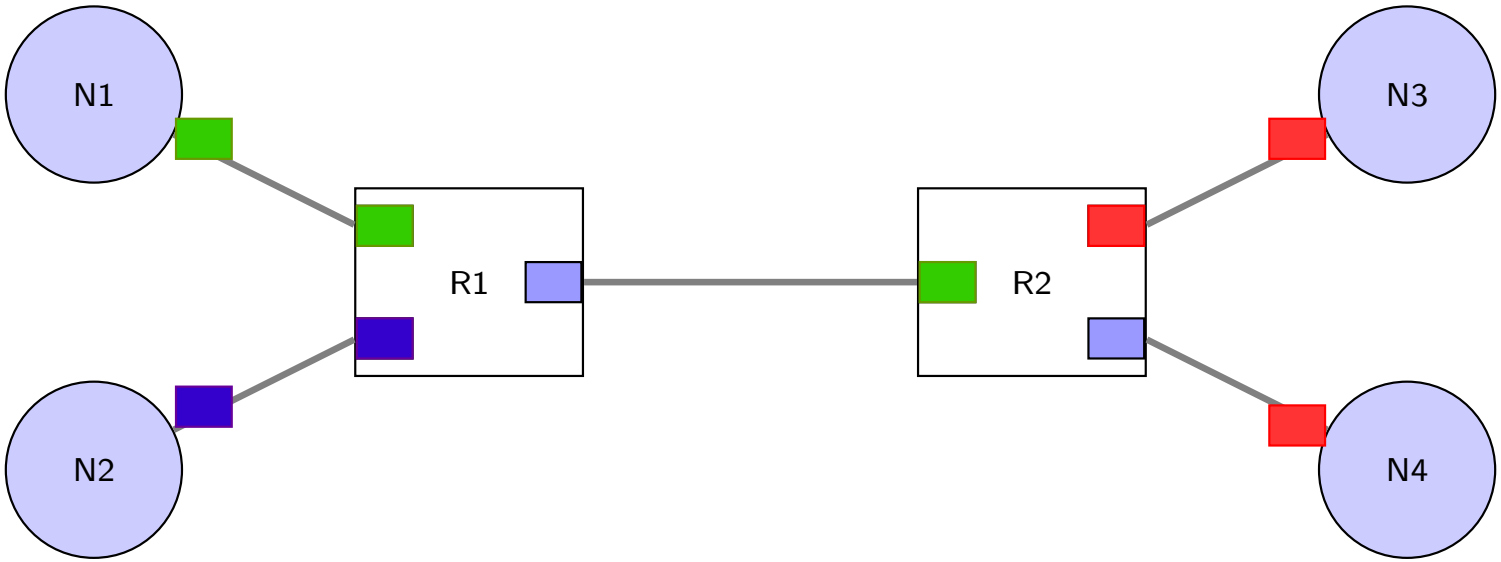
But the packet of N1 is blocked in R2

Wormhole Routing behaviour



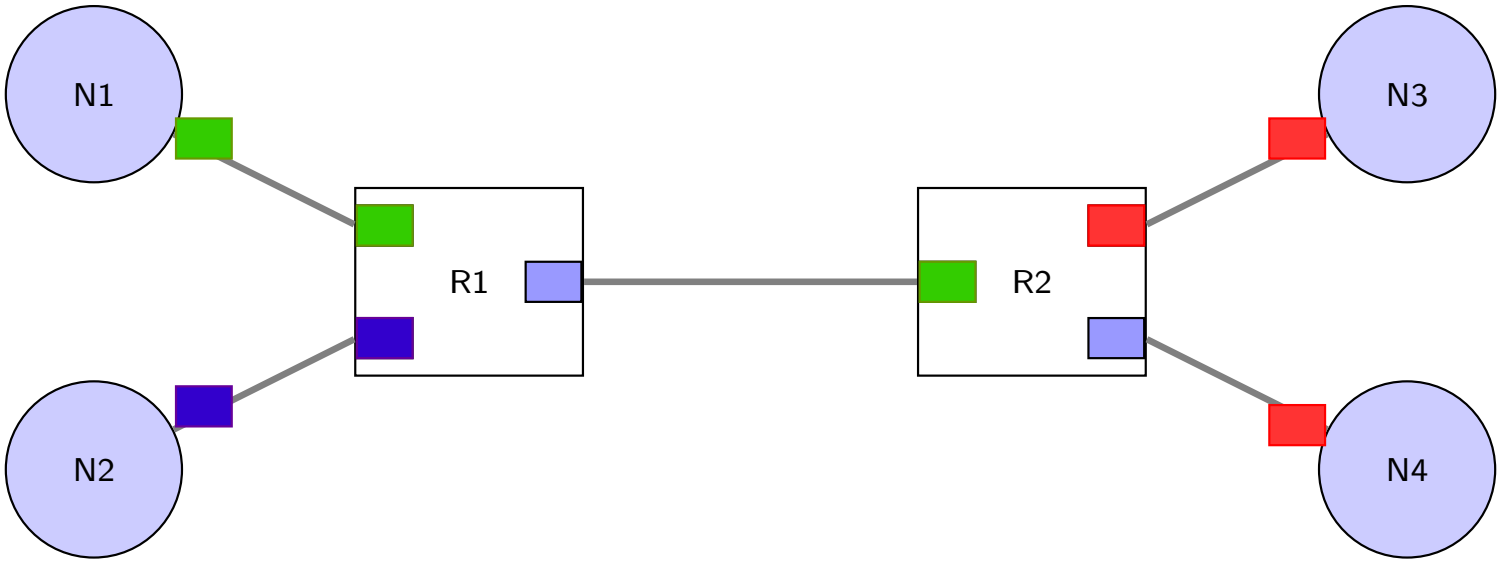
N2 sends a packet to N4

Wormhole Routing behaviour

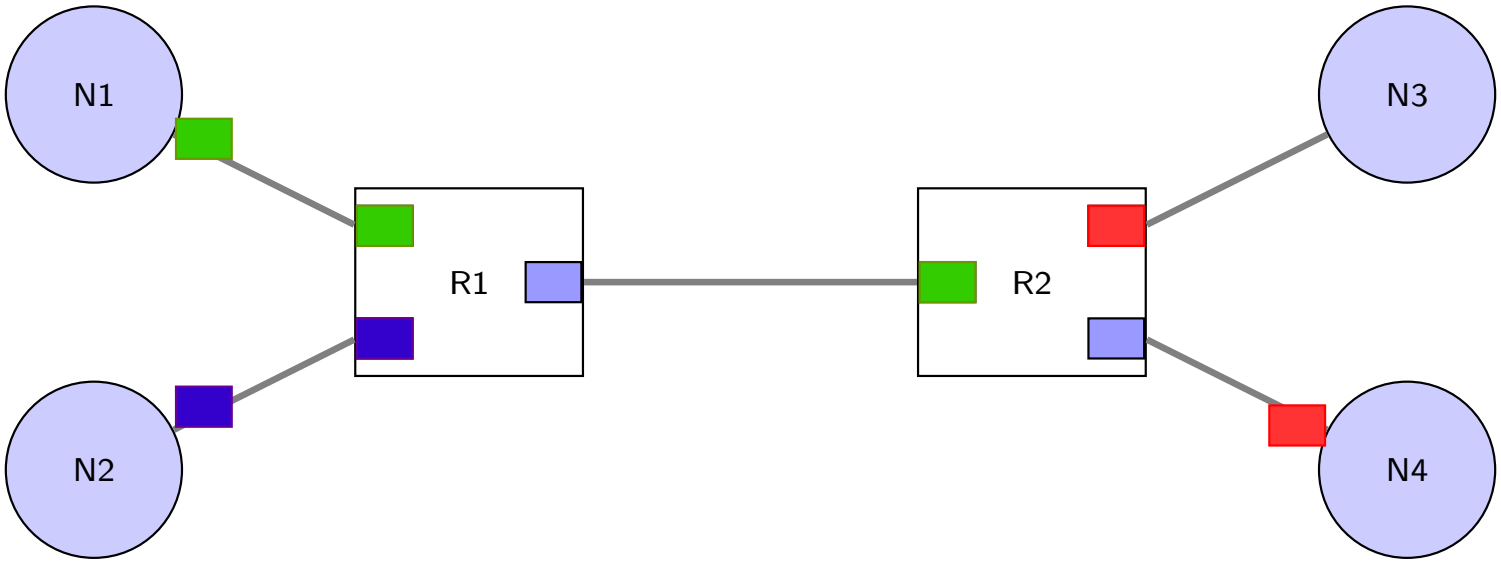


The packet from N2 is blocked in R1

Wormhole Routing behaviour

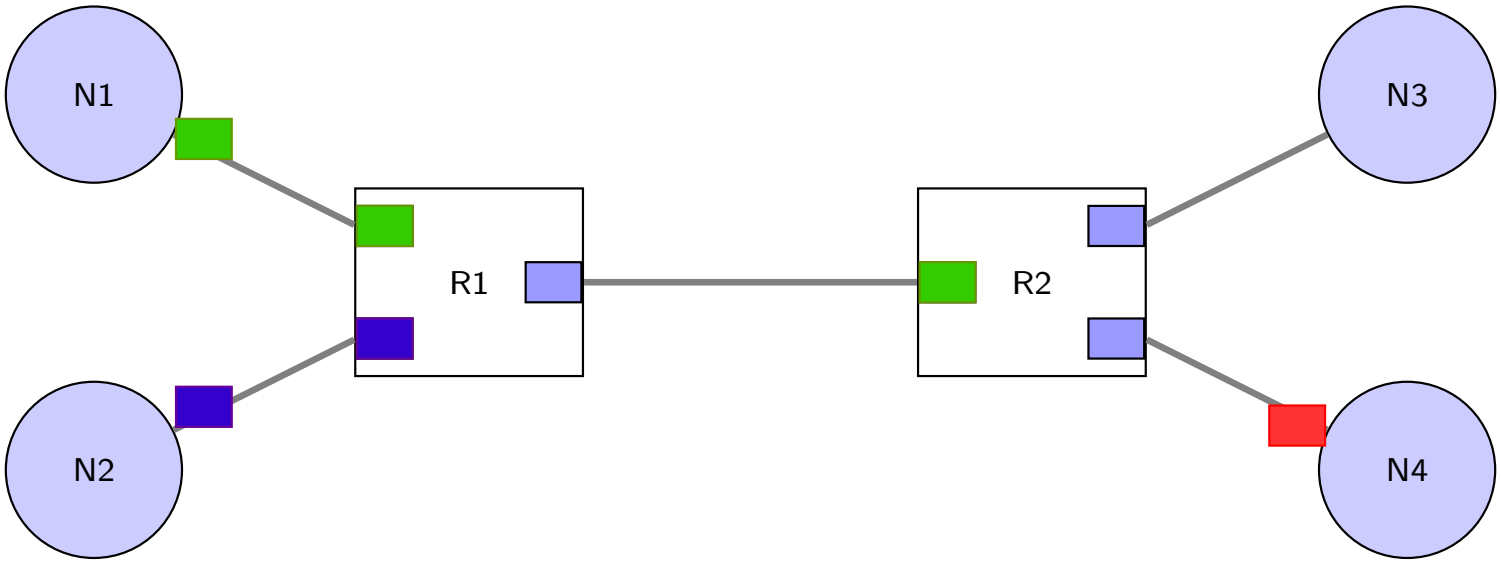


Wormhole Routing behaviour



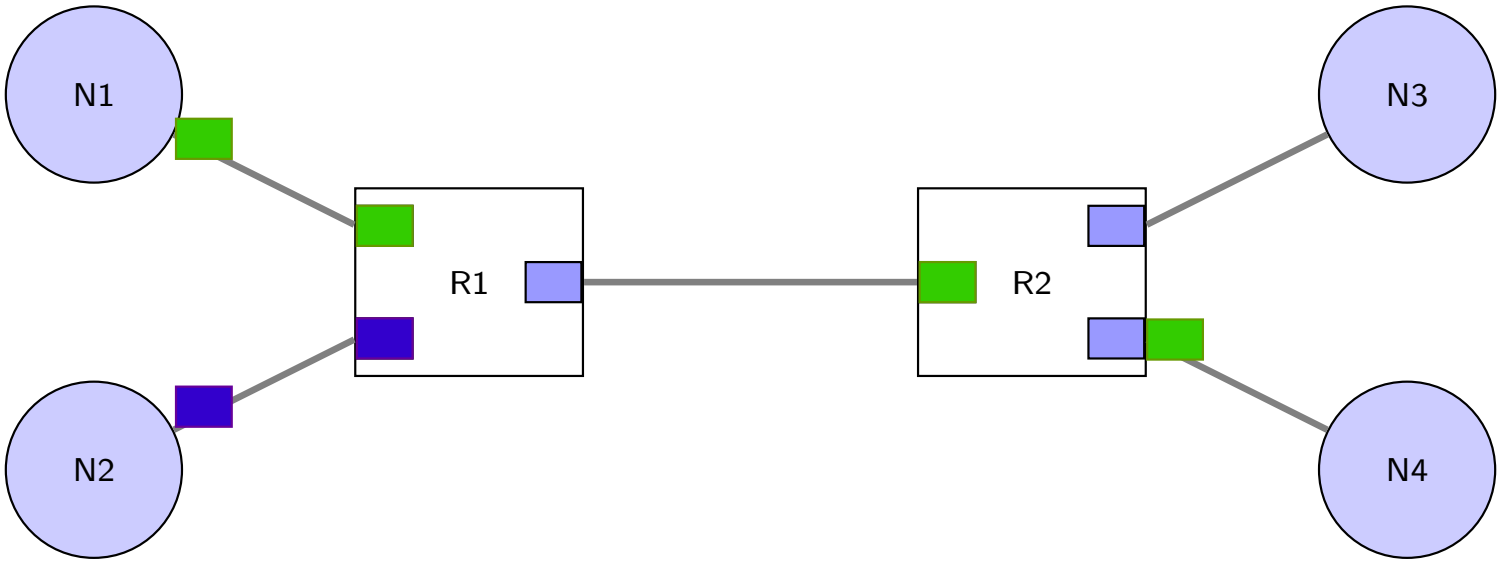
The packet from N3 is completely received

Wormhole Routing behaviour



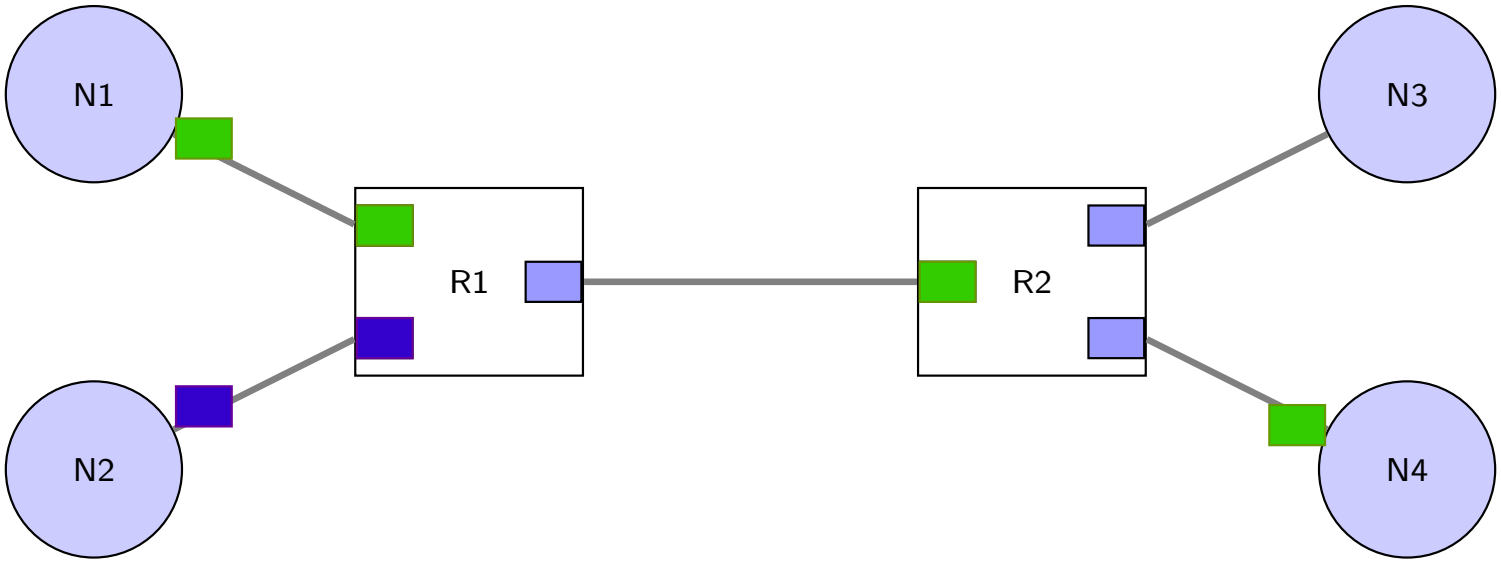
The output port of R2 becomes free $\Rightarrow$ the packet from N1 can progress in R2

Wormhole Routing behaviour

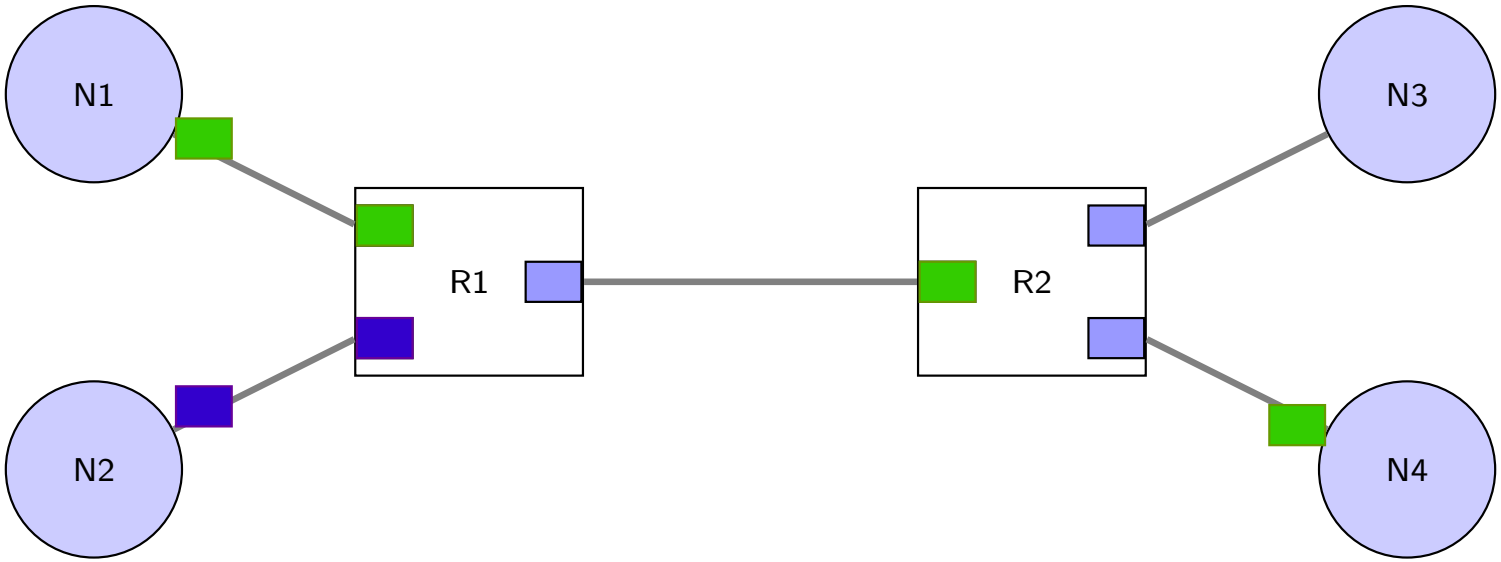


The packet from N1 is transmitted to N4

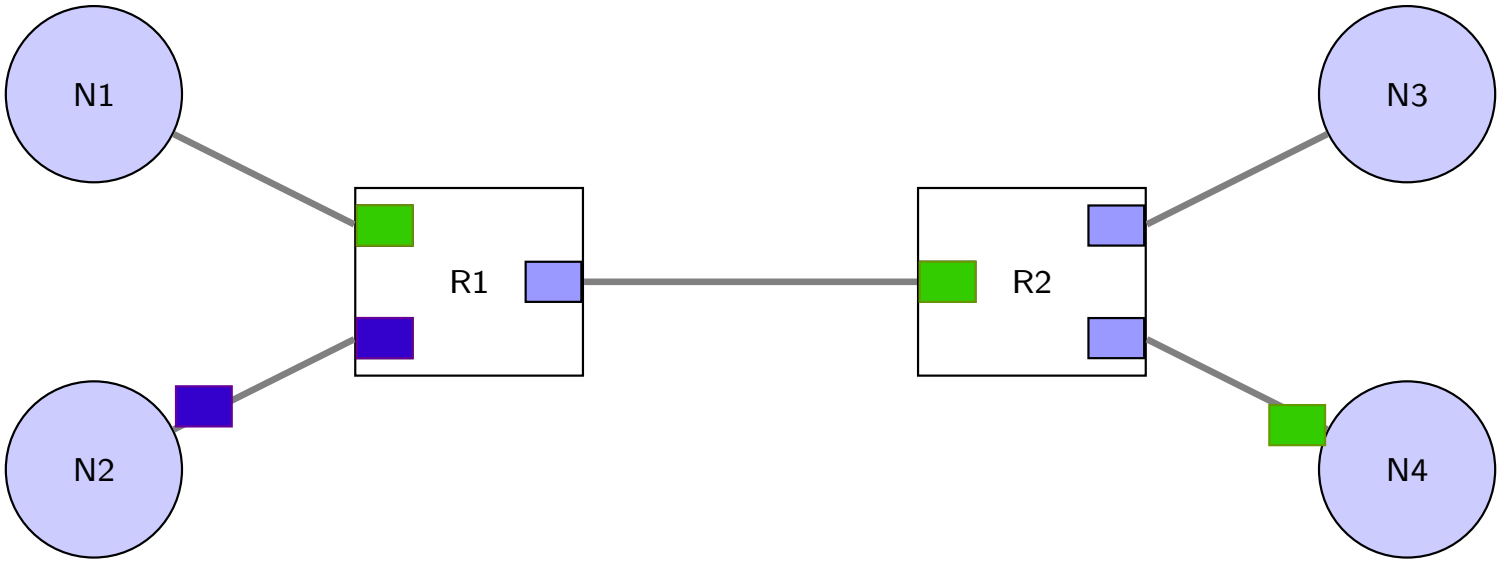
Wormhole Routing behaviour



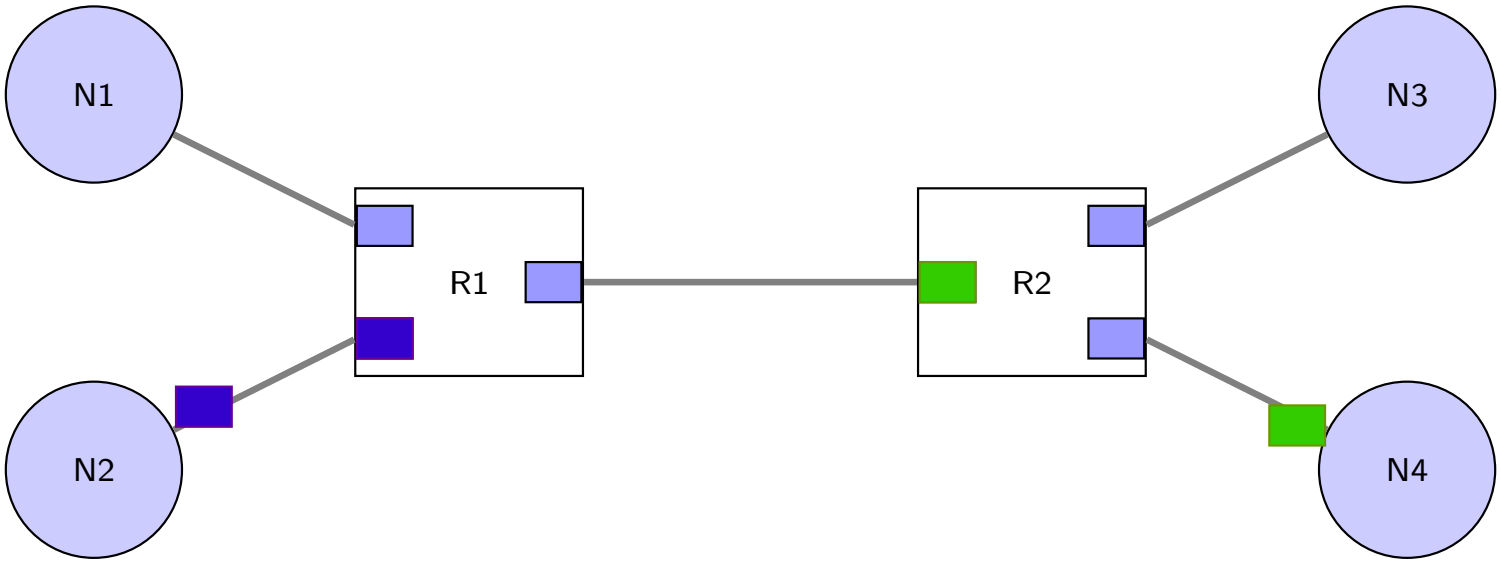
Wormhole Routing behaviour



Wormhole Routing behaviour

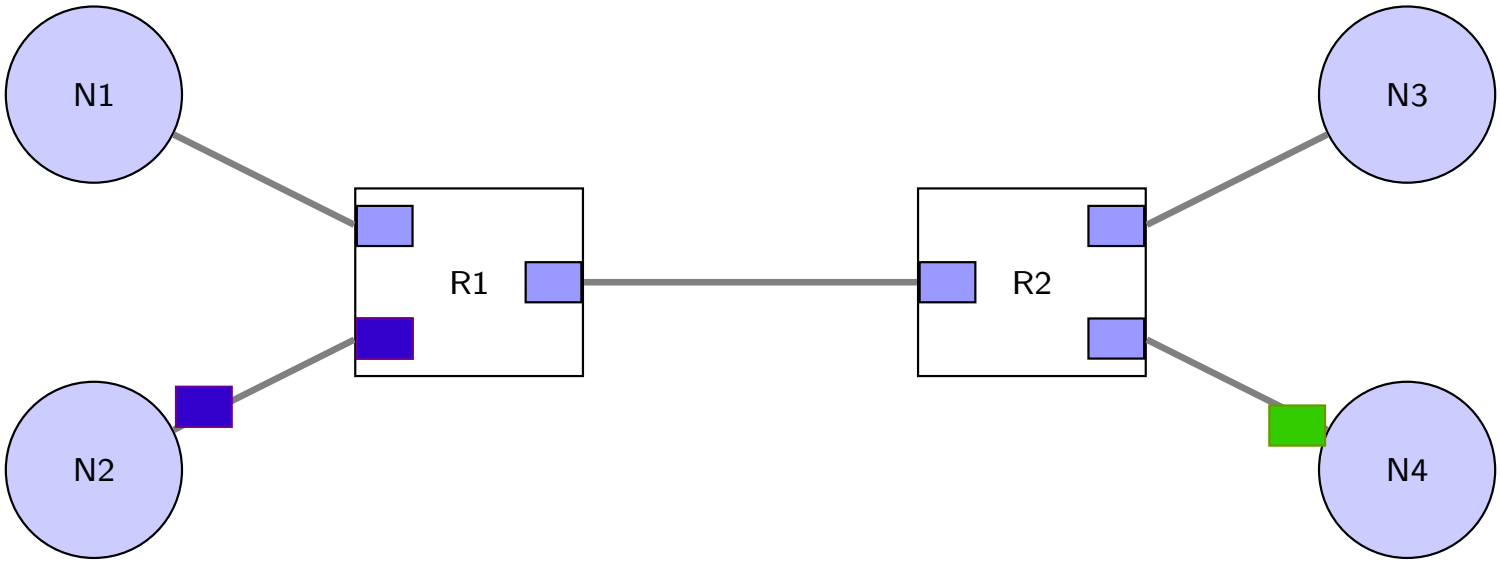


Wormhole Routing behaviour



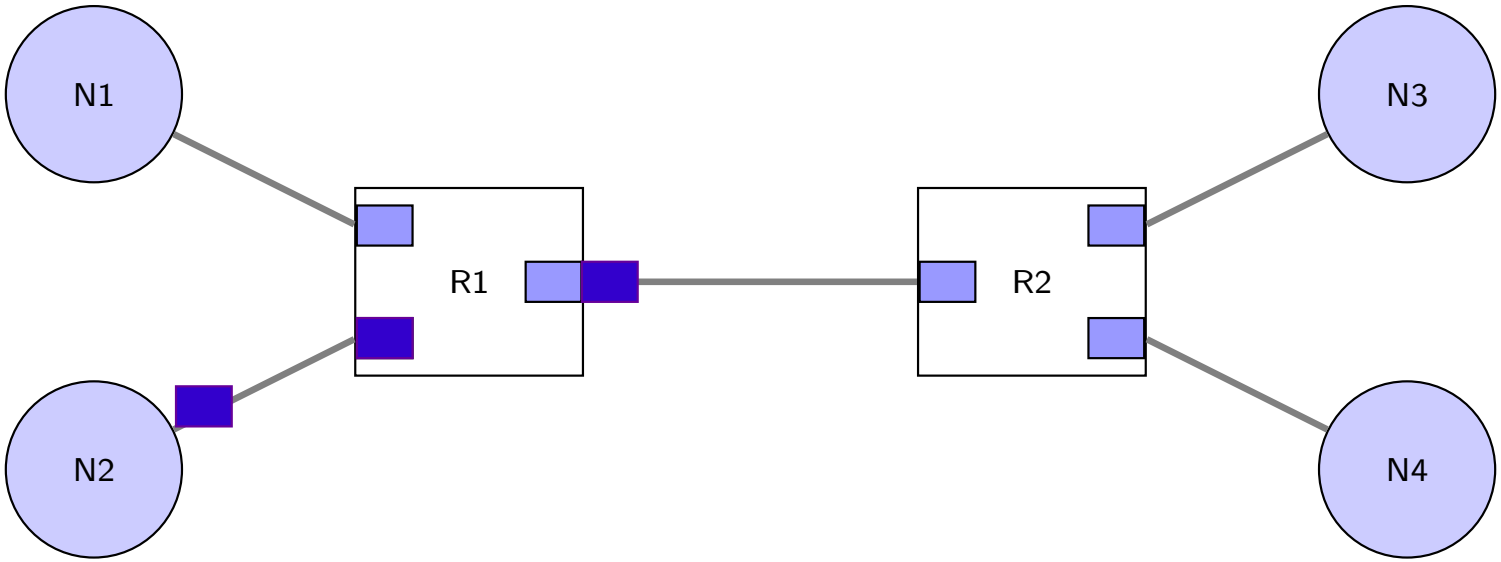
The packet from N1 is completely received

Wormhole Routing behaviour

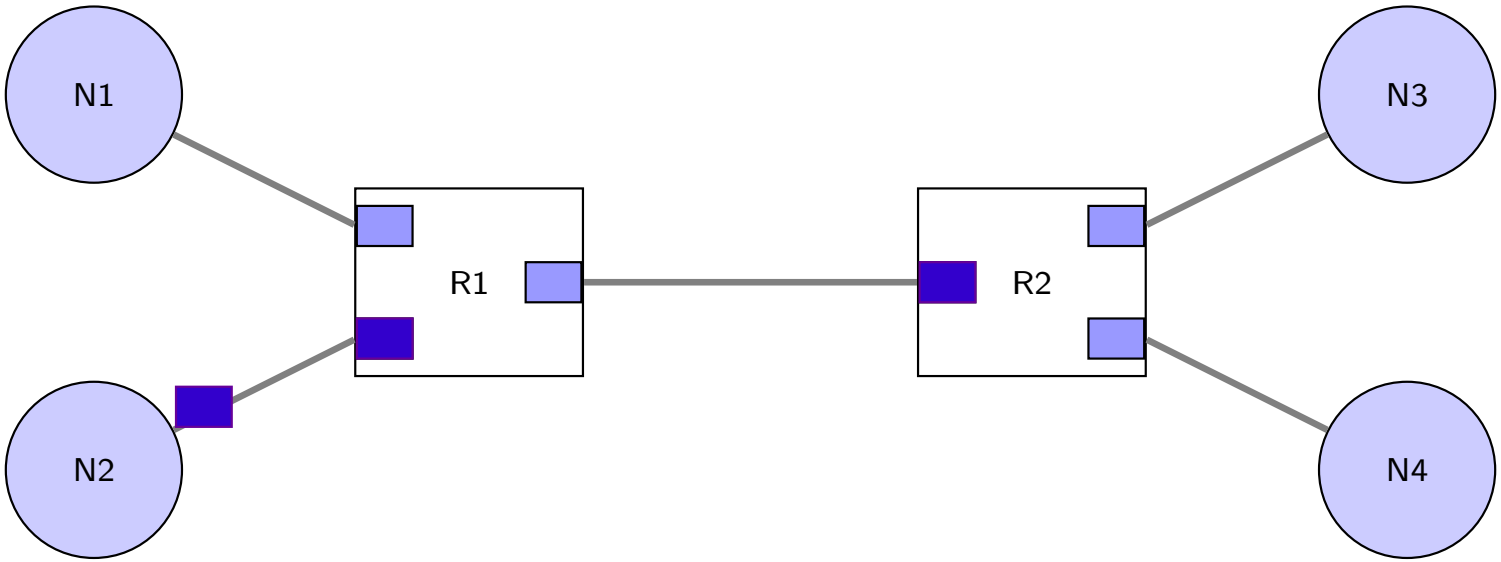


The reception buffer of R2 is now free $\Rightarrow$ the packet from N2 can progress to R2

Wormhole Routing behaviour

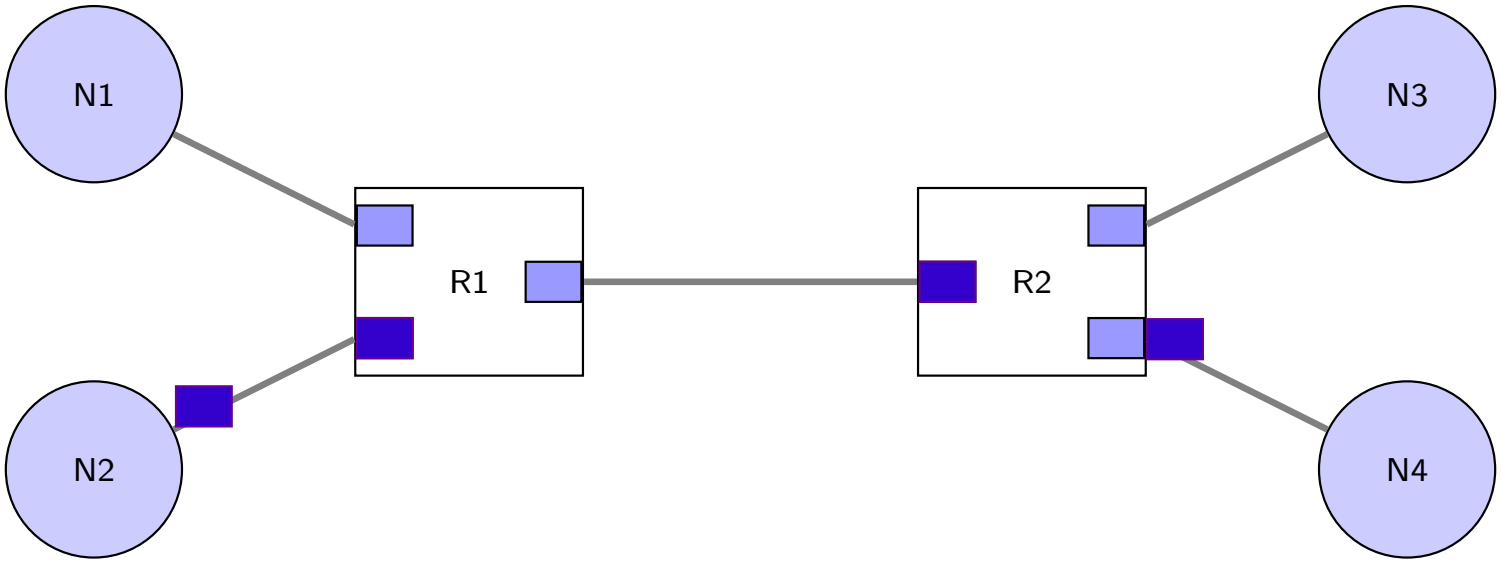


Wormhole Routing behaviour

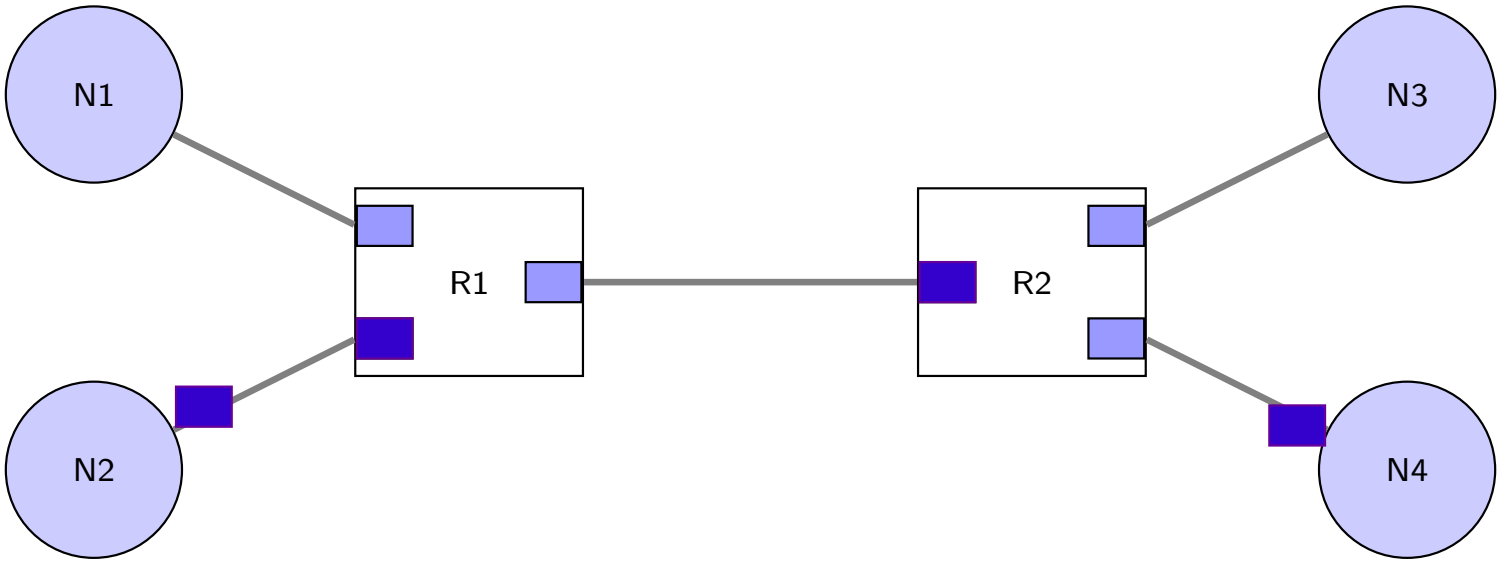


The packet from N2 is stored in the reception buffer
The output port is free, the packet progress to N4

Wormhole Routing behaviour



Wormhole Routing behaviour



The packet from N2 is transmitted to N4

Wormhole Routing

Pros

- Low memory needs
- Low latency in general
- Easy to implement

Cons

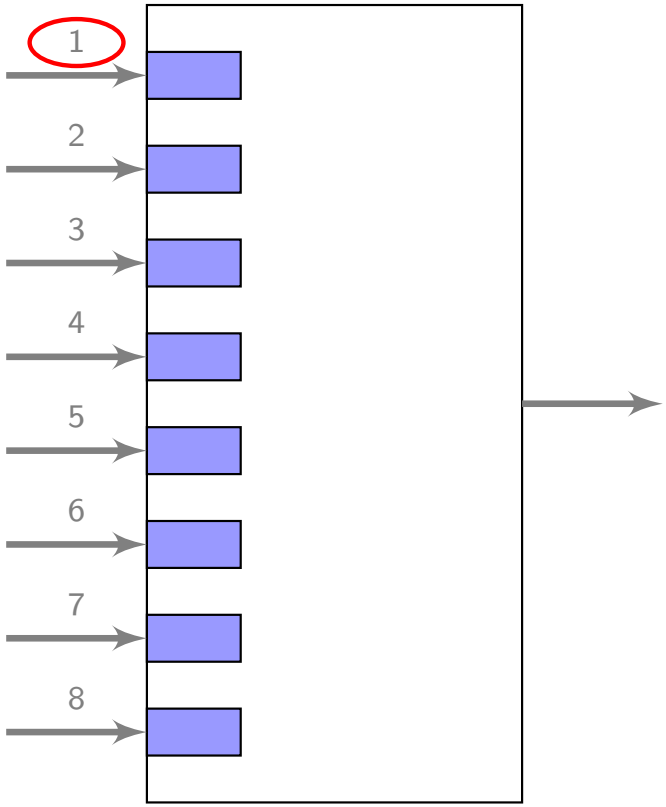
- Blocking flows
- Blocking the network
- Important and irregular delays

Output port access policy

Packets from different sources try to access to the same output port

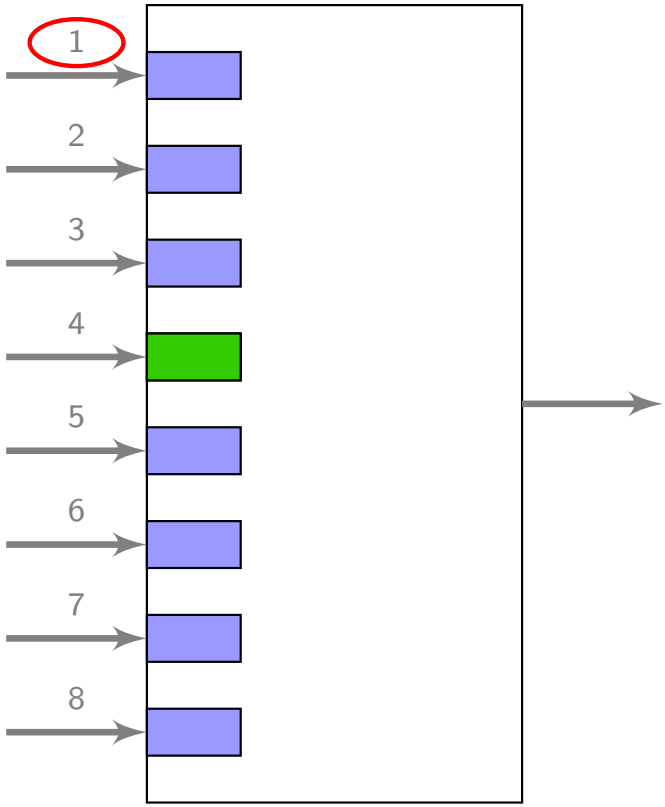
- Static priorities on destination addresses
  - 2 priority level
  - no preemption
- "rotating priority" mechanism

Rotating priorities



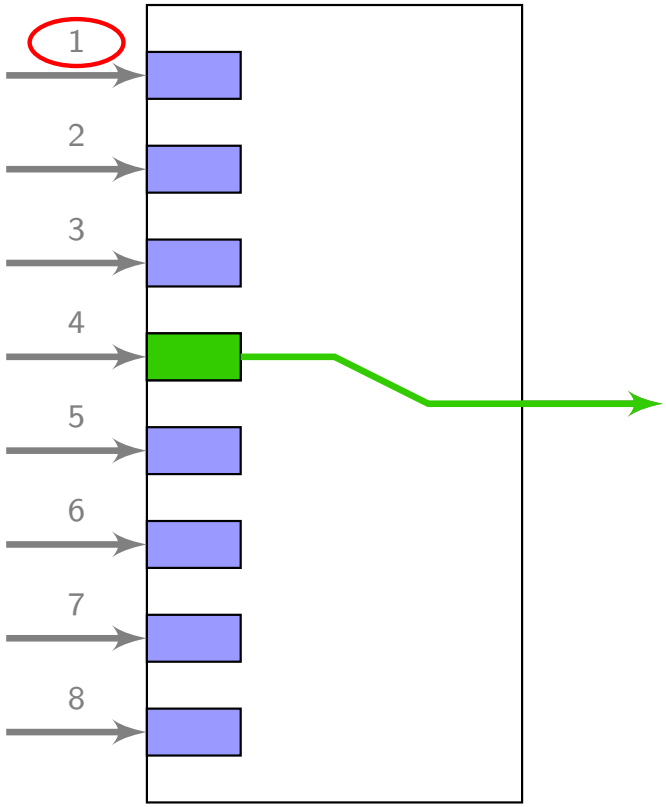
Priority to input port 1

Rotating priorities



Data in reception buffer of input port 4
Reception buffers of ports 1, 2 and 3 are free

Rotating priorities



Transmission of the green flow to the output port

Rotating priorities

Data in ports 3 and 6

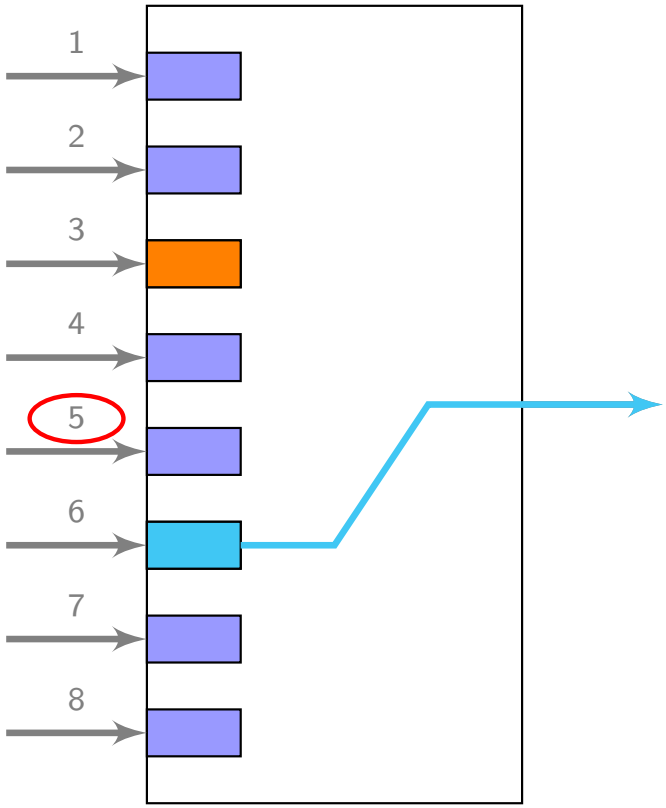
Rotating priorities

Green flow has been transmitted

The output port becomes free

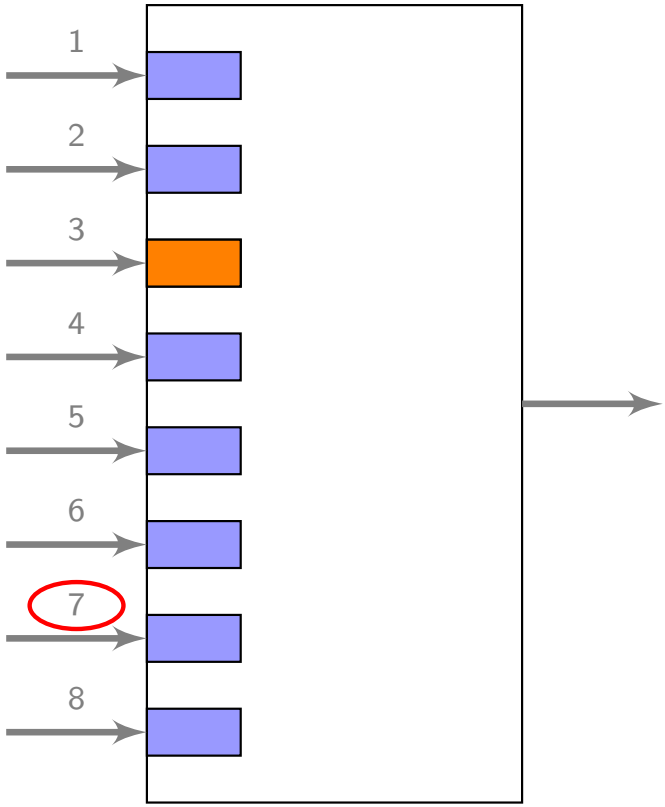
Priority to the next port following port 4

Rotating priorities



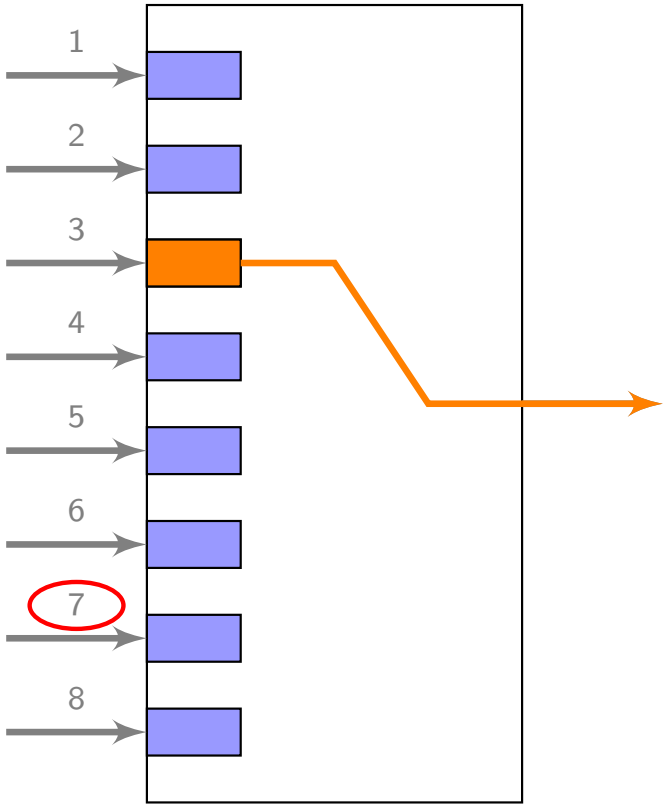
No data in port 5
Data of 6 are transmitted

Rotating priorities



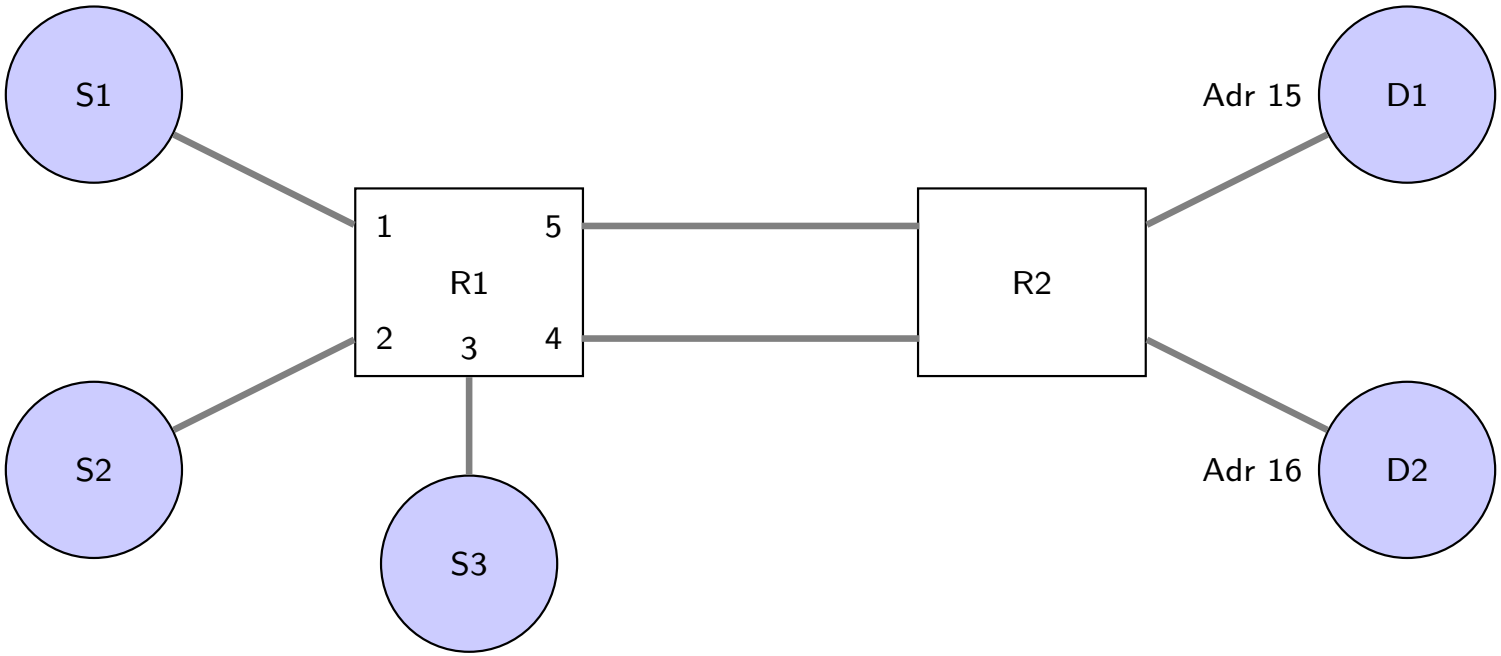
All the data of port 6 has been transmitted
Priority to port 7

Rotating priorities



Waiting data of port 3 are then transmitted

Grouping Adaptative Routing

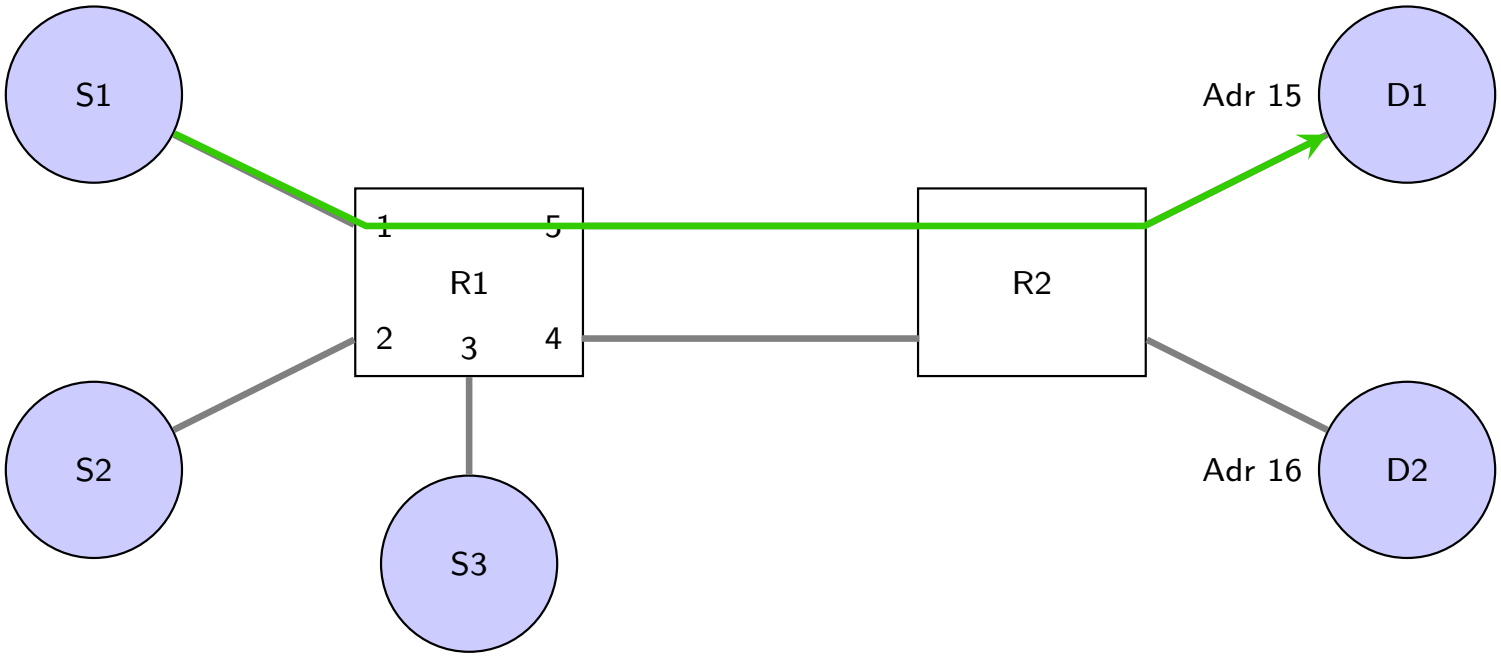


- Goal : to use all the links that interconnect 2 routers
- Routers automatically distribute the traffic

Routing table of R1

Addr	Port 1	Port 2	Port 3	Port 4	Port 5
15	0	0	0	1	1
16	0	0	0	1	1

Grouping Adaptative Routing

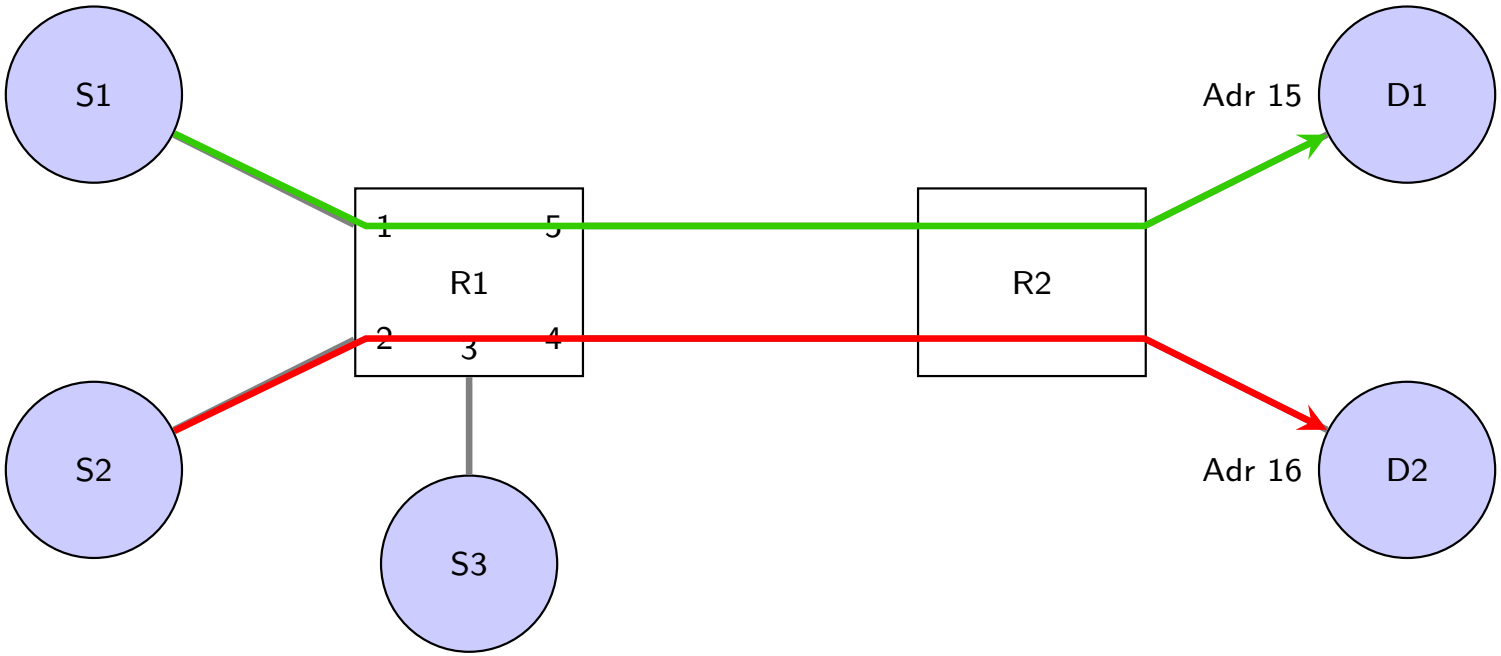


- Goal : to use all the links that interconnect 2 routers
- Routers automatically distribute the traffic

Routing table of R1

Addr	Port 1	Port 2	Port 3	Port 4	Port 5
15	0	0	0	1	1
16	0	0	0	1	1

Grouping Adaptative Routing

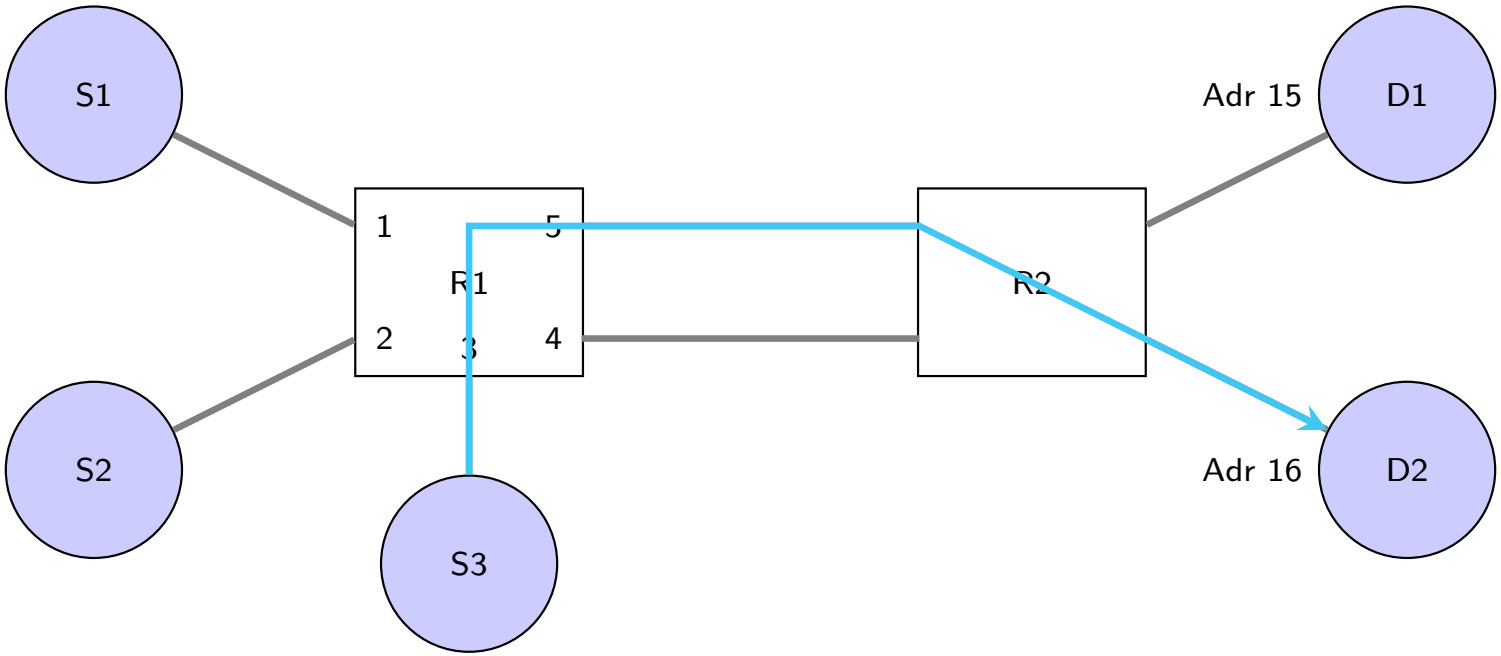


- Goal : to use all the links that interconnect 2 routers
- Routers automatically distribute the traffic

Routing table of R1

Addr	Port 1	Port 2	Port 3	Port 4	Port 5
15	0	0	0	1	1
16	0	0	0	1	1

Grouping Adaptative Routing

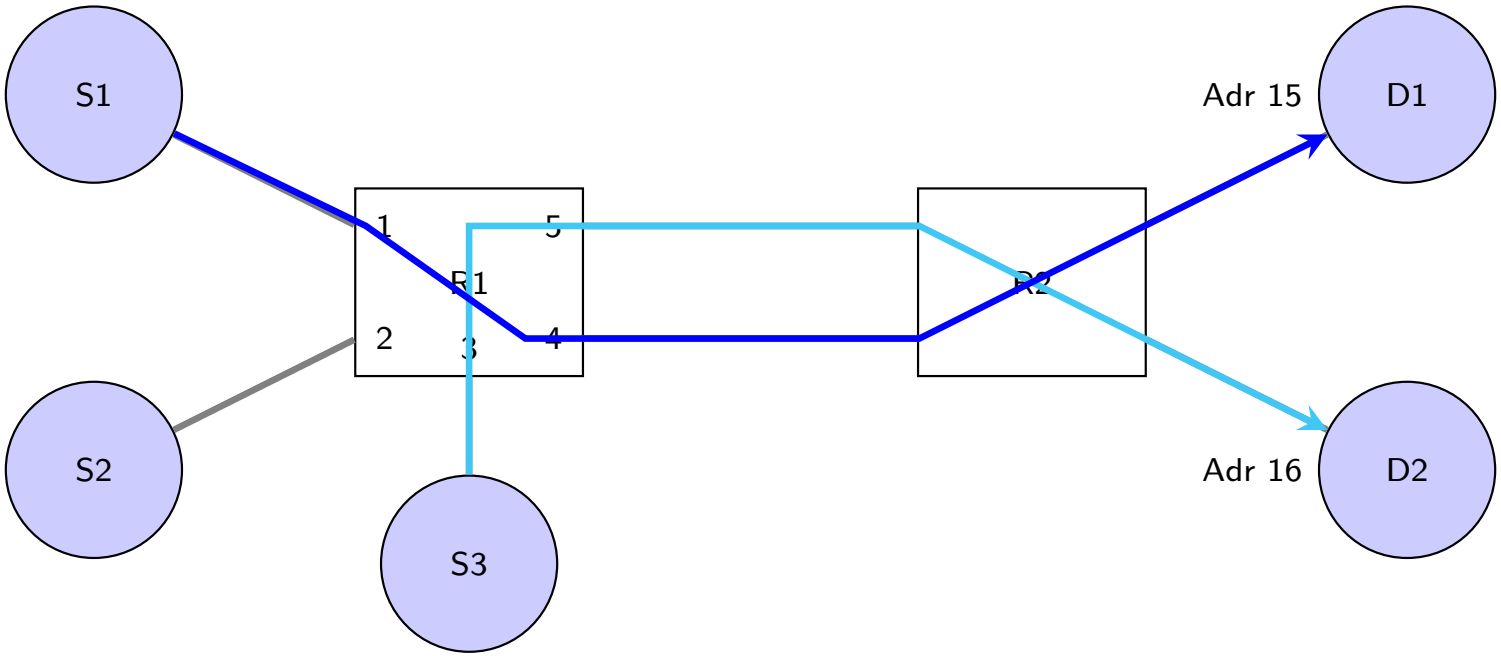


- Goal : to use all the links that interconnect 2 routers
- Routers automatically distribute the traffic

Routing table of R1

Addr	Port 1	Port 2	Port 3	Port 4	Port 5
15	0	0	0	1	1
16	0	0	0	1	1

Grouping Adaptative Routing

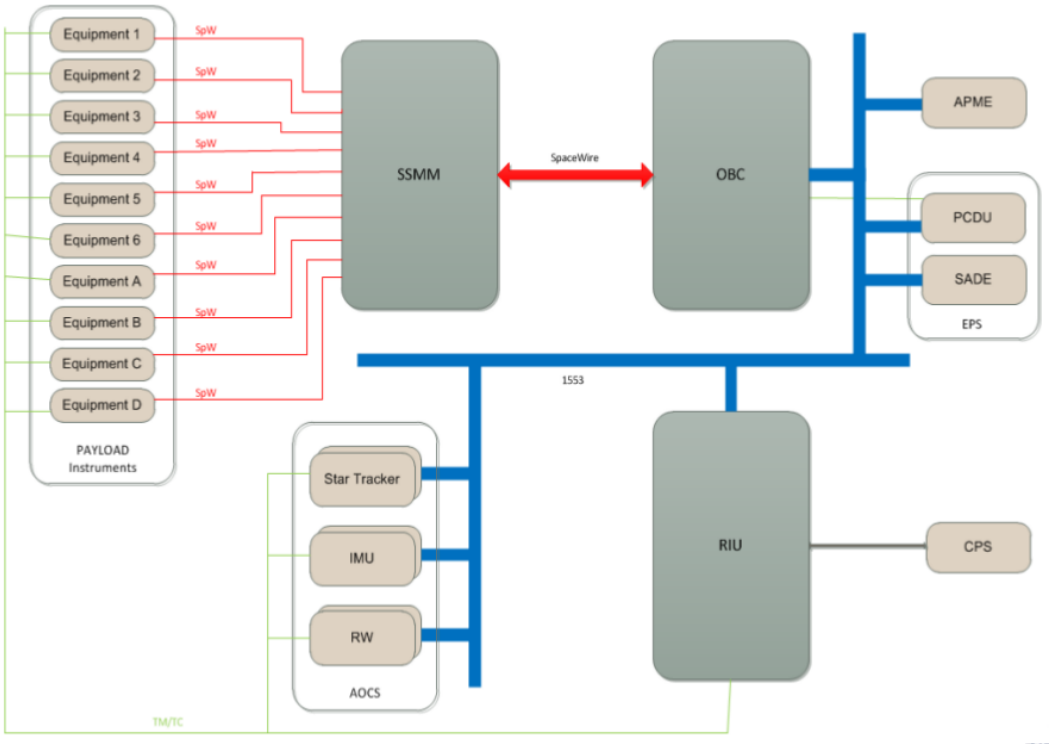


- Goal : to use all the links that interconnect 2 routers
- Routers automatically distribute the traffic

Routing table of R1

Addr	Port 1	Port 2	Port 3	Port 4	Port 5
15	0	0	0	1	1
16	0	0	0	1	1

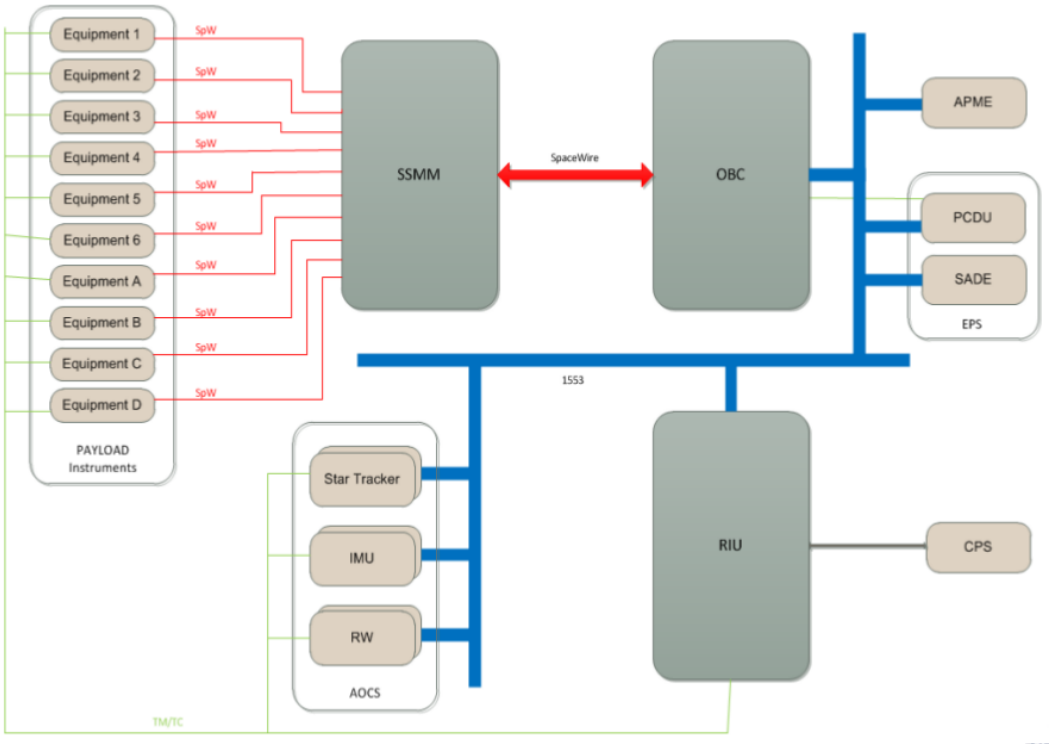
Typical spacecraft architecture, the come back



→ SpaceWire
→ 1553

(Source : O. Notebaert, Airbus Defence and Space)

And why not ?



(Source : O. Notebaert, Airbus Defence and Space)

→ SpaceWire
→ SpaceWire

SpaceWire and real time constraints

- Enough data rate to transmit:
  - Telemetry data from the instruments
  - Commands (low amount of data)

Problem: transmission delays non deterministic

- Blocking in cascade Blocages en cascade
- Important size of the scientific data

How to guarantee the real time constraints of the transmission delays

A first solution

To compute the end to end delay of each flow

Too much interactions between the flows $\rightarrow$ unfeasible

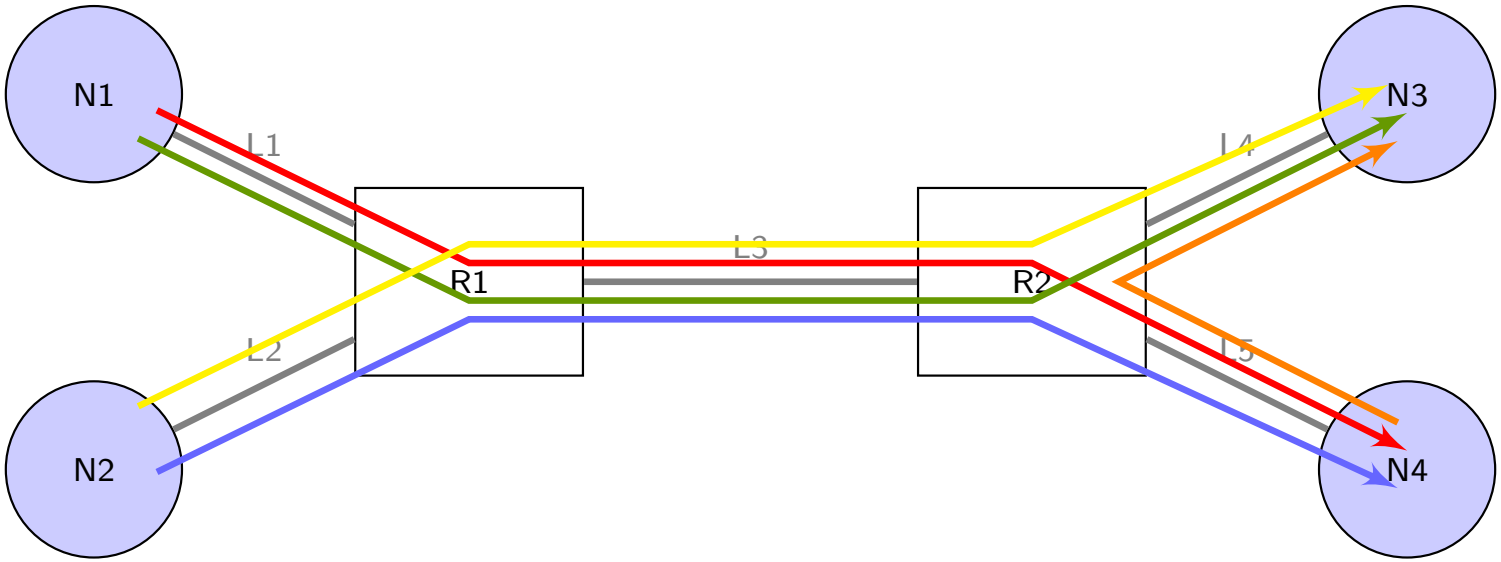
To compute the maximum worst case end to end delays

- Differents methods exist: Network Calculus, Model Checking, ...
- Recursive Calculus proposed by T. Ferrandiz [ISAE 2012]

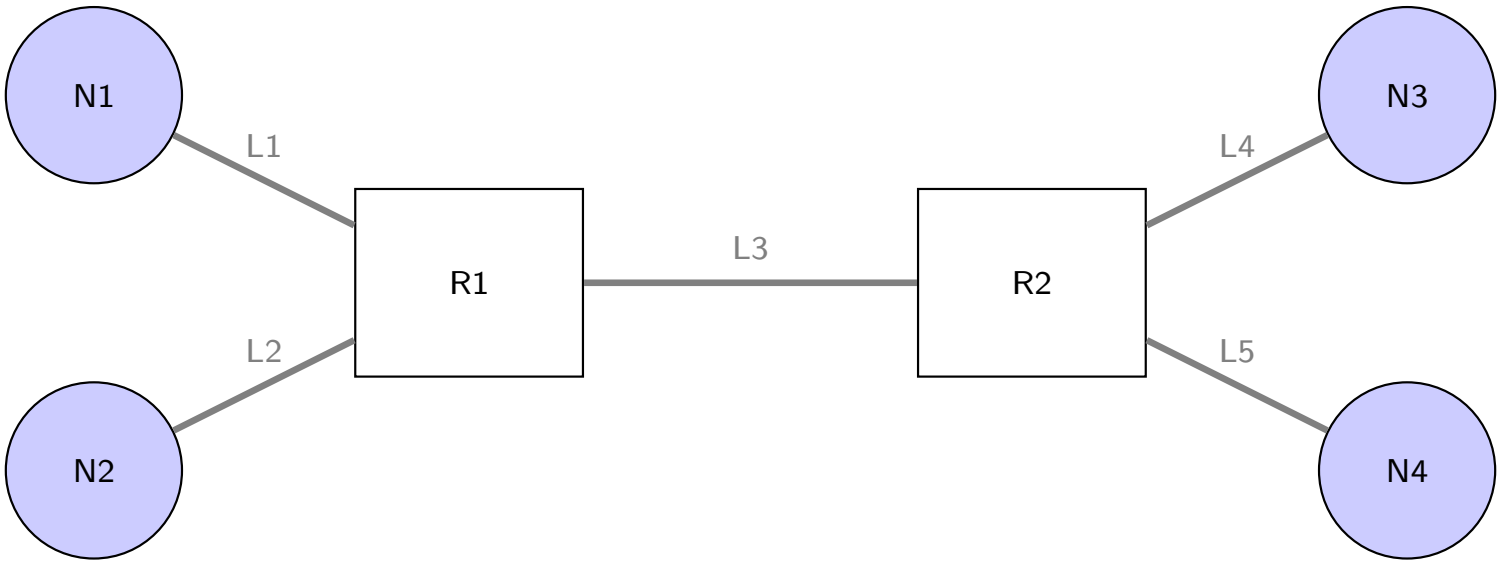
Principle of Recursive Calculus

An example

Computation of the worst case traversal time (WCTT) of the green flow

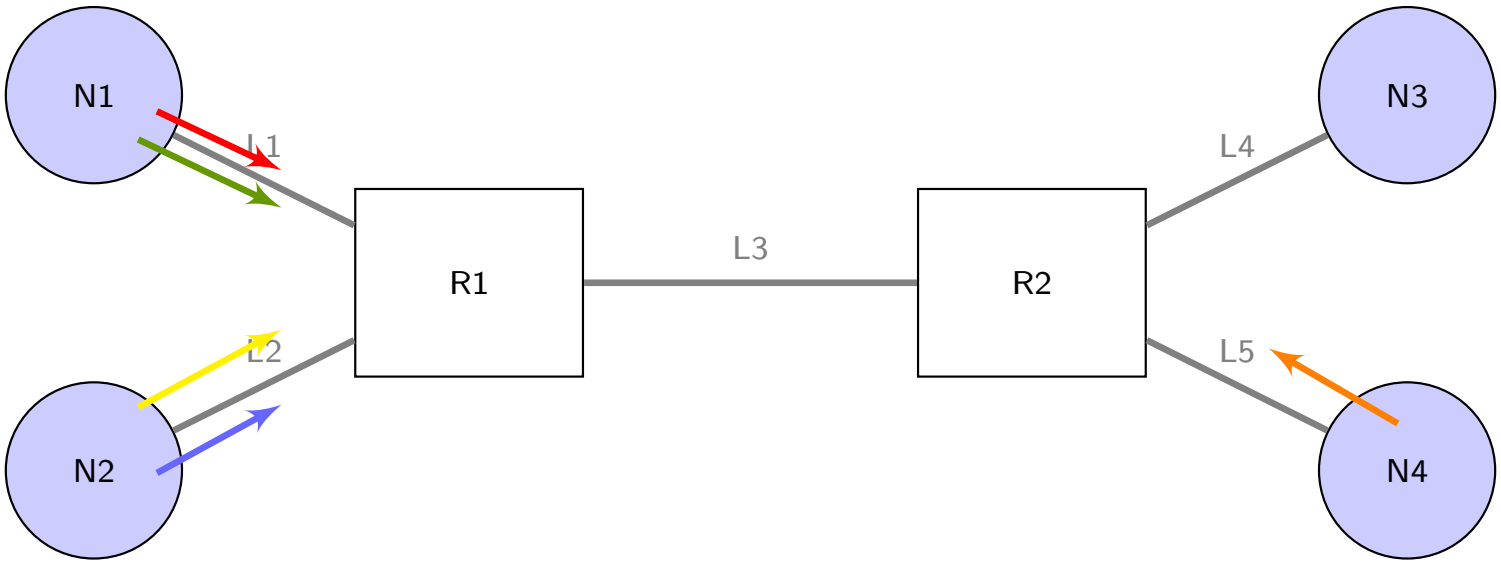


Principle of Recursive Calculus



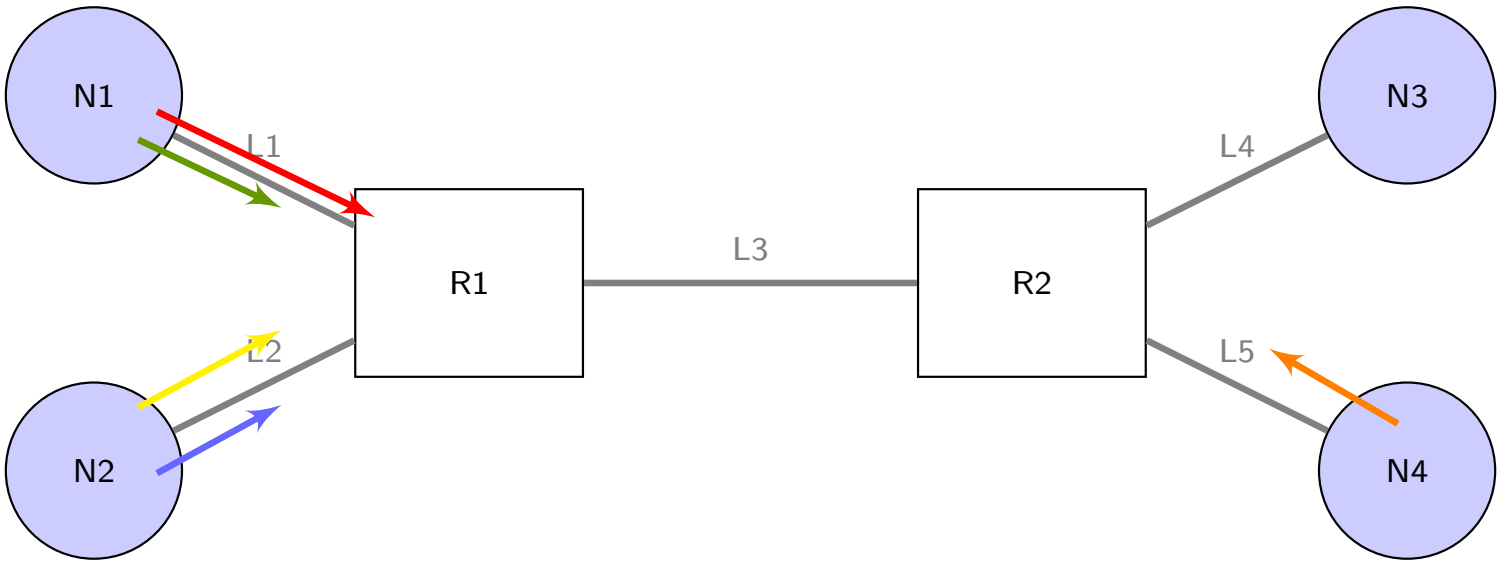
$$Delay(F_g) = d(F_g, L_1)$$

Principle of Recursive Calculus



$$Delay(F_g) = d(F_g, L_3) + d(F_r, L_3)$$

Principle of Recursive Calculus

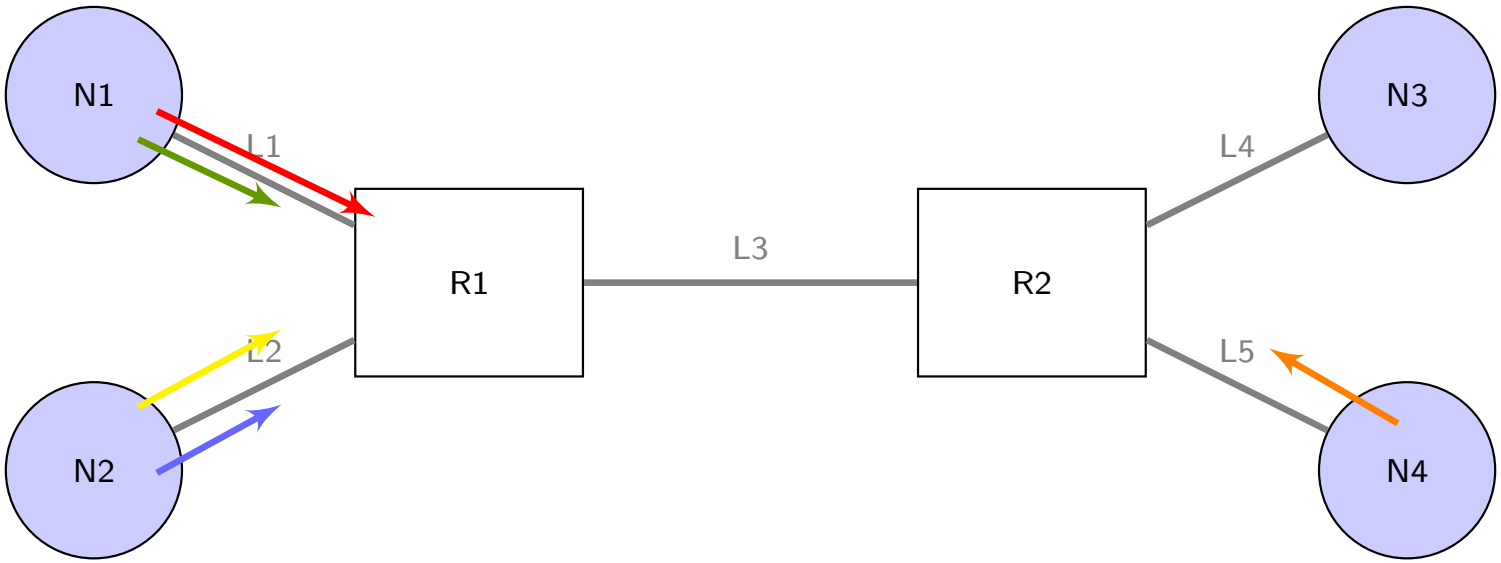


$$Delay(F_g) = d(F_g, L_3) + d(F_r, L_3) + d(F_j, L_3)$$

ou

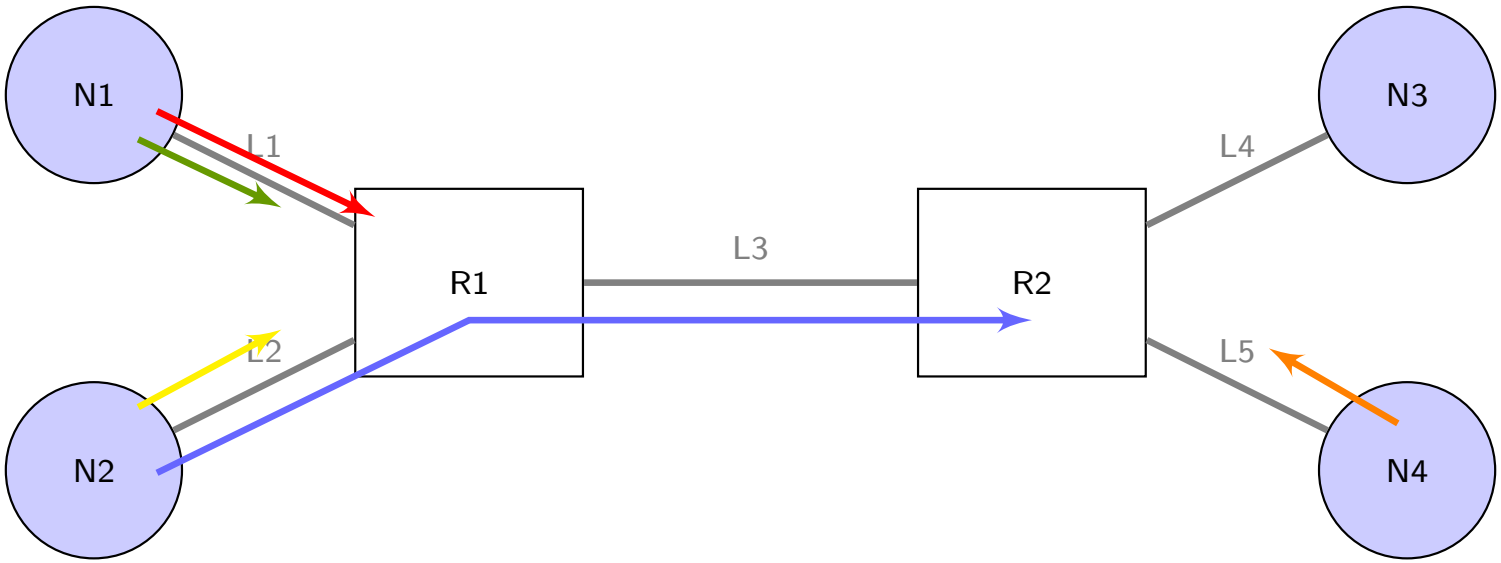
$$Delay(F_g) = d(F_g, L_3) + d(F_r, L_3) + d(F_b, L_3)$$

Principle of Recursive Calculus



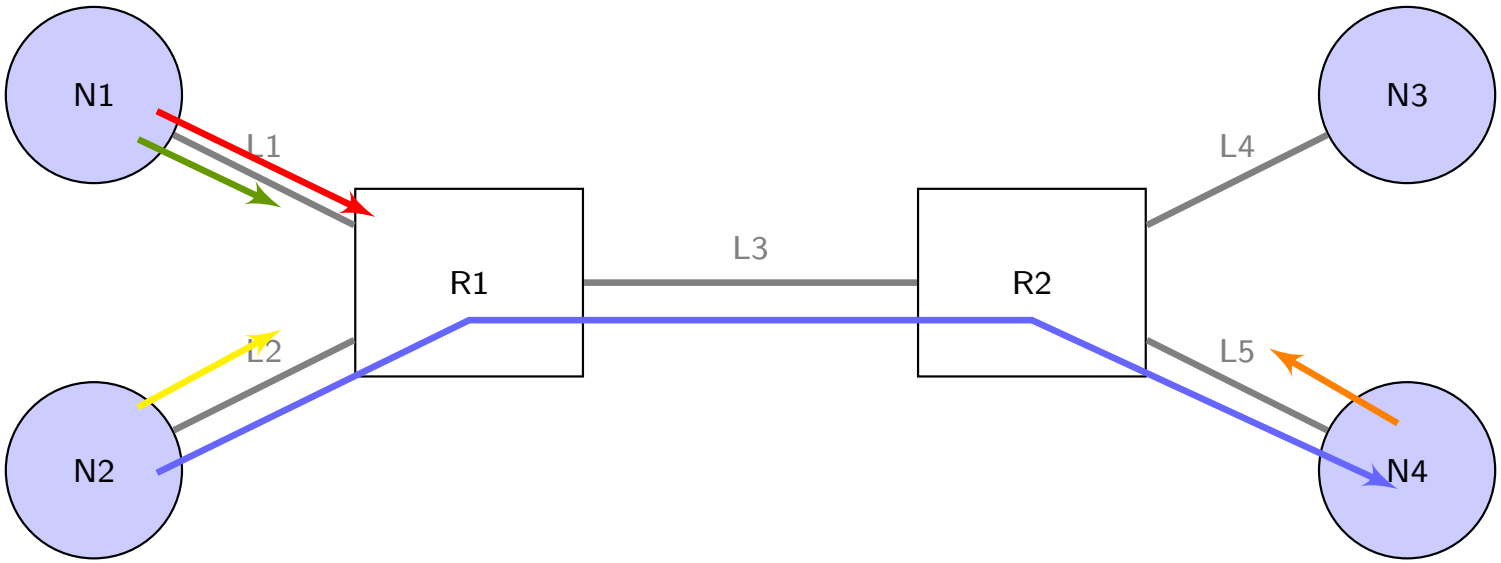
$$Delay(F_g) = d(F_g, L_3) + d(F_r, L_3) + \max\left(d(F_j, L_3), d(F_b, L_3)\right)$$

Principle of Recursive Calculus



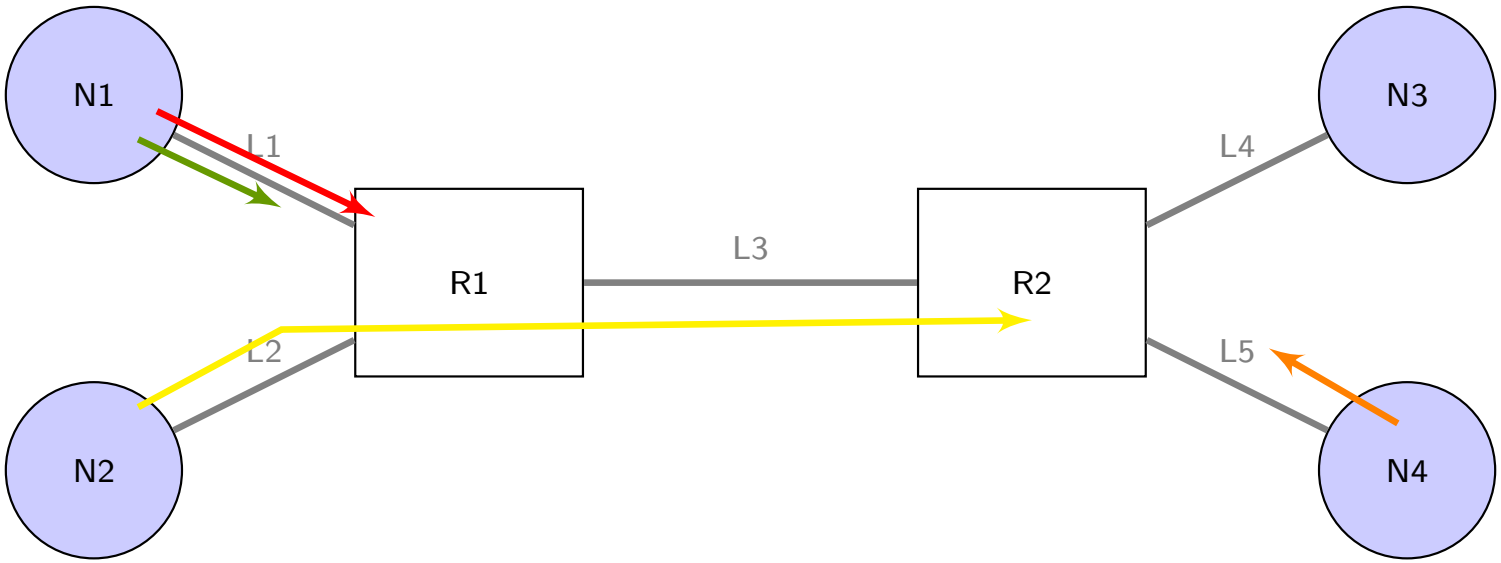
$$Delay(F_g) = d(F_g, L_3) + d(F_r, L_3) + \max\left(d(F_j, L_3), d(F_b, L_5)\right)$$

Principle of Recursive Calculus



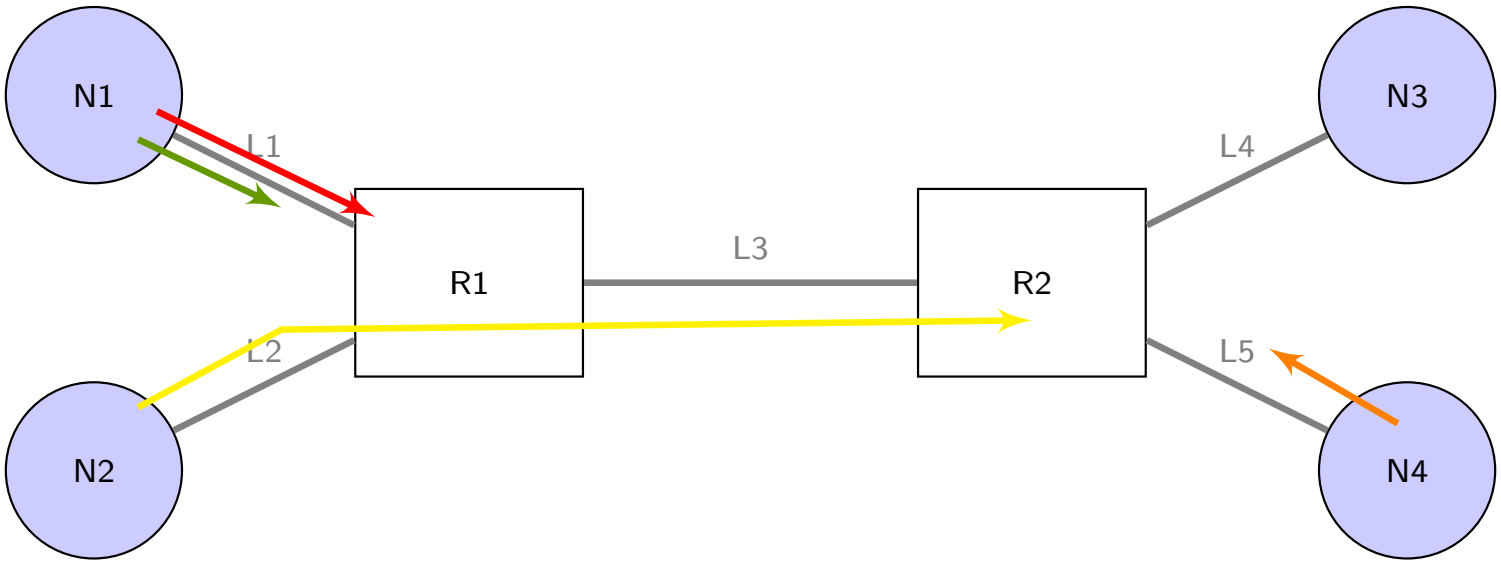
$$Delay(F_g) = d(F_g, L_3) + d(F_r, L_3) + \max\left(d(F_j, L_3), d(F_b, null)\right)$$

Principle of Recursive Calculus



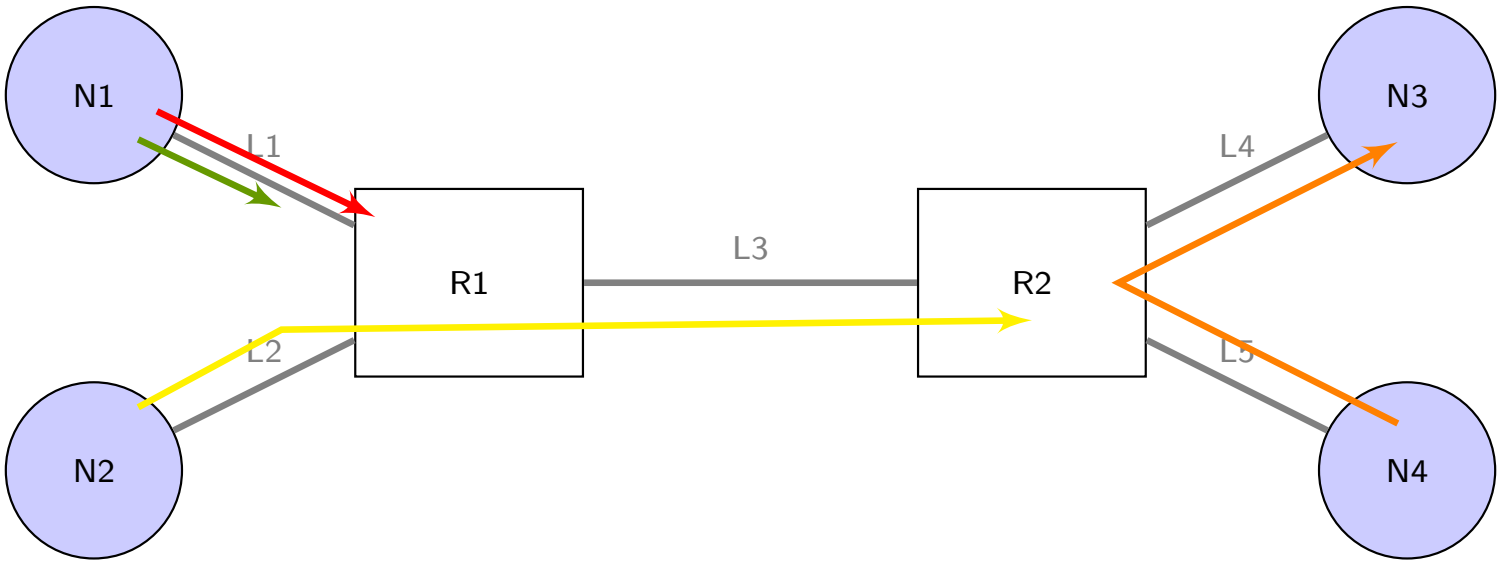
$$Delay(F_g) = d(F_g, L_3) + d(F_r, L_3) + \max\left(d(F_j, L_4), d(F_b, null)\right)$$

Principle of Recursive Calculus



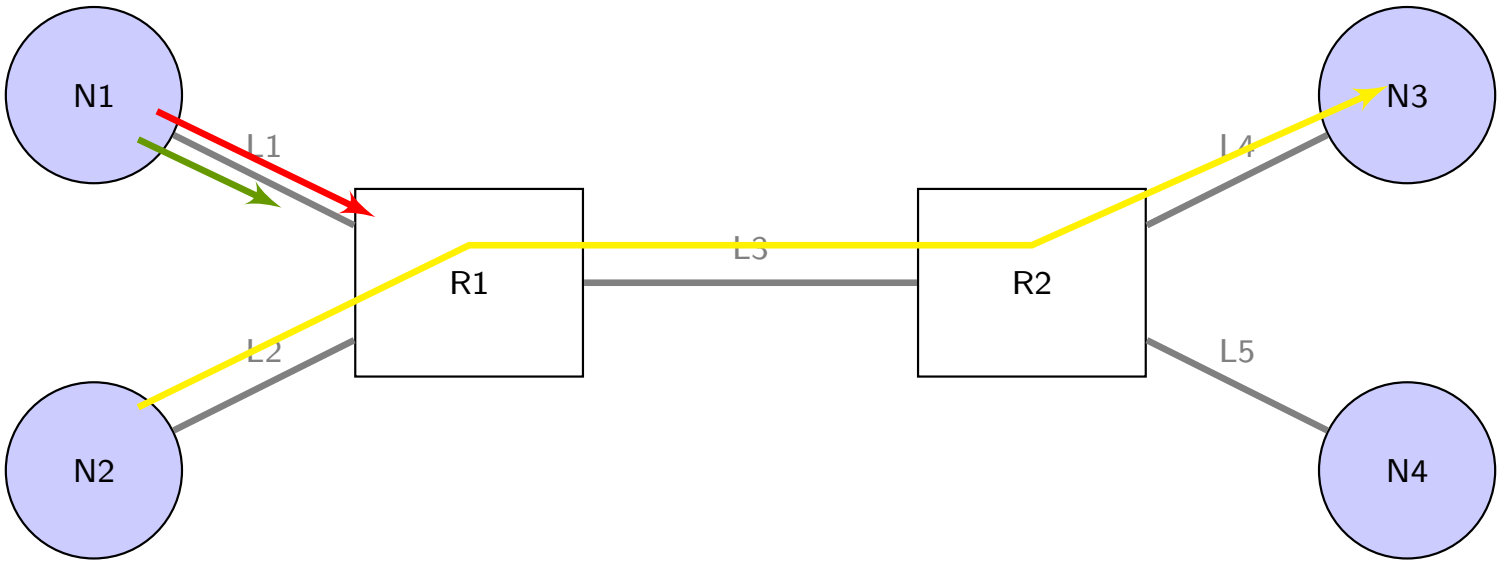
$$Delay(F_g) =$$
$$d(F_g, L_3) + d(F_r, L_3) + \max\left(d(F_j, L_4) + d(F_o, L_4), d(F_b, null)\right)$$

Principle of Recursive Calculus



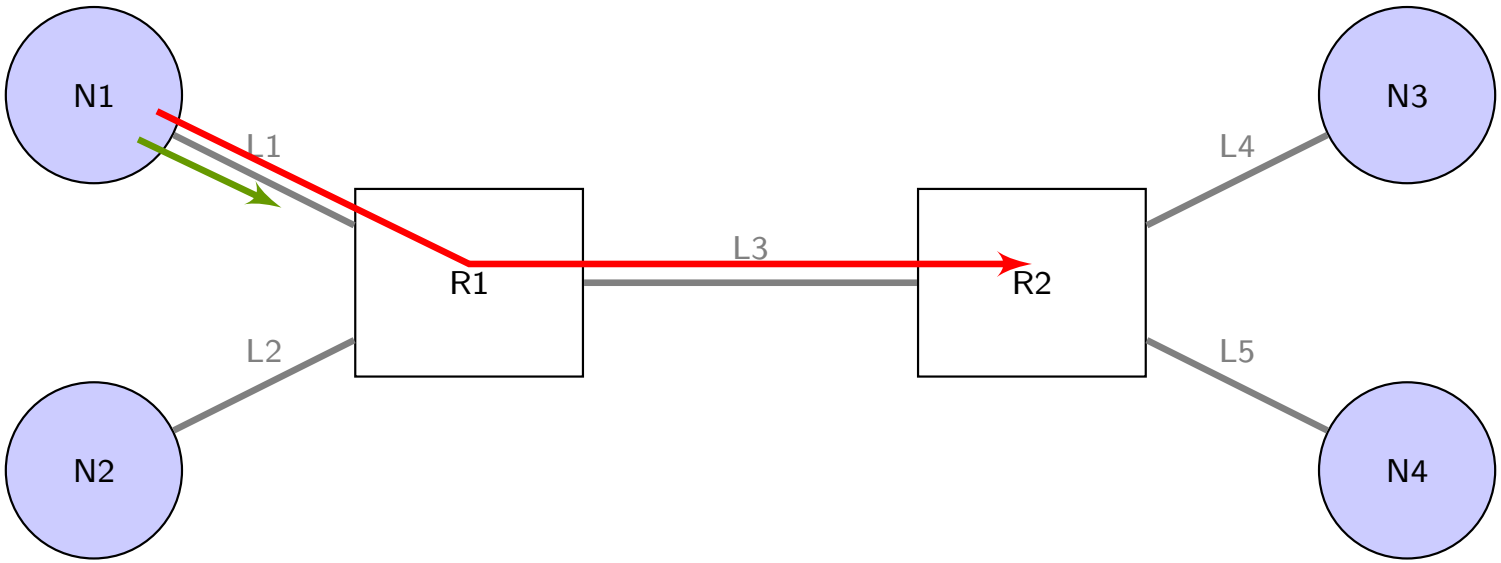
$$Delay(F_g) =$$
$$d(F_g, L_3) + d(F_r, L_3) + \max\left(d(F_j, L_4) + d(F_o, null), d(F_b, null)\right)$$

Principle of Recursive Calculus



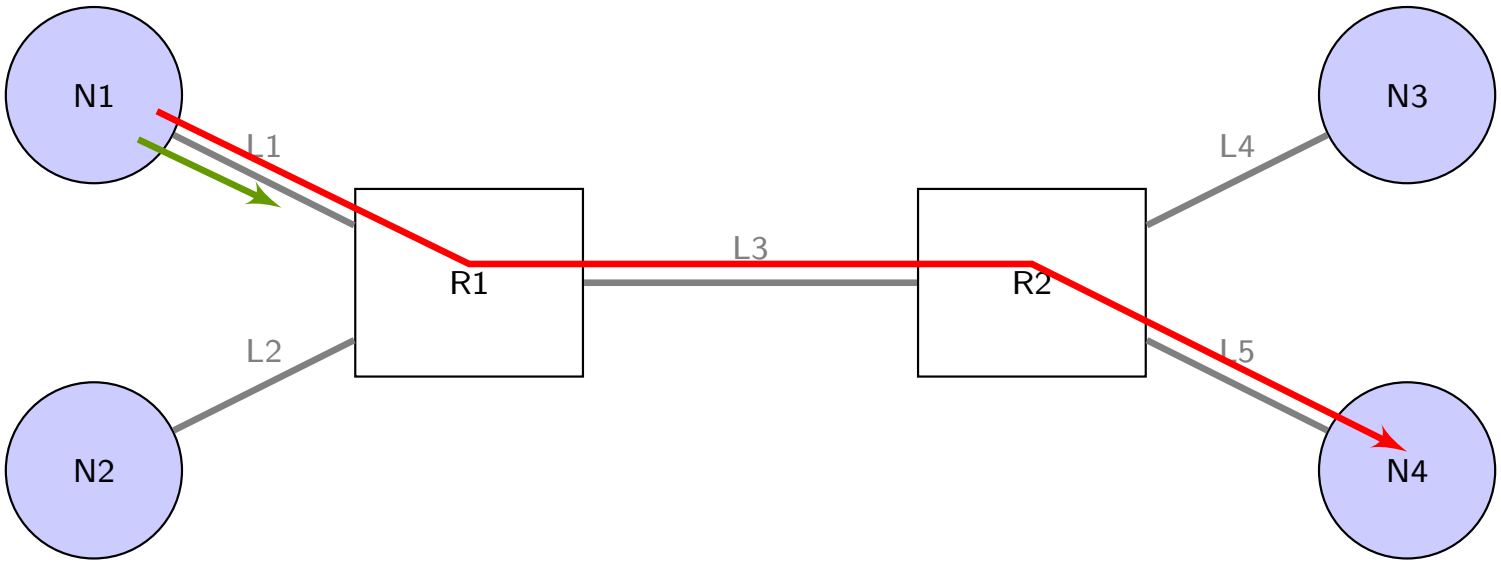
$$Delay(F_g) =$$
$$d(F_g, L_3) + d(F_r, L_3) + \max\left(d(F_j, null) + d(F_o, null), d(F_b, null)\right)$$

Principle of Recursive Calculus



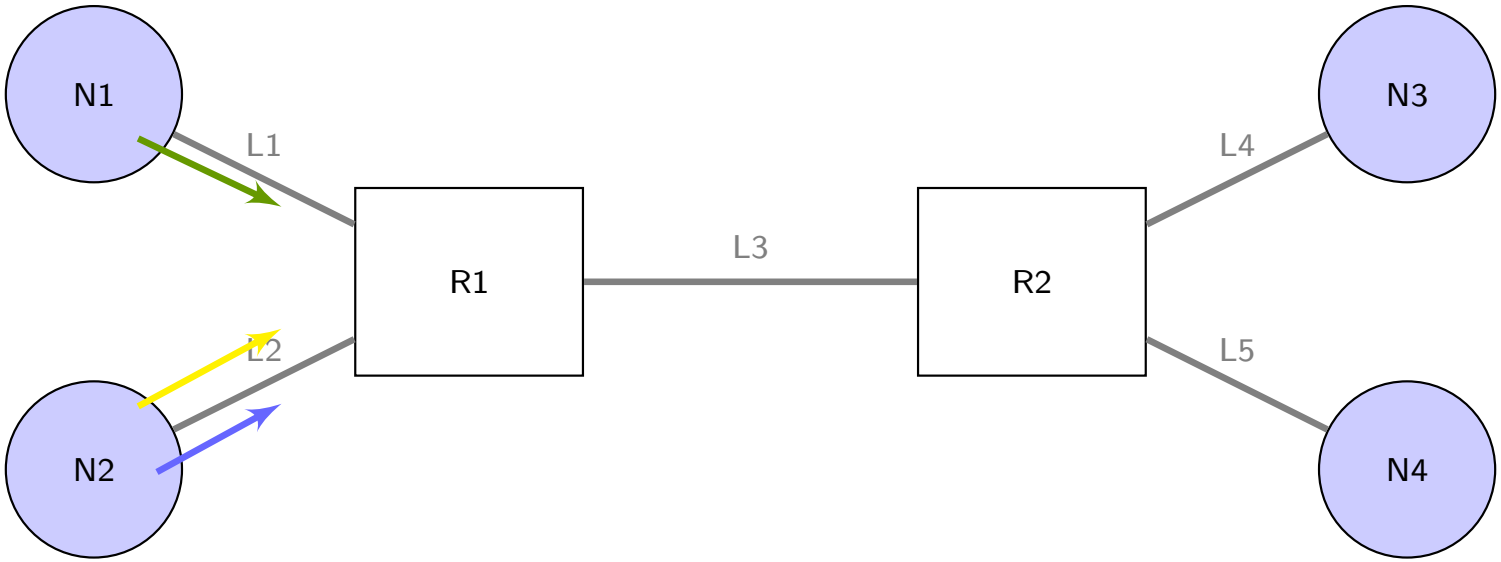
$$Delay(F_g) =$$
$$d(F_g, L_3) + d(F_r, L_5) + \max\left(d(F_j, null) + d(F_o, null), d(F_b, null)\right)$$

Principle of Recursive Calculus

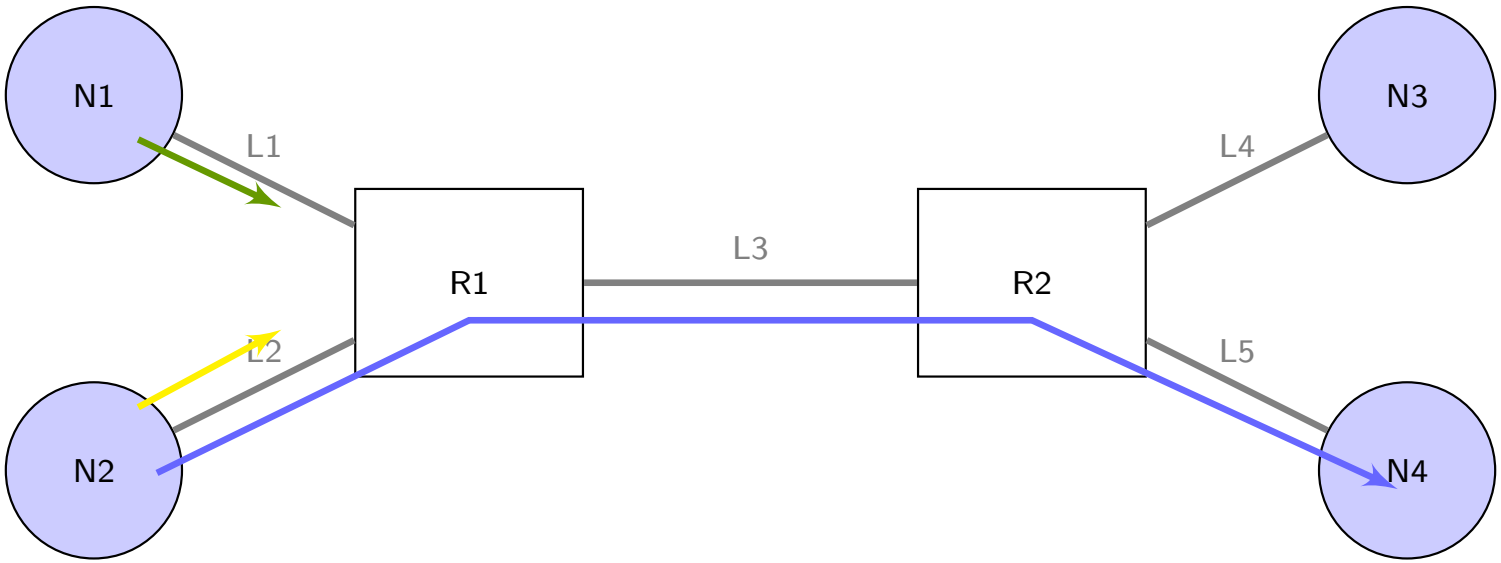


$$Delay(F_g) =$$
$$d(F_g, L_3) + d(F_r, null) + \max\left(d(F_j, null) + d(F_o, null), d(F_b, null)\right)$$

Principle of Recursive Calculus

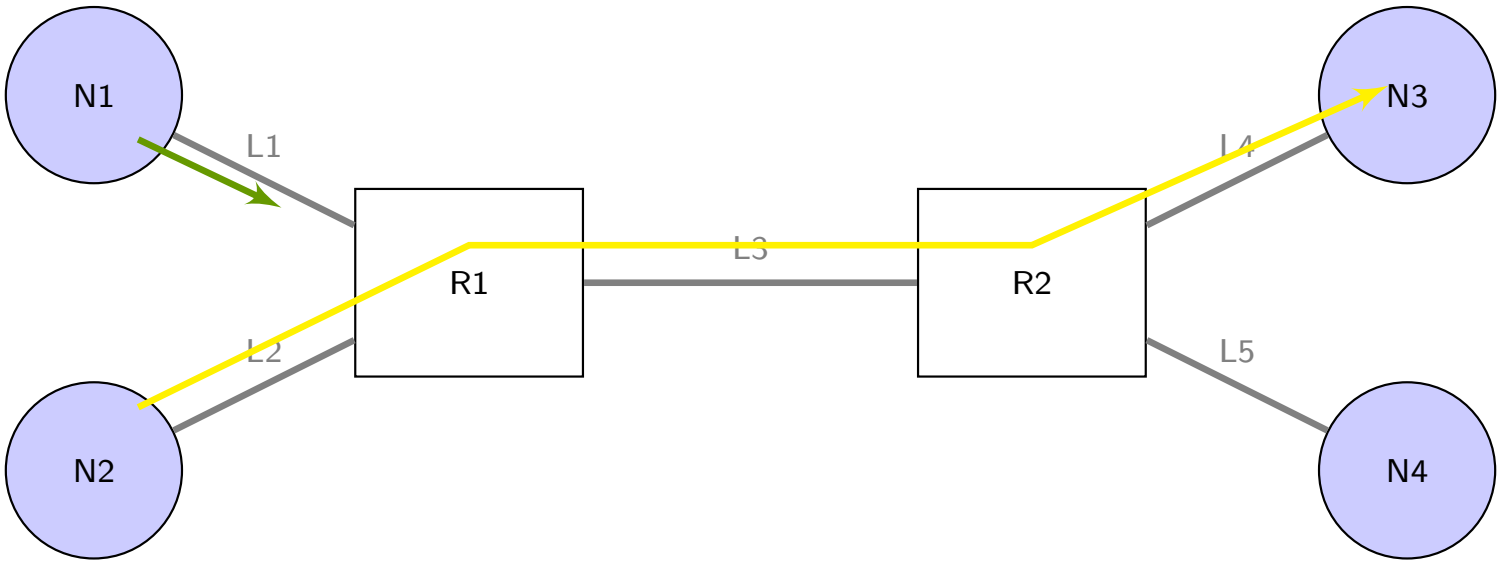


Principle of Recursive Calculus



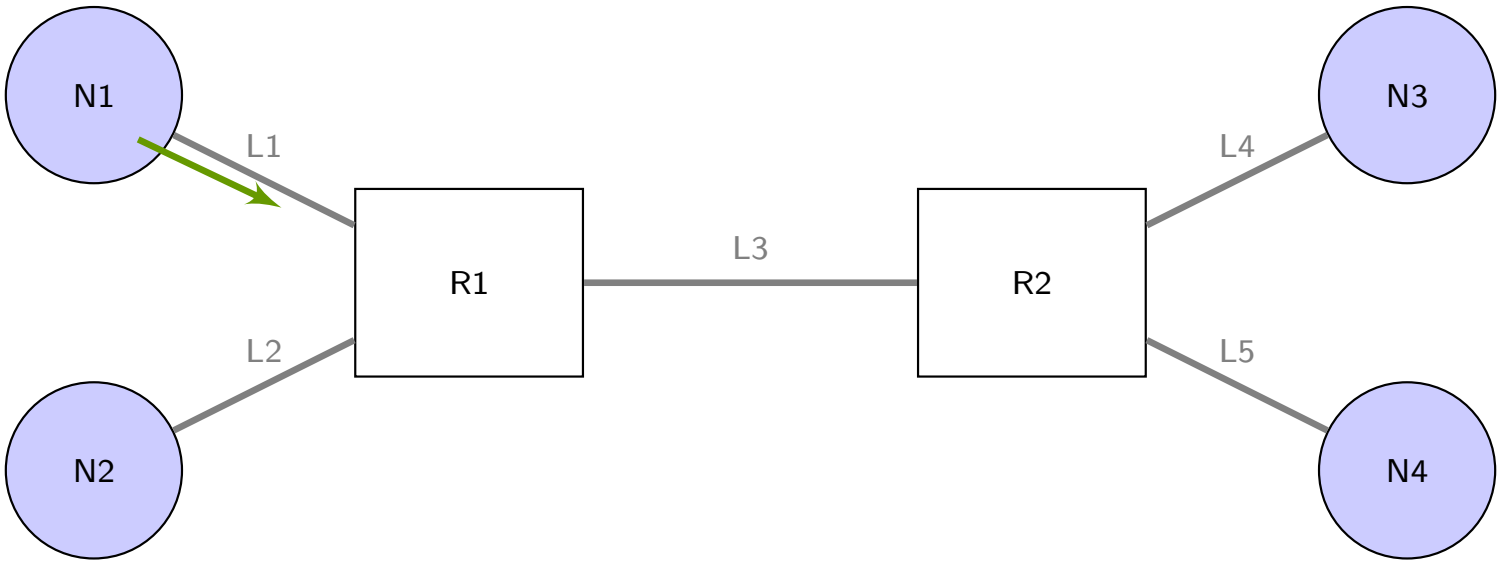
$$Delay(F_g) =$$
$$d(F_g, L_3) + d(F_r, null) + \max\left(d(F_j, null) + d(F_o, null), d(F_b, null)\right)$$

Principle of Recursive Calculus



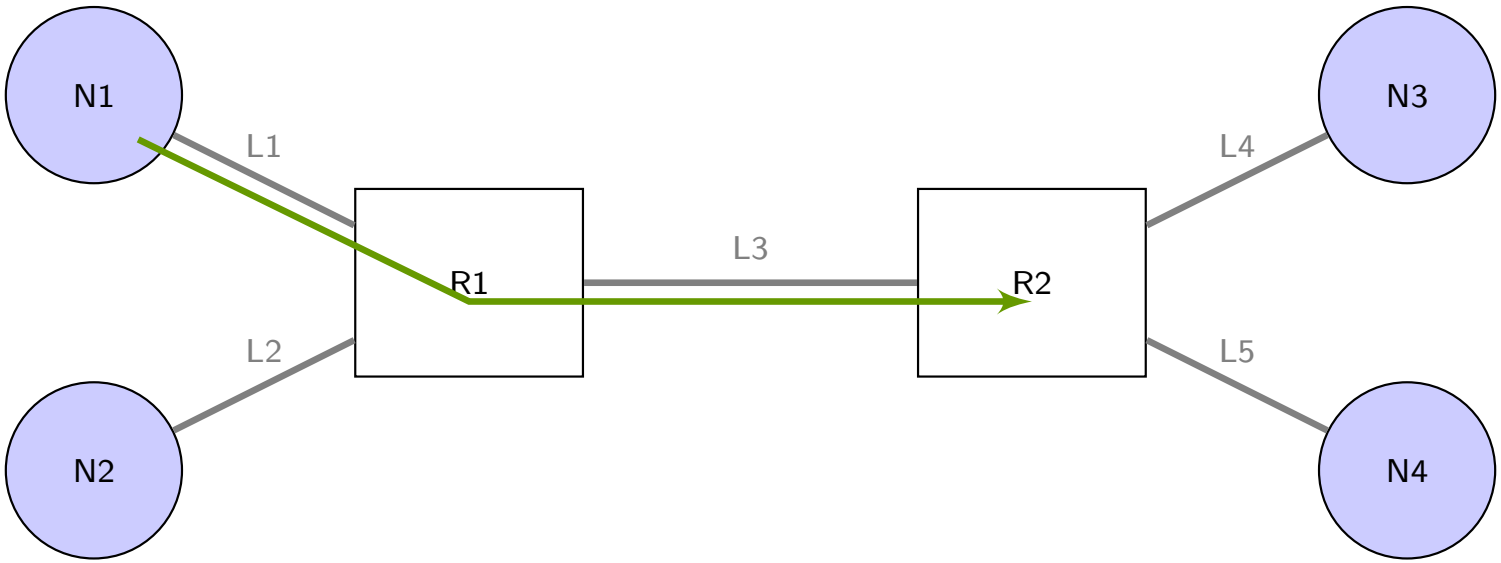
$$Delay(F_g) =$$
$$d(F_g, L_3) + d(F_r, null) + \max\left(d(F_j, null) + d(F_o, null), d(F_b, null)\right)$$

Principle of Recursive Calculus



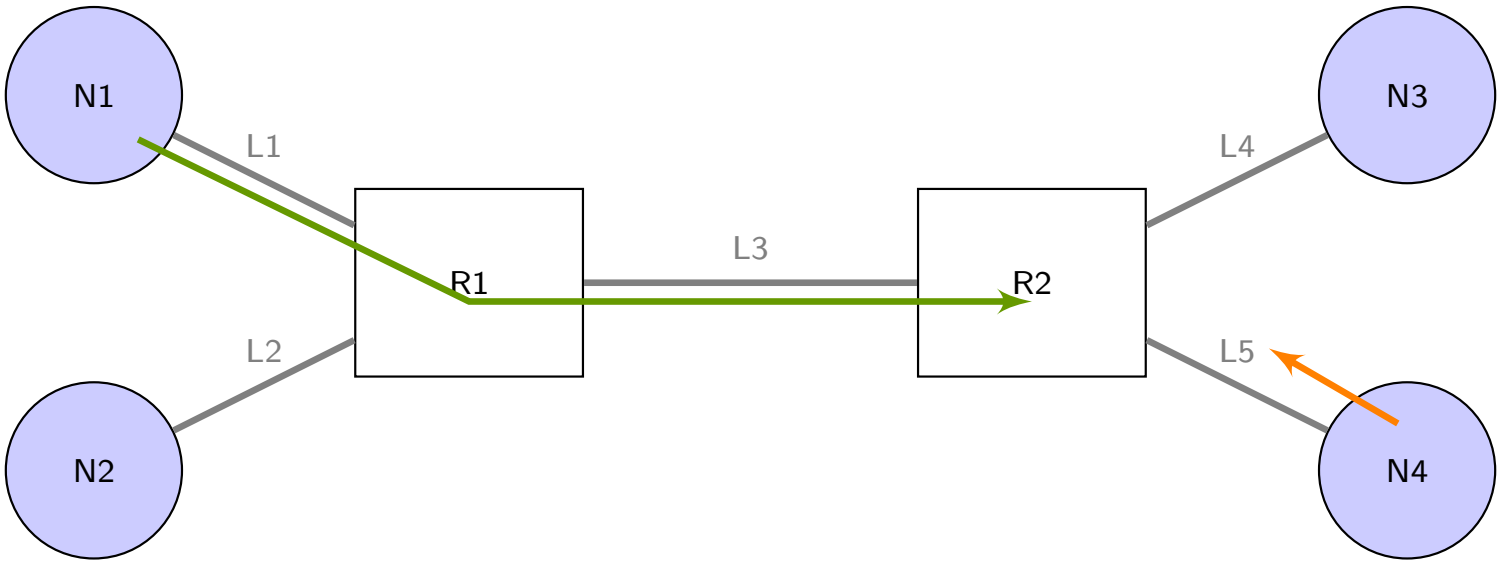
$$Delay(F_g) =$$
$$d(F_g, L_3) + d(F_r, null) + 2 \times \max\left(d(F_j, null) + d(F_o, null), d(F_b, null)\right)$$

Principle of Recursive Calculus



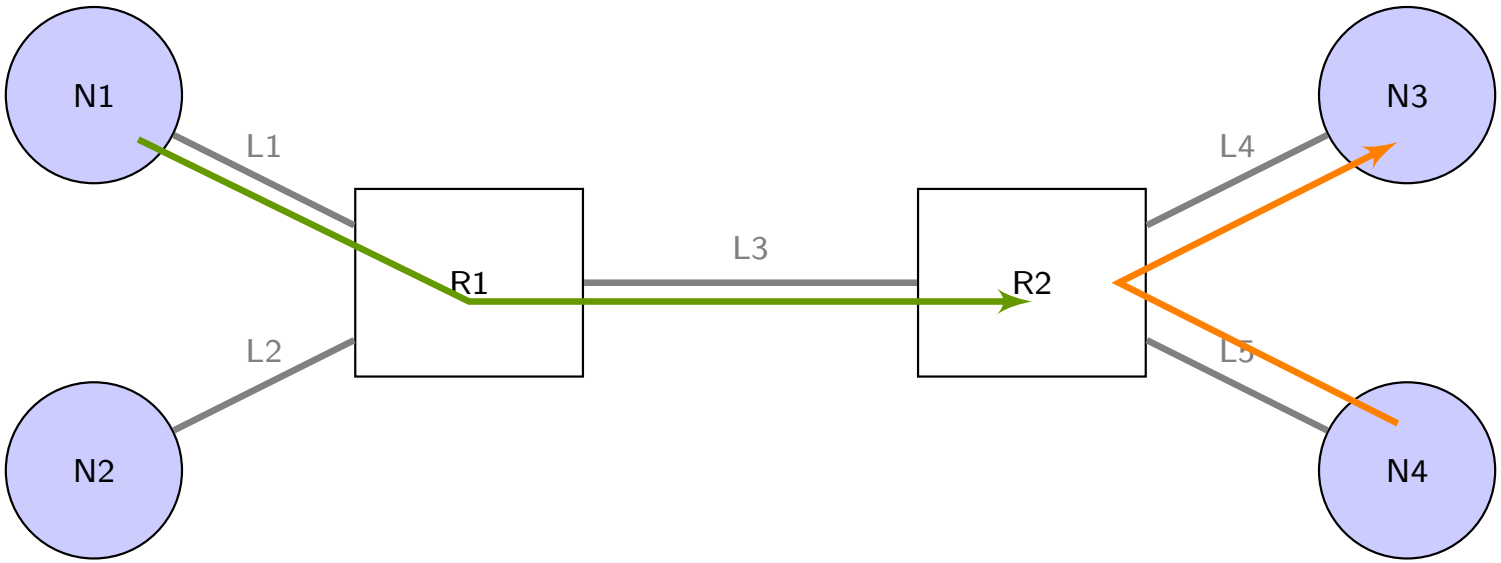
$$Delay(F_g) =$$
$$d(F_g, L_4) + d(F_r, null) + 2 \times \max\left(d(F_j, null) + d(F_o, null), d(F_b, null)\right)$$

Principle of Recursive Calculus



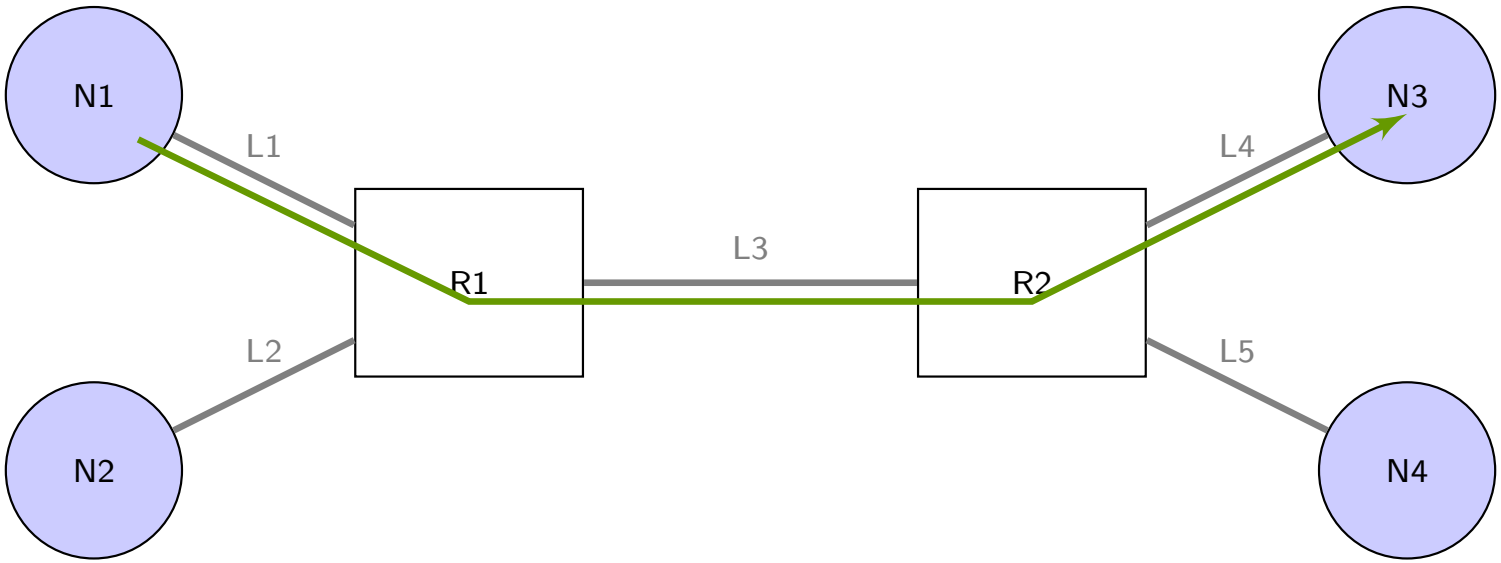
$$Delay(F_g) = d(F_g, L_4) + d(F_o, L_4) + d(F_r, null) + 2 \times \max\left(d(F_j, null) + d(F_o, null), d(F_b, null)\right)$$

Principle of Recursive Calculus



$$Delay(F_g) = d(F_g, L_4) + d(F_o, null) + d(F_r, null) + 2 \times \max\left(d(F_j, null) + d(F_o, null), d(F_b, null)\right)$$

Principle of Recursive Calculus



$$Delay(F_g) = d(F_g, null) + d(F_o, null) + d(F_r, null) + 2 \times \max\left(d(F_j, null) + d(F_o, null), d(F_b, null)\right)$$

Principle of Recursive Calculus

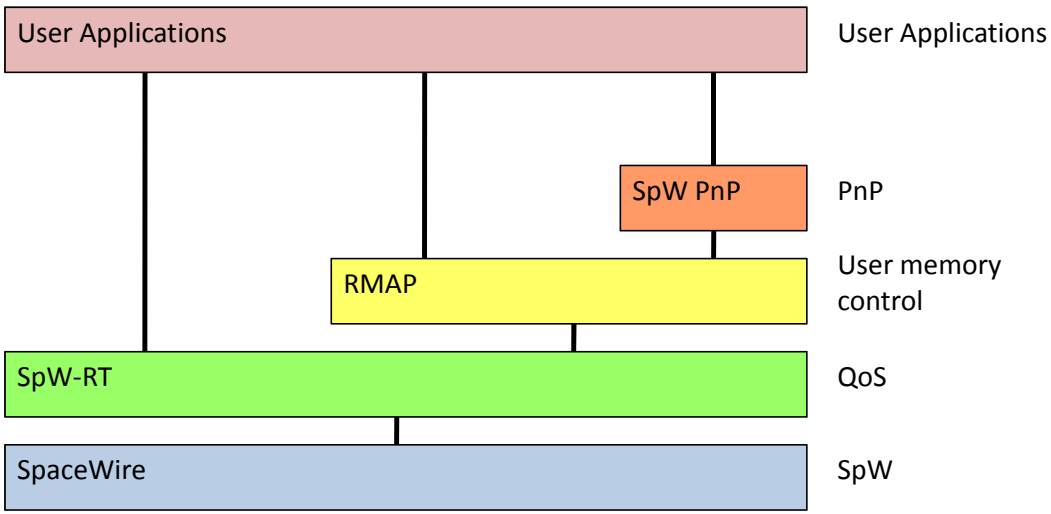
- $d(F_i, null) = \frac{S_i}{C}$ où S_i = size of the packets and C links capacity (data rate)
- $Delay(F_g) = \frac{S_v + S_o + S_r + 2 \times \max(S_j + S_o, S_b)}{C}$

Conclusion

- Simple method to compute maximum worst case end to end delay of a SpaceWire flow
- Deterministic bound of the delays

Second solution

- Add a new network layer between SpaceWire and the applications
- To guarantee the QoS of the applications
- For ex., Ethernet extension to guarantee the transmission delays: TT-Ethernet
- Solution: SpaceWire-RT (Reliability & Timeliness)



SpaceWire-RT

Goal: Synchronous and asynchronous communication with safety

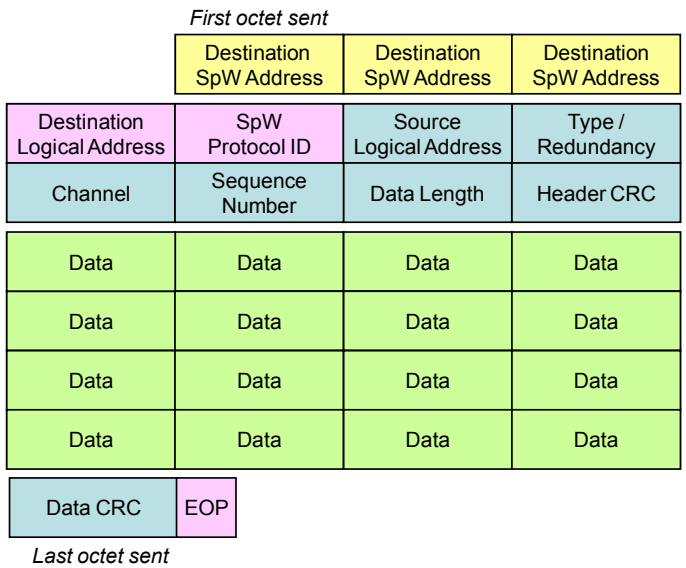
- Without Spacewire modification
- Same API as Spacewire

How ?

- Utilization of virtual channels
- 2 modes :
  - Asynchronous: no strict guarantee of delay, priorities over the virtual channels
  - Synchronous: TDMA-like mechanism to access to the network, guaranteed transmission delays

Virtual Channels

- Channel = source buffer + links + destination buffer
- Identifier : src addr + dest addr + number
- Behavior:
  - App writes in the source buffer
  - SpaceWire-RT send data through SpaceWire
  - App read data from SpaceWire-RT
- Packet format sent through SpaceWire:

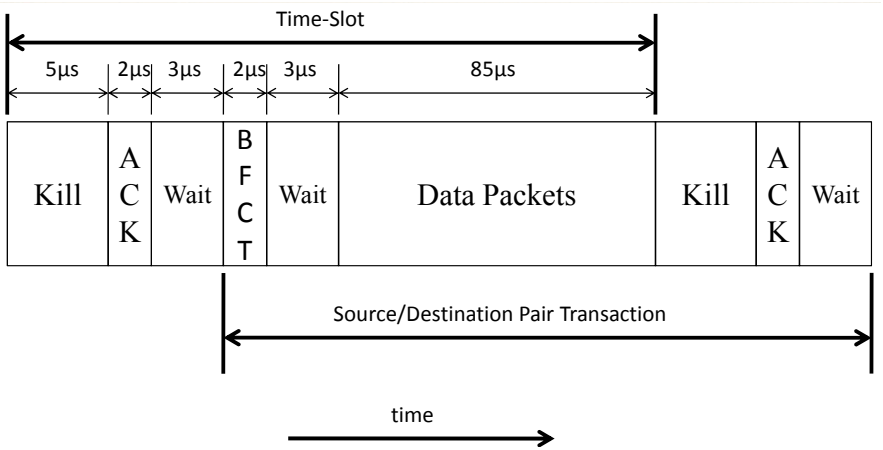


Asynchronous mode

- Unbounded transmission delays:
- Traffic control:
  - Data segmentation
    - 256 bytes fragments from the source
  - Priorities over the channels
    - low channel id = high priority
    - non preemptive
    - different than the "rotating priorities"
  - End to end flow control:
    - Ready to send via BFCT (Buffer Flow Control Token) from the destination to the source
  - Acknowledgements

Synchronous mode

- Deterministic Traversal Time
- TDMA:
  - Throughput divided in constant time slots
  - Utilization of SpaceWire Time-Codes:
    - Global synchronization
    - 64 slots cycles
  - Static allocation of the time slots to the channels
    - during a given time slot, only one link is used by a unique channel
  - Time slot format:



Conclusion on SpaceWire-RT

Advantages

- Fragmentation + priorities: urgent (short) messages can arrive before the big packets
- Synchronous mode: guaranteed bandwidth for urgent traffic

Disadvantages

- Flow control: can increase the transmission delay
- Need of a global synchronization
- Need of buffers with more important size

Agenda

1553-Spacewire, what's next ?

- Increasing needs in data rate and real-time critical communications
- High integration of systems
- Full duplex switched Ethernet is a success for aircrafts

AFDX for satellites

- MISSION (Methodology and assessment for the applicability of ARINC-664 (AFDX) in Satellite/Spacecraft on-board communicatION networks)

TT-Ethernet for launchers

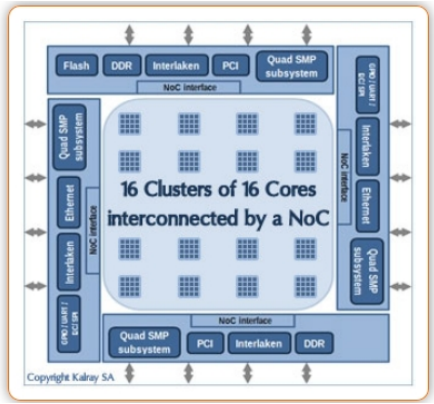
802.1 TSN

- Work in progress: EDEN project (IRT Saint Exupery, Thalès Avionics, Airbus DS, Continental, Dassault Aviation, ...)
- PhD thesis of Quentin Bailleul N7-2018

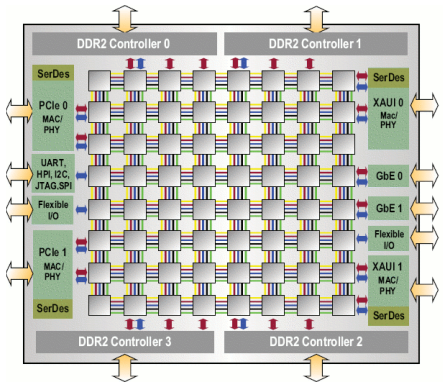
Agenda

Processors evolution ...

Many-Cores processors



Kalray MPPA 256



Tilera TILE 64

- Simple processors: good for certification
- Interconnection: Network on Chip (NoC)
  - Routers with few memory
  - Wormhole routing
  - QoS in order to guarantee real-time constraints

Processors evolution ...

NoC for Real-time communication

- Need to guarantee a bounded WCTT
- Close to Spacewire mechanisms
- Recursive Calculus for NoC