SpaceWire et SpaceWire-RT

Thomas Ferrandiz

thomas.ferrandiz@isae.fr
http://personnel.isae.fr/thomas-ferrandiz

janvier 2012

Contexte

- Deux types de trafic dans un satellite
- Trafic commande/contrôle (plate-forme)
  - peu de données
  - contraintes temporelles strictes
- Trafic scientifique (charge utile)
  - grande quantité de données
  - contraintes temporelles faibles
  - contrainte sur le débit moyen à assurer à chaque flux

Contexte

- Historiquement, utilisation de deux réseaux pour les deux types de trafic
- Trafic CC : bus de contrôle

  (ex: MIL-STD-1553B)

  - bus synchrone → bien adapté au trafic TR
  - bande passante faible (1 Mbps)
- ⇒ ajout de liens point-à-point haut débit pour trafic SC

Contexte

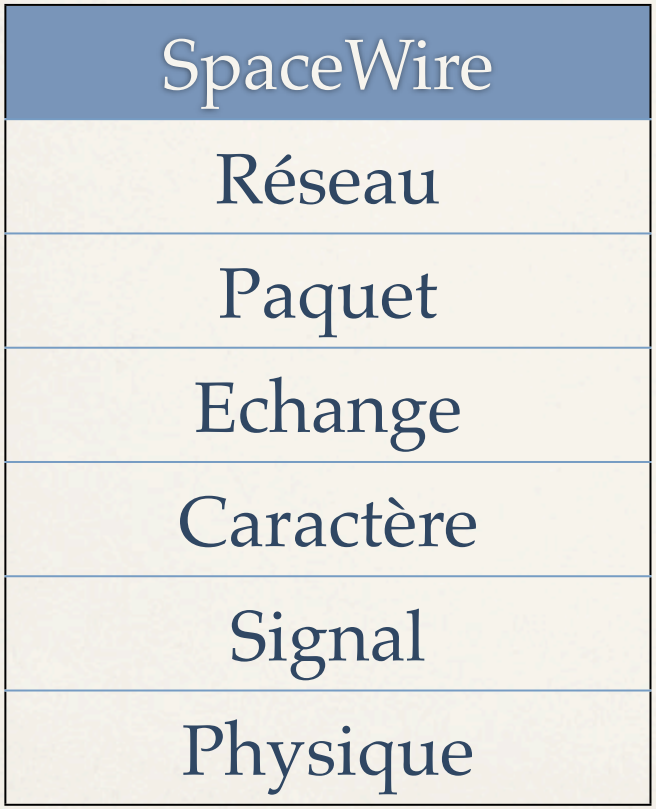
- Problème : les quantités de données augmentent
- ⇒ multiplication des liens point-à-point
- augmentation de la complexité et du coût du satellite
- Solution : un réseau bord unique

  → architecture routée

SpaceWire

- Réseau embarqué pour satellite, développé par l'ESA et l'Université de Dundee
- liens série point-à-point à haut débit (200 Mbps)
- faible consommation électrique
- adapté aux contraintes spatiales
- interface réseau standard qui permette de développer des composants génériques

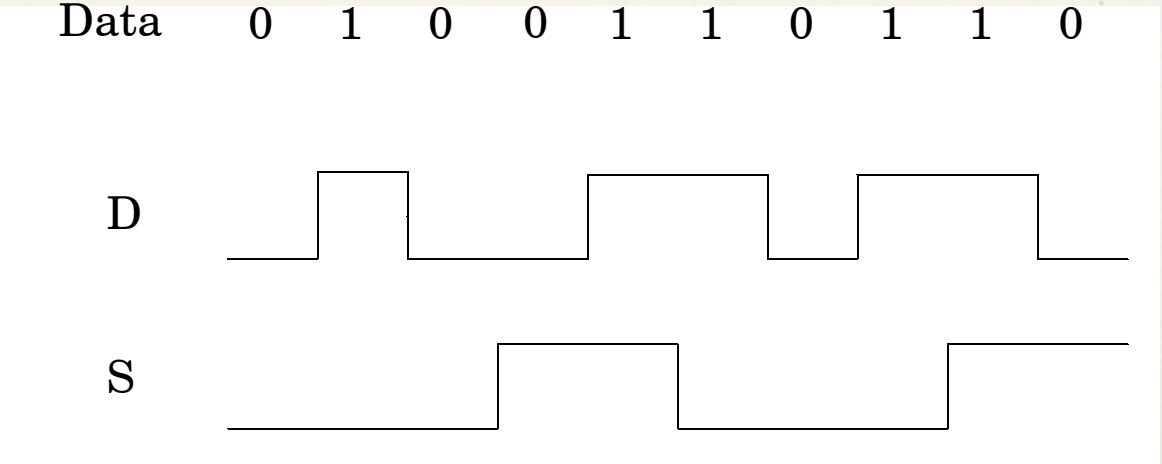
Les six niveaux de SpaceWire



- correspondent au 3 couches basses du modèle OSI (physique, liaison de données, réseau)
- chaque niveau utilise les fonctionnalités du niveau inférieur

Niveaux physique et signal

- LVDS (Low Voltage Differential Signalling)
  ⇒ faible consommation
- encodage Data-Strobe
- Full duplex ⇒ 4 paires de câbles
  ⇒ masse importante des câbles



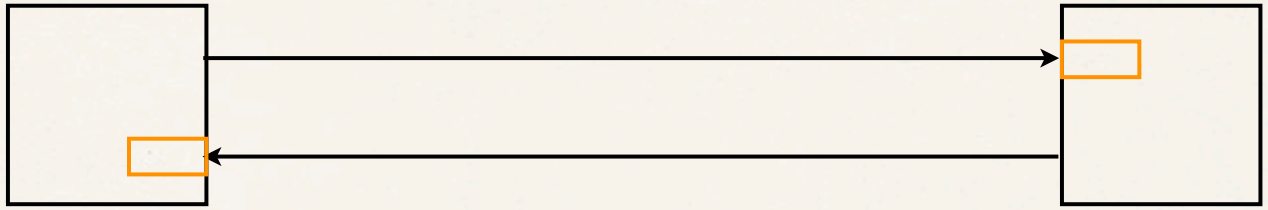
Niveau caractère

- Unité de transmission d'un réseau SpaceWire : le caractère

  ⇒ on peut entrelacer certains caractères au milieu d'un paquet

- 2 types de caractères :
  - Données (forment les paquets) : 10 bits
  - Contrôle (gestion du réseau) : 4 bits
    - FCT : Flow Control Token / ESC : Escape
    - EOP : End Of Packet / EEP : Error End Of Packet

Contrôle de flux



- propre à chaque lien
- récepteur envoie des autorisations d'émission à l'émetteur
- une autorisation (FCT) pour 8 octets
- au maximum, 56 octets peuvent être autorisés
  => buffer de 64 o dans chaque port d'entrée
- FCT prioritaires sur les caractères de données

Format des paquets et adressage

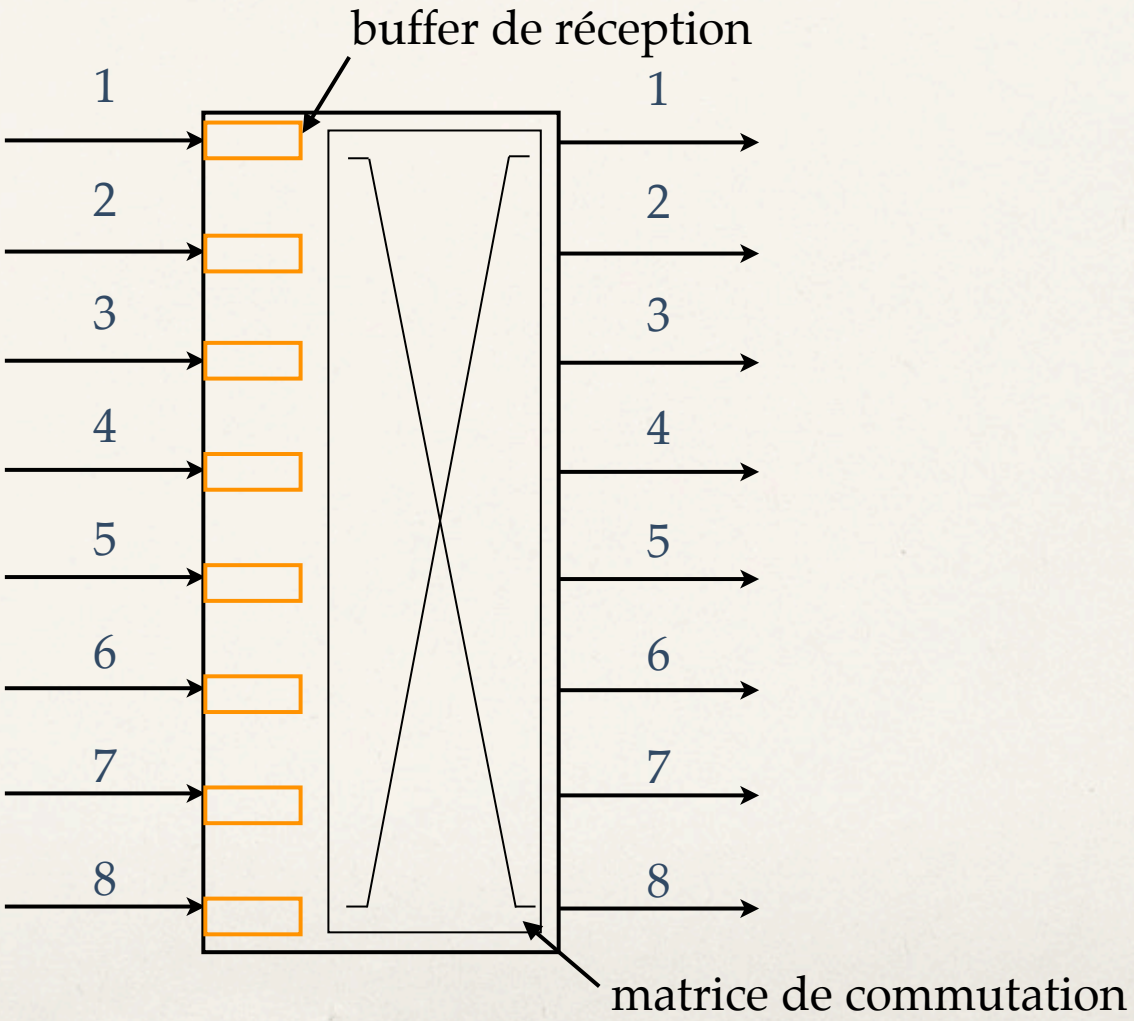
Adresse	Données	EOP

- Adresse et données de longueur quelconque
- Ici, adressage logique
  - chaque port relié au réseau à 1 numéro sur un octet (224 adresses possibles)

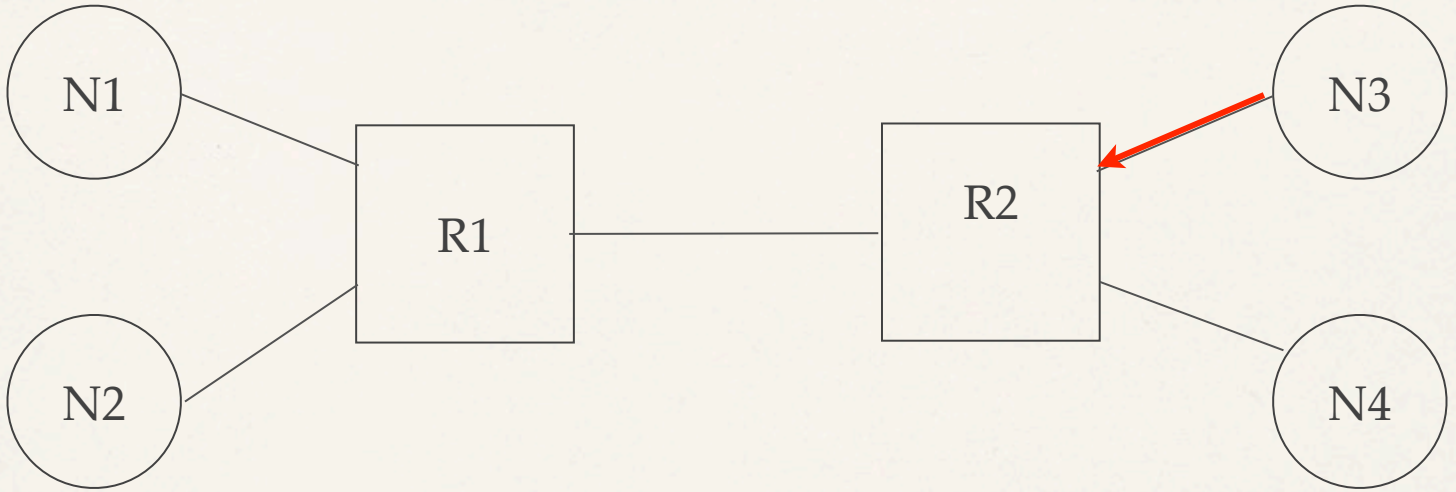
Routage Wormhole

- les liens point-à-point ne suffisent pas à suivre l'augmentation de la charge
- on veut donc interconnecter les terminaux à l'aide de routeurs
- Contraintes :
  - faible utilisation mémoire (très couteuse dans le spatial)
  - compatibilité avec les liens point-à-point

Routeur SpaceWire

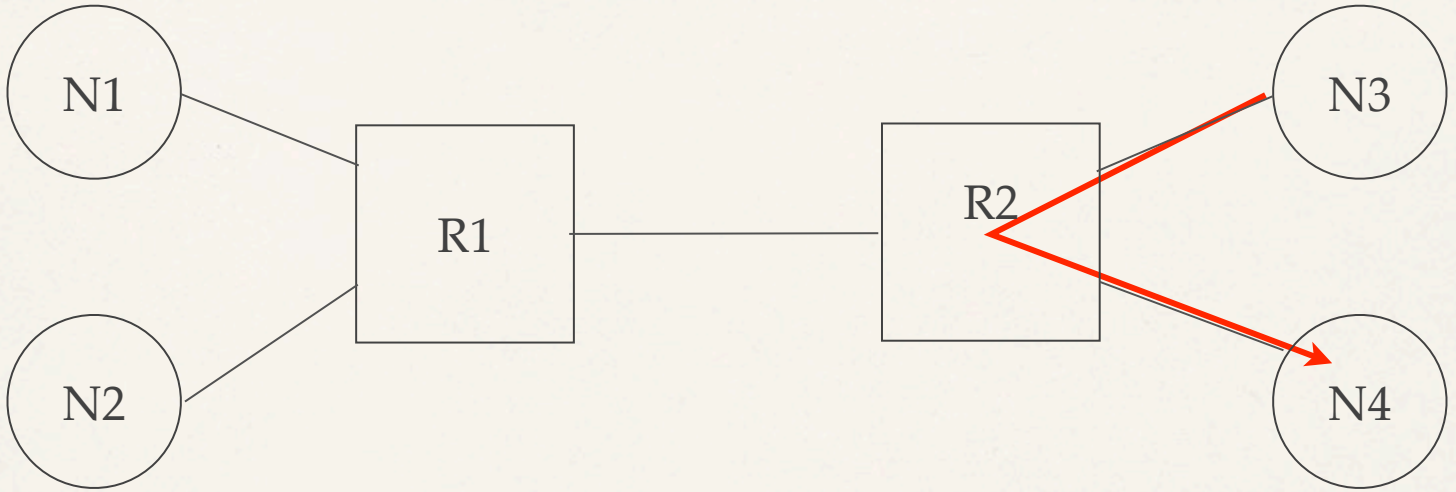


Routage Worhole



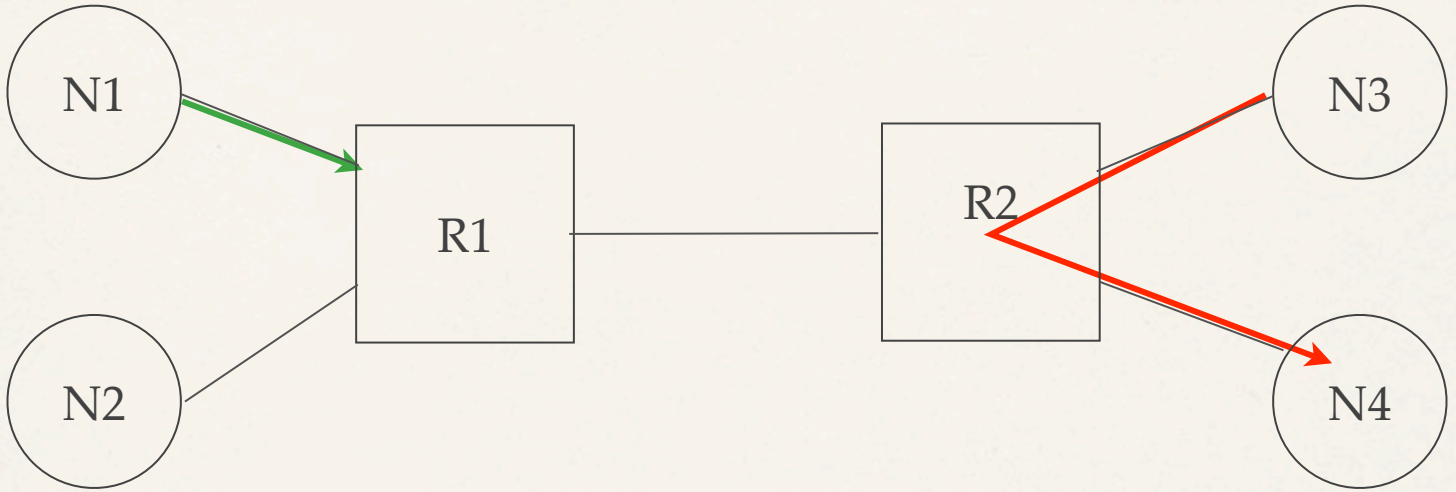
N3 envoie un paquet à N4

Routage Wormhole



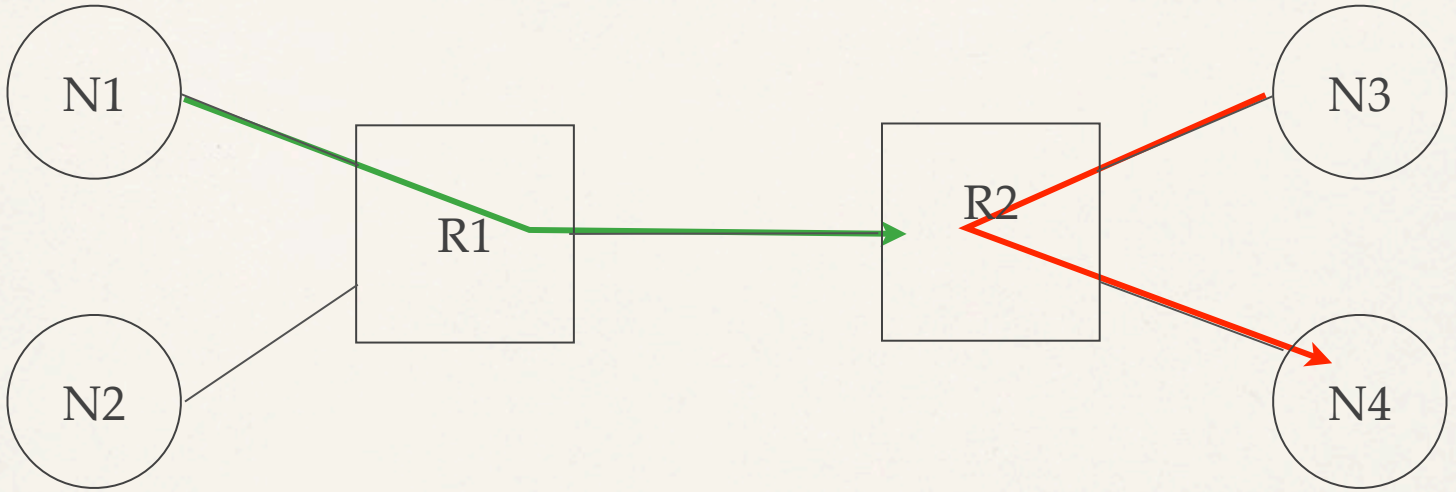
R2 retransmet les données au fur et à mesure de leur arrivée

Routage Wormhole



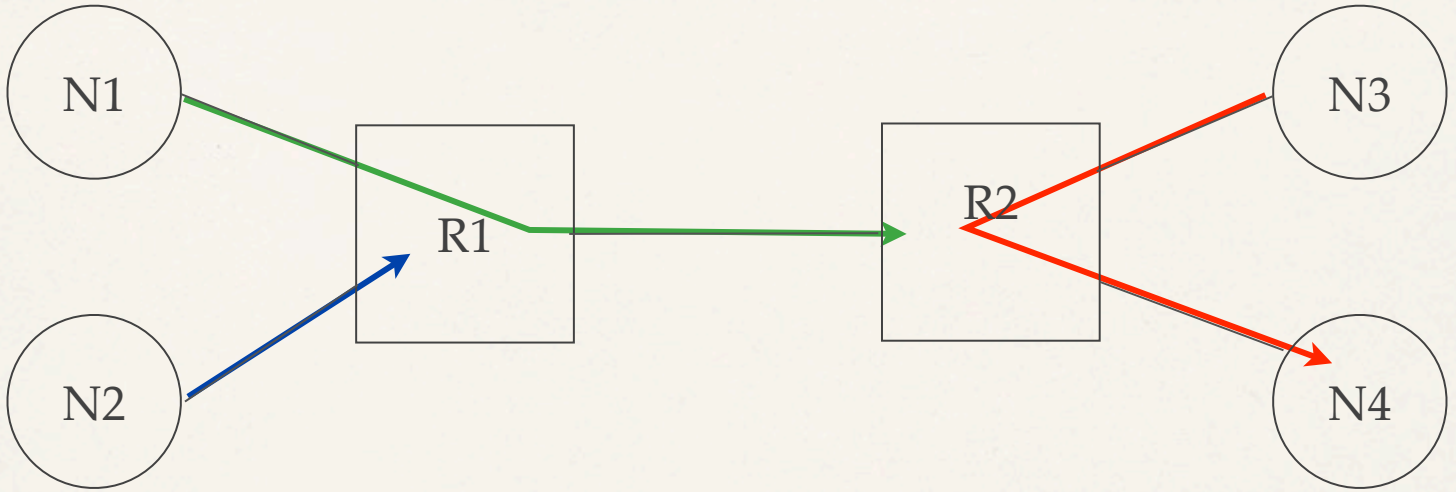
N1 commence à émettre un paquet vers N4

Routage Wormhole



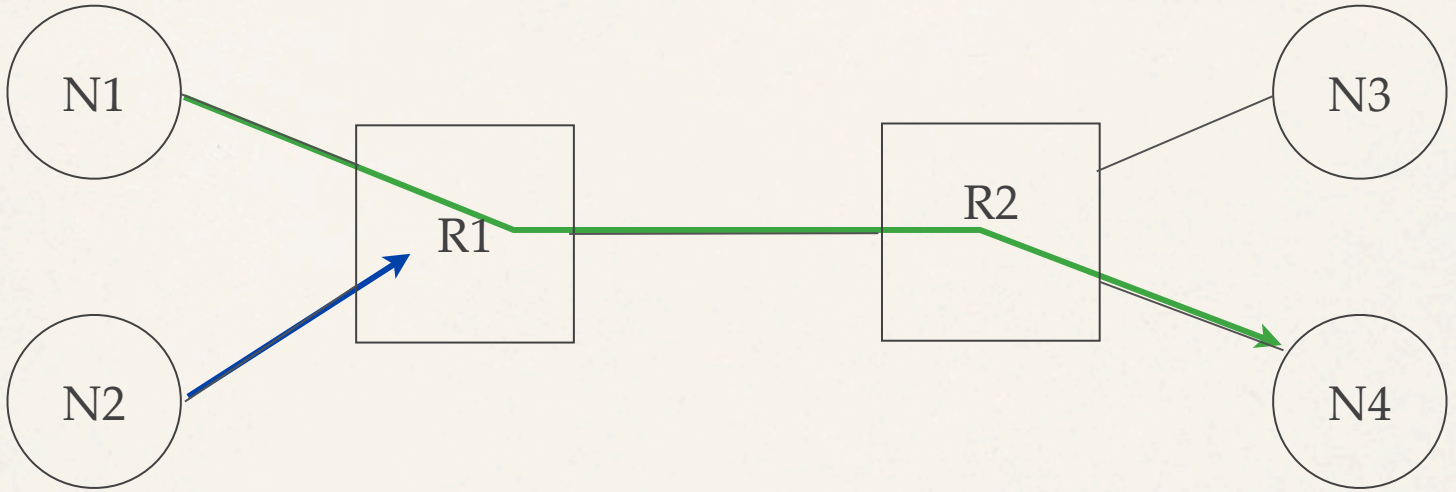
Le paquet de N1 reste bloqué dans R2

Routage Wormhole



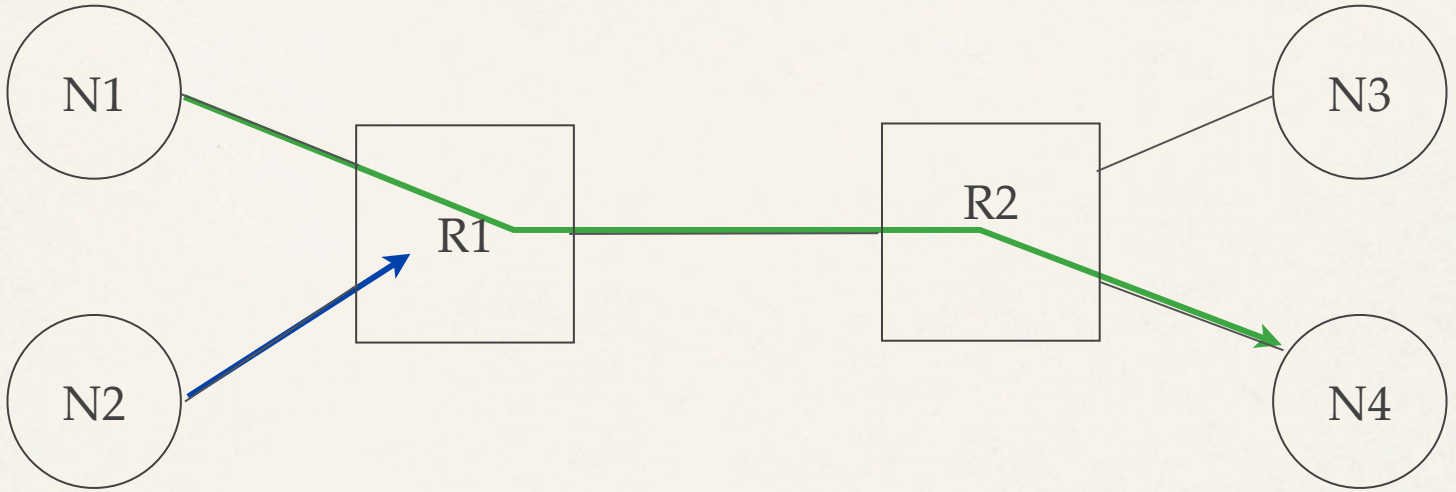
N2 tente d'envoyer un paquet à N3.
Il reste bloqué dans R1

Routage Wormhole



Une fois le paquet de N3 transmis à N4,
le paquet de N1 est transmis à N4.
Le paquet de N2 attend encore.

Routage Wormhole



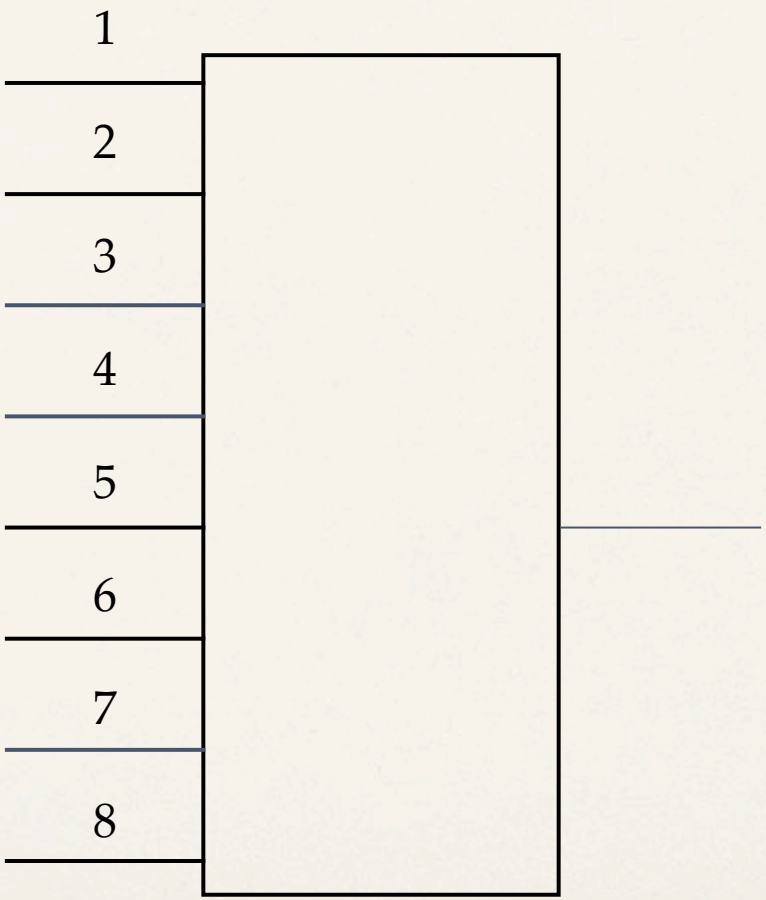
Routage Wormhole

- Avantages du WH Routing :
  - nécessite peu de mémoire
  - latence faible dans les bons cas
  - implémentation simple
- Problèmes :
  - engendre des blocages
  - le réseau peut s'écrouler
  - délais importants et irréguliers

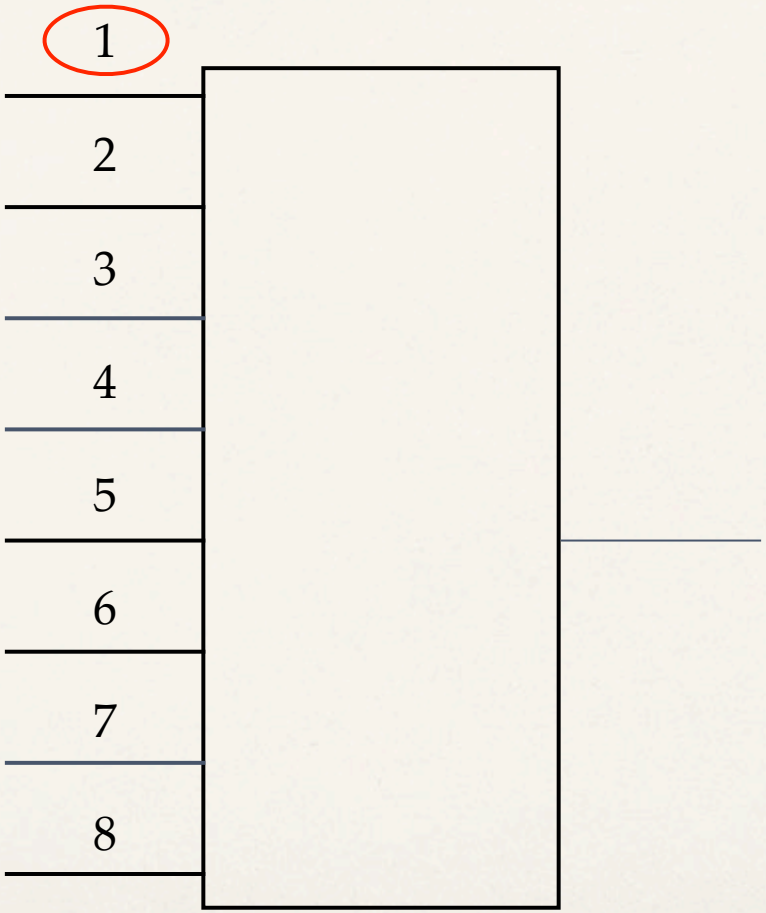
Conflits d'accès aux ports de sortie

- Si plusieurs paquets attendent la libération du même port :
  - priorités statiques sur les adresses de destination
    - deux niveaux de priorités
    - non-préemptives
  - Mécanisme de «priorités tournantes» en cas de niveaux identiques
- chaque port d'entrée peut envoyer un paquet par le port de sortie à chaque cycle

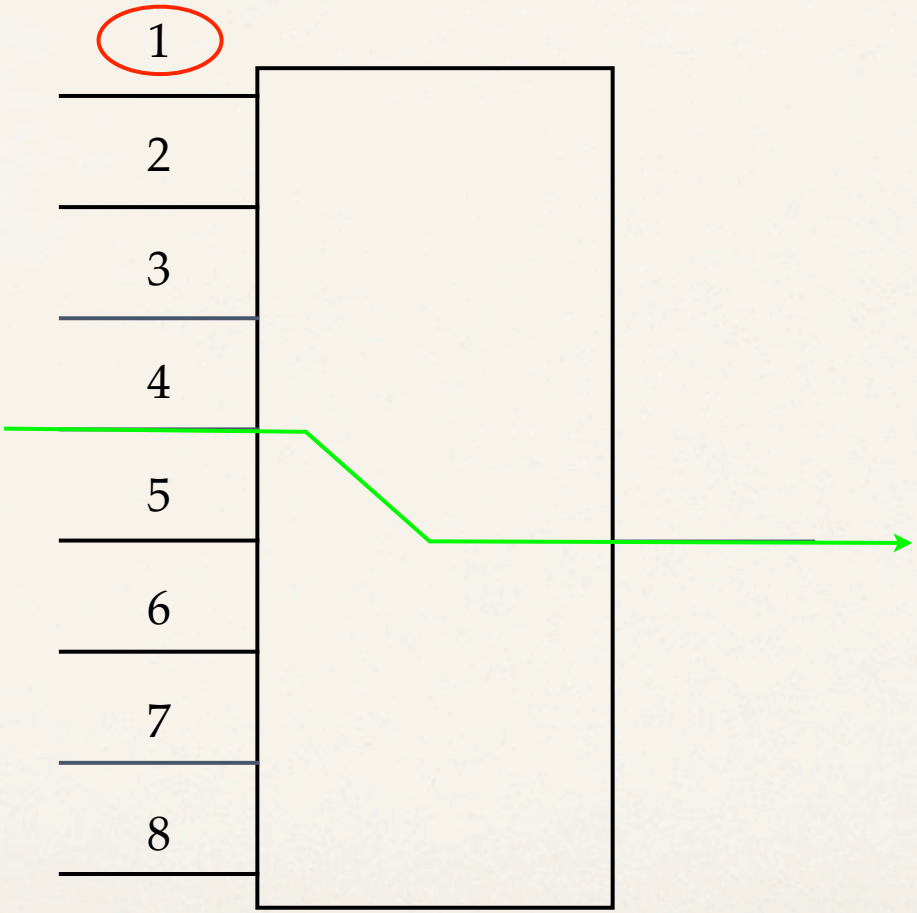
Priorités tournantes



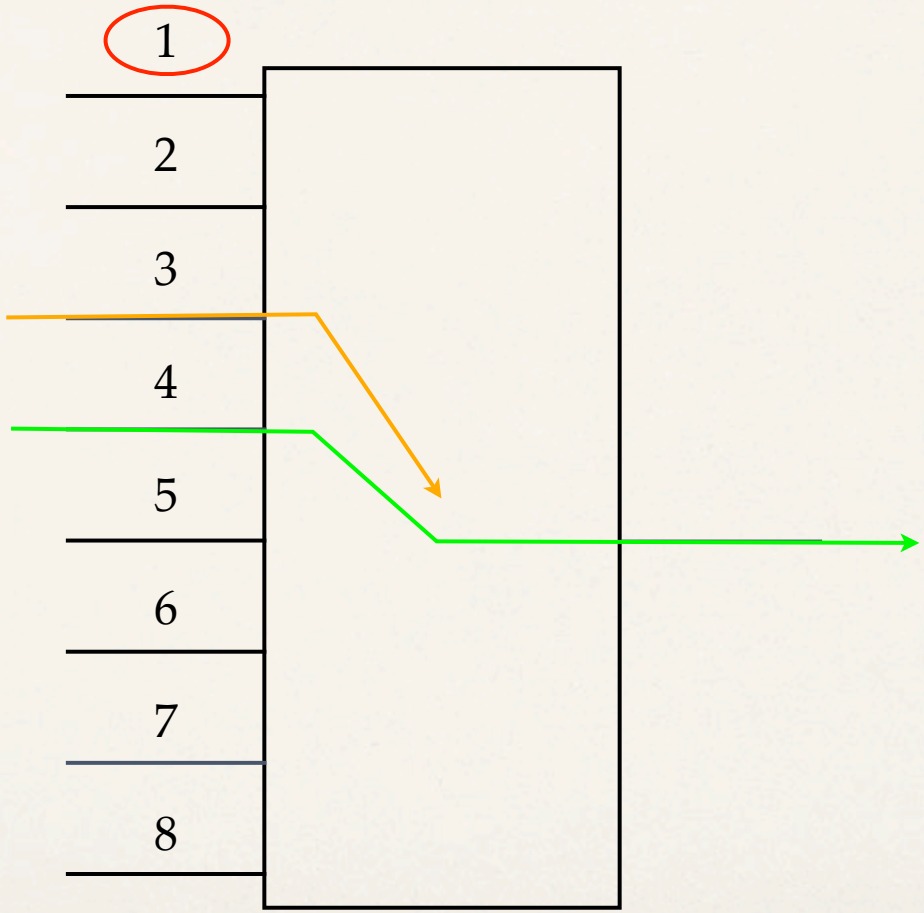
Priorités tournantes



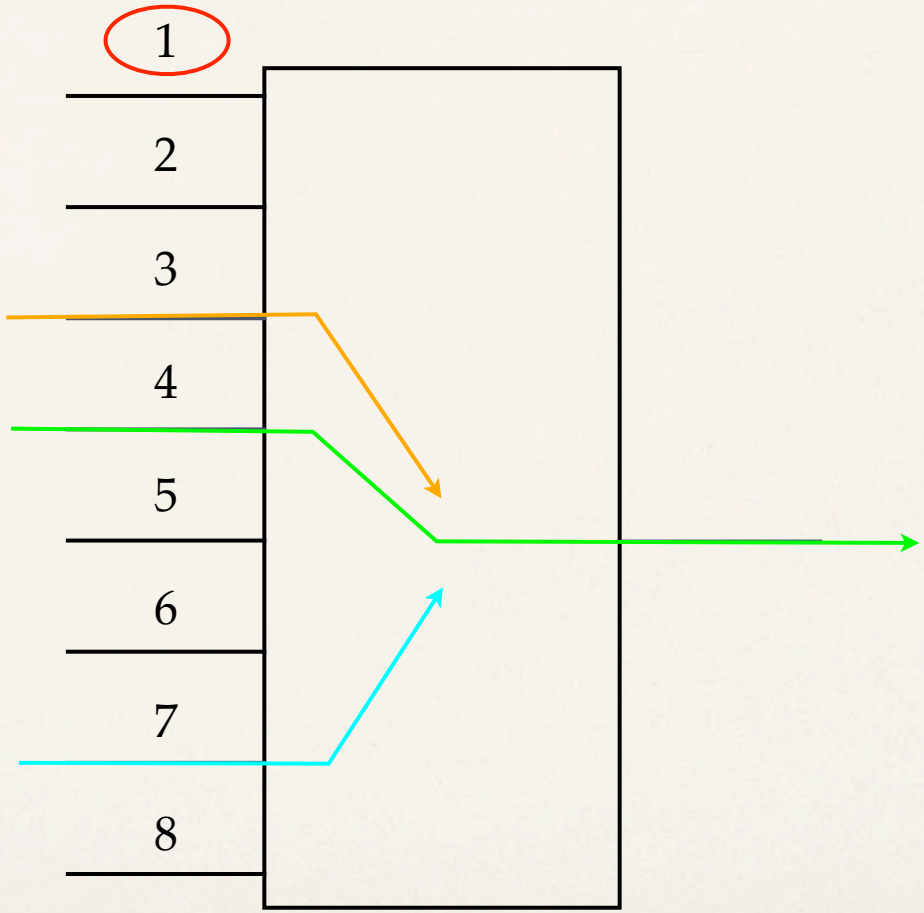
Priorités tournantes



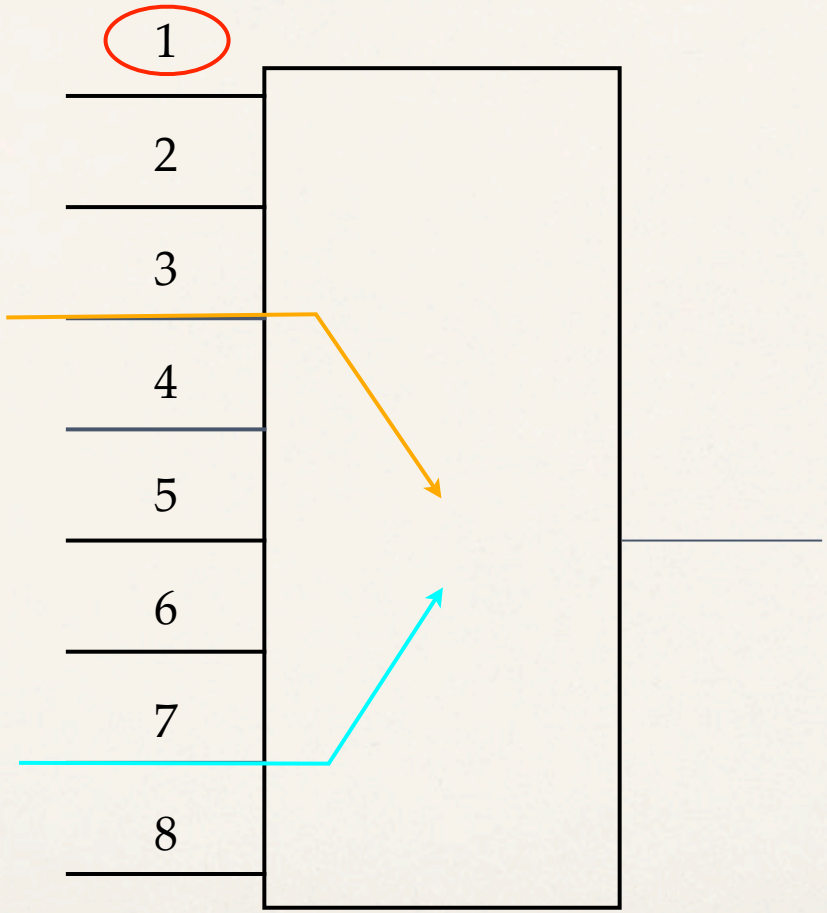
Priorités tournantes



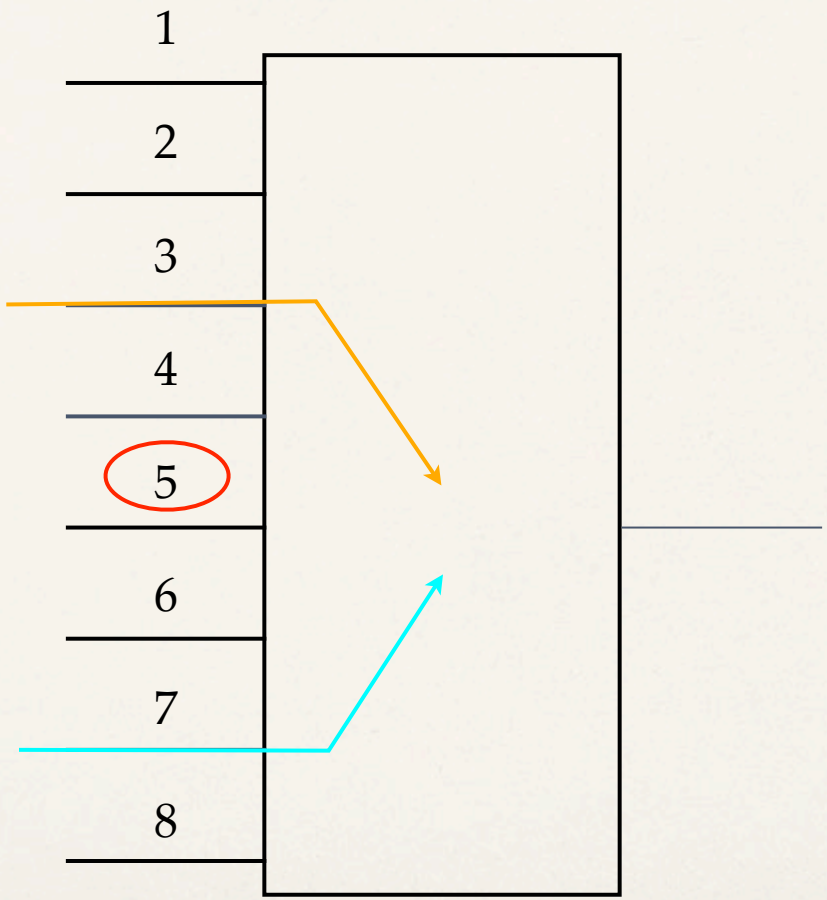
Priorités tournantes



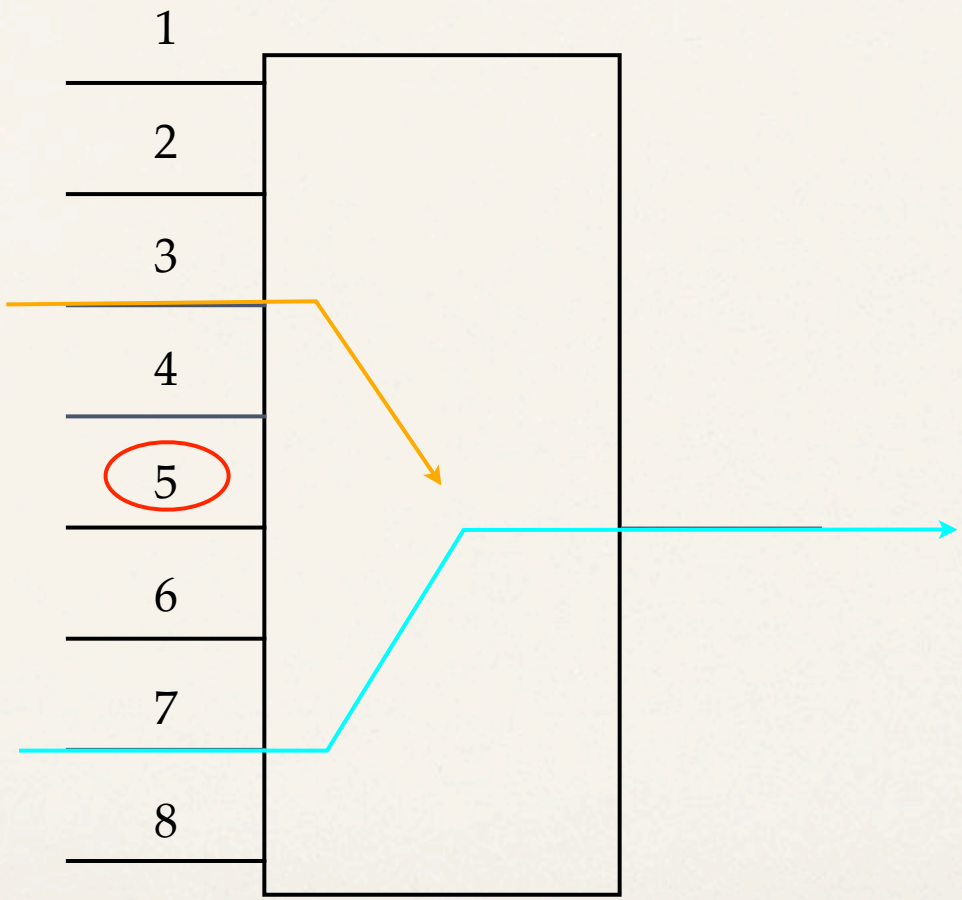
Priorités tournantes



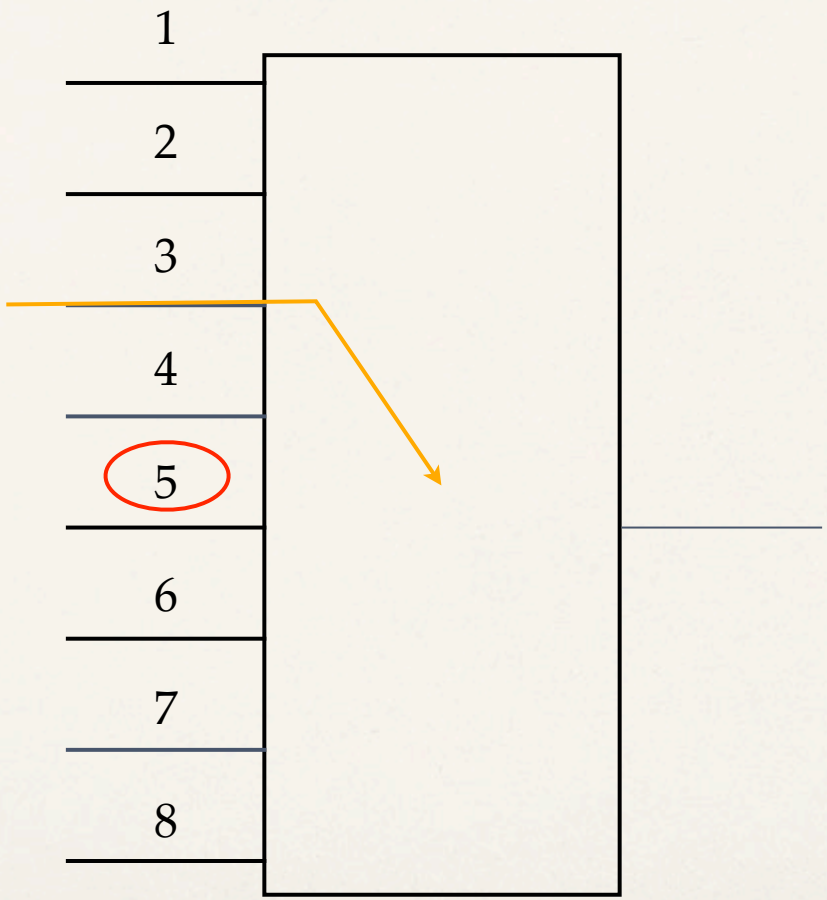
Priorités tournantes



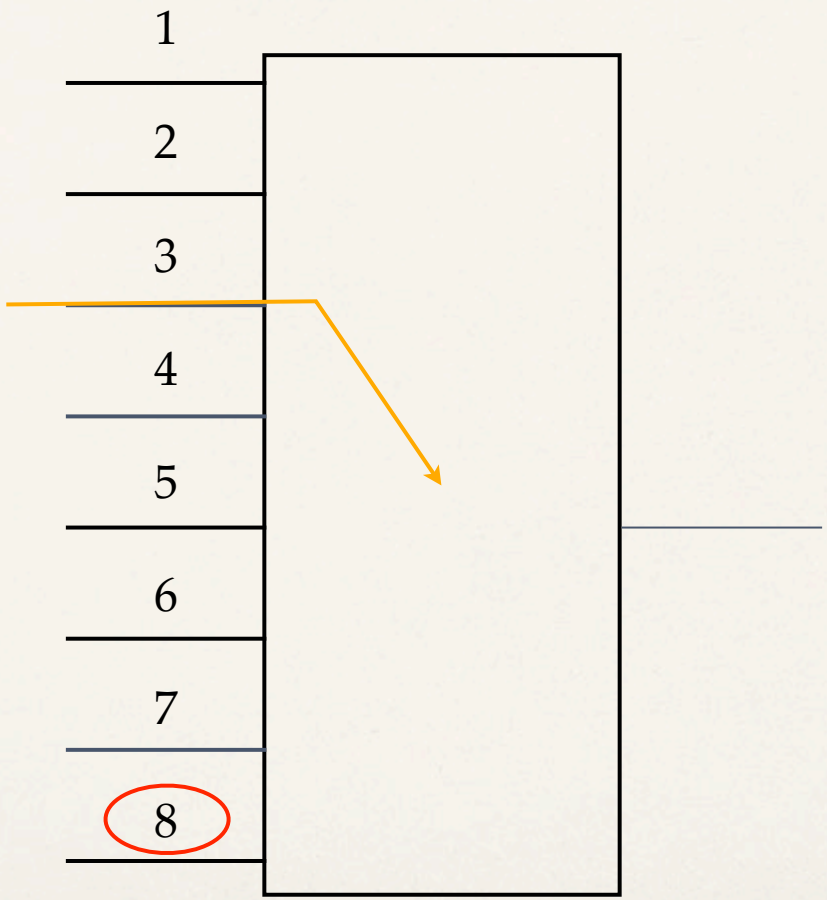
Priorités tournantes



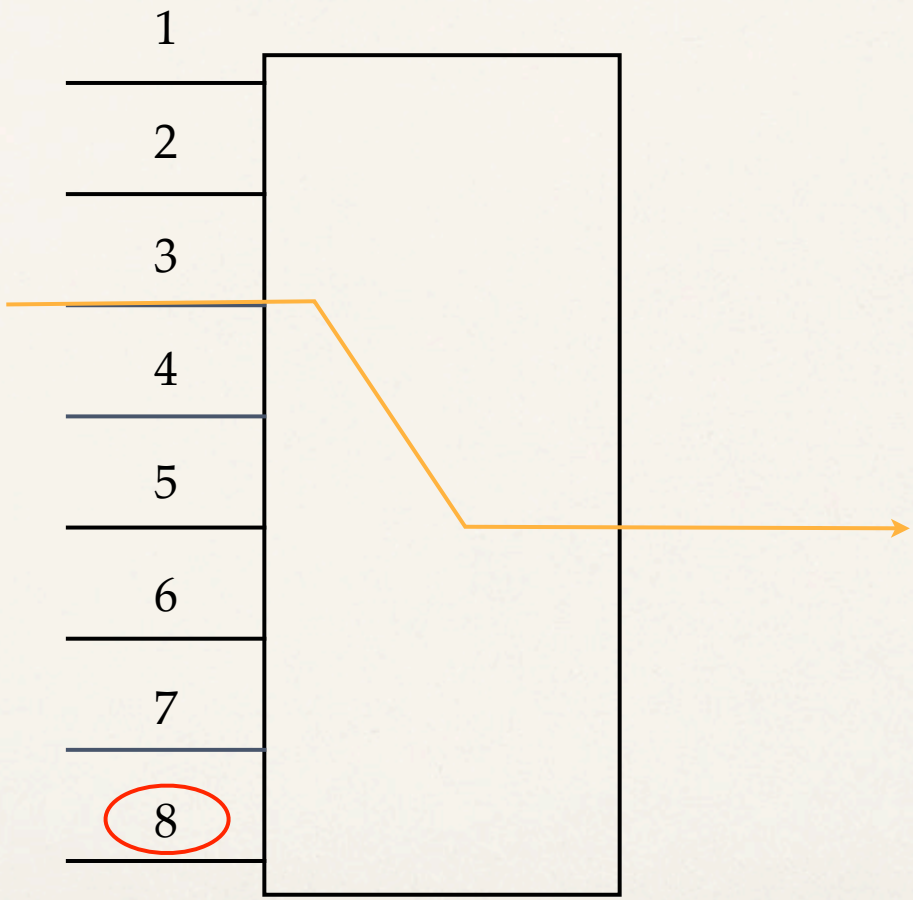
Priorités tournantes



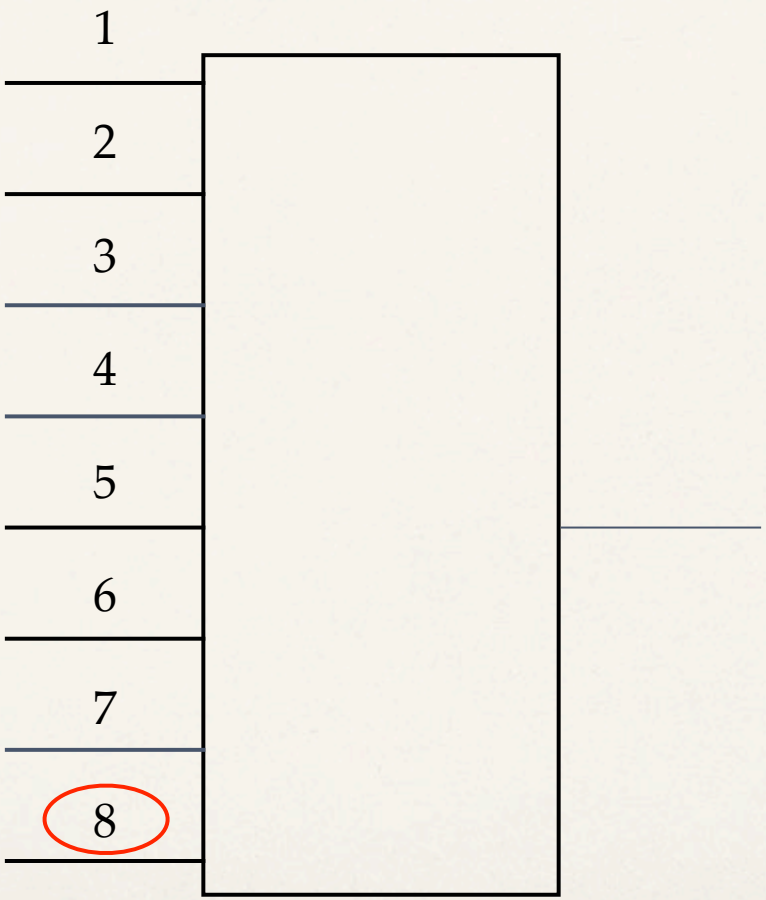
Priorités tournantes



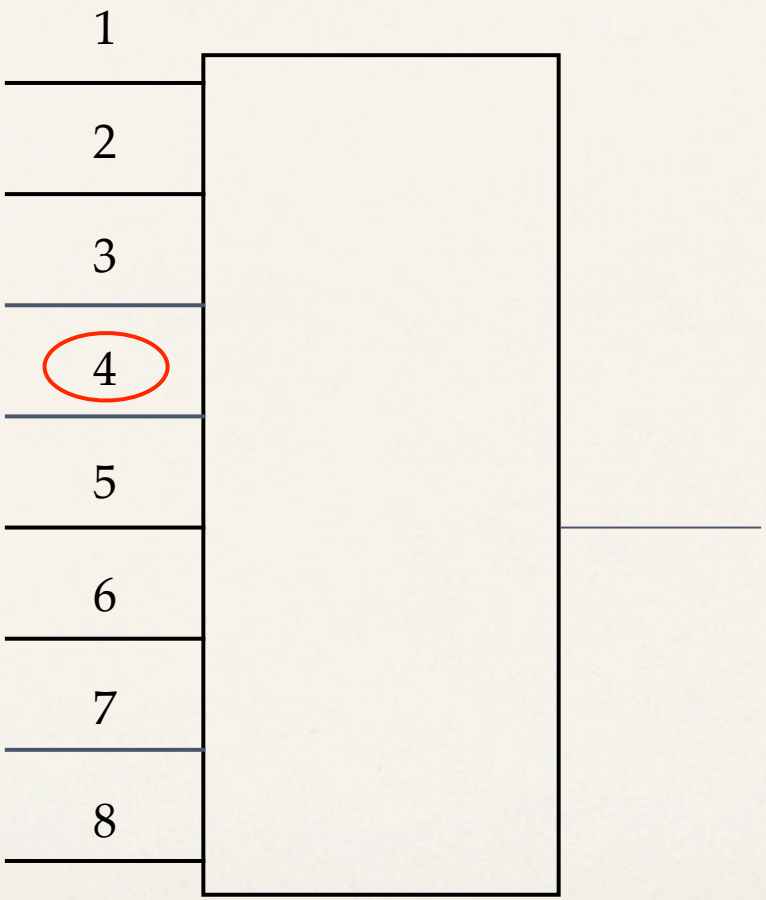
Priorités tournantes



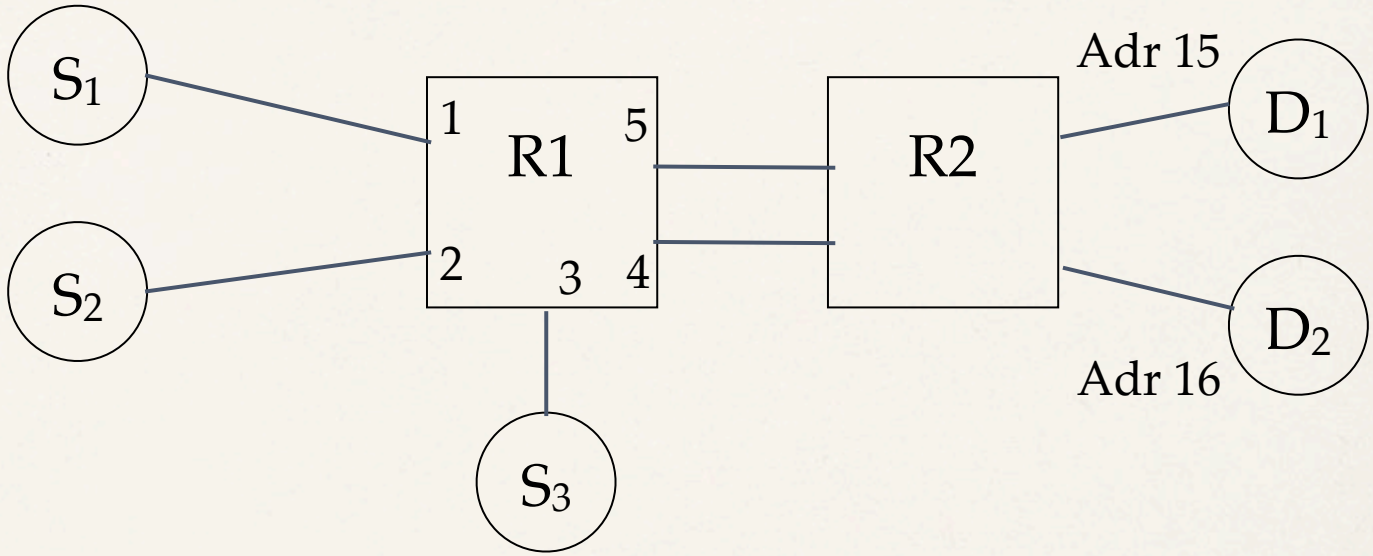
Priorités tournantes



Priorités tournantes



Group Adaptative Routing

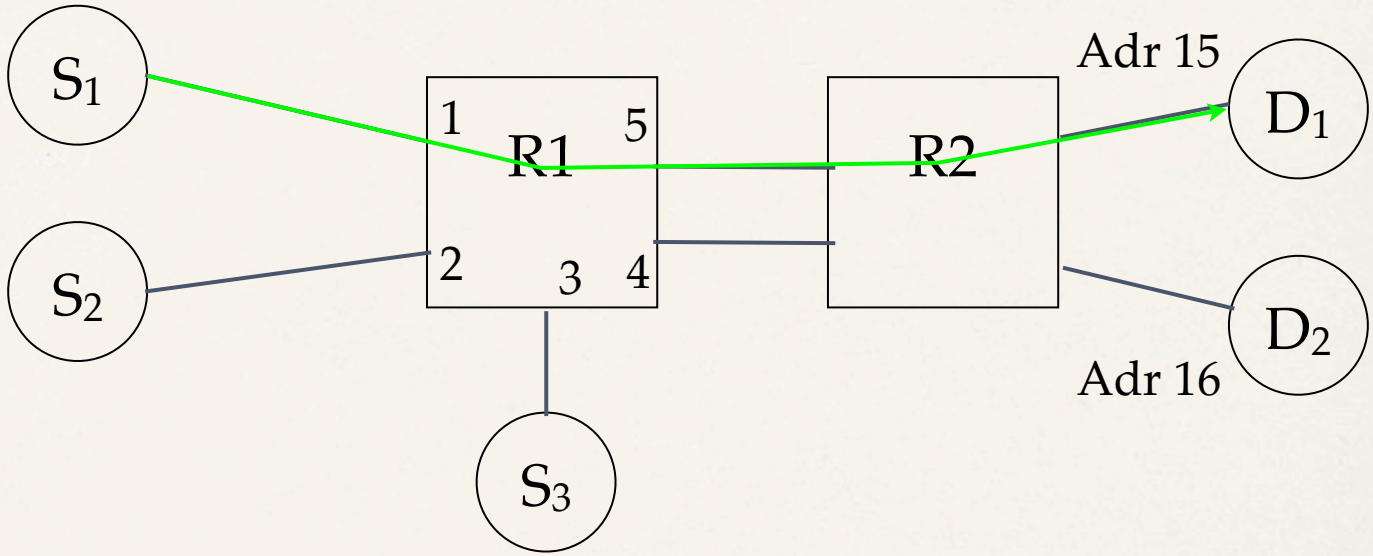


- But : utiliser plusieurs liens en parallèle entre deux routeurs
- Routeurs répartissent automatiquement le trafic

Table de routage de R1

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

Group Adaptative Routing

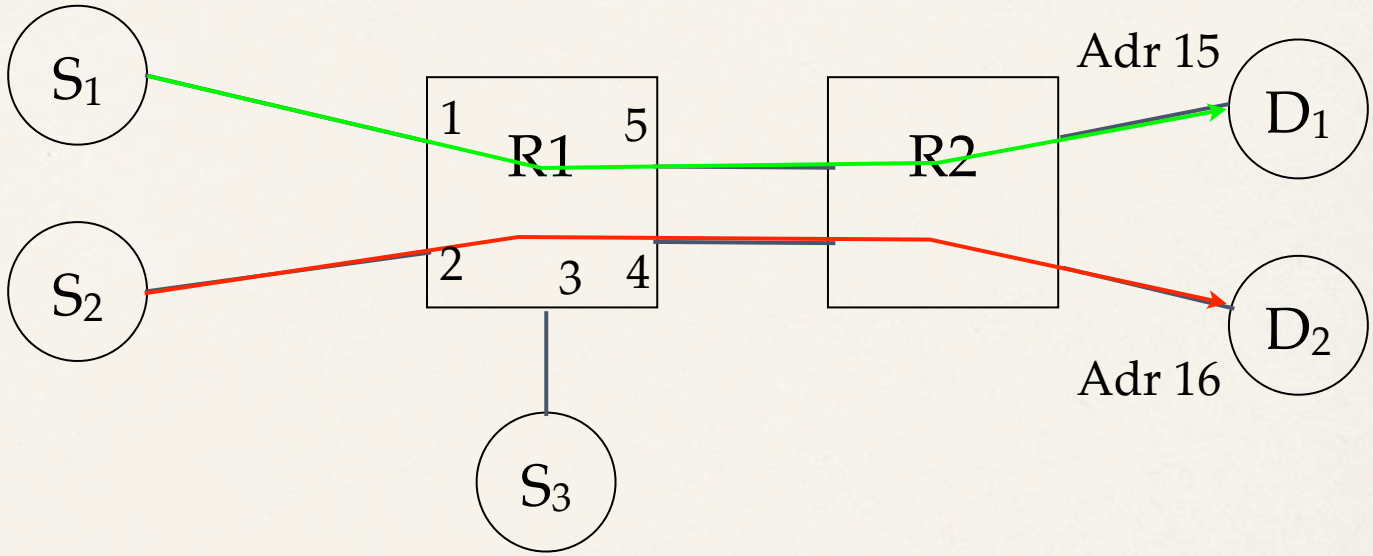


- But : utiliser plusieurs liens en parallèle entre deux routeurs
- Routeurs répartissent automatiquement le trafic

Table de routage de R1

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

Group Adaptative Routing

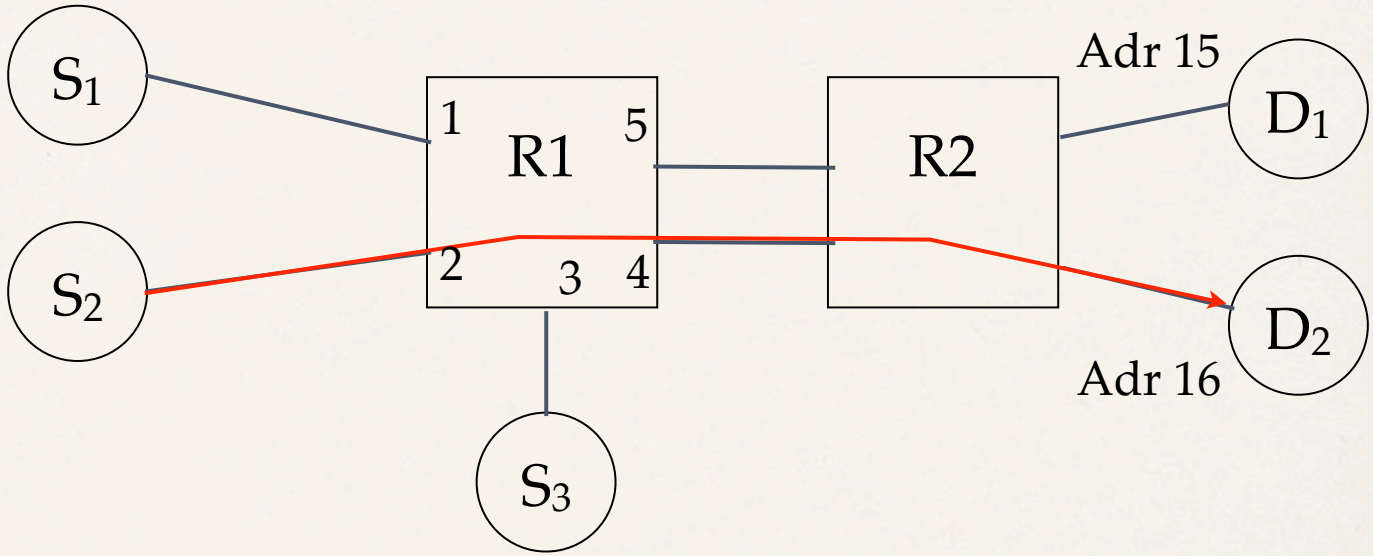


- But : utiliser plusieurs liens en parallèle entre deux routeurs
- Routeurs répartissent automatiquement le trafic

Table de routage de R1

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

Group Adaptative Routing

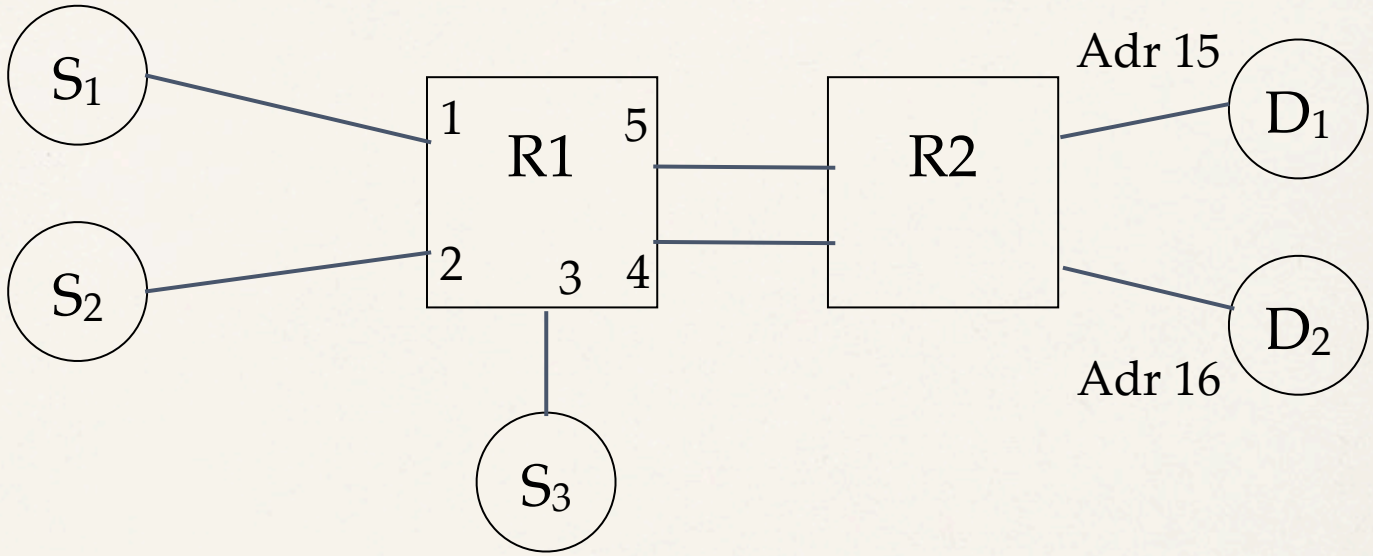


- But : utiliser plusieurs liens en parallèle entre deux routeurs
- Routeurs répartissent automatiquement le trafic

Table de routage de R1

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

Group Adaptative Routing

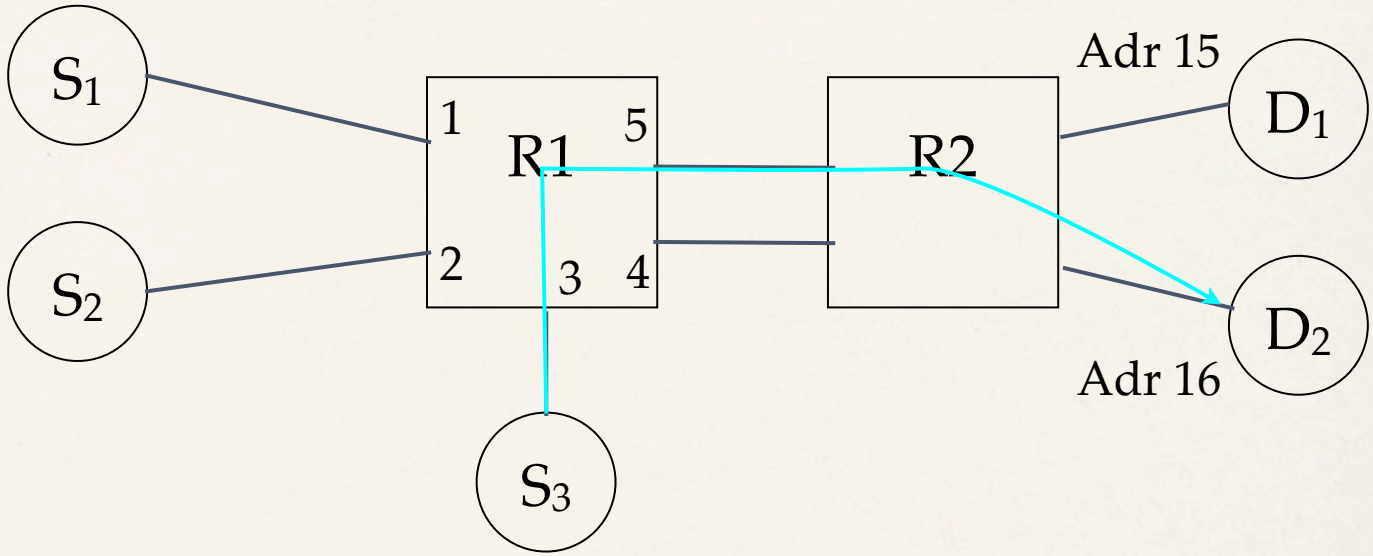


- But : utiliser plusieurs liens en parallèle entre deux routeurs
- Routeurs répartissent automatiquement le trafic

Table de routage de R1

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

Group Adaptative Routing

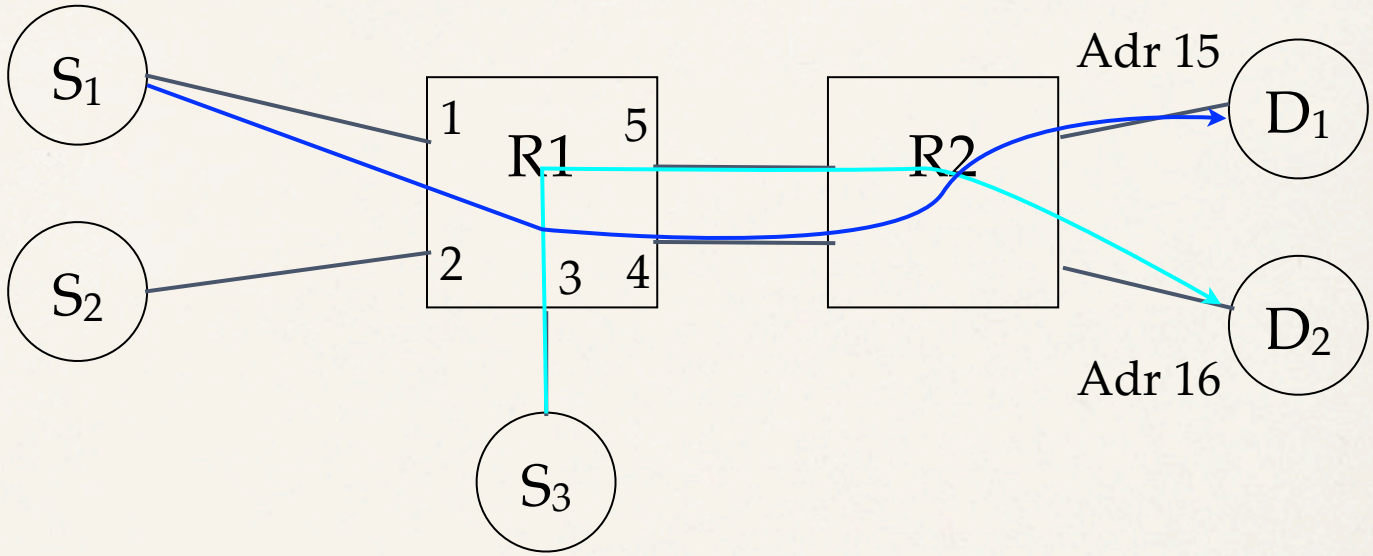


- But : utiliser plusieurs liens en parallèle entre deux routeurs
- Routeurs répartissent automatiquement le trafic

Table de routage de R1

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

Group Adaptative Routing

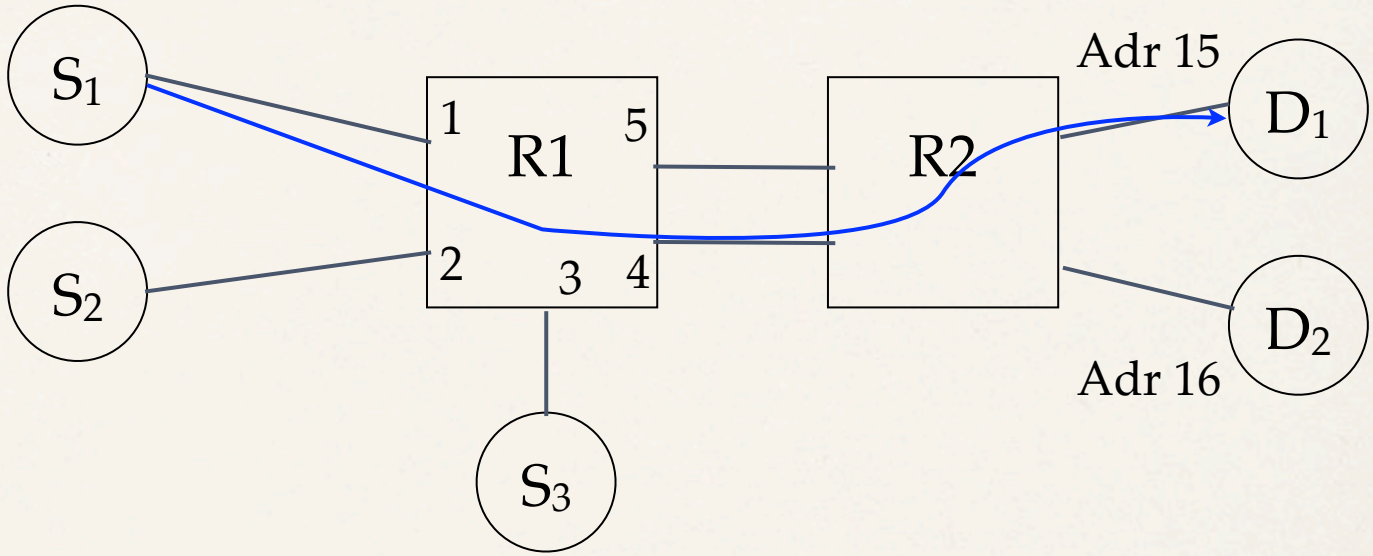


- But : utiliser plusieurs liens en parallèle entre deux routeurs
- Routeurs répartissent automatiquement le trafic

Table de routage de R1

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

Group Adaptative Routing

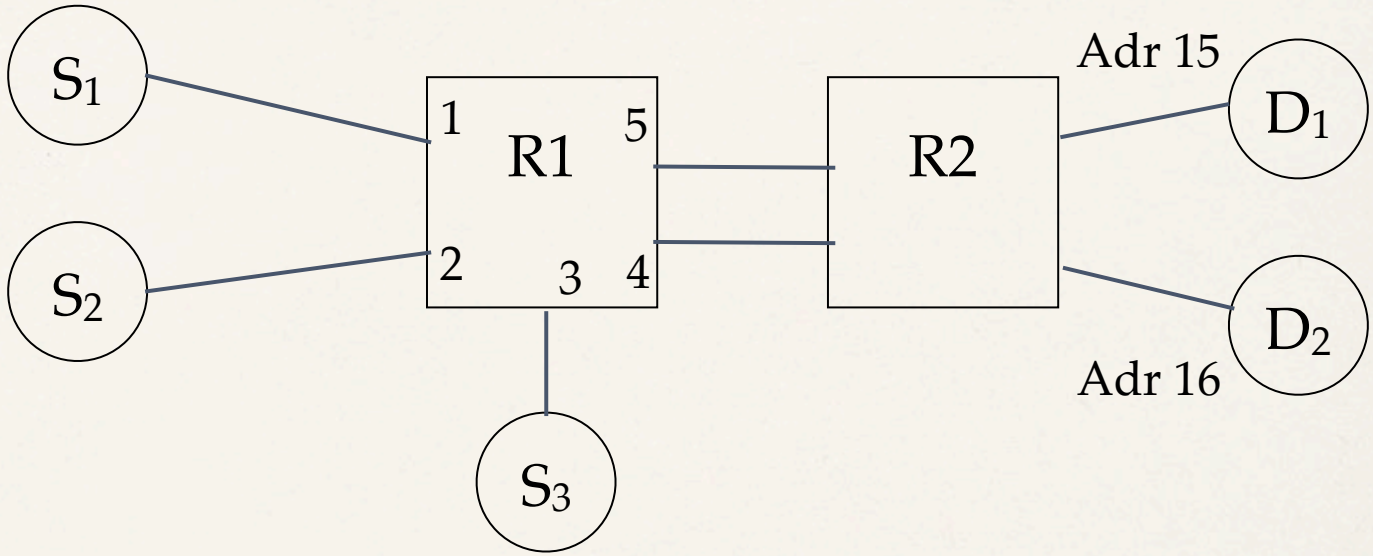


- But : utiliser plusieurs liens en parallèle entre deux routeurs
- Routeurs répartissent automatiquement le trafic

Table de routage de R1

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

Group Adaptative Routing



- But : utiliser plusieurs liens en parallèle entre deux routeurs
- Routeurs répartissent automatiquement le trafic

Table de routage de R1

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

Time-Codes

- Code de contrôle générée par les interfaces SpaceWire
- Format : caractère ESC + 1 car de données
  Prioritaires sur tous les autres caractères
- assurent la synchronisation d'un compteur sur 6 bits dans chaque nœud/routeur (⇒ 64 valeurs possibles)
- un nœud maître incrémente son compteur périodiquement et transmet un TC sur chaque interface
- les routeurs les répercutent de même en évitant les boucles infinies

SpaceWire et temps-réel

- **Problématique TR** : transmettre les trafics scientifique et commande/contrôle sur un même réseau en respectant leurs contraintes propres
- SpaceWire fournit une bande passante suffisante
- Problème : délais de livraison non déterministes
  - blocages en cascade
  - taille importante des paquets scientifiques

Comment garantir le respect des contraintes temporelles ?

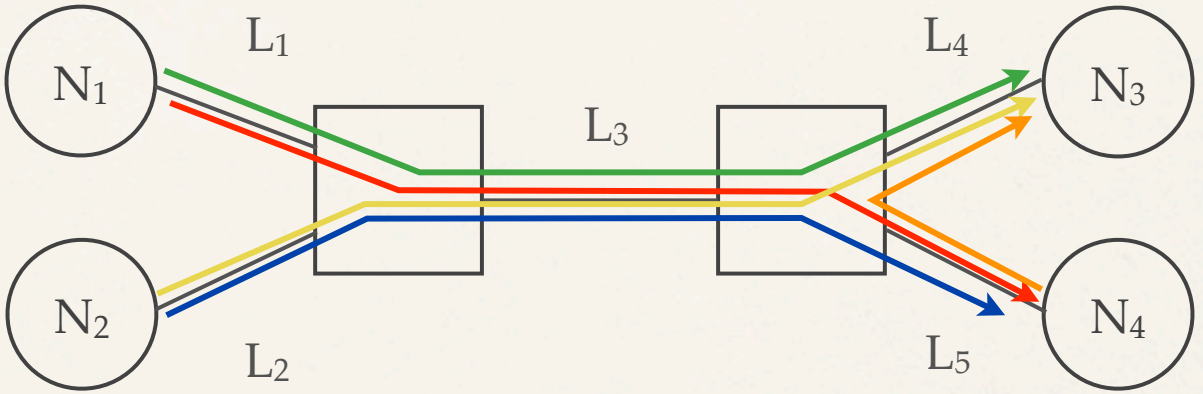
1ère solution

- calculer les délais de bout en bout pour chaque flux
  - impossible de façon analytique à cause des interactions complexes entre les paquets
  - résultats probabilistes insuffisants
- **calculer une borne supérieure sur le délai pire-cas de bout en bout de chaque flux**

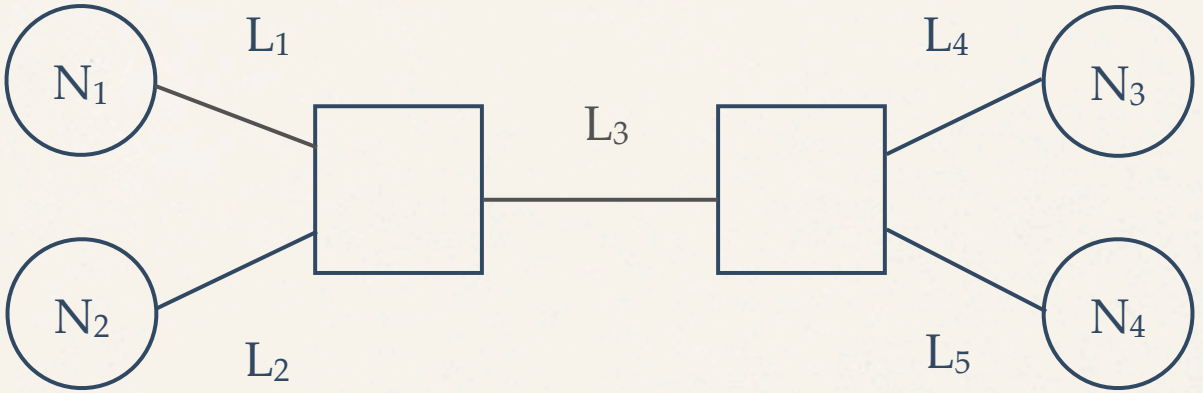
Principe du calcul

- la transmission d'un paquet a lieu en 2 phases
- 1ère phase : l'en-tête du paquet crée un «circuit virtuel» de la source à la destination
  - délais à la source si plusieurs paquets sont émis simultanément
  - délais dans chaque routeur traversé si d'autres paquets occupent déjà le port de sortie
- 2ème phase : le corps du paquet est transféré sur le circuit virtuel
  - même délai que sur un lien point-à-point : T/C

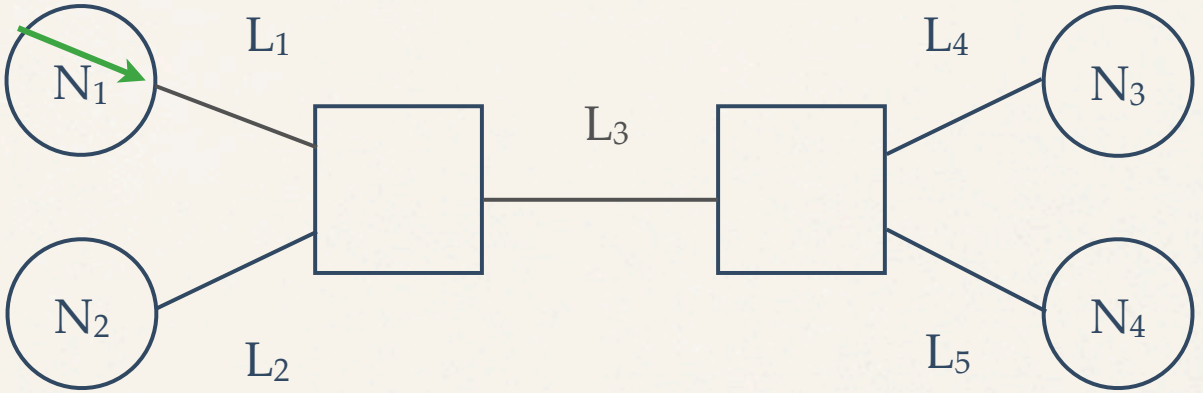
Un exemple pour commencer



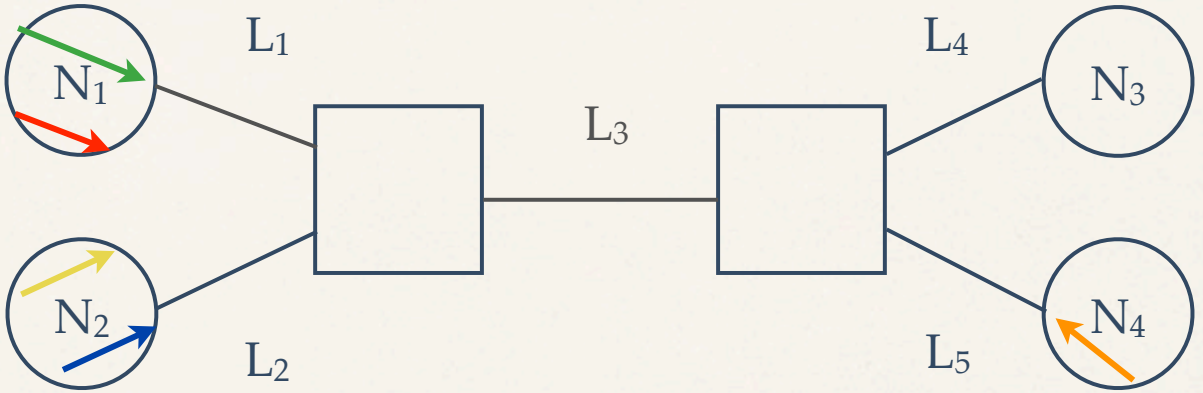
Calculons le délai du flux vert



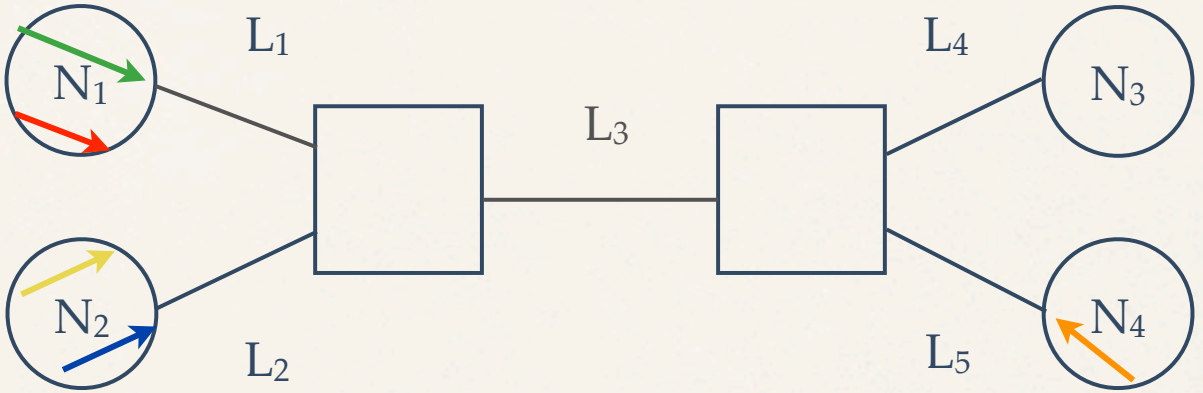
Calculons le délai du flux vert



Calculons le délai du flux vert

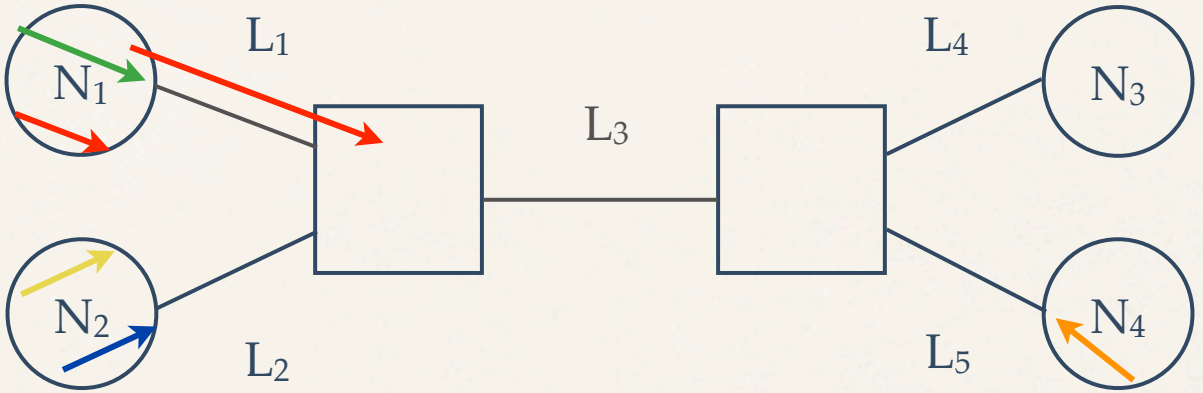


Calculons le délai du flux vert



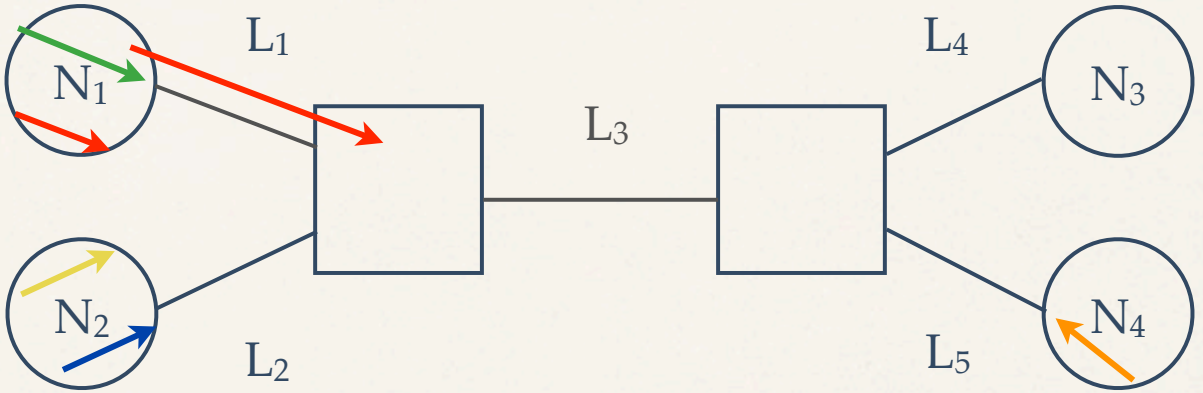
$$d(f_v) = d(f_v, l_1)$$

Calculons le délai du flux vert



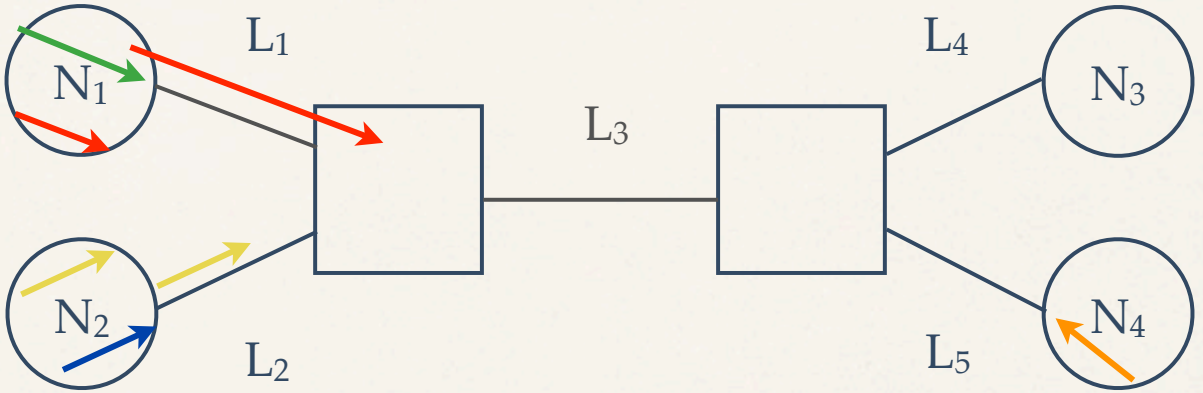
$$d(f_v) = d(f_v, l_1)$$

Calculons le délai du flux vert



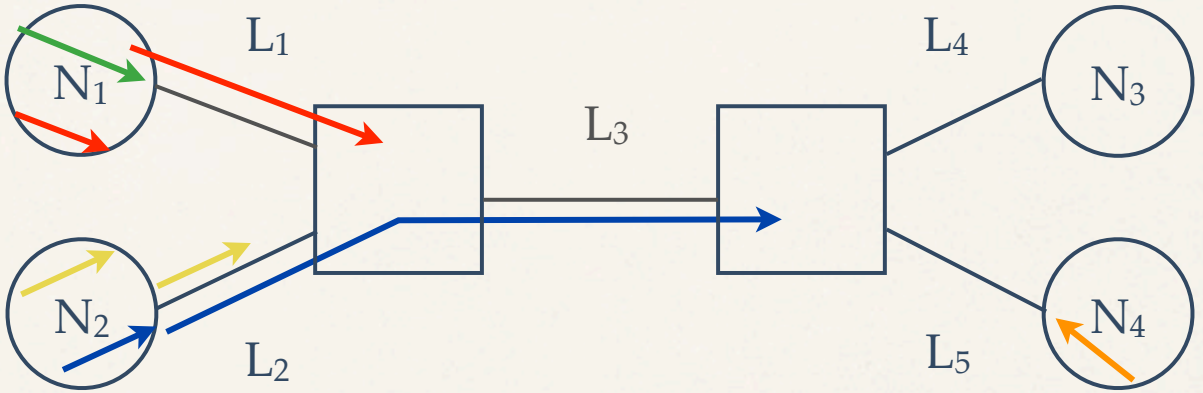
$$d(f_v) = d(f_v, l_3) + d(f_r, l_3)$$

Calculons le délai du flux vert



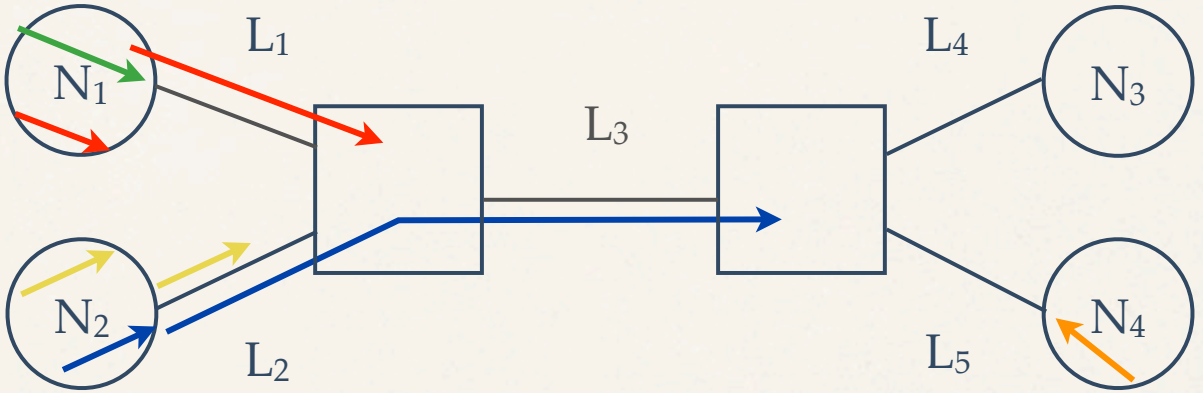
$$d(f_v) = d(f_v, l_3) + d(f_r, l_3)$$

Calculons le délai du flux vert



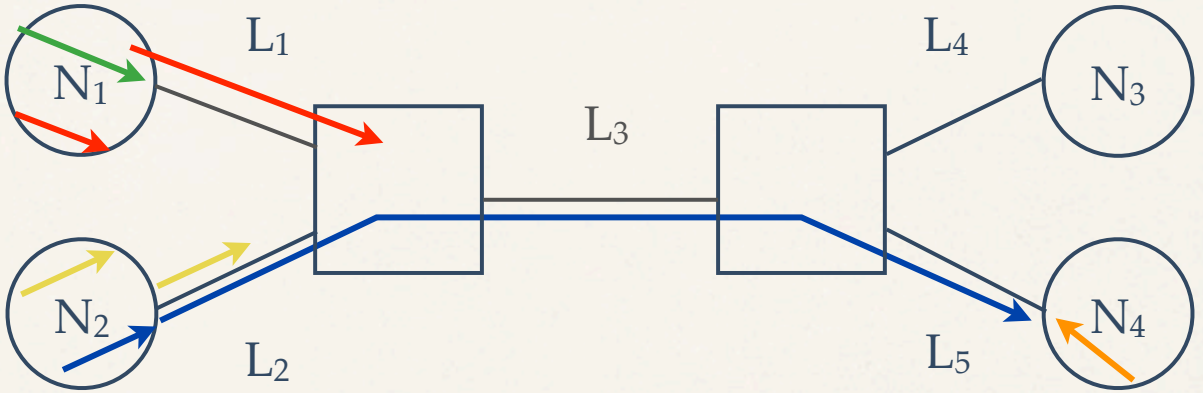
$$d(f_v) = d(f_v, l_3) + d(f_r, l_3)$$

Calculons le délai du flux vert



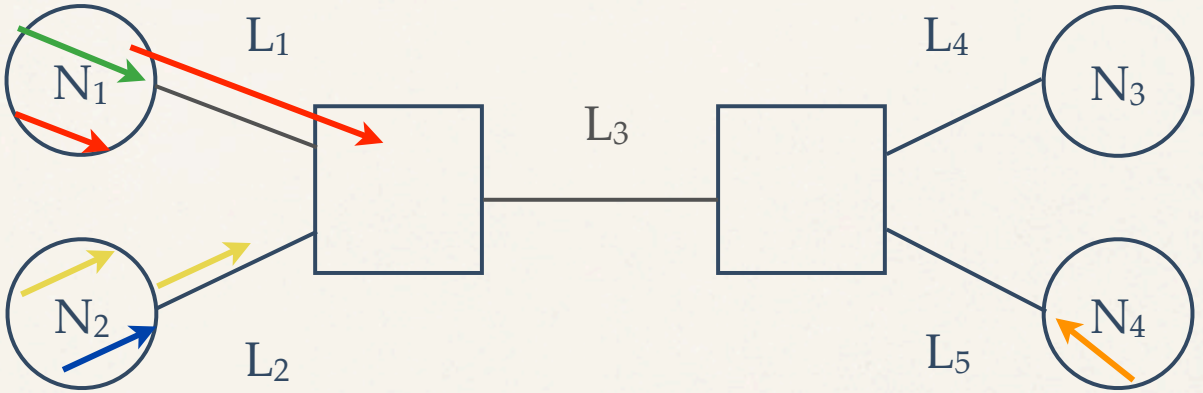
$$d(f_v) = d(f_v, l_3) + d(f_r, l_5) + \max[d(f_j, l_4), d(f_b, l_5)]$$

Calculons le délai du flux vert



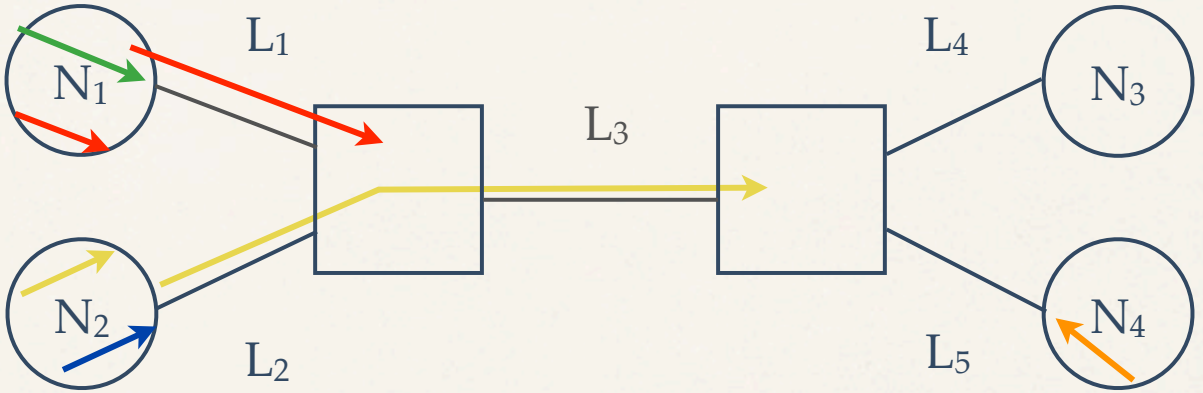
$$d(f_v) = d(f_v, l_3) + d(f_r, l_5) + \max[d(f_j, l_4), d(f_b, l_5)]$$

Calculons le délai du flux vert



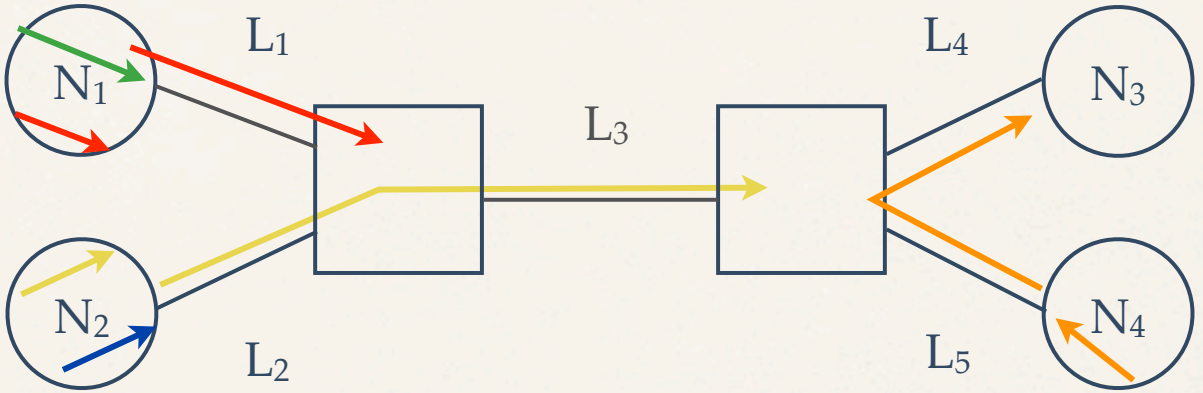
$$d(f_v) = d(f_v, l_3) + d(f_r, l_5) + \max[d(f_j, l_4), d(f_b, l_5)]$$

Calculons le délai du flux vert



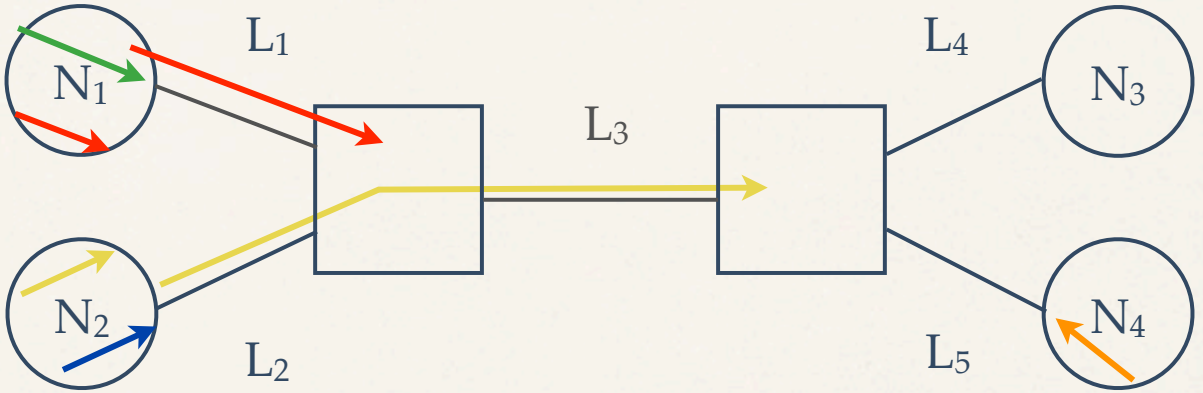
$$d(f_v) = d(f_v, l_3) + d(f_r, l_5) + \max[d(f_j, l_4), d(f_b, l_5)]$$

Calculons le délai du flux vert



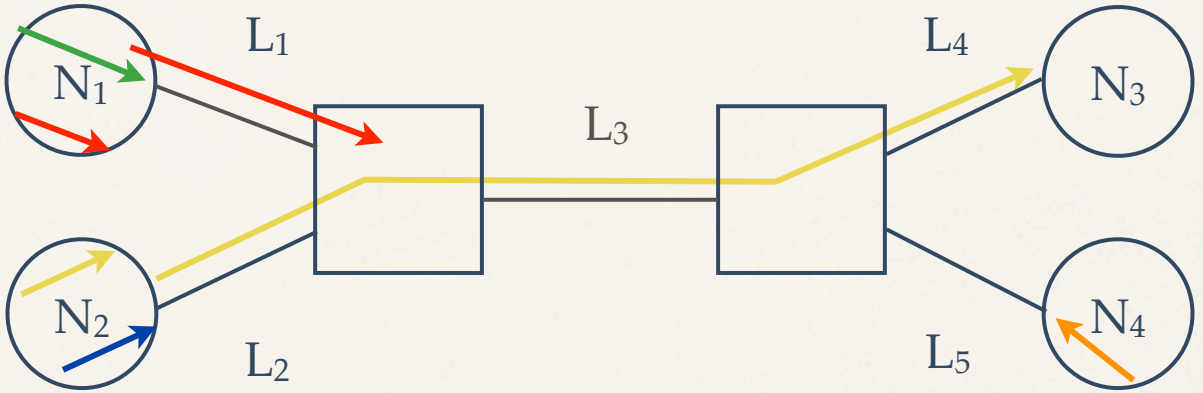
$$d(f_v) = d(f_v, l_3) + d(f_r, l_5) + \max[d(f_j, l_4), d(f_b, l_5)]$$

Calculons le délai du flux vert



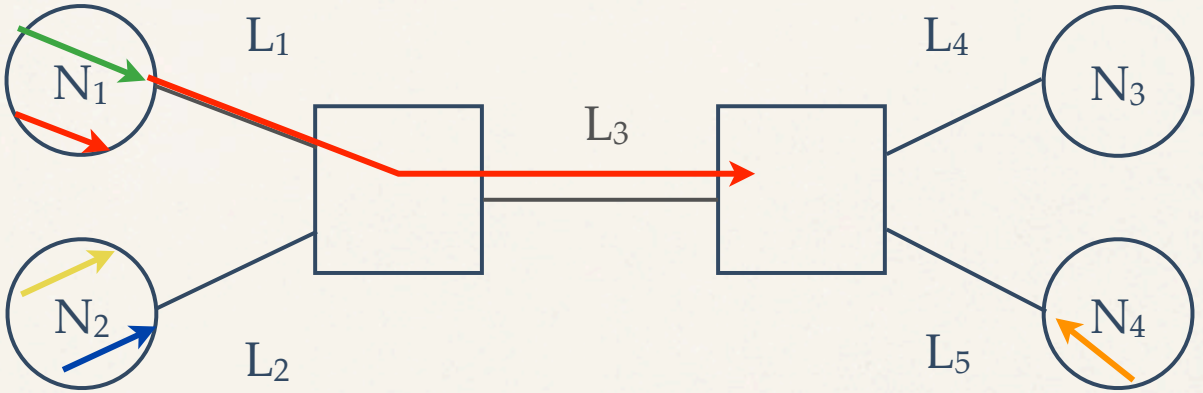
$$d(f_v) = d(f_v, l_3) + d(f_r, l_5) + \max[d(f_j, l_4), d(f_b, l_5)]$$

Calculons le délai du flux vert



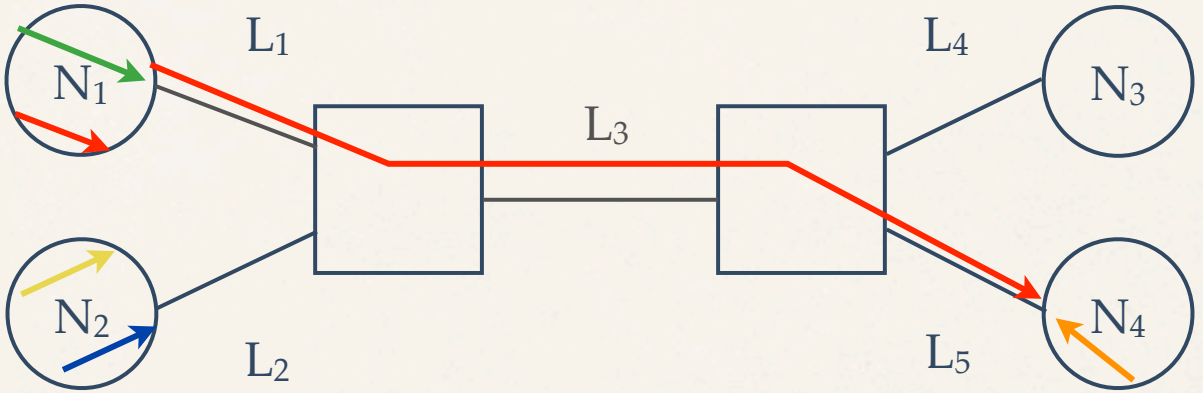
$$d(f_v) = d(f_v, l_3) + d(f_r, l_5) + \max[d(f_j, l_4), d(f_b, l_5)]$$

Calculons le délai du flux vert



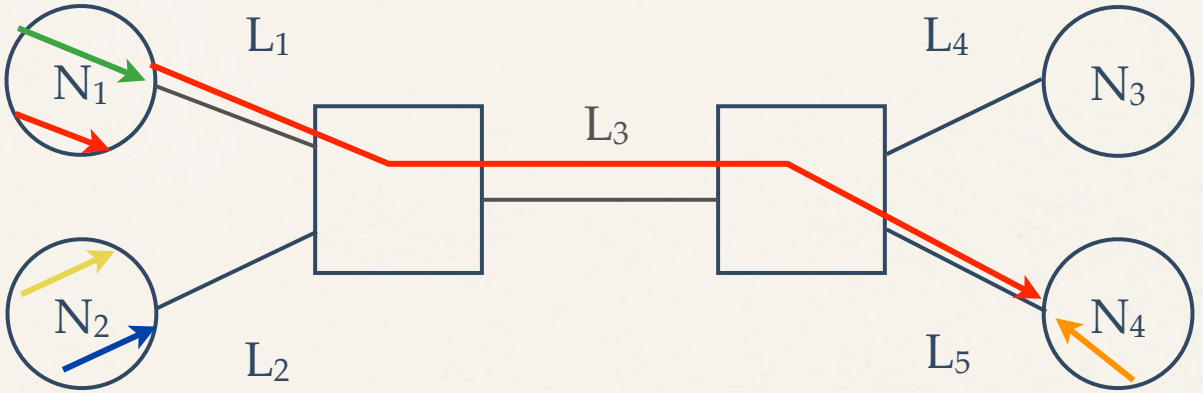
$$d(f_v) = d(f_v, l_3) + d(f_r, l_5) + \max[d(f_j, l_4), d(f_b, l_5)]$$

Calculons le délai du flux vert



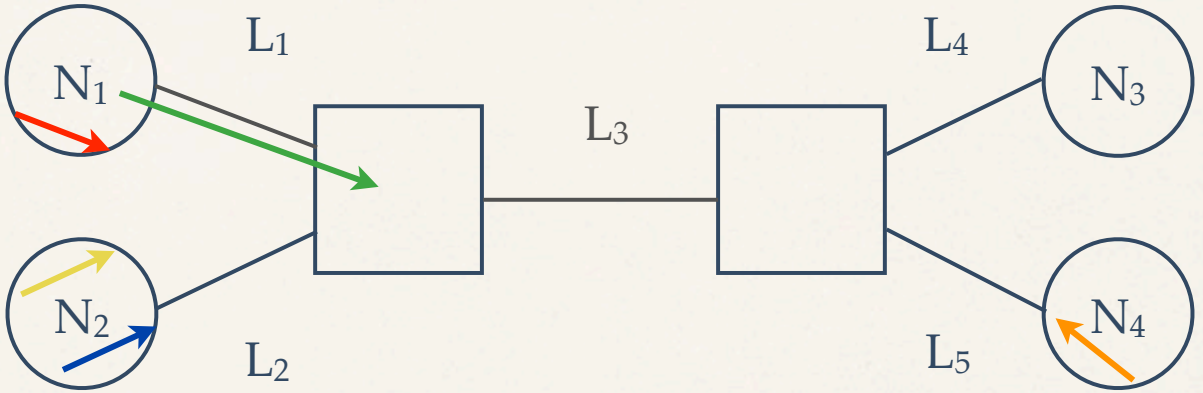
$$d(f_v) = d(f_v, l_3) + d(f_r, l_5) + \max[d(f_j, l_4), d(f_b, l_5)]$$

Calculons le délai du flux vert



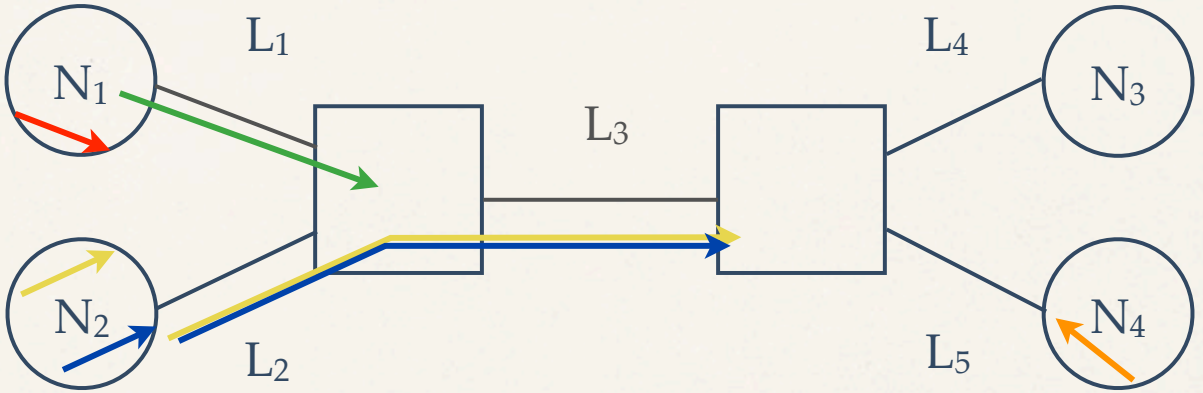
$$d(f_v) = d(f_v, l_3) + d(f_r, null) + \max[d(f_j, null) + d(f_o, null), d(f_b, null)]$$

Calculons le délai du flux vert



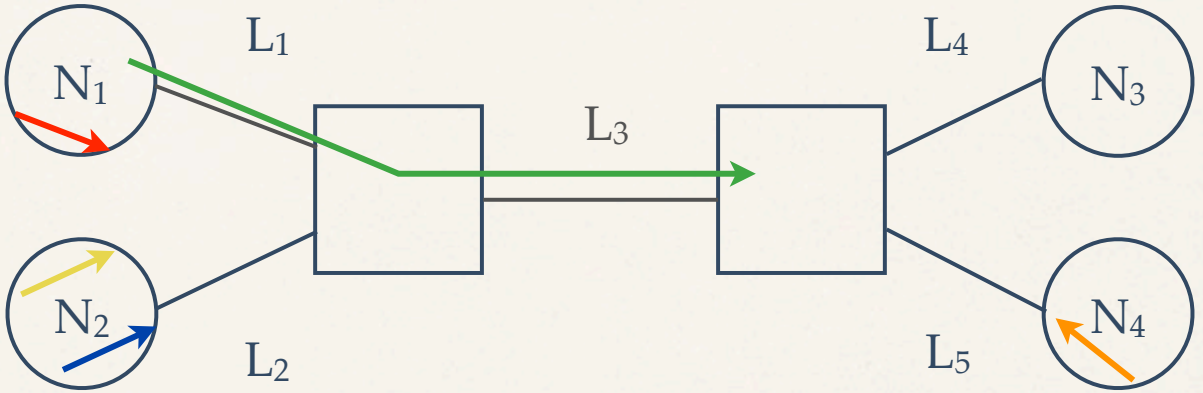
$$d(f_v) = d(f_v, l_3) + d(f_r, null) + \max[d(f_j, null) + d(f_o, null), d(f_b, null)]$$

Calculons le délai du flux vert



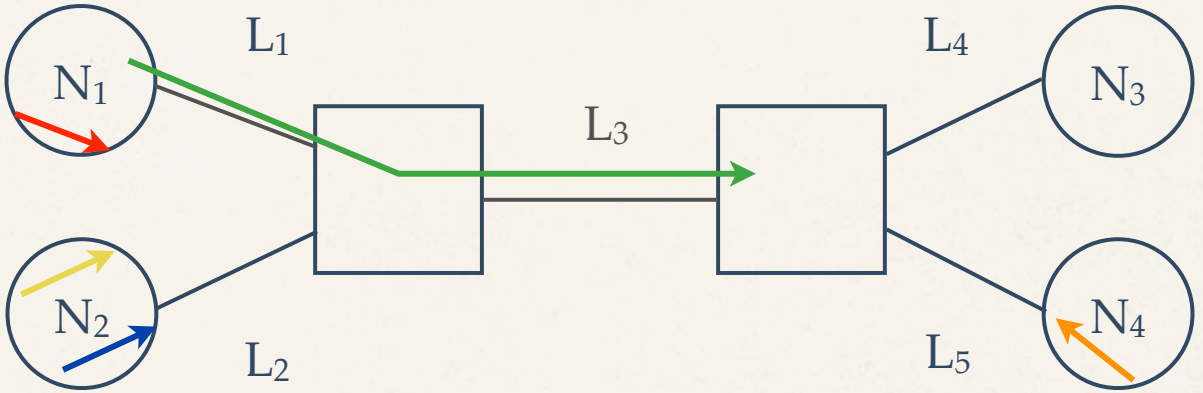
$$d(f_v) = d(f_v, l_3) + d(f_r, null) + \max[d(f_j, null) + d(f_o, null), d(f_b, null)]$$

Calculons le délai du flux vert



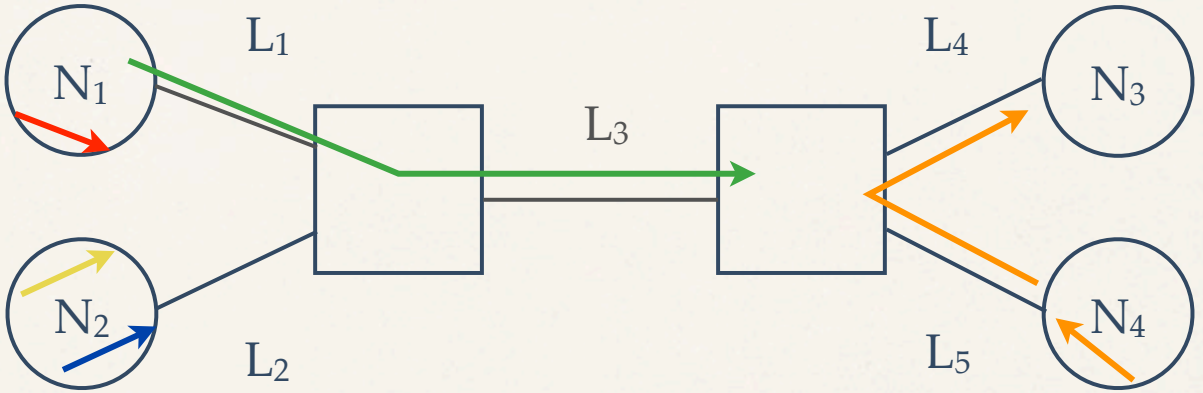
$$d(f_v) = d(f_v, l_3) + d(f_r, null) + \max[d(f_j, null) + d(f_o, null), d(f_b, null)]$$

Calculons le délai du flux vert



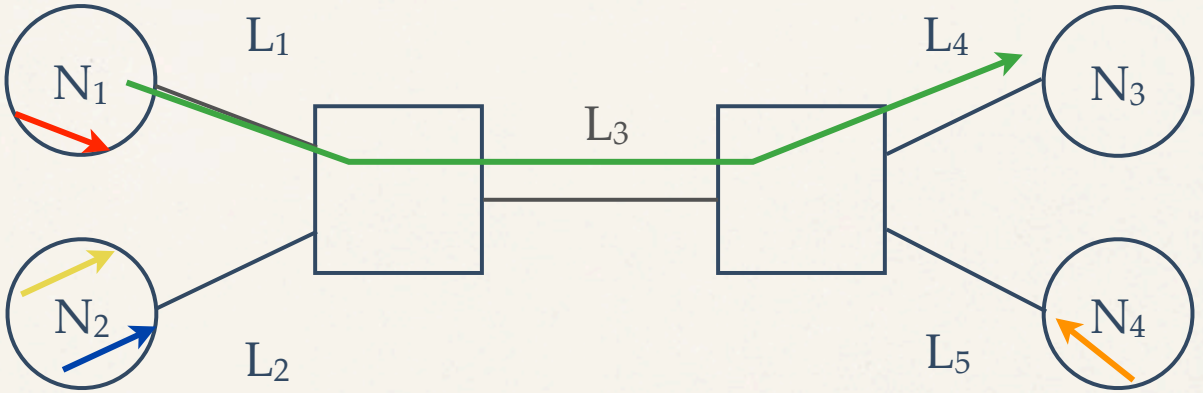
$$d(f_v) = d(f_v, l_4) + d(f_r, null) + 2.\max[d(f_j, null) + d(f_o, null), d(f_b, null)]$$

Calculons le délai du flux vert



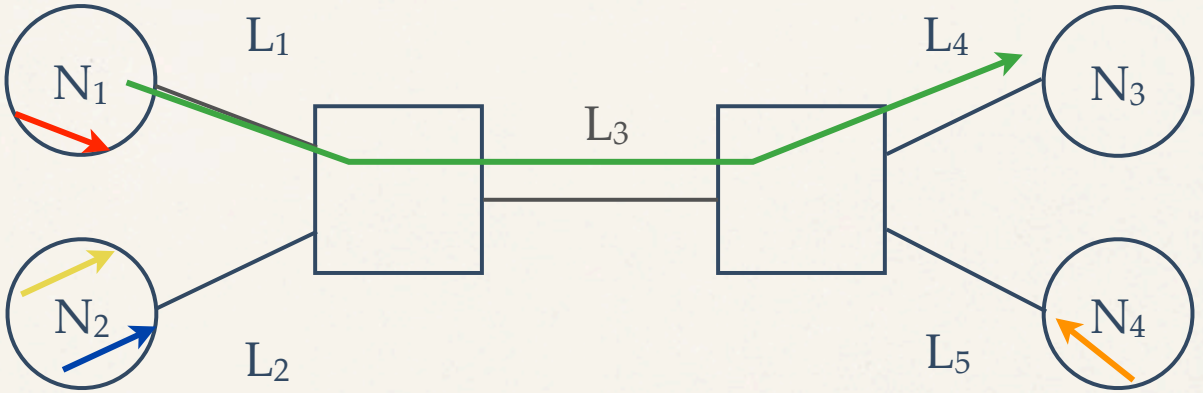
$$d(f_v) = d(f_v, l_4) + d(f_r, null) + 2.\max[d(f_j, null) + d(f_o, null), d(f_b, null)]$$

Calculons le délai du flux vert



$$d(f_v) = d(f_v, l_4) + d(f_r, null) + 2.\max[d(f_j, null) + d(f_o, null), d(f_b, null)]$$

Calculons le délai du flux vert



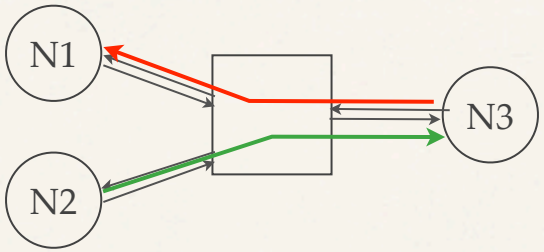
$$d(f_v) = d(f_v, null) + d(f_o, null) + d(f_r, null) + 2.\max[d(f_j, null) + d(f_o, null), d(f_b, null)]$$

Résultat

Fin de la récursion : $d(f_i, null) = \frac{T_i}{C}, \ \forall i$
avec T_i : taille max des paquets du flux i
et C : capacité des liens

$$d(f_v) = \frac{T_v + T_r + T_o + 2.\max(T_j + T_o, T_b)}{C}$$

Modèle du réseau



- réseau = graphe orienté
- flux = chemin dans le graphe d'une source vers une destination et une taille maximale de paquet
- links(f) = liste ordonnées des liens empruntés par f
- Autres notations : first(f), next(f,l), prev(f,l), null

Hypothèses pour le calcul

- routage statique
- tous les liens ont la même capacité C
- tous les flux ont la même priorité
- la source émet ses paquets à la vitesse maximale
- la destination traite chaque paquet dès qu'il arrive
- pour chaque flux f, $T_f \geq$ nb_routeurs * taille_buffer (56 octets)
- les paquets d'un même flux n'interagissent pas entre eux

Délai dans un routeur

Phase 1

Routeur

Source

Phase 2

- Délai pour un paquet p :
  - seuls les paquets sortants par le même port que p ont un impact
  - 1 paquet par autre port d'entrée peut passer avant p (accès équitable)
  - si plusieurs flux entrent par le même port, on prend le délai max
  - chacun de ces paquets peut lui-même être bloqué ensuite
- ⇒ formulation récursive du délai

Délai dans un routeur

$$d(f,l) = \sum_{l_{in} \in B_{f,l}} [\max_{f_{in} \in U_{l_{in}}} d(f_{in}, next(f_{in}, l)) + d_{sw}]$$
$$+ d(f, next(f,l)) + d_{sw}$$

où

$$B_{f,l} = \{l_{in} \in L,\ l_{in} \neq prev(l),$$
$$pour\ lesquels\ \exists f_{in} \in F | next(f_{in}, l_{in}) = l\}$$

($B_{f,l}$ est l'ensemble des liens utilisés immédiatement avant l par un flux f_{in} et qui ne sont pas $prev(f,l)$)

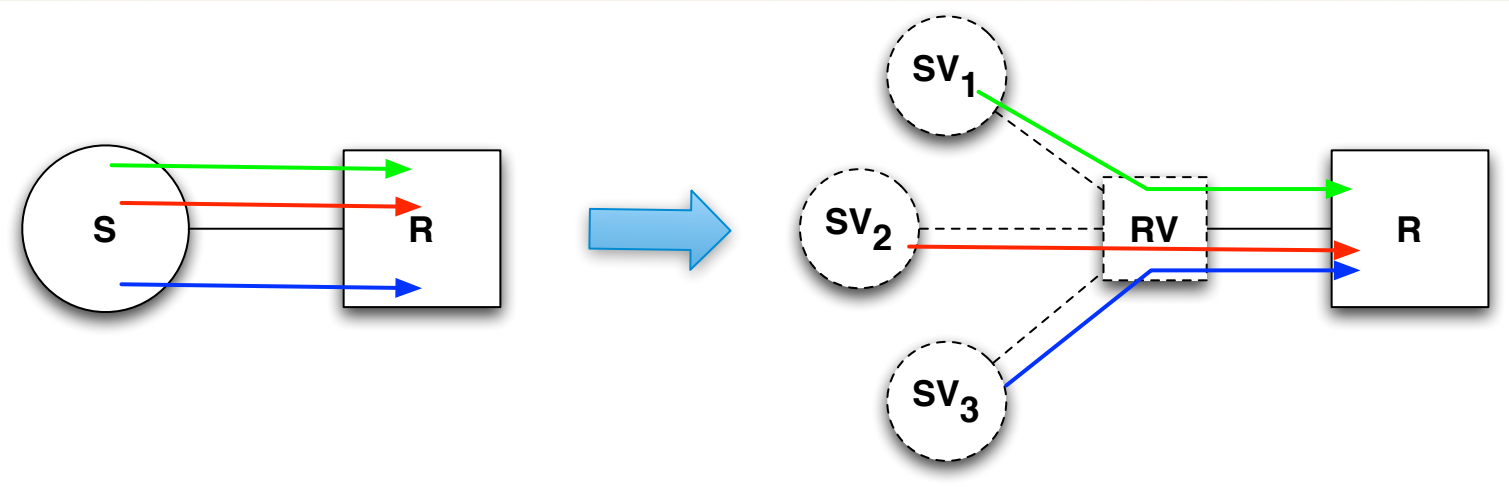
et

$$U_{l_{in}} = \{f_{in} \in F | l_{in} \in links(f_{in})\}$$

($U_{l_{in}}$ est l'ensemble des flux qui utilisent l_{in})

Délai à la source

- causé par les flux de même source que p
- accès au lien de sortie équitable : au pire, un paquet de chaque flux peut passer avant p
- Modèle : routeur virtuel (ramène au cas précédent)



Délai pendant la phase 2

- connexion équivalente à un lien point-à-point entre la source et la destination
- Par hypothèse, la destination ne retarde pas le paquet

$$d(f, null) = \frac{T_f}{C}$$

Bilan

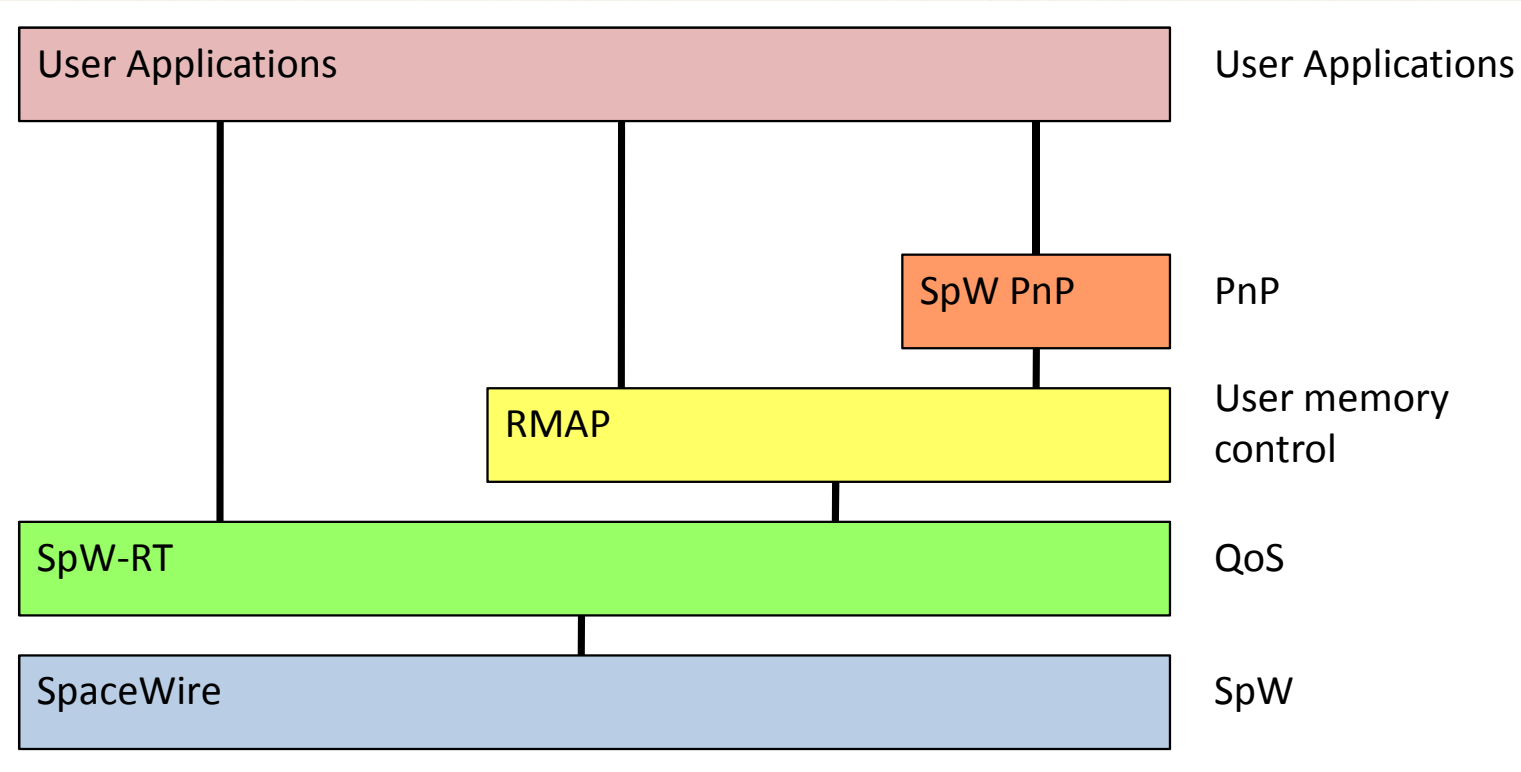
- méthode simple permettant de calculer une borne supérieure sur le délai pire-cas d'un flux dans un réseau SpaceWire
- borne déterministe, pas probabiliste
- fonctionne sur un réseau de topologie quelconque, à condition qu'il n'y ait pas d'interblocages ni d'**auto-dépendance**

2ème solution

- ajouter une couche supplémentaire entre le réseau et les applications
- assure une certaine qualité de service aux applications
- Quelle QoS est nécessaire ? Avec quel coût ?

SpaceWire-RT

- RT pour Reliabilty & Timeliness
- standard en cours de rédaction (draft)
- Surcouche de SpaceWire garantissant une qualité de service aux applications



Aperçu

- But : fournir des communications (a)synchrones, éventuellement fiable
- pas de modifications de SpaceWire
- Doit avoir une interface applicative similaire à celle de SpaceWire
- ⇒ modèle de communication identique :
  liens point-à-point entre applications
- ici les liens sont virtuels : canaux virtuels

Deux modes

- Asynchrone : pas de garanties temporelles strictes, priorités entre les CV pour gérer l'accès au réseau
- Synchrone : mécanisme TDMA (Time Division Multiple Access) pour gérer l'accès au réseau, délais de transmission garantis

Classes de trafic prévues

	Asynchrone	Synchrone
Non Fiable	Best Effort	Reserved
Fiable	Assured	Guaranteed

La fiabilité en question

- 3 mécanismes prévus :
  - Acquittements
  - Réémission automatique
  - Redondance
- Problèmes :
  - pas utile dans tous les cas
  - augmente le coût des équipements
  - augmente la durée nécessaire à la standardisation

Etat actuel du draft

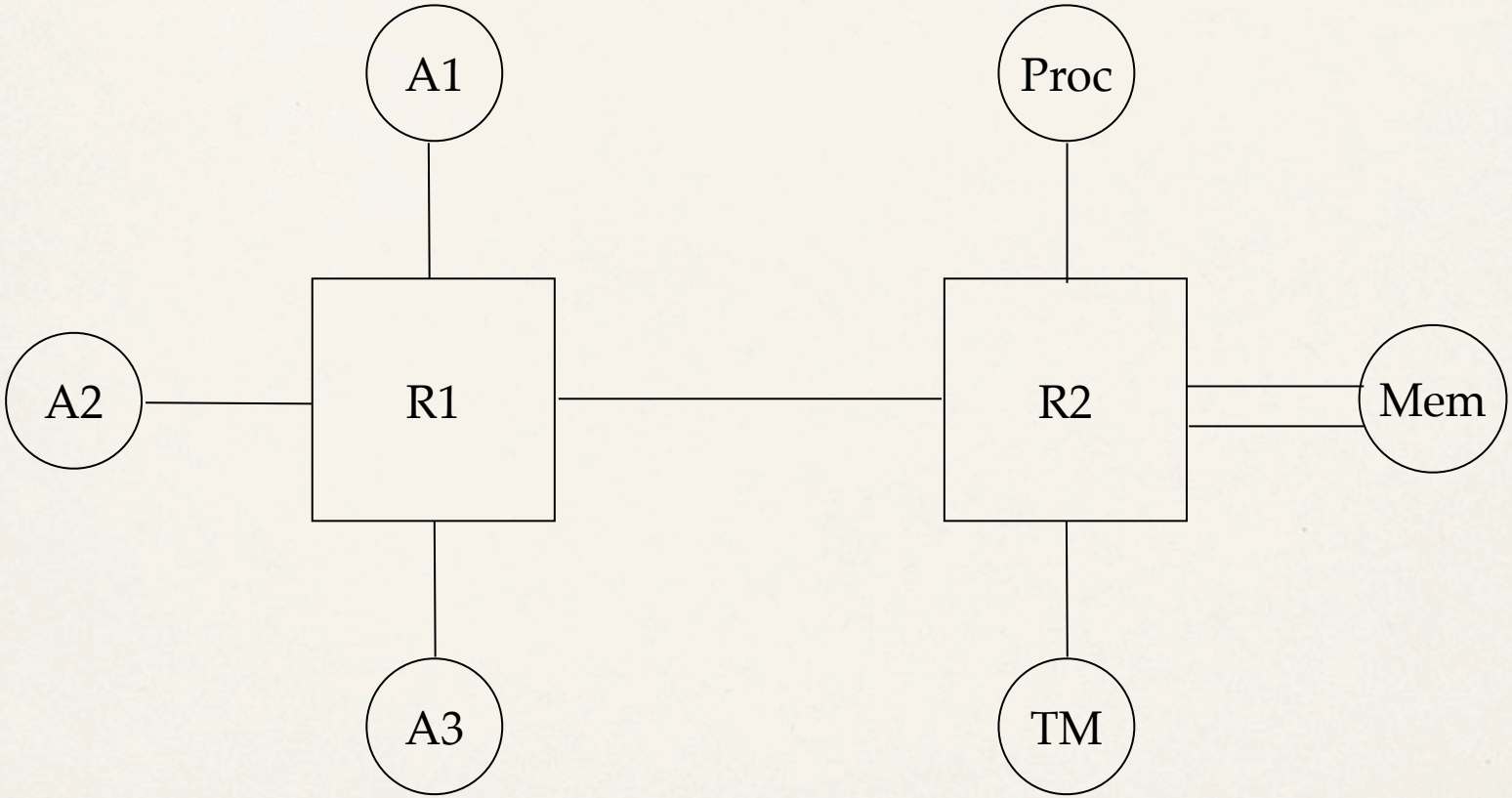
	Asynchrone	Synchrone
Non Fiable	Best Effort	Reserved
Fiable	Assured	Guaranteed

SpaceWire-RT → SpaceWire-T

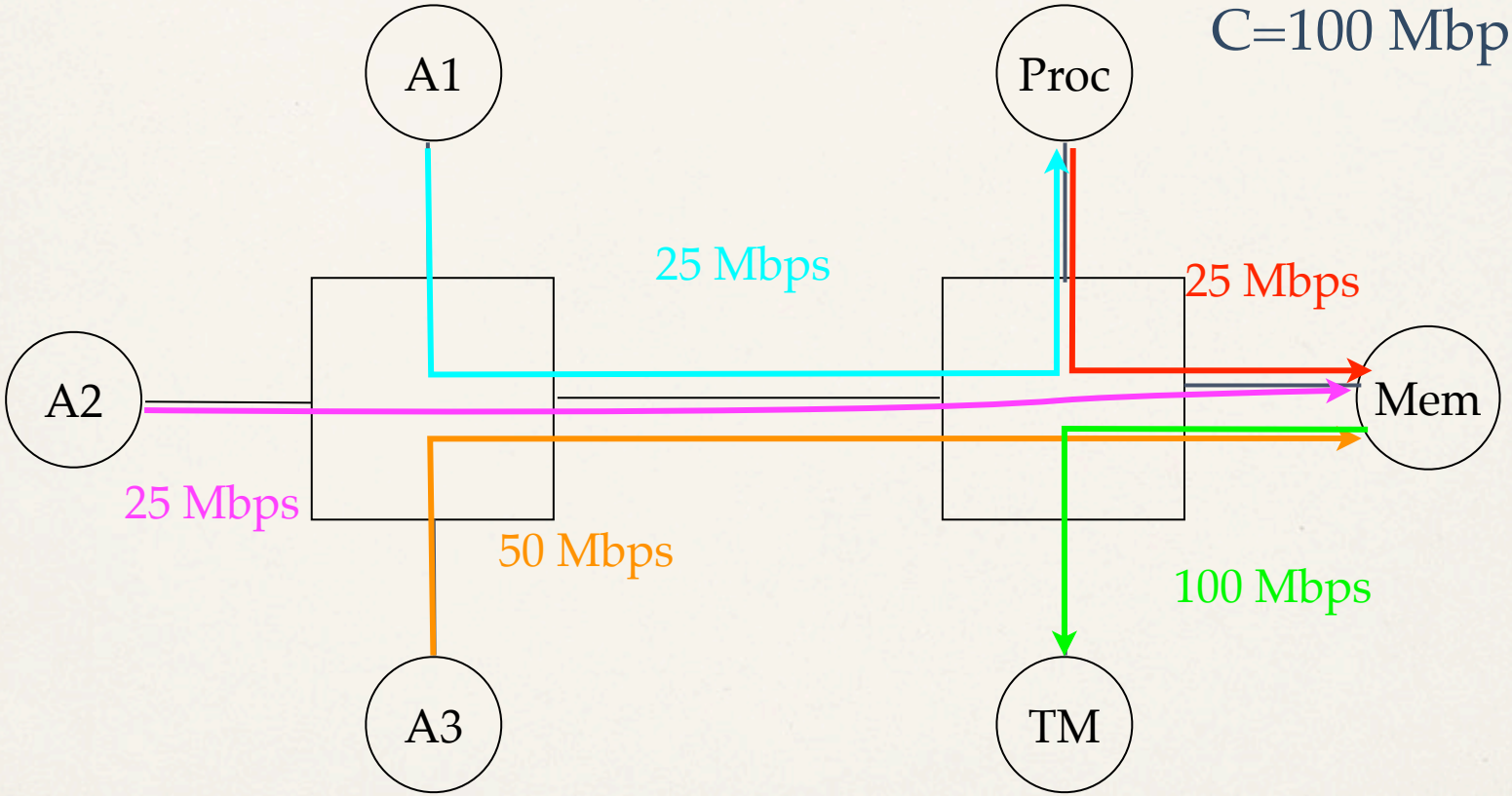
Canaux virtuels

- canal = buffer source + liens + buffer destination
- identifiant : adr source et destination + n°
- Fonctionnement :
  - appli écrit dans le buffer source
  - SpW-T transfère les données *via* SpaceWire
  - appli distante récupère les données sous leur forme initiale

Exemple de réseau



Définition des canaux



+ Proc vers tous les Ai
+ TM vers tous les Ai

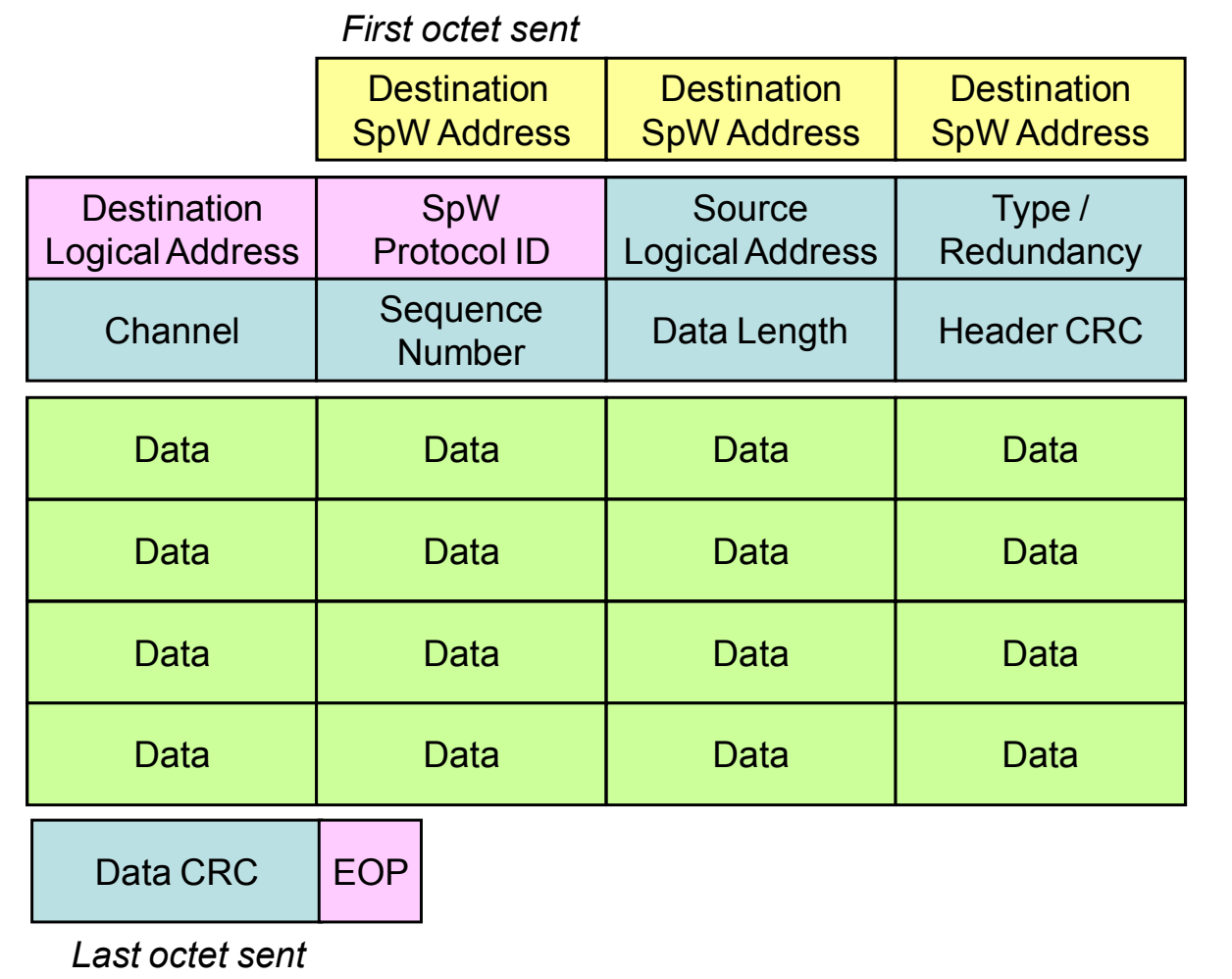
Mode asynchrone

- Délais de communication non bornés
- données délivrées dans l'ordre, sans duplication
- Gestion du trafic :
  - segmentation des données
  - priorités entre canaux
  - contrôle de flux de bout en bout
  - acquittements (optionnels)

Segmentation des données

- Appli émettrice écrit des données de taille quelconque dans le buffer source
- Découpage en fragments de 256 octets (User Data Segment), transmis dans des paquets SpW distincts (Data Packet)
- Reconstitution de façon transparente sur le nœud distant

Format des DP



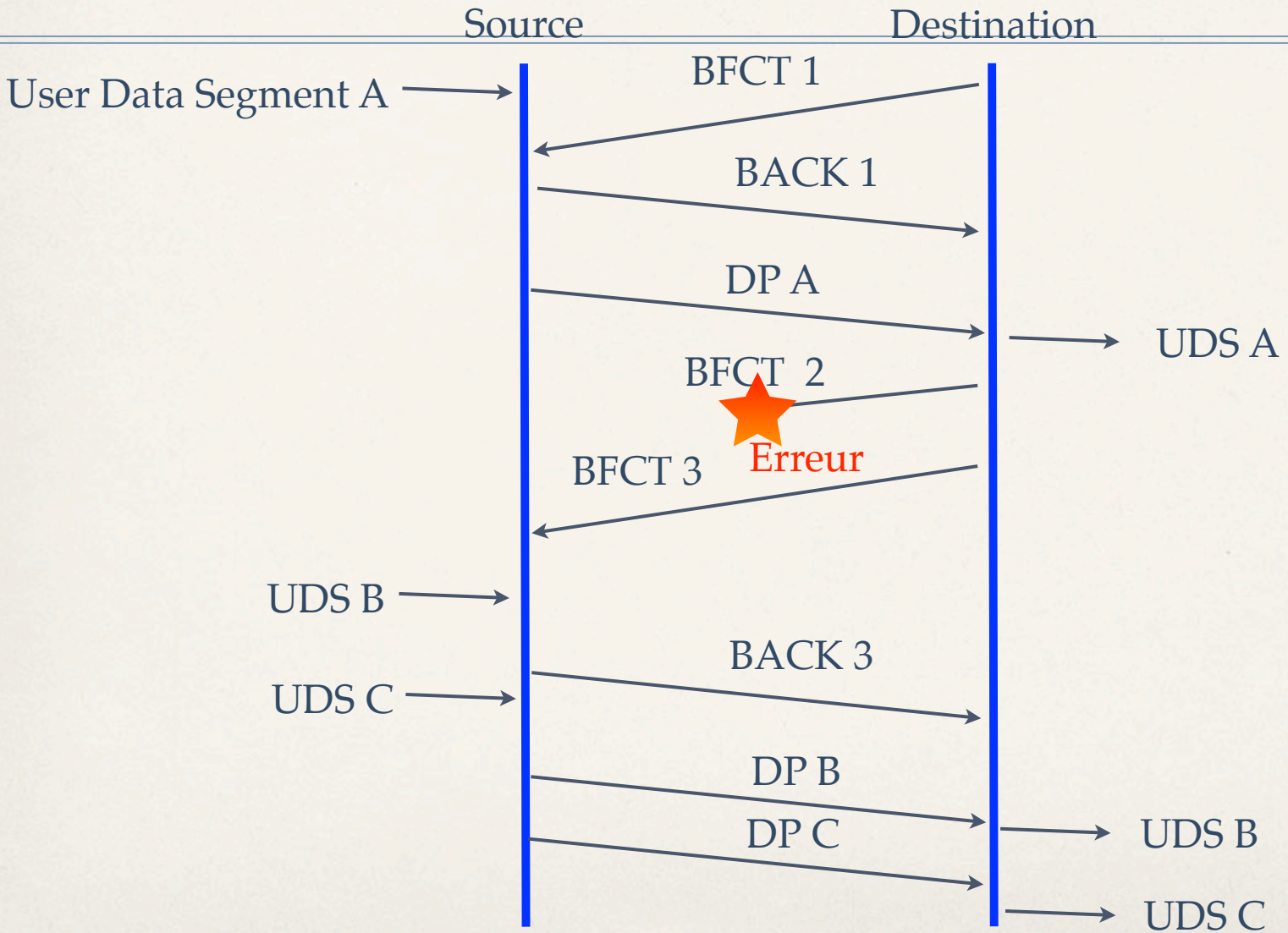
Priorités entre les canaux

- Indépendants des priorités entre adresses logiques (au niveau des routeurs)
- priorités statiques sur les numéros des canaux
- appliquées à la source
- plus le numéro d'un canal est faible, plus ce canal est prioritaire
- non préemptives

Contrôle de flux de bout en bout (1)

- but : éviter qu'un DP reste bloqué à l'entrée d'un nœud
- mécanisme similaire à celui des liens
- destination envoie des jetons BFCT (Buffer Flow Control Token) à la source pour l'autoriser à émettre des DP
- BFCT doivent être acquittés par des BACK sinon ils sont réémis
- possibilité d'émettre plusieurs BFCT d'avance

Contrôle de flux (2)



Acquittements

- choix canal par canal
- si ACK activés :
  - source déclenche un timer à l'émission de chaque UDS
  - la destination émet un ACK lorsqu'elle reçoit un UDS
  - si tous les UDS d'un bloc de données sont délivrés, l'appli est alertée du succès
  - si au moins un des UDS n'est pas délivré, l'appli est avertie de l'échec et les UDS restants sont jetés

Bilan sur la classe Best Effort

- Ne fournit pas de garanties strictes
- Mais segmentation + priorités permettent aux messages urgents de doubler les paquets longs, non-urgents
- Contrôle de flux réduit aussi les risques de congestion du réseau

Mode synchrone

- Communications synchrones, non fiables
- Temps de transfert des données déterministe (sauf erreurs de transmission)
- Mécanisme TDMA pour le contrôle d'accès au réseau

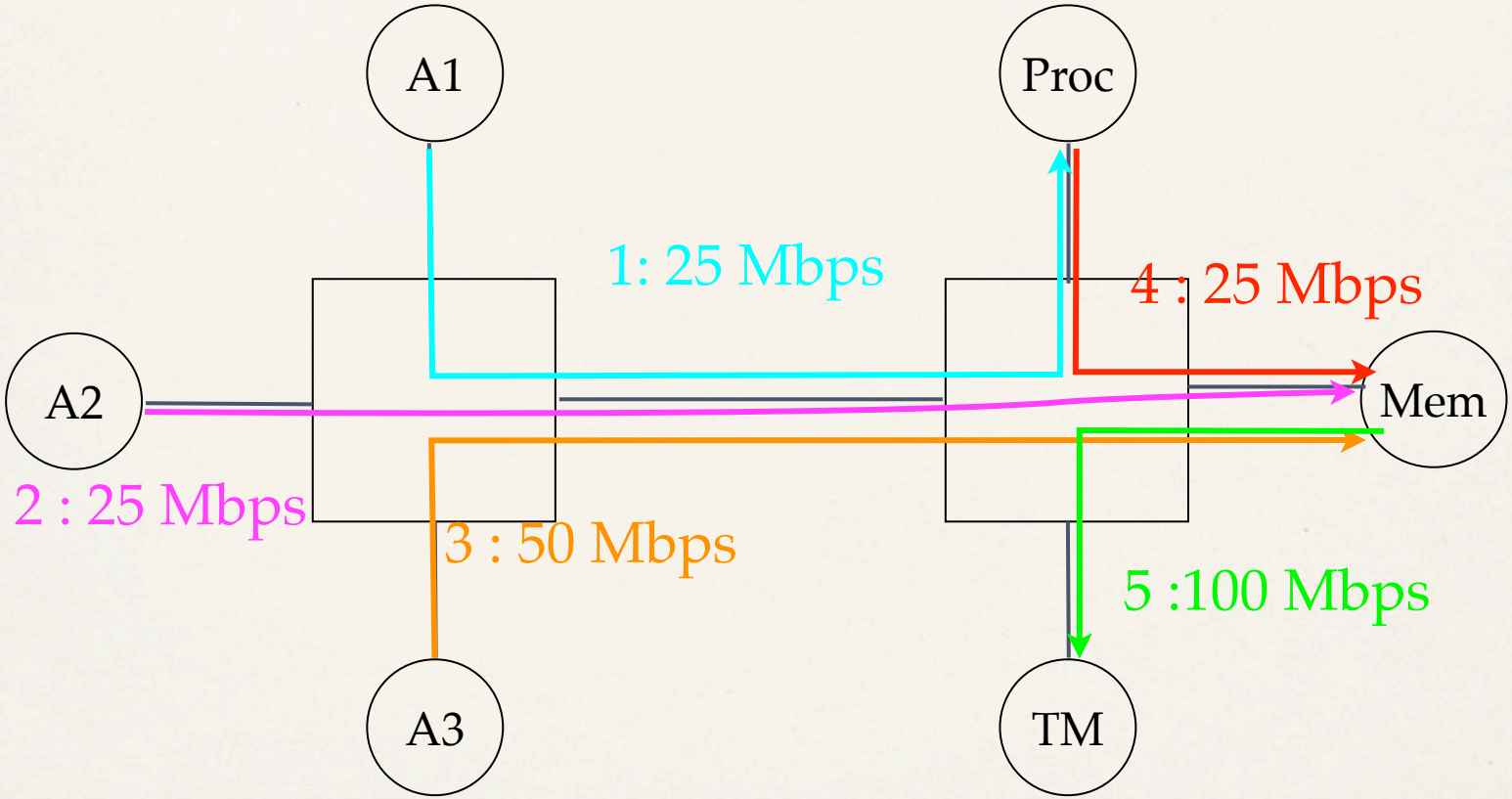
TDMA

- Time Division Multiple Access
- Ordonnancement des DP pour garantir les délais de transmission
- Bande passante divisée en slots temporels de durée constante
- Utilisation des Time-Codes SpaceWire :
  - cycles de 64 slots
  - synchronisation globale

TDMA

- Slots alloués statiquement aux canaux
- Contrainte : aucun lien ne peut être partagé par plusieurs canaux durant un slot
- Chaque nœud a sa table d'allocation
- Les routeurs ne participent pas au contrôle

Définition des canaux (rappel)



+ 6 : Proc vers tous les Ai

+ 7 : TM vers tous les Ai

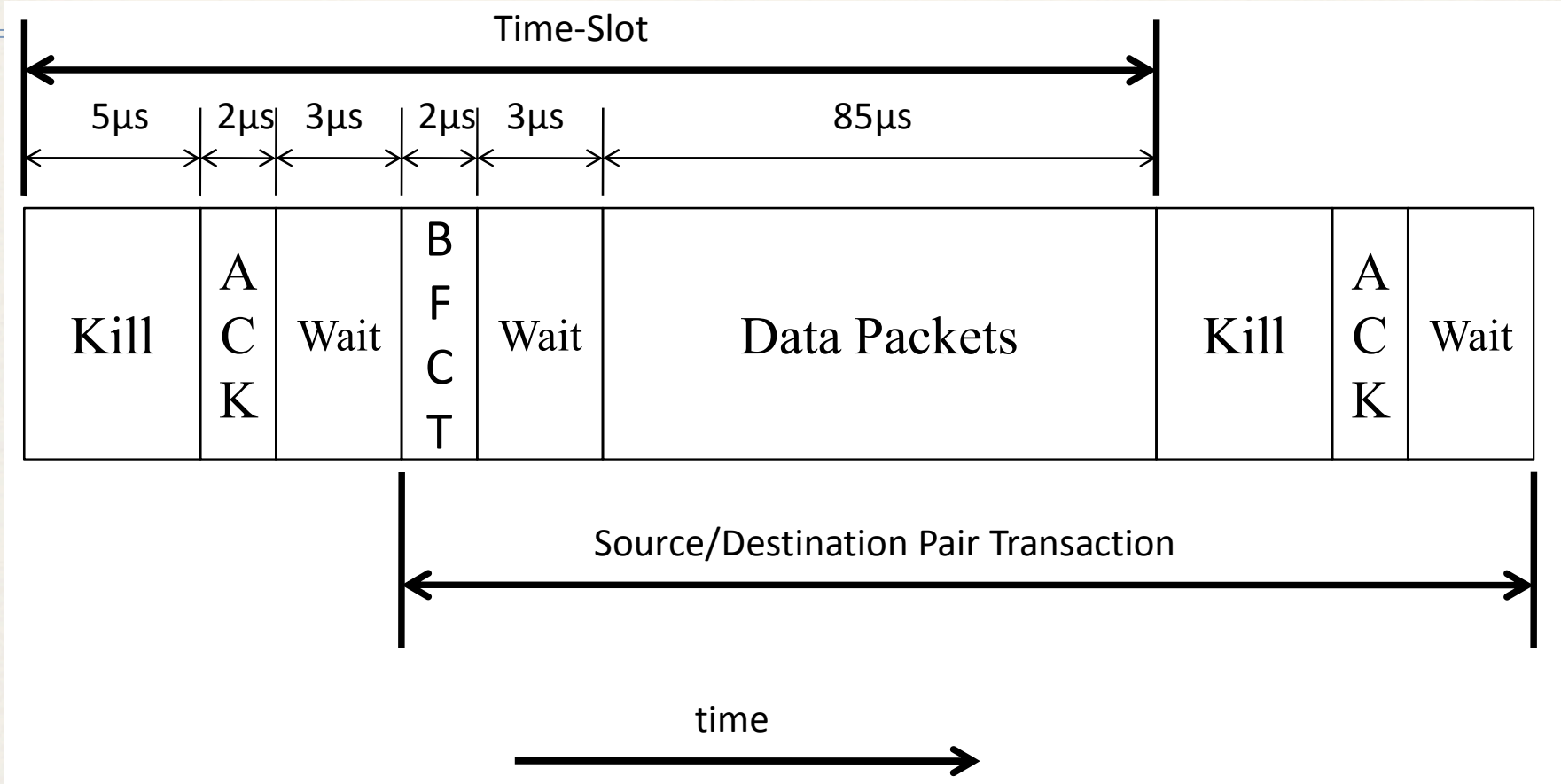
Allocation des slots

	Slot 0	Slot 1	Slot 2	Slot 4
Canal 1	X			
Canal 2		X		
Canal 3			X	X
Canal 4	X			
Canal 5	X	X	X	X
Canal 6		X		X
Canal 7	X		X	

Contrôle de flux synchrone

- les BFCT ne peuvent pas circuler librement
- regroupés en début de slot
- 1 seul BFCT par paire (source, destination) pour tous les canaux
- pas d'acquittements (émission régulière)

Structure d'un slot



- Durée recommandée : 100 µs (=> cycle: 6,4 ms)
- Permet d'émettre 6 DP par slot

Bilan sur le mode synchrone

- réservation de la bande passante
- séparation stricte des types de trafics
- oblige à synchroniser tout le réseau
- alloue trop de bande passante aux canaux prioritaires

Bilan sur SpaceWire-T

- fragmentation + priorités ⇒ paquets urgents peuvent doubler paquets longs
- mode synchrone permet de réserver la bp pour le trafic urgent

Bilan sur SpaceWire-T

- importante taille des buffers (très coûteux)
- contrôle de flux peut augmenter les délais de livraison
- pour l'instant, impossible de modifier l'allocation des slots
- interface applicative non compatible avec celle de SpaceWire ?

Questions ?