Placement des tâches temps-réel dur sur des multi-coeurs en réseau-sur-puce (NoC)

Chawki Benchehida

Directeurs :

Pr. Giuseppe Lipari

Pr. Kamel Benhaoua

Encadrant :

Dr. Houssam Zahaf

09 Novembre 2021

مخبر الهياكل المتوازية، المحمولة والحساب المكثف
LAPECI
Laboratoire des Architectures Parallèles, Embarquées et du Calcul Intensif.

CRIStAL
Centre de Recherche en Informatique, Signal et Automatique de Lille

جامعة وهران 1
Université d'Oran 1 أحمد بن بلة

Université de Lille

Plan

1 Contexte de l’étude

2 Étude comparative des stratégies d’arbitrage dans les routeurs

3 Transformation et analyse du modèle de tâche en DAG sur NoC

4 Placement efficace de tâches sur NoC

5 Conclusion et perspectives

Plan

1 Contexte de l’étude

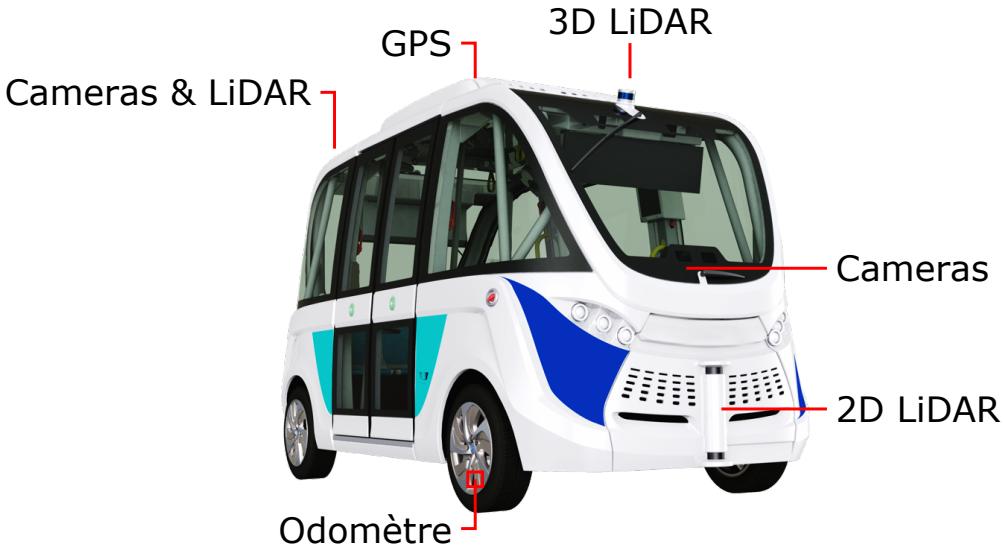
2 Étude comparative des stratégies d’arbitrage dans les routeurs

3 Transformation et analyse du modèle de tâche en DAG sur NoC

4 Placement efficace de tâches sur NoC

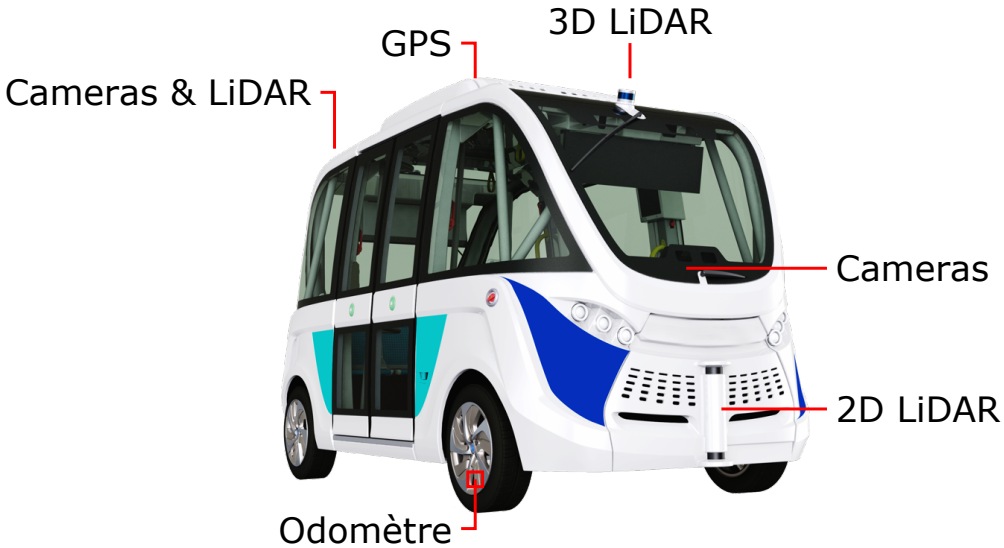
5 Conclusion et perspectives

Systèmes Cyber-Physiques Modernes



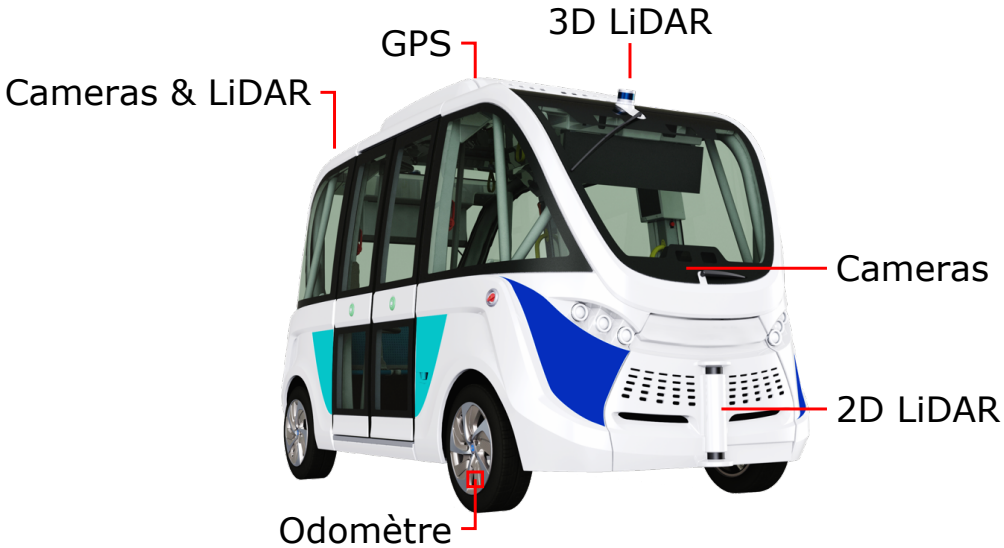
- Traiter de grandes quantités de données en continue

Systèmes Cyber-Physiques Modernes



- Traiter de grandes quantités de données en continue
- Gérer des systèmes critiques

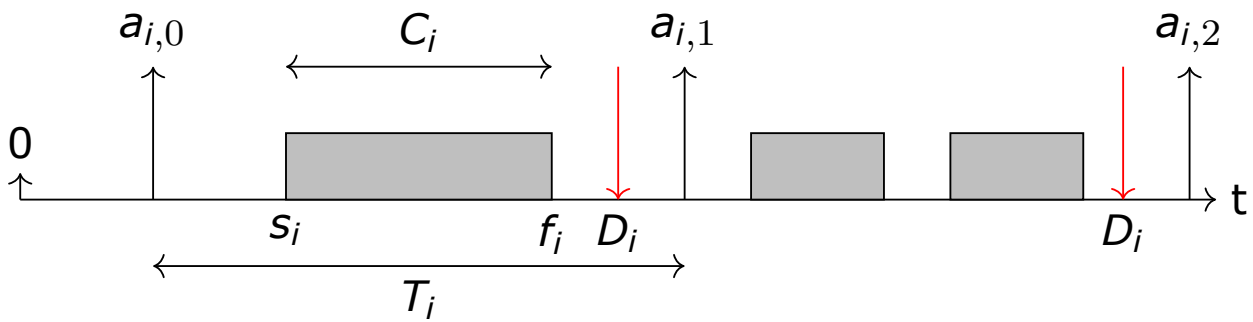
Systèmes Cyber-Physiques Modernes



- Traiter de grandes quantités de données en continue
- Gérer des systèmes critiques
- Problème :
  - Contraintes temps-réel

Systèmes temps-réel : introduction

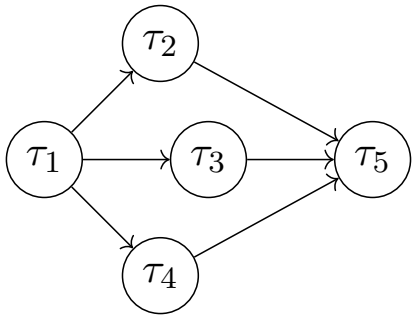
- Les résultats doivent être corrects logiquement et délivrés à temps.
  - Délivrés à temps $\neq$ rapidement
- Les tâches temps réels sont récurrentes (périodiques, sporadiques)



Le modèle de Liu and Layland

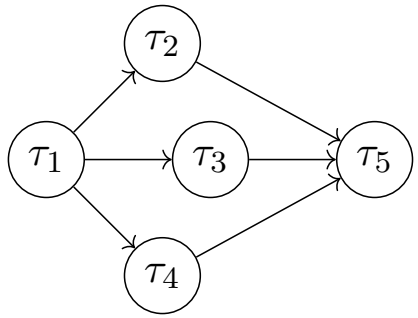
Besoin de paralléliser

- Les systèmes temps-réel modernes sont parallélisés
  - Exemple du modèle Fork-join
  - 1 tâche parallélisée en **5 sous-tâches**

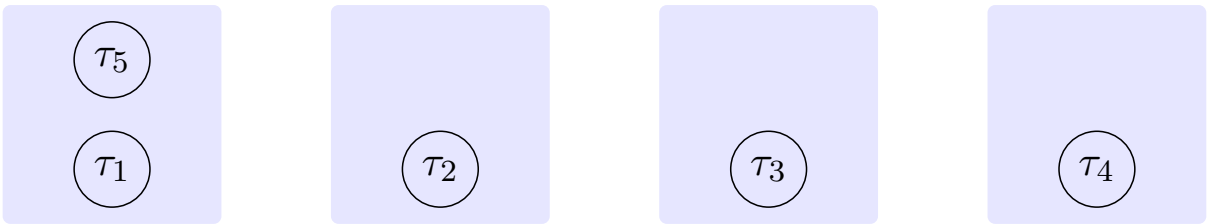


Besoin de paralléliser

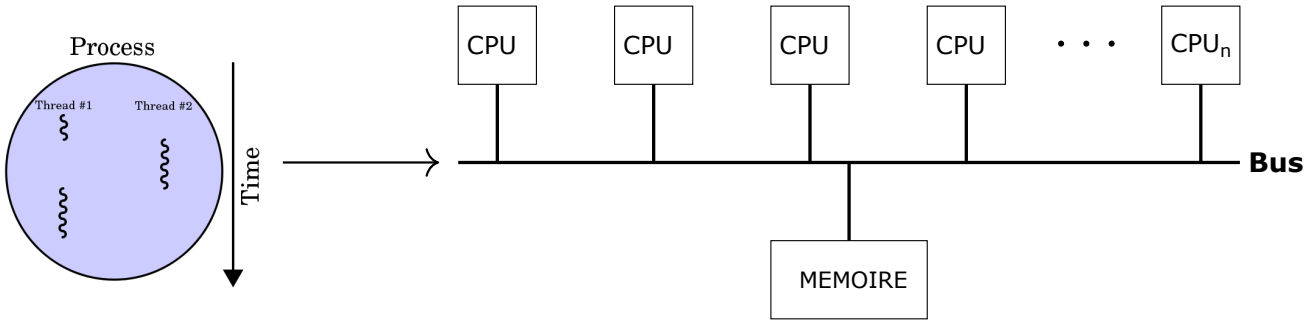
- Les systèmes temps-réel modernes sont parallélisés
  - Exemple du modèle Fork-join
  - 1 tâche parallélisée en **5 sous-tâches**



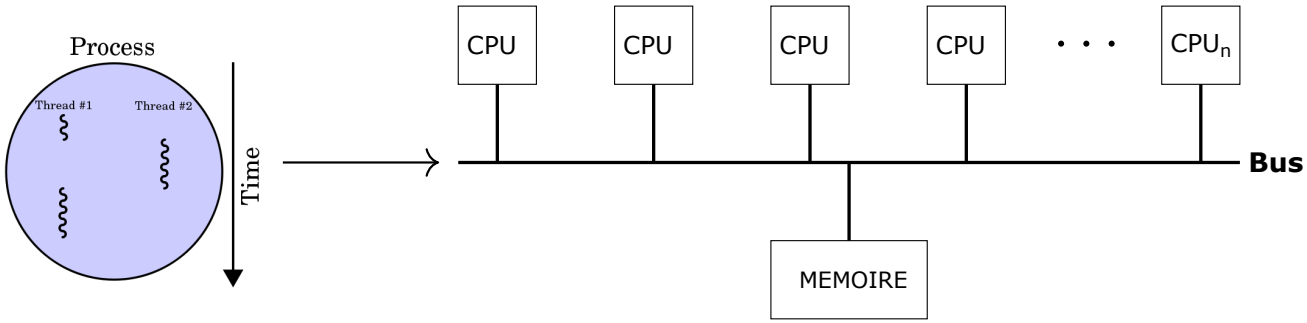
- Une exécution sur multiprocesseur à **4 coeurs**



Les multiprocesseurs actuels



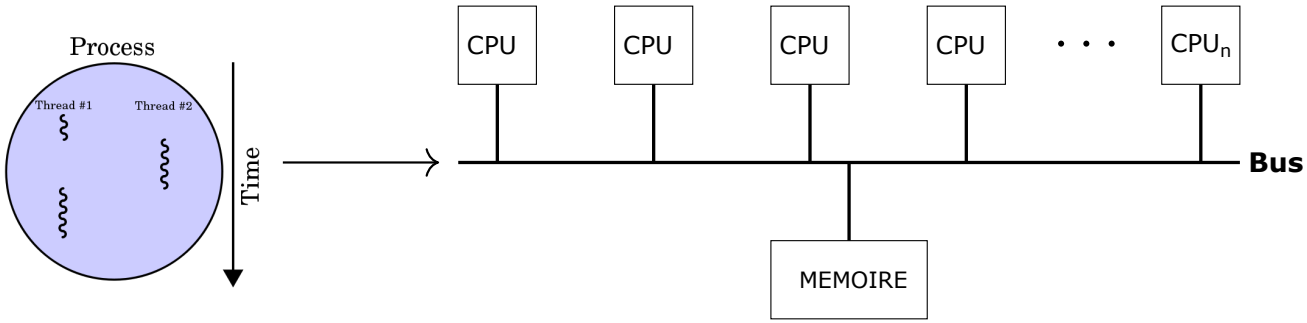
Les multiprocesseurs actuels



Limitations

- Scalabilité

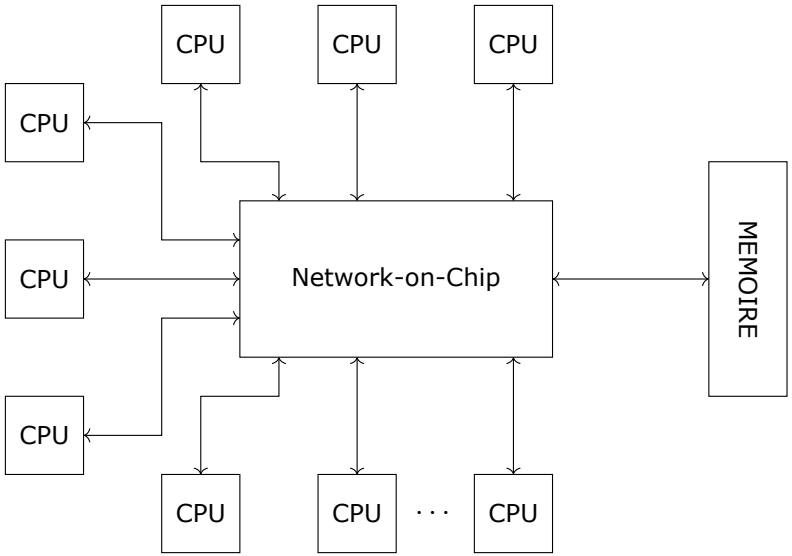
Les multiprocesseurs actuels



Limitations

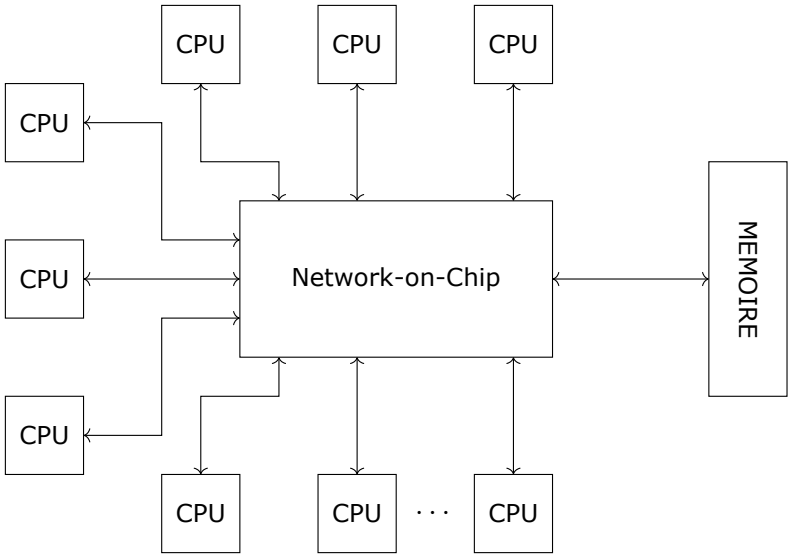
- Scalabilité
- Congestion du bus
  - Communication CPU-CPU / CPU-Mémoire
  - Latence de communication élevée

Alternative : Réseau-sur-puce (NoC)



- Diversification des chemins d'interconnexion (Path diversity)
- Scalabilité de l'architecture (jusqu'à 1024 coeurs)

Alternative : Réseau-sur-puce (NoC)

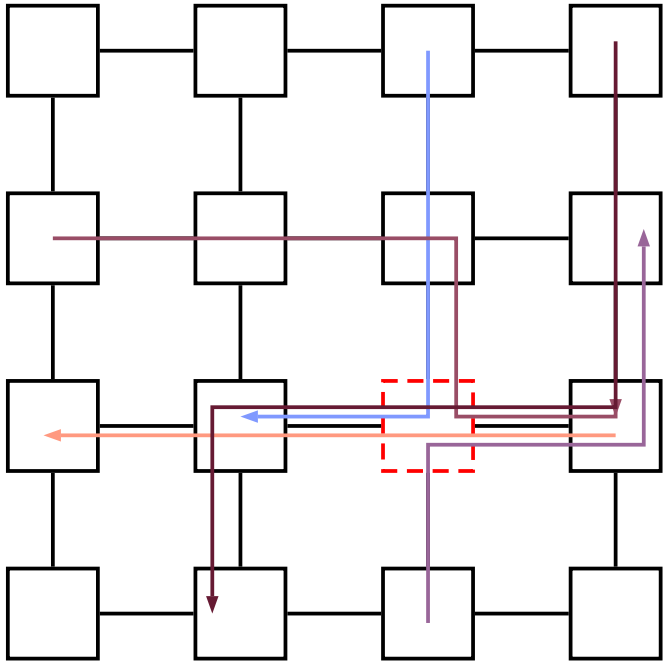


- Diversification des chemins d'interconnexion (Path diversity)
- Scalabilité de l'architecture (jusqu'à 1024 coeurs)

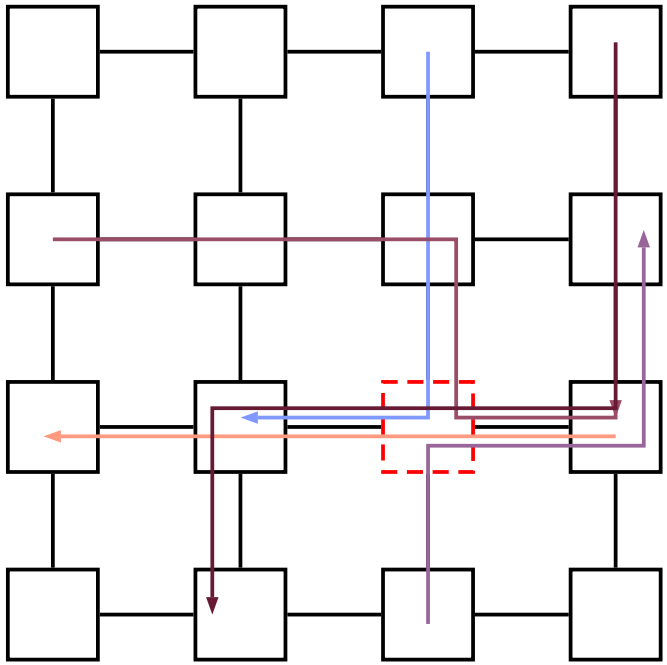
Cependant, certains **problèmes** sont à considérer :

- Complexité du réseau : Routage, gestion des ressources, etc.
- Latence de communication : Dépend de plusieurs paramètres et de l'emplacement des tâches

Communications dans les NoC



Communications dans les NoC



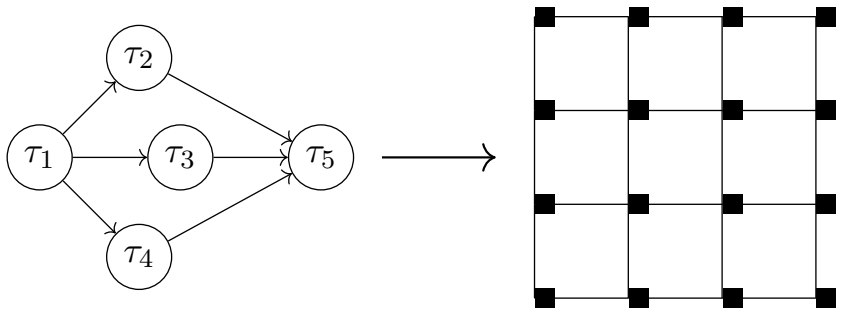
Question 1

Quelle est la bonne stratégie à adopter pour garantir des latences de communication prédictibles ?

Analyse des systèmes temps-réel sur NoC

L'ordonnancement du système dépend :

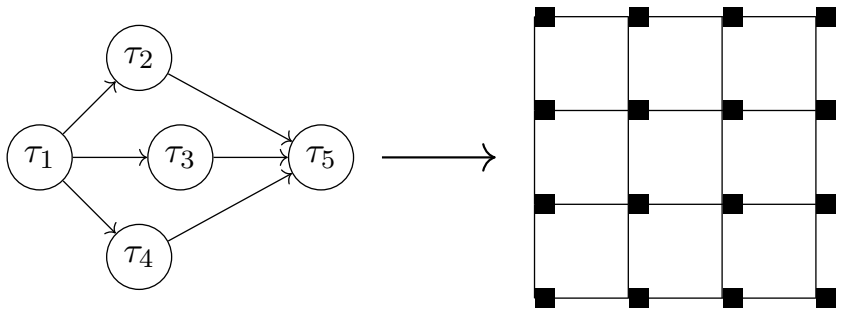
- Caractéristiques des tâches
- État des liens entre les routeurs
- Longueur des communications



Analyse des systèmes temps-réel sur NoC

L'ordonnancement du système dépend :

- Caractéristiques des tâches
- État des liens entre les routeurs
- Longueur des communications

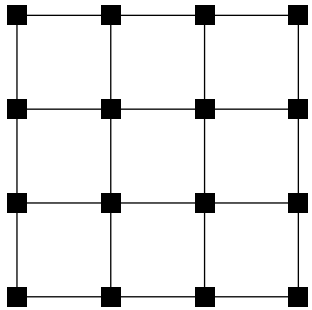


Question 2

Comment procéder à une analyse d'ordonnançablité des tâches devant la complexité du NoC ?

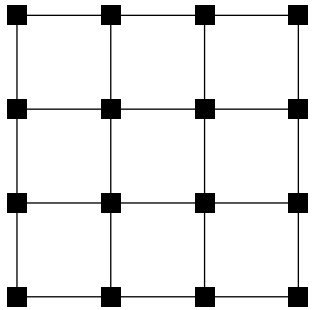
Allocation de tâches sur NoC

Le problème d’allocation de tâches est hautement combinatoire. Il est défini comme **NP-complet**



Allocation de tâches sur NoC

Le problème d'allocation de tâches est hautement combinatoire. Il est défini comme **NP-complet**



Question 3

Comment explorer efficacement l'espace des solutions sans pour autant aller vers une explosion combinatoire ?

Contributions

Q1) Comment garantir des latences de communication prédictibles ?

→ Étude comparative des stratégies d'arbitrage dans les routeurs

Q2) Comment procéder à une analyse des tâches temps-réel sur NoC ?

→ Transformation et analyse du modèle de tâche en DAG sur NoC

Q3) Comment explorer l'espace des solutions (allocations) ?

→ Placement efficace de tâches sur NoC

Plan

1 Contexte de l'étude

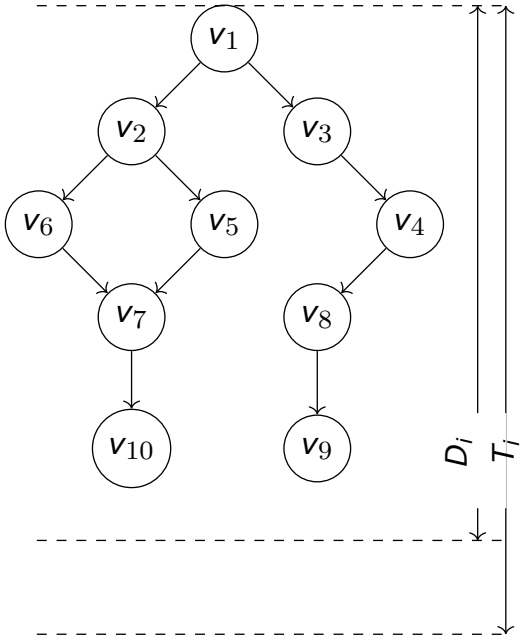
2 Étude comparative des stratégies d'arbitrage dans les routeurs

3 Transformation et analyse du modèle de tâche en DAG sur NoC

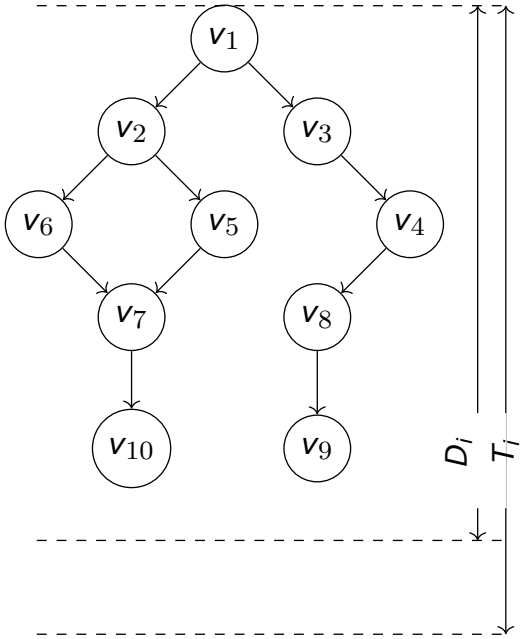
4 Placement efficace de tâches sur NoC

5 Conclusion et perspectives

Modèle de tâches



Modèle de tâches



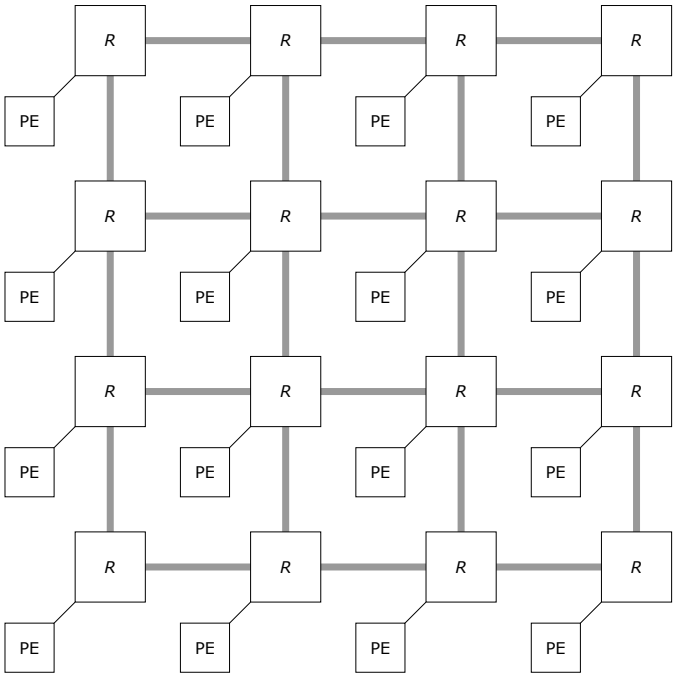
$\rightarrow$ Au niveau spécification, sont données :

- Les caractéristiques d'une tâche $\tau = \{\mathsf{T}, \mathsf{D}, \mathcal{V}, \mathcal{E}\}$
  - Sous-tâches $\mathcal{V} = \{v_1, \cdots, v_k\}$
  - Communications $\mathcal{E} = \{e(n_i, n_j), \cdots\}$
  - Échéance et période de bout-en-bout

Gestion des communications dans les NoC

Dans un NoC à :

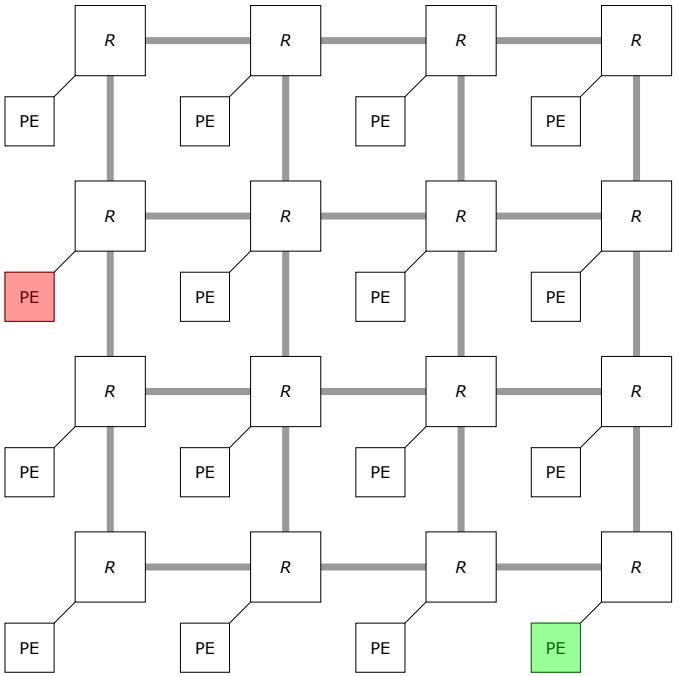
- Topologie en Mesh-2D
- Routage déterministe



Gestion des communications dans les NoC

Dans un NoC à :

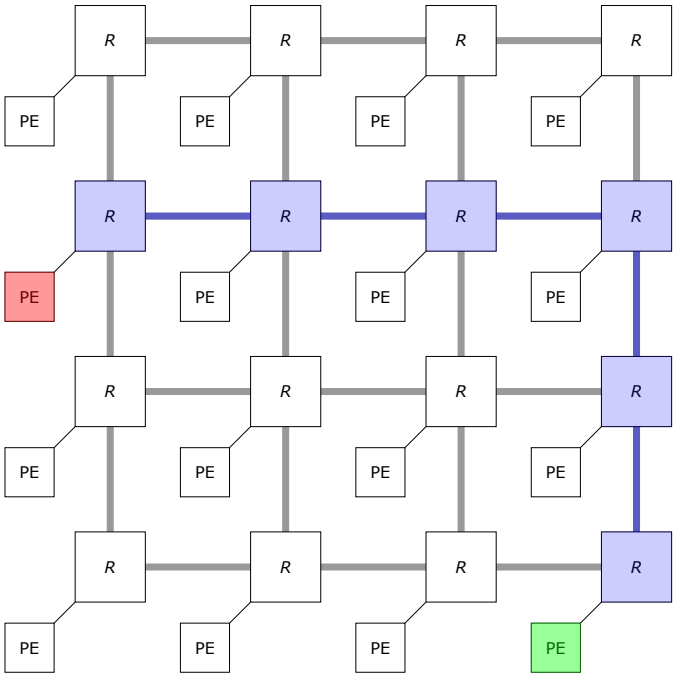
- Topologie en Mesh-2D
- Routage déterministe



Gestion des communications dans les NoC

Dans un NoC à :

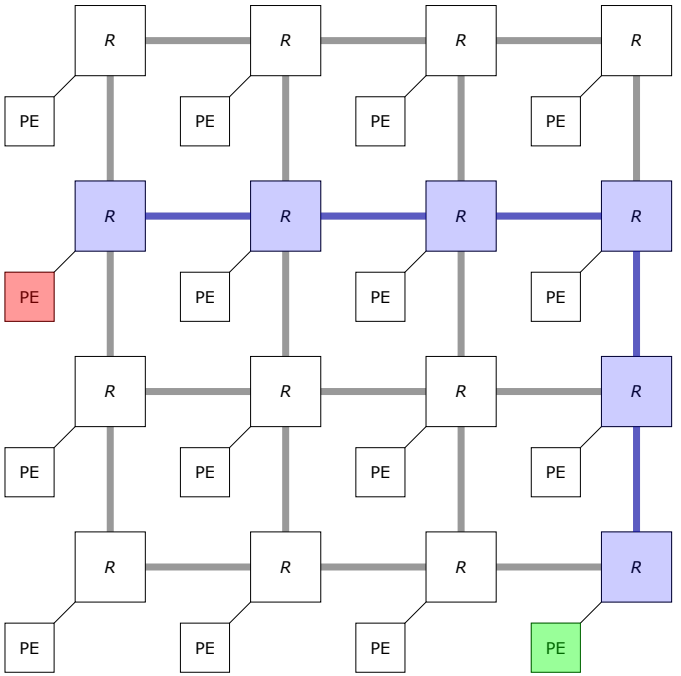
- Topologie en Mesh-2D
- Routage déterministe



Gestion des communications dans les NoC

Dans un NoC à :

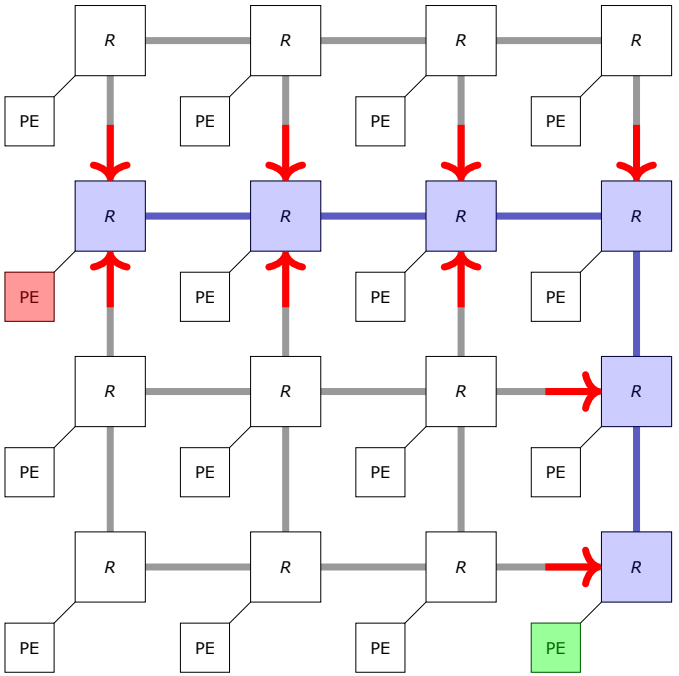
- Topologie en Mesh-2D
- Routage déterministe



Gestion des communications dans les NoC

Dans un NoC à :

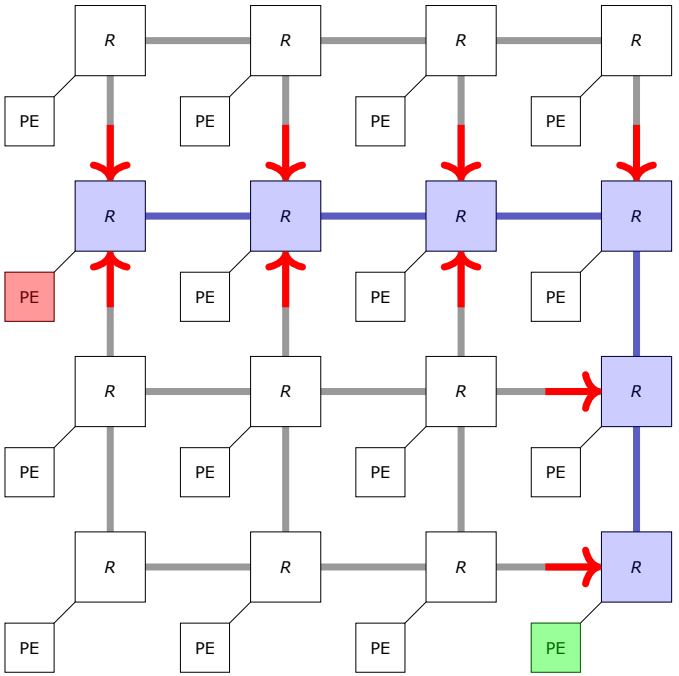
- Topologie en Mesh-2D
- Routage déterministe



Gestion des communications dans les NoC

Dans un NoC à :

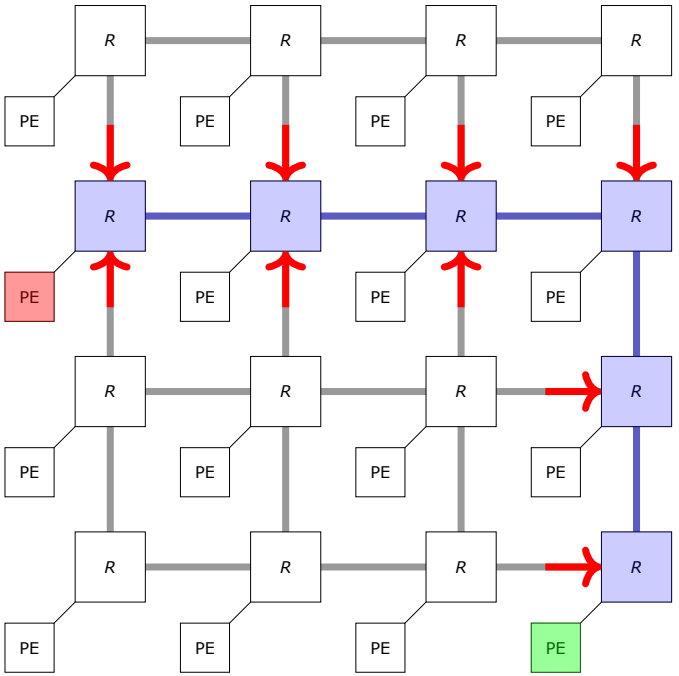
- Topologie en Mesh-2D
- Routage déterministe



Gestion des communications dans les NoC

Dans un NoC à :

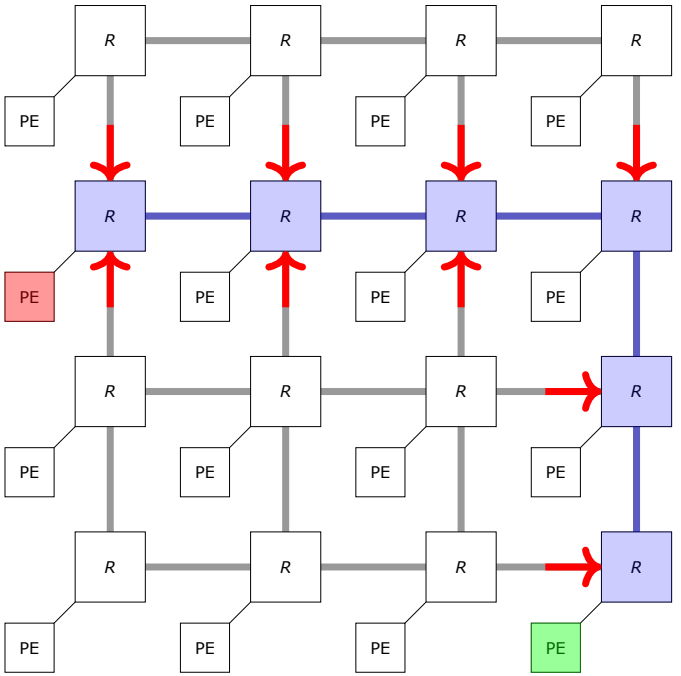
- Topologie en Mesh-2D
- Routage déterministe



Gestion des communications dans les NoC

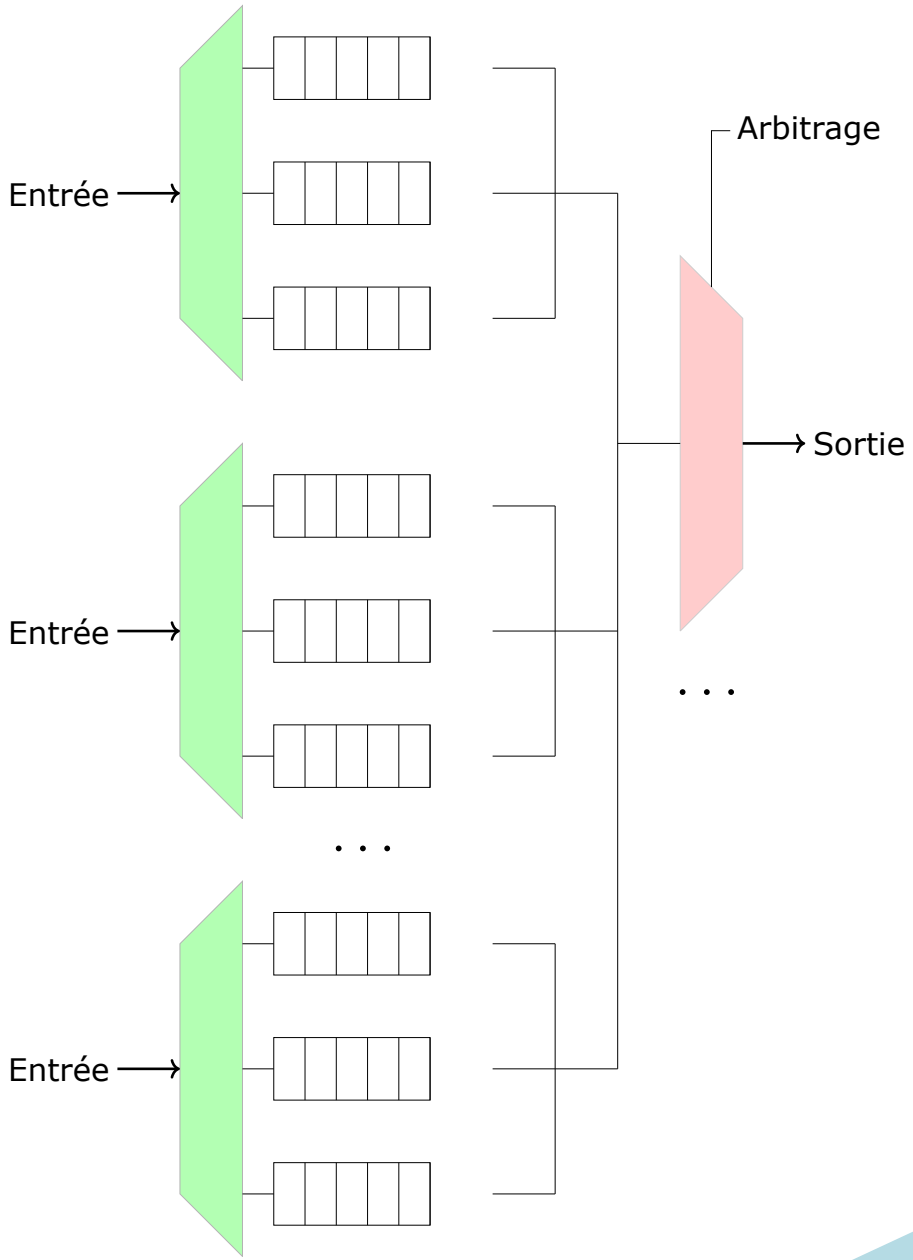
Dans un NoC à :

- Topologie en Mesh-2D
- Routage déterministe



- Déterminer une politique de gestion des communications adaptées au temps-réel

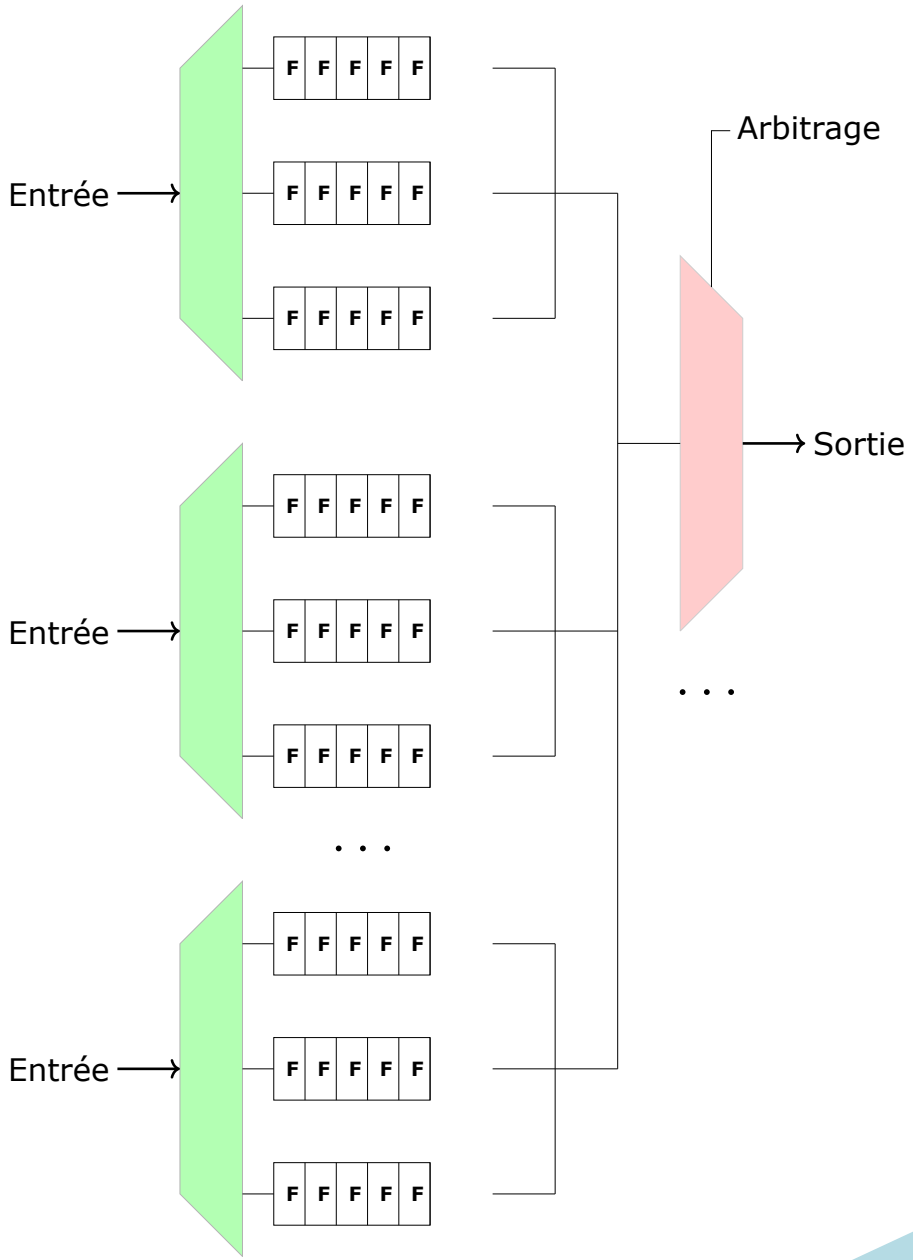
Arbitrage par priorité VS TDMA



Arbitrage par priorité VS TDMA

Wormhole Switching

- Message $\rightarrow$ Paquet $\rightarrow$ Flit



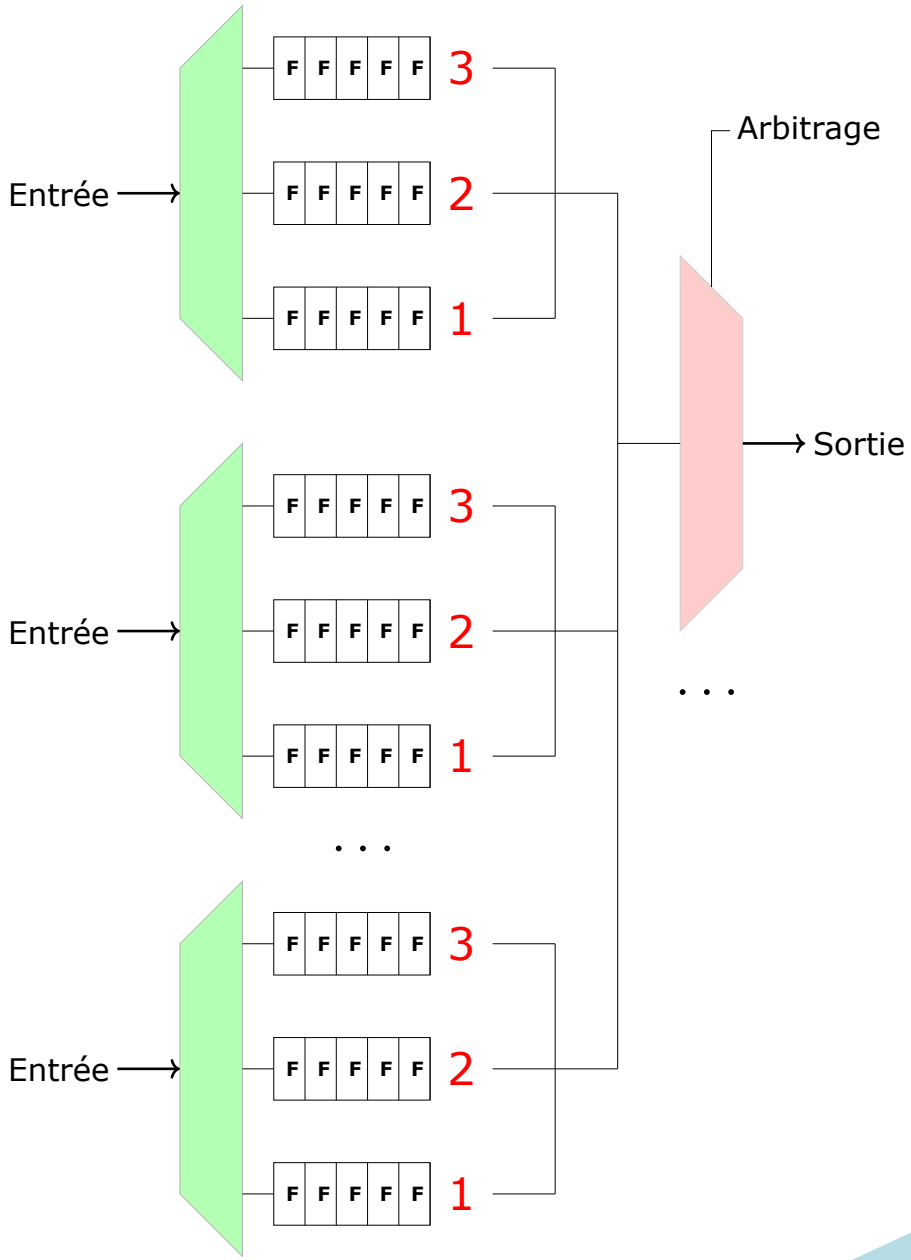
Arbitrage par priorité VS TDMA

Wormhole Switching

- Message $\longrightarrow$ Paquet $\longrightarrow$ Flit

Stratégie 1 :

- VCs par priorités
- Arbitrage par échéance



Arbitrage par priorité VS TDMA

Wormhole Switching

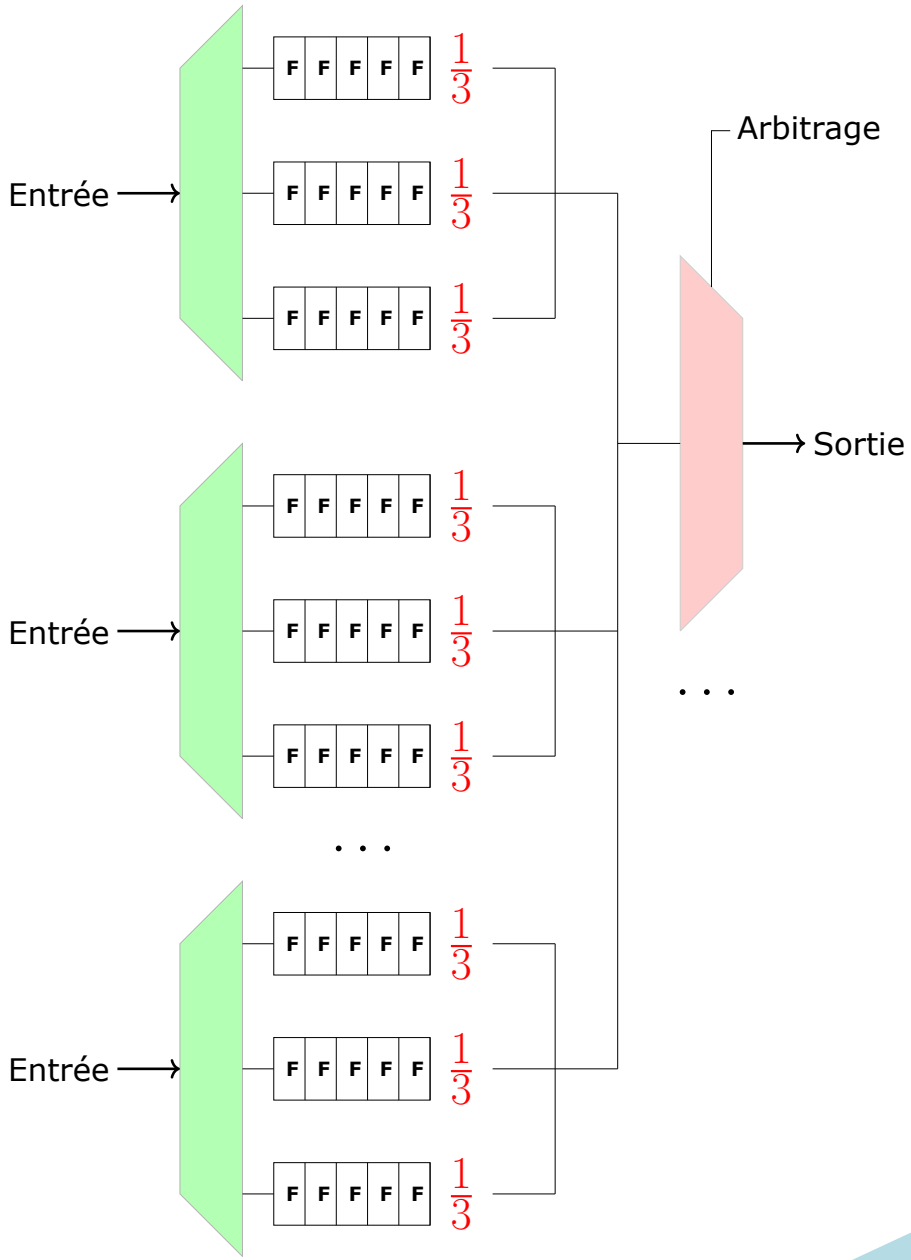
- Message $\rightarrow$ Paquet $\rightarrow$ Flit

Stratégie 1 :

- VCs par priorités
- Arbitrage par échéance

Stratégie 2 :

- Ordonnancement TDMA (Time-division Multiple Access)
- Attribution de slots



Choisir une politique

→ Caractéristiques recherchées

- Capacité à garantir une bande passante
- Empêchement du phénomène de famine parmi les communications
- Facilité de mise en oeuvre sur NoC réel

Choisir une politique

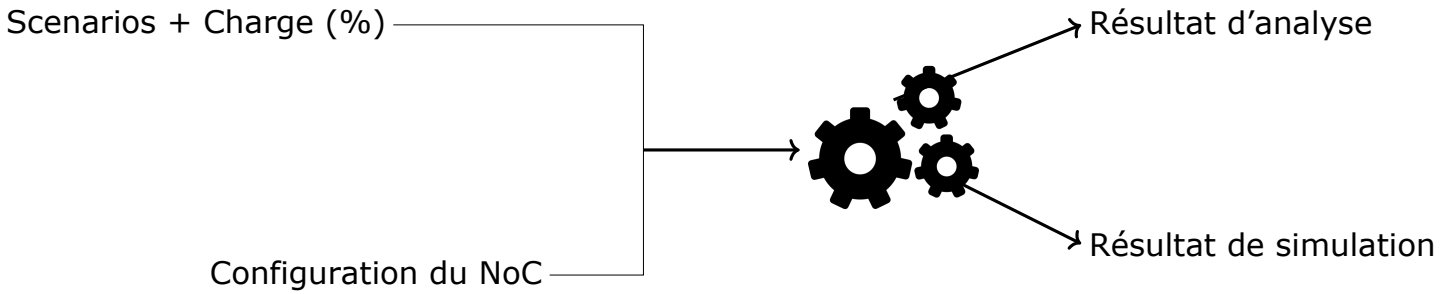
→ Caractéristiques recherchées

- Capacité à garantir une bande passante
- Empêchement du phénomène de famine parmi les communications
- Facilité de mise en oeuvre sur NoC réel

→ Évaluer les deux stratégies (Par priorité & TDMA)

- Par analyse : évaluation au pire-cas
- Par simulation : évaluation par des scénarios

Simulateur et analyseur



Besoin

- Un simulateur NoC accès sur les modèles de tâches
- Ne dépend pas d'une architecture matérielle spécifique

$\rightarrow$ Caractéristiques de ReTINAS [1] :

- Construire différentes topologies du NoC
- Offre plusieurs modèles de tâches
- Tracer le déroulement d'un scénario jusqu'à l'hyperperiode
- Open source et extensible

1. Chawki Benchehida, et al. 2020. An analysis and simulation tool of real-time communications in on-chip networks : a comparative study. SIGBED Rev

Scénarios pour l'expérimentation

On spécifie

- Nombre de processeurs m en Mesh-2D $(m = n \times n)$
- Répartition des slots aux entrées des routeurs
- Taille et nombre des communications selon la charge désirée

Scénarios pour l'expérimentation

On spécifie

- Nombre de processeurs m en Mesh-2D $(m = n \times n)$
- Répartition des slots aux entrées des routeurs
- Taille et nombre des communications selon la charge désirée

Paramètres de Simulation

Topologie	4x4 2D-Mesh
VC par InputPort	6
Taille des VC	10 Flits
Periodes (cycle)	[1000, 1500, 2000, 3000, 4000, 6000]
Slots TDMA	[4, 2, 3, 5, 3, 3]
Message	8 Packets (10 Flits each)

Scénarios pour l'expérimentation

On spécifie

- Nombre de processeurs m en Mesh-2D $(m = n \times n)$
- Répartition des slots aux entrées des routeurs
- Taille et nombre des communications selon la charge désirée

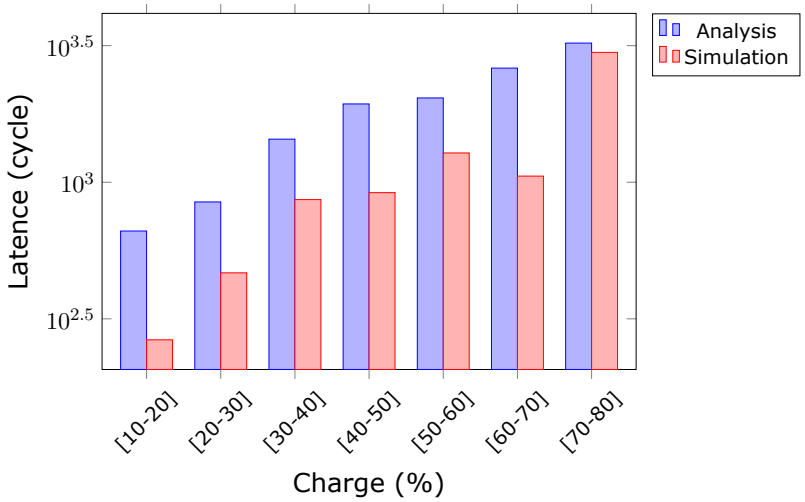
Paramètres de Simulation

Topologie	4x4 2D-Mesh
VC par InputPort	6
Taille des VC	10 Flits
Periodes (cycle)	[1000, 1500, 2000, 3000, 4000, 6000]
Slots TDMA	[4, 2, 3, 5, 3, 3]
Message	8 Packets (10 Flits each)

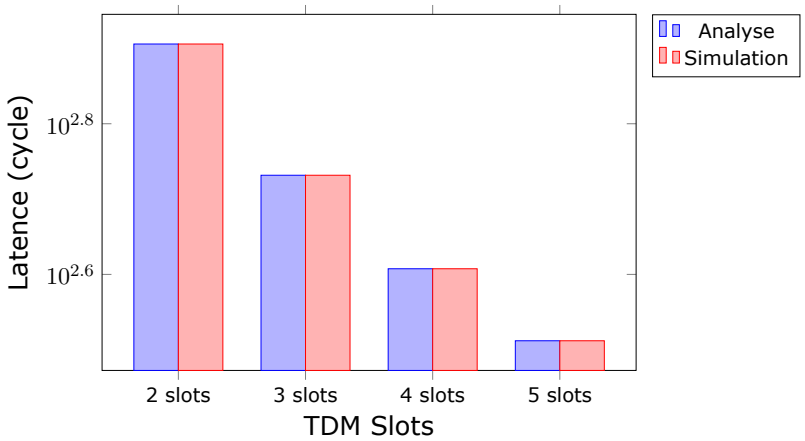
- Générer une tâche témoin
- Ajouter des tâches conflictuelles pour créer une charge (stress)

Résultat : Latence théorique et pratique

→ Arbitrage par priorité



→ Arbitrage TDMA



Ce que nous retenons

→ Difficile de comparer les deux stratégies

Ce que nous retenons

→ Difficile de comparer les deux stratégies

Arbitrage par priorité

- Difficile à réaliser en pratique
  - Adapter le nombre de VC aux nombre de priorités

Ce que nous retenons

→ Difficile de comparer les deux stratégies

Arbitrage par priorité

- Difficile à réaliser en pratique
  - Adapter le nombre de VC aux nombre de priorités
- Préemption des communications coûteuse (changement de priorité)

Ce que nous retenons

→ Difficile de comparer les deux stratégies

Arbitrage par priorité

- Difficile à réaliser en pratique
  - Adapter le nombre de VC aux nombre de priorités
- Préemption des communications coûteuse (changement de priorité)

Cependant, **TDMA**

- Facilité de mise en oeuvre
- Facilité du calcul de latence
- Utilisé par les COTS NoC

Plan

1 Contexte de l'étude

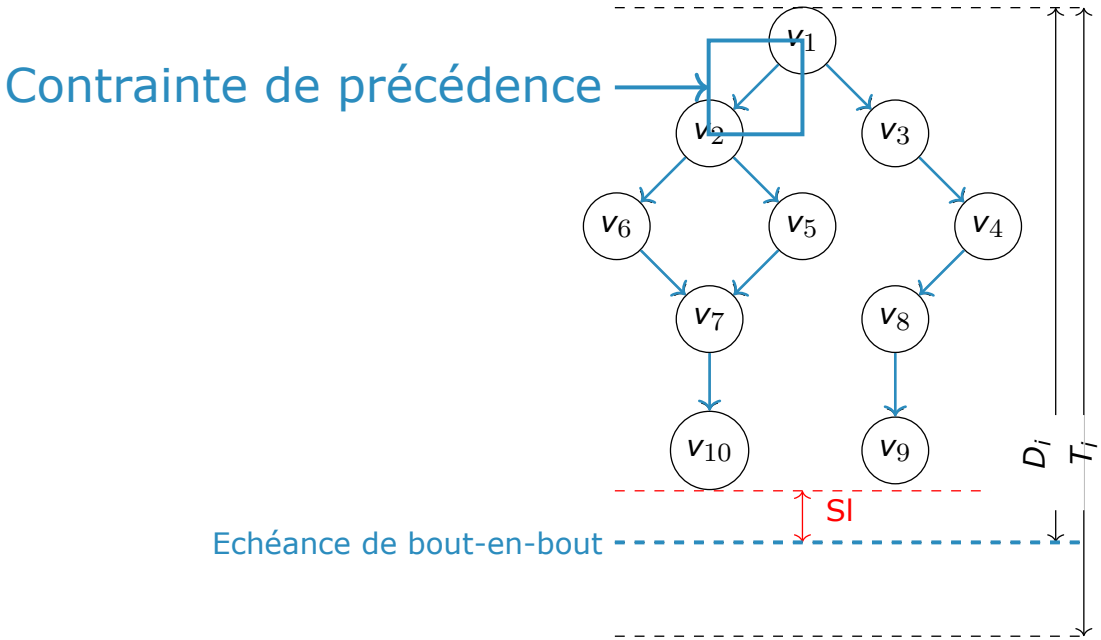
2 Étude comparative des stratégies d'arbitrage dans les routeurs

3 Transformation et analyse du modèle de tâche en DAG sur NoC

4 Placement efficace de tâches sur NoC

5 Conclusion et perspectives

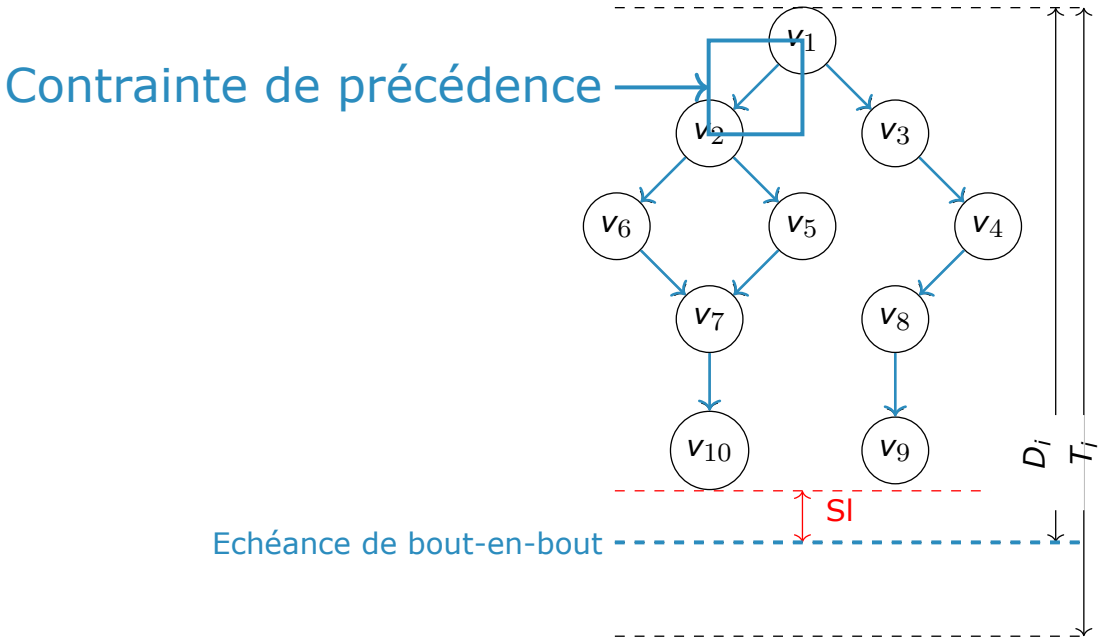
Placement de tâches : problème d'analyse



$\longrightarrow$ Un schéma de placement est constitué d'un ensemble de :

- Couples processeur/sous-tâches (proc_i, v_j)
- Couples communication/VC (VC_k, δ_l)

Placement de tâches : problème d'analyse



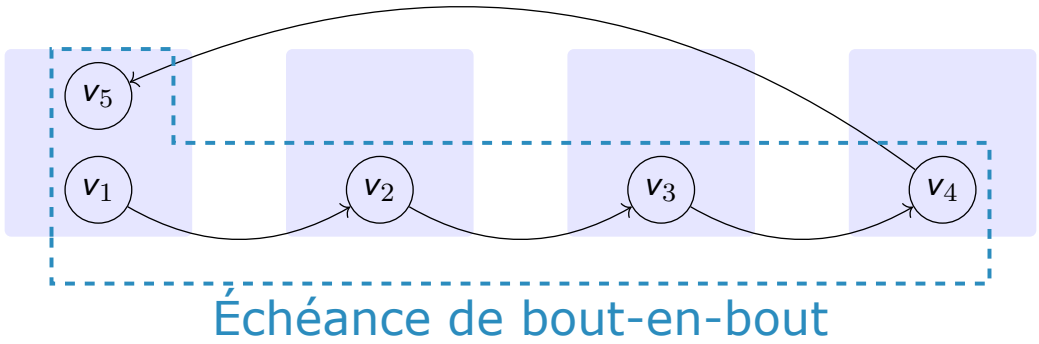
→ Un schéma de placement est constitué d'un ensemble de :

- Couples processeur/sous-tâches (proc_i, v_j)
- Couples communication/VC (VC_k, δ_l)

→ Le schéma donné est-t-il faisable ? → Analyse d'ordonnançabilité

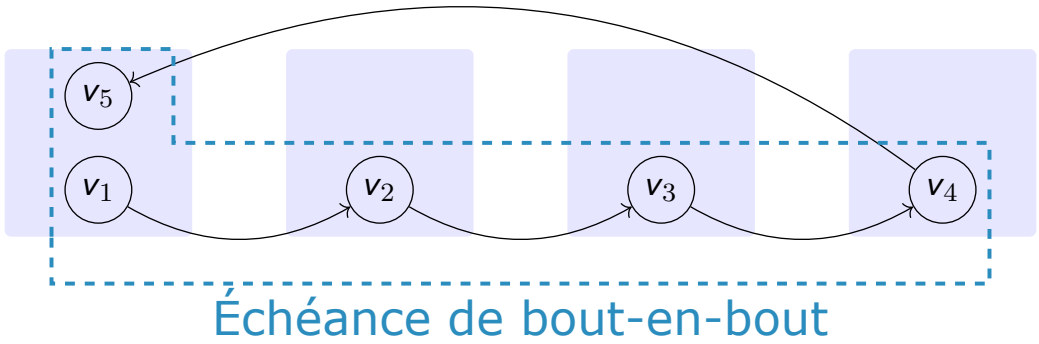
Analyse de tâches en DAG

$\rightarrow$ Une structure fortement couplée. **Difficilement analysable**



Analyse de tâches en DAG

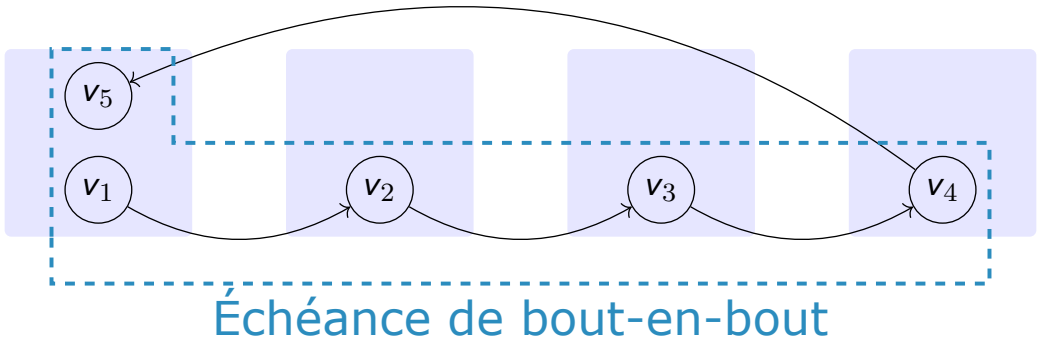
$\rightarrow$ Une structure fortement couplée. **Difficilement analysable**



$\rightarrow$ Transformer en un problème d'analyse d'ordonnançabilité sur mono-processeur

Analyse de tâches en DAG

$\rightarrow$ Une structure fortement couplée. **Difficilement analysable**



$\rightarrow$ Transformer en un problème d'analyse d'ordonnançabilité sur mono-processeur

$\rightarrow$ Analyse d'ordonnancement sur mono-processeur $\rightarrow$ Un problème déjà résolu dans la littérature

Transformation du problème

$\rightarrow$ Étendre une technique déjà utilisée [2]

- Incorporer la latence de communication

2. Zahaf et al., ”Preemption-Aware Allocation, Deadline Assignment for Conditional DAGs on Partitioned EDF” RTCSA’2020

Transformation du problème

$\rightarrow$ Étendre une technique déjà utilisée [2]

- Incorporer la latence de communication

Comment procéder ?

Méthodes :

$\rightarrow$ Échéances (B-en-B) : distribuer des échéances intermédiaires

$\rightarrow$ Contraintes de précédence : affecter des offsets aux sous-tâches

2. Zahaf et al., ”Preemption-Aware Allocation, Deadline Assignment for Conditional DAGs on Partitioned EDF” RTCSA’2020

Transformation du problème

→ Étendre une technique déjà utilisée [2]

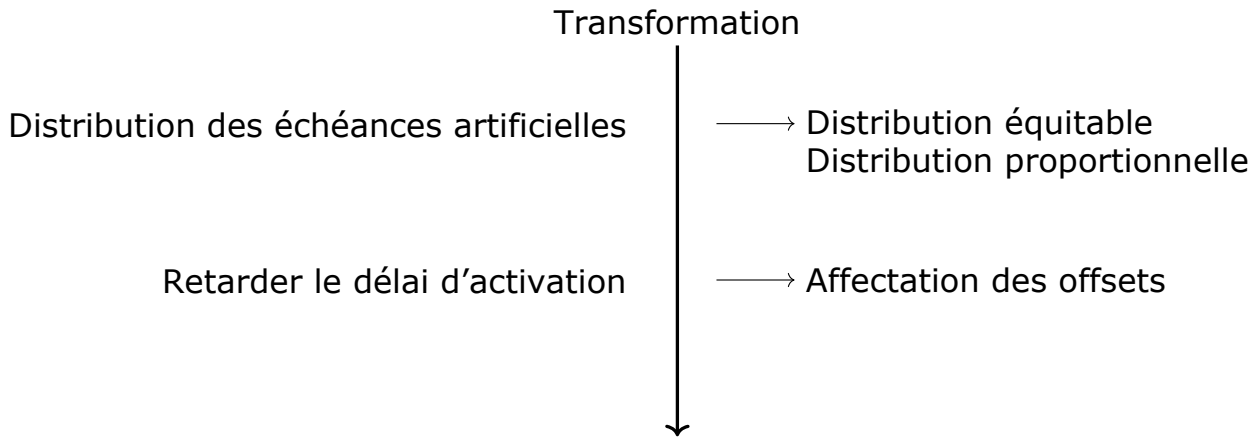
- Incorporer la latence de communication

Comment procéder ?

Méthodes :

→ Échéances (B-en-B) : distribuer des échéances intermédiaires

→ Contraintes de précédence : affecter des offsets aux sous-tâches



2. Zahaf et al., "Preemption-Aware Allocation, Deadline Assignment for Conditional DAGs on Partitioned EDF" RTCSA'2020

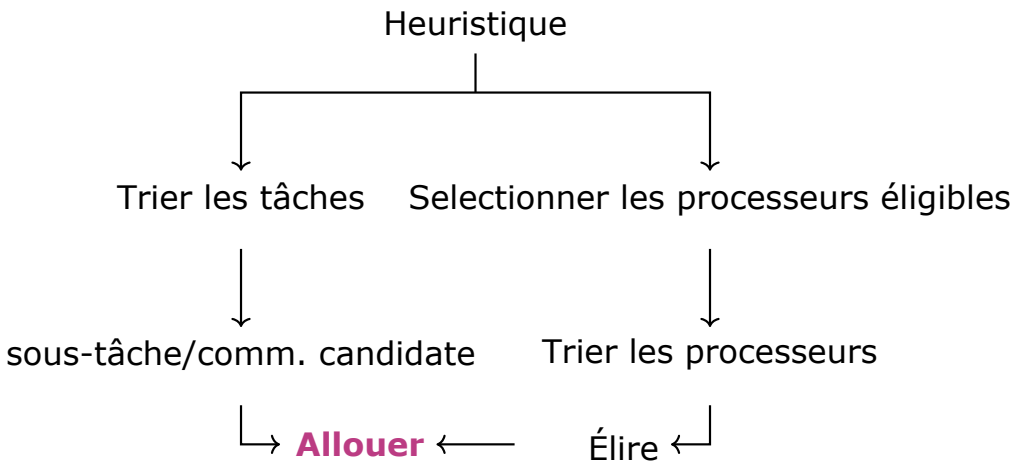
Allocation de tâche

Pour appliquer la méthode, il faut d’abord **Allouer** les sous-tâches et les communications pour construire un schéma de placement

Allocation de tâche

Pour appliquer la méthode, il faut d'abord **Allouer** les sous-tâches et les communications pour construire un schéma de placement

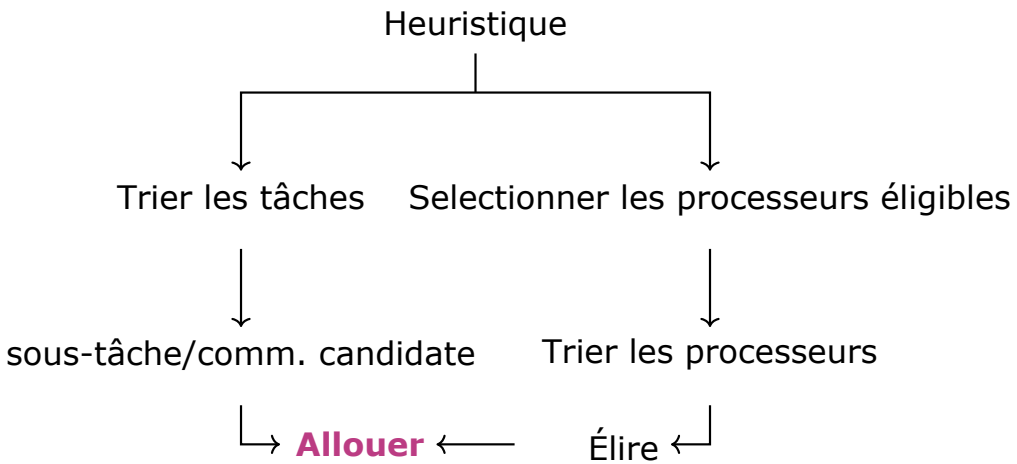
$\longrightarrow$ Utiliser les heuristiques **bin-packing**



Allocation de tâche

Pour appliquer la méthode, il faut d'abord **Allouer** les sous-tâches et les communications pour construire un schéma de placement

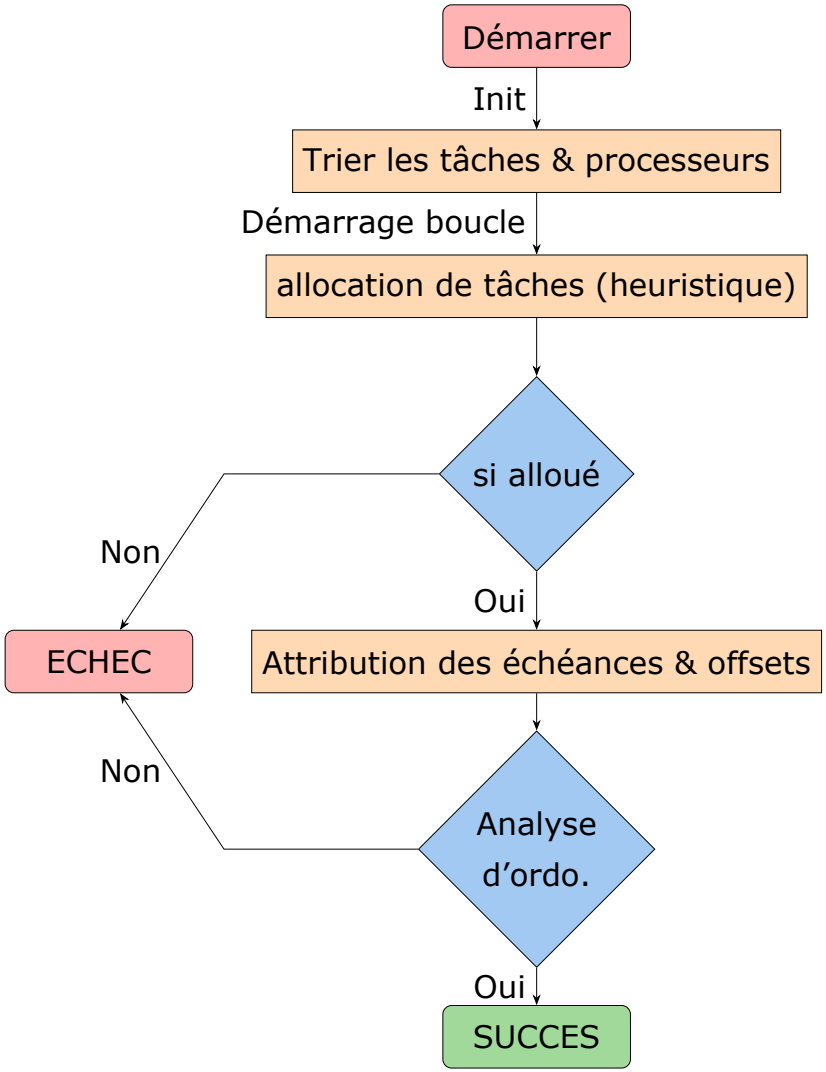
$\longrightarrow$ Utiliser les heuristiques **bin-packing**



À ce stade

Entamer la procédure de transformation

Algorithme d'allocation de tâches



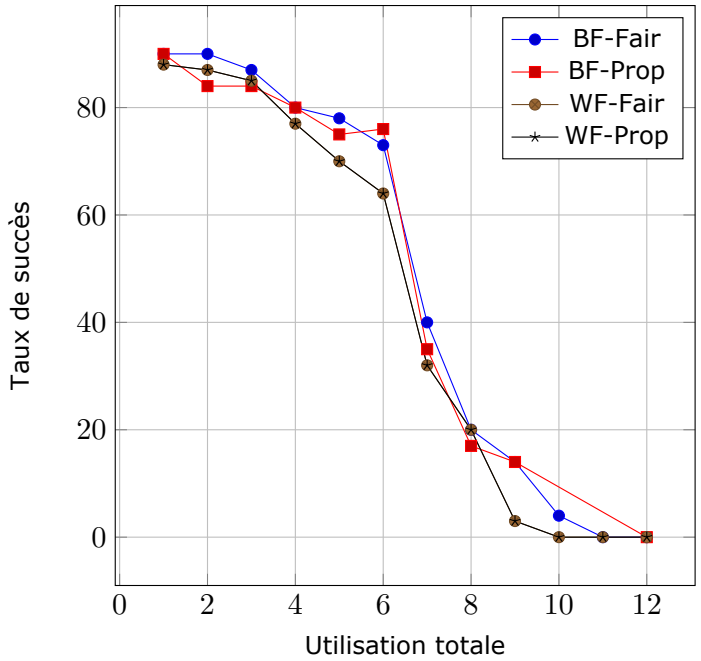
Protocole expérimental

Topologie	4x4 2D-Mesh
VC par Input Port	6
Taille des VC	10 Flits
Periodes	[1000, 1500, 2000, 3000, 4000, 6000]
TDM Slots	[4, 2, 3, 5, 3, 3]
Message	8 Packets (10 Flits chaque)

- Varier l'utilisation totale de 1 à 12 par pas de 1
- EDF comme ordonnanceur
  - Priorité aux sous-tâches qui se rapprochent le plus de leur échéance
- Comparer les heuristiques
- Bin-packing : Best-Fit (BF) et Worst-Fit (WF)
  - Combiné avec une technique de distribution des échéances
  - XX-Prop (Proportionnelle) XX-Fair (Équitable)

→ Analyse d'ordonnançabilité sur mono-processeur

Résultat - Taux d'ordonnançabilité



Ce que nous avons réalisé

$\rightarrow$ Réduire la complexité de l'analyse d'ordonnançabilité [3]

3. Benchehida et al. "Task and Communication Allocation for Real-time Tasks to Networks-on-Chip Multiprocessors" EDiS'2020

Ce que nous avons réalisé

$\rightarrow$ Réduire la complexité de l'analyse d'ordonnançabilité [3]

En partant

D'un problème d'analyse de tâches en DAG sur NoC

3. Benchehida et al. "Task and Communication Allocation for Real-time Tasks to Networks-on-Chip Multiprocessors" EDiS'2020

Ce que nous avons réalisé

$\rightarrow$ Réduire la complexité de l'analyse d'ordonnançabilité [3]

En partant

D'un problème d'analyse de tâches en DAG sur NoC

Jusqu'à

Un problème d'ordonnancement de tâches indépendantes sur mono-processeur

3. Benchehida et al. "Task and Communication Allocation for Real-time Tasks to Networks-on-Chip Multiprocessors" EDiS'2020

Ce que nous avons réalisé

$\rightarrow$ Réduire la complexité de l'analyse d'ordonnançabilité [3]

En partant

D'un problème d'analyse de tâches en DAG sur NoC

Jusqu'à

Un problème d'ordonnancement de tâches indépendantes sur mono-processeur

Remarque

- Le placement des tâches jusqu'ici n'obéit à aucun objectif

3. Benchehida et al. "Task and Communication Allocation for Real-time Tasks to Networks-on-Chip Multiprocessors" EDiS'2020

Plan

1 Contexte de l'étude

2 Étude comparative des stratégies d'arbitrage dans les routeurs

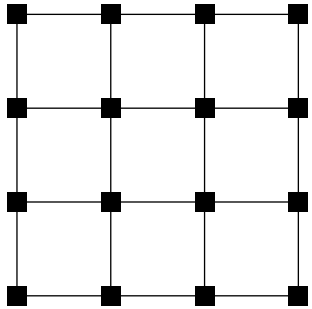
3 Transformation et analyse du modèle de tâche en DAG sur NoC

4 Placement efficace de tâches sur NoC

5 Conclusion et perspectives

NoC : Complexité du problème de placement

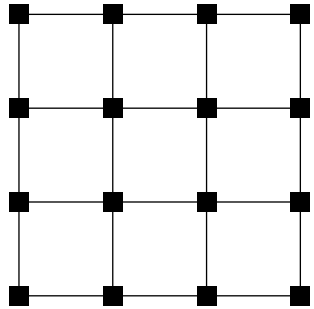
Réduire la complexité du problème de placement par la division du NoC en plusieurs sous-régions



Considérons l'allocation de **16 sous-tâches** (regroupés 4 DAGs) à **16 coeurs**

NoC : Complexité du problème de placement

Réduire la complexité du problème de placement par la division du NoC en plusieurs sous-régions

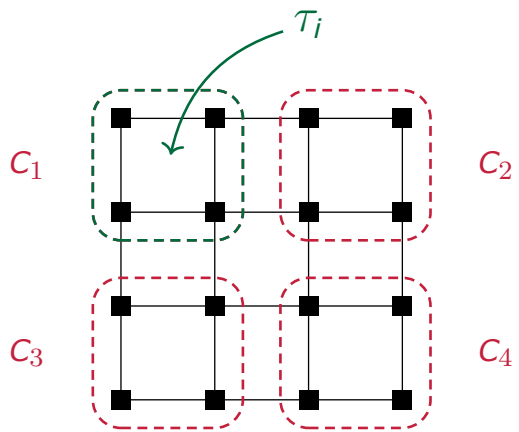


Considérons l'allocation de **16 sous-tâches** (regroupés 4 DAGs) à **16 coeurs**

- Nombre de combinaisons $\rightarrow 16^{16}$
  - Cela, sans compter les communications

NoC : Complexité du problème de placement

Réduire la complexité du problème de placement par la division du NoC en plusieurs sous-régions



Considérons l'allocation de **16 sous-tâches** (regroupés 4 DAGs) à **16 coeurs**

- Nombre de combinaisons $\rightarrow 16^{16}$
  - Cela, sans compter les communications

Après la division du NoC en plusieurs sous-régions

- 4 clusters pour 4 DAGs
- Nombre de combinaisons $\rightarrow (4^4) \times 4 = 1024$

Un problème multi-objectifs

But

Considérer la minimisation de temps de réponse de chaque tâche comme un seul objectif $\rightarrow$ Optimisation multi-objectifs

Un problème multi-objectifs

But

Considérer la minimisation de temps de réponse de chaque tâche comme un seul objectif $\rightarrow$ Optimisation multi-objectifs

Nous avons :

- Un ensemble de tâches en DAG composés de :
  - Ensemble de sous-tâches (noeuds)
  - Leurs communications (arêtes)

Un problème multi-objectifs

But

Considérer la minimisation de temps de réponse de chaque tâche comme un seul objectif $\rightarrow$ Optimisation multi-objectifs

Nous avons :

- Un ensemble de tâches en DAG composés de :
  - Ensemble de sous-tâches (noeuds)
  - Leurs communications (arêtes)

Chercher :

- Un placement ordonnançable qui offre le meilleur temps de réponse possible

Un problème multi-objectifs

But

Considérer la minimisation de temps de réponse de chaque tâche comme un seul objectif $\rightarrow$ Optimisation multi-objectifs

Nous avons :

- Un ensemble de tâches en DAG composés de :
  - Ensemble de sous-tâches (noeuds)
  - Leurs communications (arêtes)

Chercher :

- Un placement ordonnançable qui offre le meilleur temps de réponse possible

Comment ?

Choisir une méthode qui explore rapidement de larges possibilités

Techniques d'optimisation

$\rightarrow$ Un problème multi-objectif impliquant plusieurs tâches

- Méthode exactes $\rightarrow$ temps de calcul important

Techniques d'optimisation

$\rightarrow$ Un problème multi-objectif impliquant plusieurs tâches

- Méthode exactes $\rightarrow$ temps de calcul important

Les métaheuristiques

Présentent un compromis entre vitesse d'exploration et précision de la solution (fournissent un résultat proche de l'optimal)

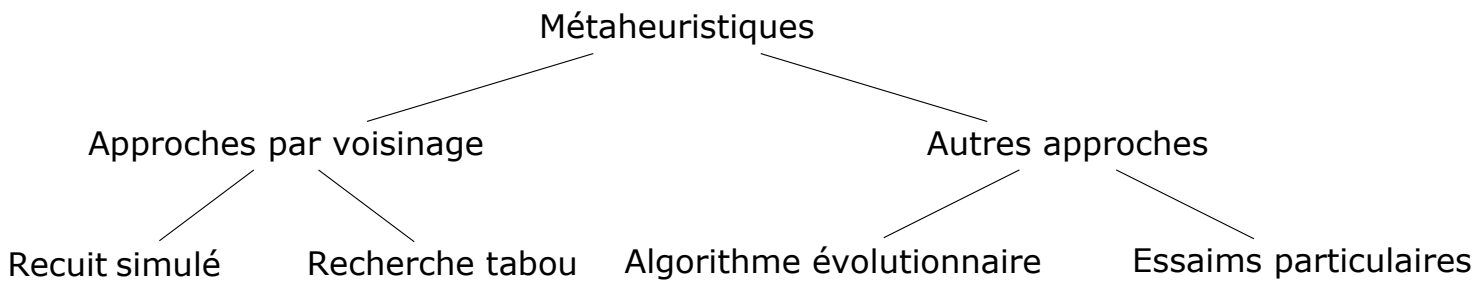
Techniques d'optimisation

$\rightarrow$ Un problème multi-objectif impliquant plusieurs tâches

- Méthode exactes $\rightarrow$ temps de calcul important

Les métaheuristiques

Présentent un compromis entre vitesse d'exploration et précision de la solution (fournissent un résultat proche de l'optimal)



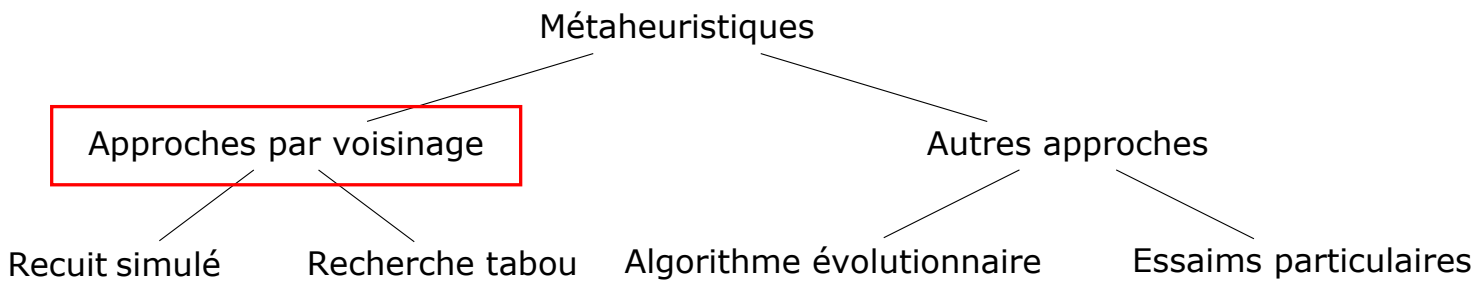
Techniques d'optimisation

$\rightarrow$ Un problème multi-objectif impliquant plusieurs tâches

- Méthode exactes $\rightarrow$ temps de calcul important

Les métaheuristiques

Présentent un compromis entre vitesse d'exploration et précision de la solution (fournissent un résultat proche de l'optimal)



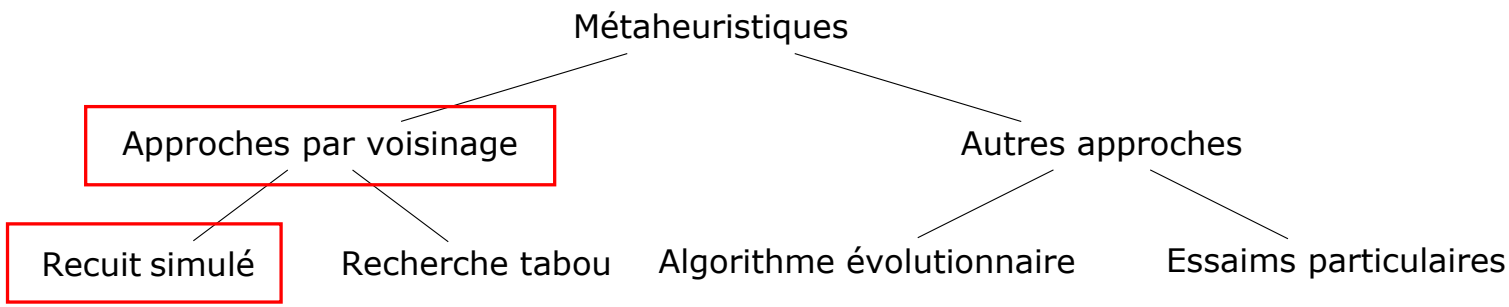
Techniques d'optimisation

$\rightarrow$ Un problème multi-objectif impliquant plusieurs tâches

- Méthode exactes $\rightarrow$ temps de calcul important

Les métaheuristiques

Présentent un compromis entre vitesse d'exploration et précision de la solution (fournissent un résultat proche de l'optimal)



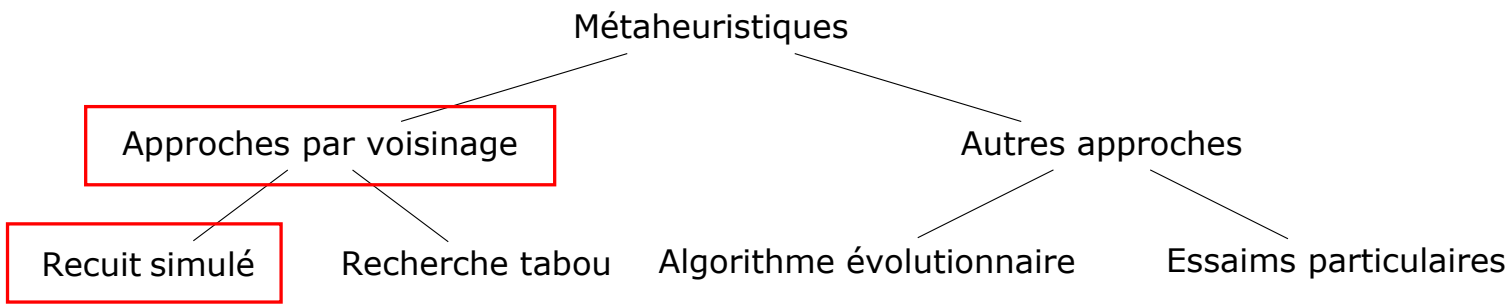
Techniques d'optimisation

→ Un problème multi-objectif impliquant plusieurs tâches

- Méthode exactes → temps de calcul important

Les métaheuristiques

Présentent un compromis entre vitesse d'exploration et précision de la solution (fournissent un résultat proche de l'optimal)



→ Constituer un *Front Pareto*

Placement par recuit simulé

Déterminer une fonction d’exploration du voisinage qui tente, premièrement, un changement local

Placement par recuit simulé

Déterminer une fonction d'exploration du voisinage qui tente, premièrement, un changement local

$\rightarrow$ Solution initiale S_o : trouver aléatoirement un schéma d'allocation **ordonnançable** dans l'espace des solutions

Placement par recuit simulé

Déterminer une fonction d'exploration du voisinage qui tente, premièrement, un changement local

$\rightarrow$ Solution initiale S_o : trouver aléatoirement un schéma d'allocation **ordonnançable** dans l'espace des solutions

Exploration du voisinage

1. Désigner aléatoirement, une communication
   1. Déplacer localement vers une VC d'un slot plus grand
   2. Si, impossible $\rightarrow$ 2
2. Sinon, allouer la sous-tâche et sa communication sur un autre processeur

Placement par recuit simulé

Déterminer une fonction d'exploration du voisinage qui tente, premièrement, un changement local

$\rightarrow$ Solution initiale S_o : trouver aléatoirement un schéma d'allocation **ordonnançable** dans l'espace des solutions

Exploration du voisinage

1. Désigner aléatoirement, une communication
   1. Déplacer localement vers une VC d'un slot plus grand
   2. Si, impossible $\rightarrow$ 2
2. Sinon, allouer la sous-tâche et sa communication sur un autre processeur

Enfin, **Comparer** la nouvelle solution par rapport à la précédente

N.B : Une solution moins optimale peut être acceptée

Vers un modèle réaliste

$\rightarrow$ Dans notre modèle

- On suppose que les données sont déjà prêtes dans le NoC (**Pas réaliste**)

4. Benchehida, et al. Memory-Processor co-scheduling for Real-time Tasks on Network-on-chip manycore architectures. IJHPSA

Vers un modèle réaliste

$\rightarrow$ Dans notre modèle

- On suppose que les données sont déjà prêtes dans le NoC (**Pas réaliste**)

Étendre :

- Prendre en compte le temps de copie depuis la DRAM vers la mémoire locale du processeur
- Aussi, la copie retour après exécution [4]

4. Benchehida, et al. Memory-Processor co-scheduling for Real-time Tasks on Network-on-chip manycore architectures. IJHPSA

Vers un modèle réaliste

$\rightarrow$ Dans notre modèle

- On suppose que les données sont déjà prêtes dans le NoC (**Pas réaliste**)

Étendre :

- Prendre en compte le temps de copie depuis la DRAM vers la mémoire locale du processeur
- Aussi, la copie retour après exécution [4]

Inclure :

- Des sous-tâches pour la copie DRAM
- Un modèle de DRAM avec un NoC

4. Benchehida, et al. Memory-Processor co-scheduling for Real-time Tasks on Network-on-chip manycore architectures. IJHPSA

Modèle de tâches (AER)

Étendre le modèle 3-Phases [5]

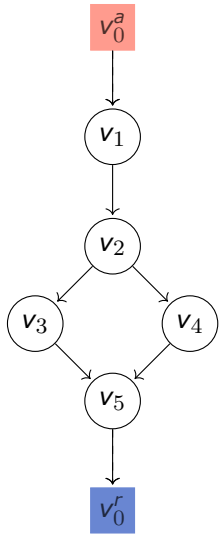
- Pour exprimer les communications entre les processeurs par le biais du réseau (On-chip)

5. Guy Durrieu et al. Predictable Flight Management System Implementation on a Multicore Processor. ERTS'14

Modèle de tâches (AER)

Étendre le modèle 3-Phases [5]

- Pour exprimer les communications entre les processeurs par le biais du réseau (On-chip)

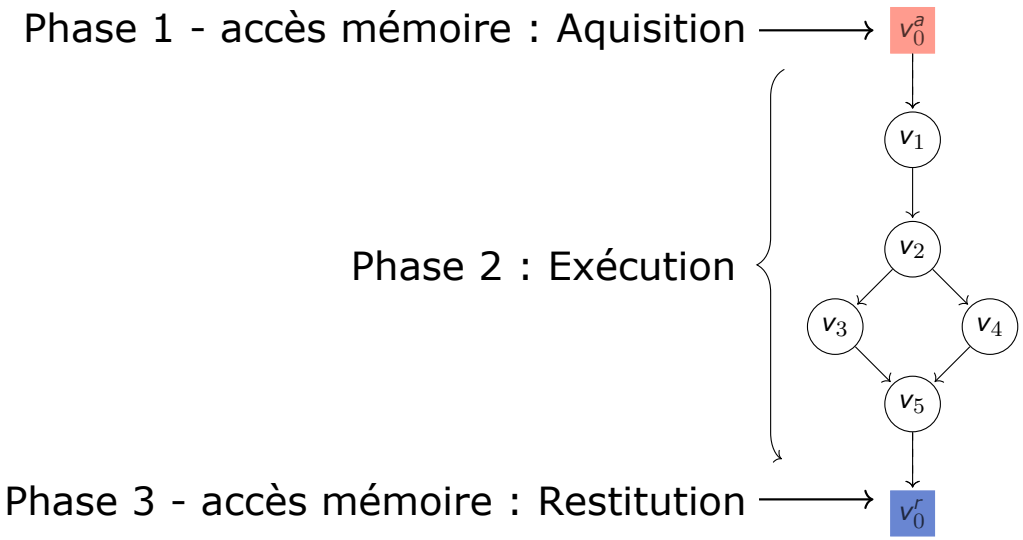


5. Guy Durrieu et al. Predictable Flight Management System Implementation on a Multicore Processor. ERTS'14

Modèle de tâches (AER)

Étendre le modèle 3-Phases [5]

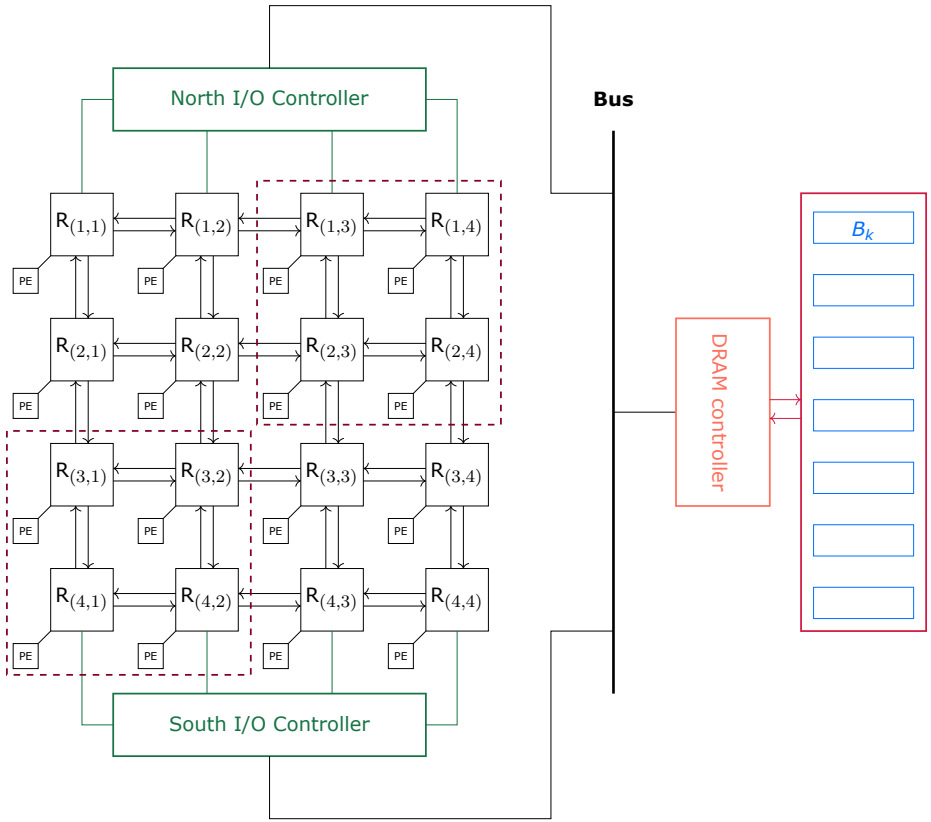
- Pour exprimer les communications entre les processeurs par le biais du réseau (On-chip)



- Sous-tâches virtuelles d'accès mémoire
  - 2 phases mémoires séparées par le bloc de sous-tâches calcul

5. Guy Durrieu et al. Predictable Flight Management System Implementation on a Multicore Processor. ERTS'14

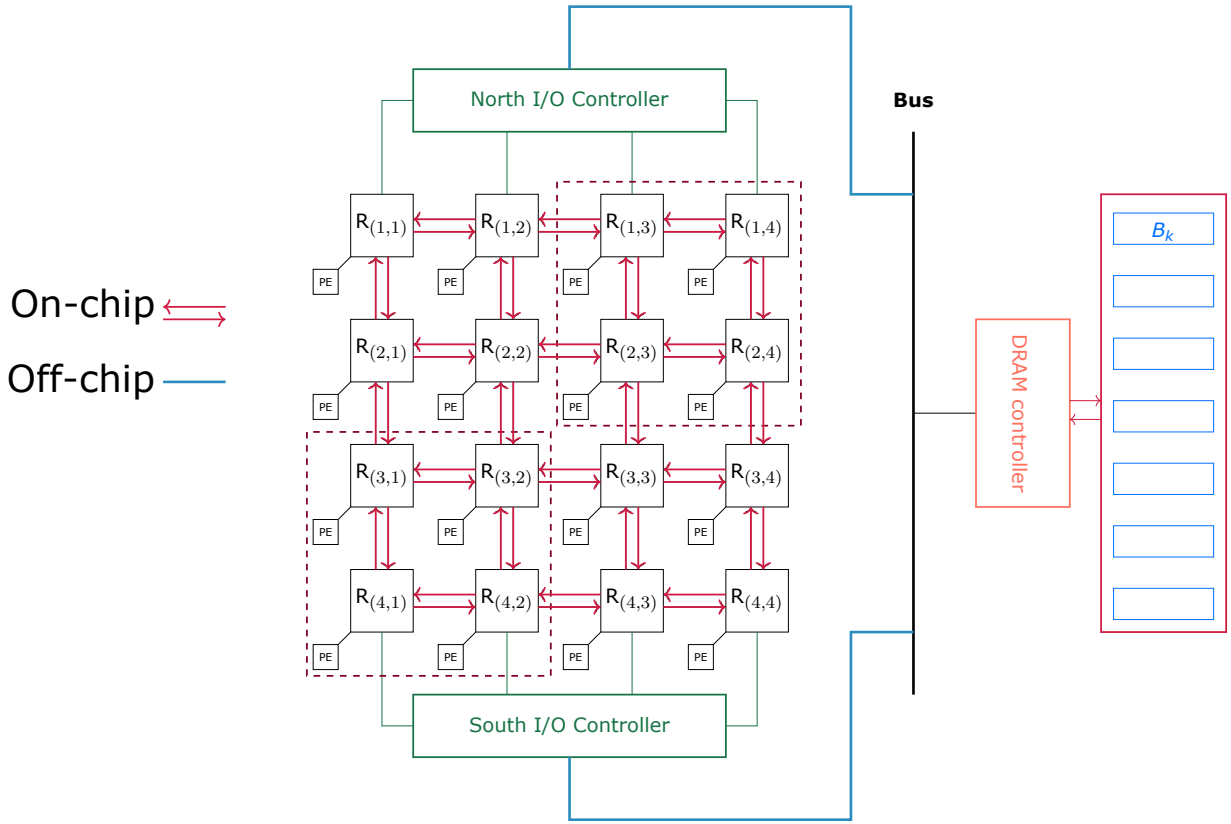
Architecture matérielle



$\rightarrow$ Dans un NoC (4×4) :

- Contrôleurs d'accès mémoire
- DRAM + Contrôleur DRAM

Architecture matérielle



$\rightarrow$ Dans un NoC (4×4) :

- Contrôleurs d'accès mémoire
- DRAM + Contrôleur DRAM

$\rightarrow$ 2 types de communications : on-chip et off-chip

Objectif

L’objectif se résume à :

- Trouver un placement proche de l’optimum
- Dans un NoC formé par des *clusters* de processeurs
- Qui prend en compte le temps de copie DRAM-NoC

Objectif

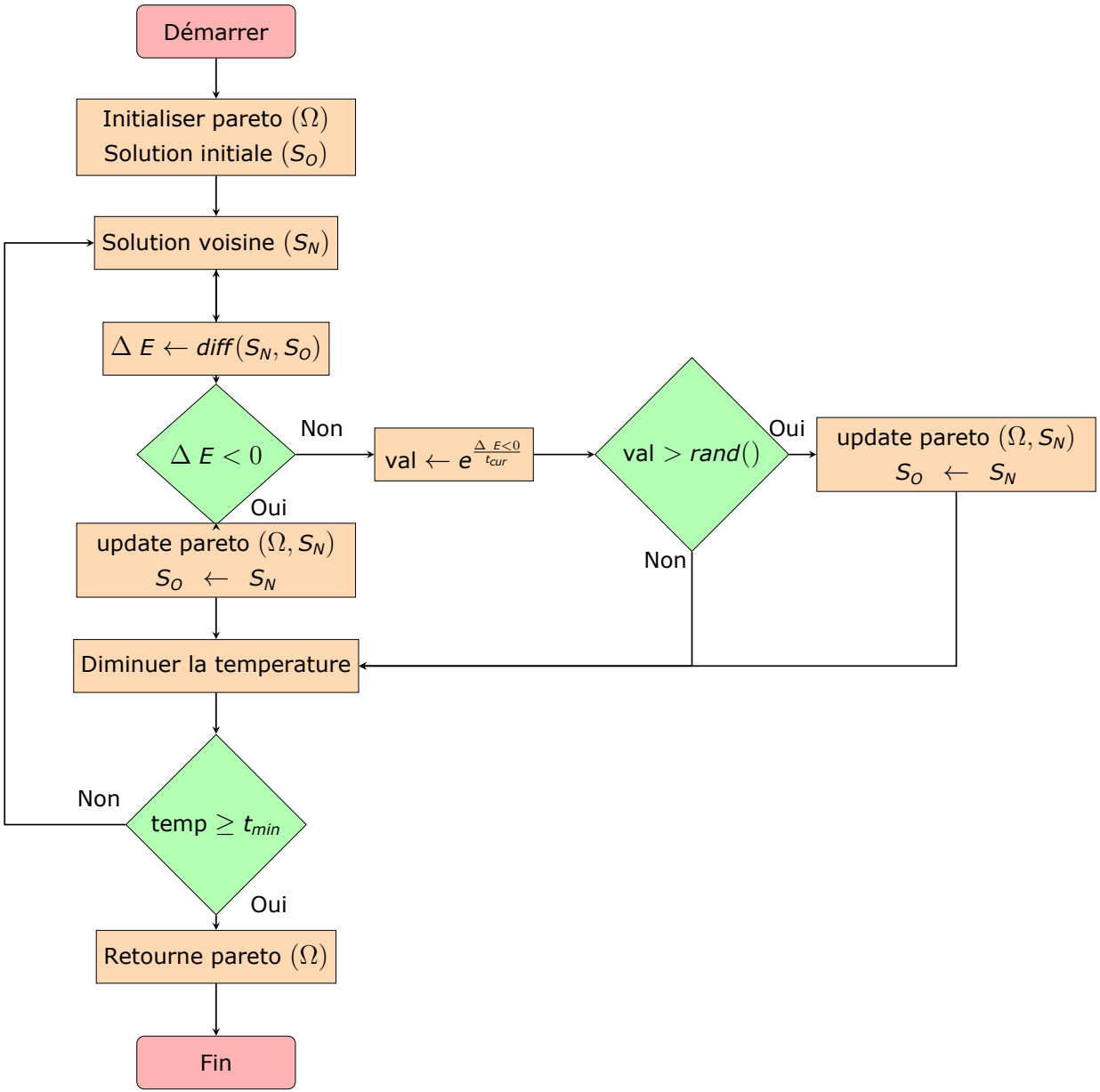
L'objectif se résume à :

- Trouver un placement proche de l'optimum
- Dans un NoC formé par des *clusters* de processeurs
- Qui prend en compte le temps de copie DRAM-NoC

À la fin

Analyser le *Front pareto* pour choisir un schéma de placement

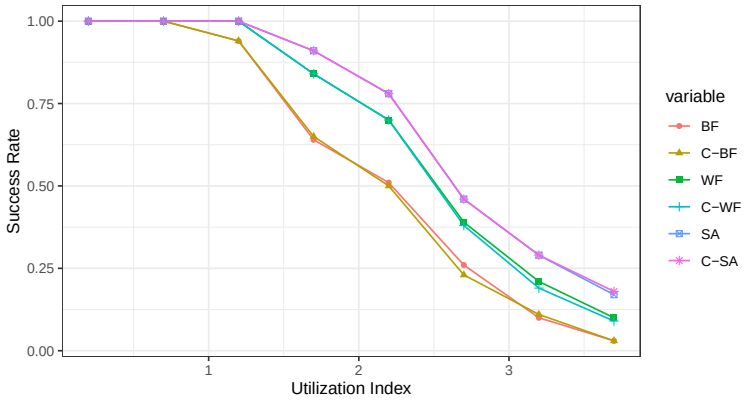
Algorithme d'allocation - Recuit simulé MO



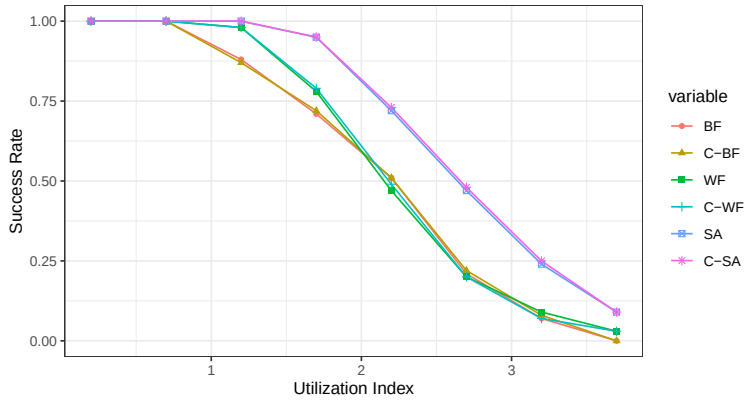
Protocole expérimental

- Varier l'utilisation totale de 1 à 4 par pas de 0.5
- Varier la charge de communication de 20% à 100%
- Ordonnancement à priorité fixe
  - RM (Rate Monotonic) : les tâches avec les périodes les plus courtes sont les plus prioritaires
- DRAM avec partitionnement de Banks
  - Chaque processeur à un accès exclusif à ses Banks
- Comparer les heuristiques
  - Bin-packing (BF et WF) avec le recuit simulé (SA)
  - Avec **core clustering** (C-XX) et sans (XX)
- Produire des expériences avec des paramètres de recuit simulé différents

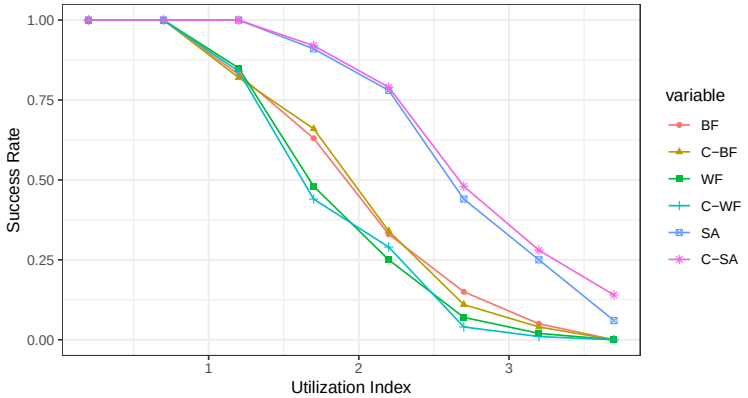
Taux d'odonnançabilité



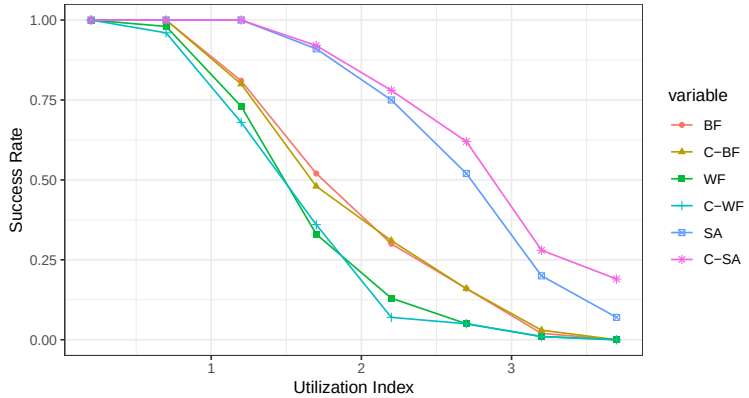
(a) 20% de charge



(b) 40% de charge



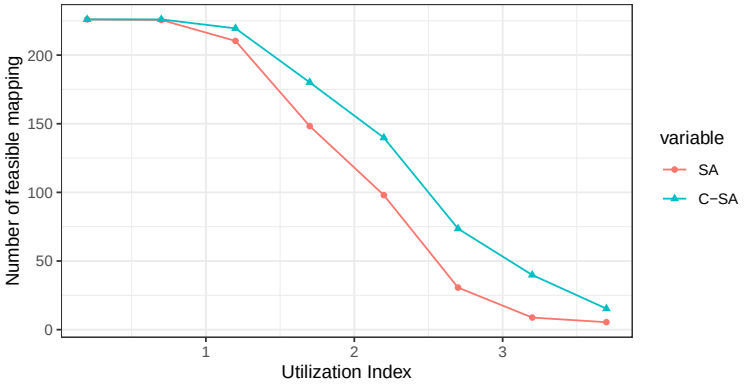
(c) 80% de charge



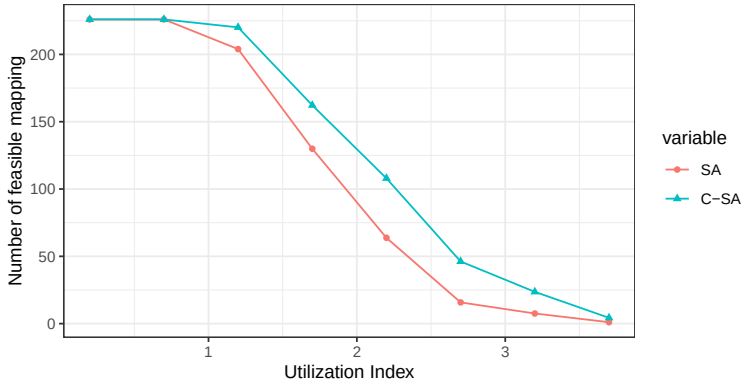
(d) 100% de charge

$\rightarrow t_{init} = 1.0, t_{min} = 10^{-3}, t_{\alpha} = 0.90$

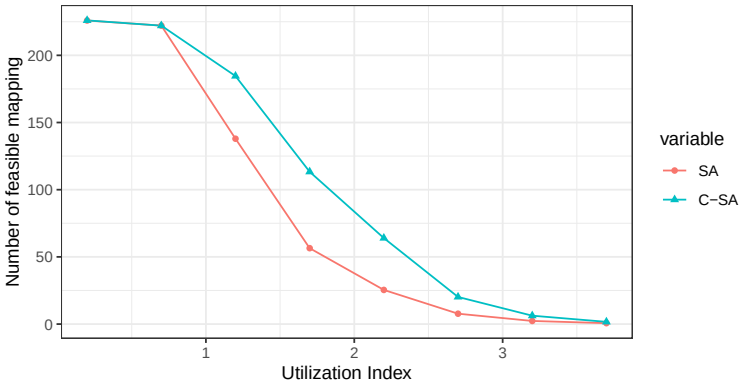
Nombre de solutions faisables



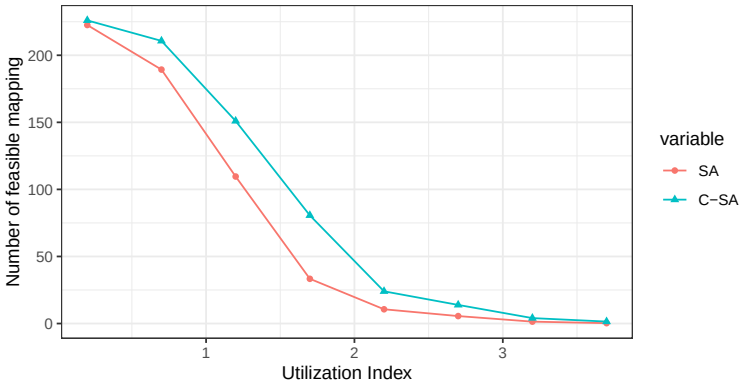
(a) 20% de charge



(b) 40% de charge

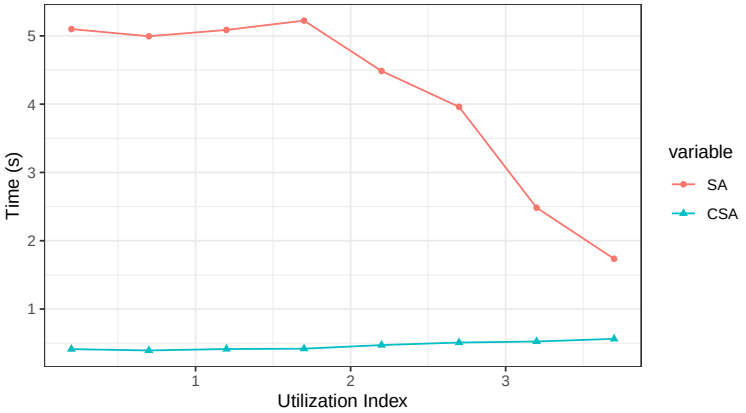


(c) 80% de charge

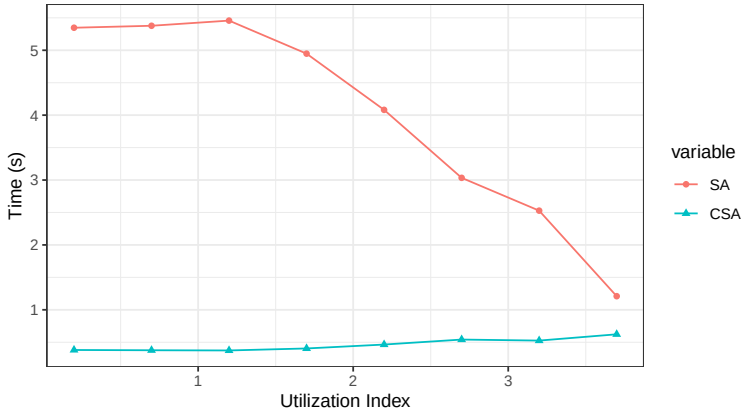


(d) 100% de charge

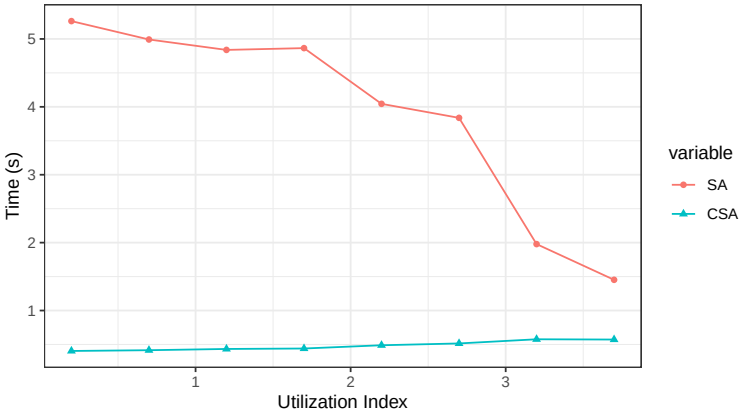
Temps pris pour le placement initial



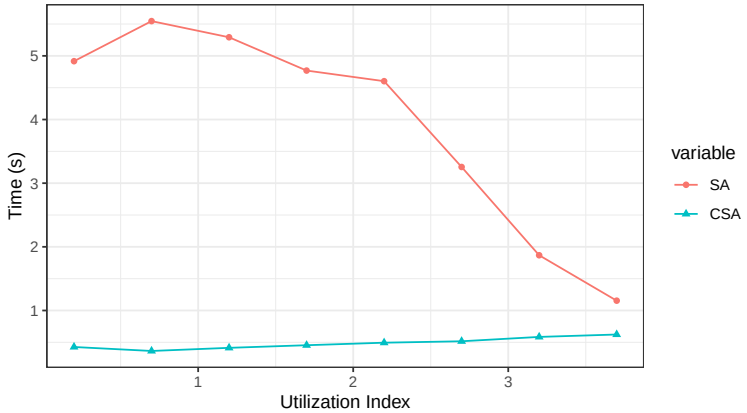
(a) 20% de charge



(b) 40% de charge



(c) 80% de charge



(d) 100% de charge

Plan

1 Contexte de l'étude

2 Étude comparative des stratégies d'arbitrage dans les routeurs

3 Transformation et analyse du modèle de tâche en DAG sur NoC

4 Placement efficace de tâches sur NoC

5 Conclusion et perspectives

Conclusion

Dans cette thèse, nous avons proposé :

- Un NoC avec un contrôle de flux de communications prédictible
- Une analyse d'oronnançabilité simplifiée de tâches en DAG sur NoC
- Un algorithme de placement efficace sur un NoC

Perspectives

- Étendre le problème de placement sur un NoC hétérogène
- Proposer des techniques de core clustering adaptées aux besoins des tâches
- Proposer un algorithme de placement pour les tâches à criticité mixte sur NoC

Merci

Des questions ?

Productions scientifiques

Journaux

- Chawki Benchehida, et al. Memory-Processor co-scheduling for Real-time Tasks on Network-on-chip manycore architectures. Int. J. of High Performance Systems Architecture
- Chawki Benchehida, et al. 2020. An analysis and simulation tool of real-time communications in on-chip networks : a comparative study. SIGBED Rev. 17, 1 (February 2020), 5–11.

Conférences & Workshop

- Chawki Benchehida, et al. ”Task and Communication Allocation for Real-time Tasks to Networks-on-Chip Multiprocessors,” 2020 Second International Conference on Embedded & Distributed Systems (EDiS), Oran, Algeria, 2020, pp. 9-14.
- Chawki Benchehida, et al. ”An analysis and Simulation Tool of Real-Time Communications in On-Chip Networks”, The Embedded Operating Systems Workshop EWiLi’19, New York, USA.
- Chawki Benchehida, et al. ”Real-time Communications in On-Chip Networks”, The 12th Junior Researcher Workshop on Real-Time Computing JRWRTC 2018, Poitiers, France.

En cours

- DAG tasks mapping on cluster-based Network-on-Chip with off-chip communications