

# Agrégation d'arbres Multicast

Joanna Moulierac

Equipe ARMOR : Miklós Molnár,  
Bernard Cousin,  
Raymond Marie,  
Alexandre Guiton.

IRISA, Université de Rennes 1

# Plan de la présentation

1. Contexte de travail
2. L'agrégation rapide : STA
3. L'agrégation d'arbres dans un grand domaine : TALD

# I- Le Contexte du travail

# Multicast

- Proposé par Steve Deering en 1991
- Communication de groupes sur Internet
- Une seule copie du message envoyée, duplication aux routeurs de branchement
- Applications : Vidéoconférences, jeux vidéos en ligne...

# Multicast

Problèmes de passage à l'échelle :

- Un groupe  $g \leftrightarrow$  un arbre  $t \leftrightarrow |t|$  entrées de routage
- Si le nombre de groupes est grand :
  - ◆ Beaucoup d'entrées de routage :
    - Mémoire des routeurs saturée
    - Routage ralenti
  - ◆ Beaucoup de messages de contrôle pour maintenir ces entrées de routage

# L'agrégation d'arbres

- L'agrégation d'arbres proposée en 2001 par M.Gerla, J.-H. Cui et L. Lao de UCLA, Los Angeles
  - ◆ *Aggregated Multicast*
- Proposée pour résoudre le problème de passage à l'échelle

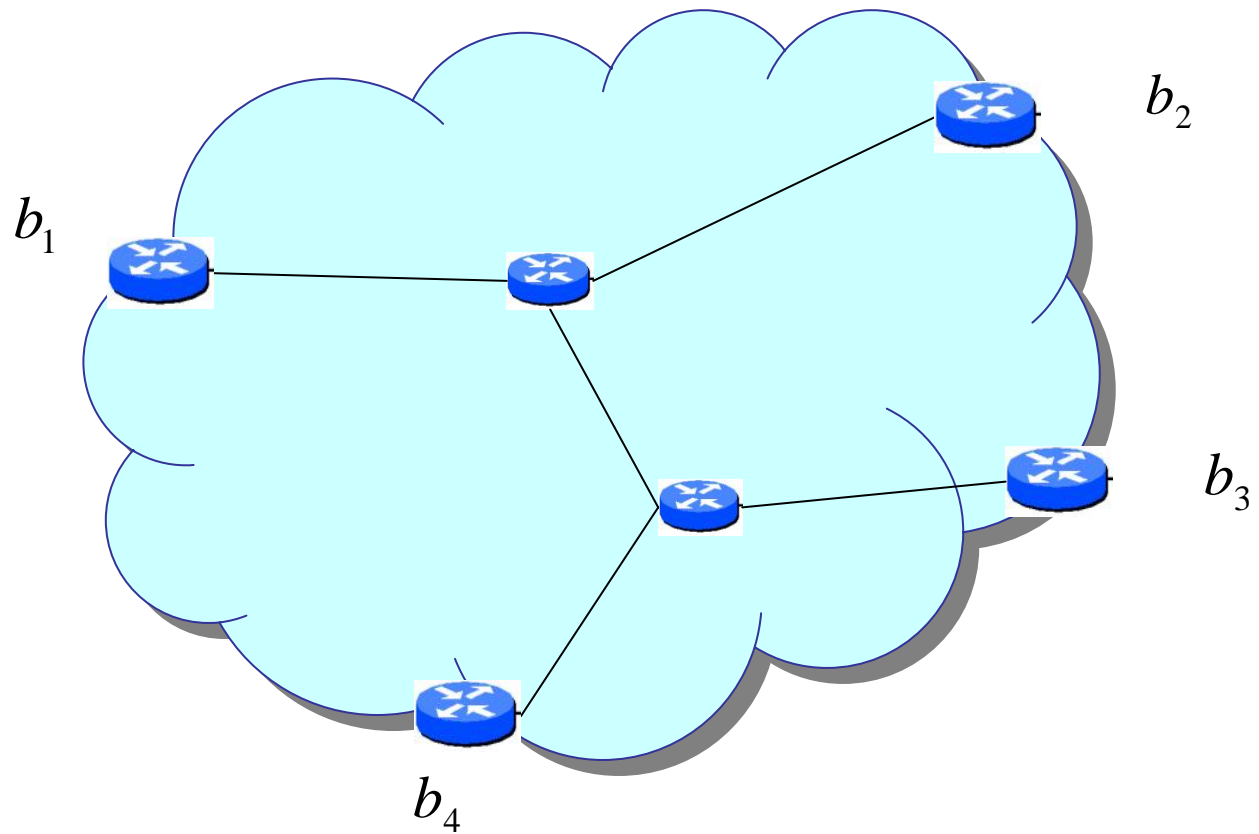
## Principe :

permettre à plusieurs groupes d'utiliser le même arbre

**Multicast** : un groupe, un arbre

**Agrégation** : plusieurs groupes, un arbre

# L'agrégation d'arbres

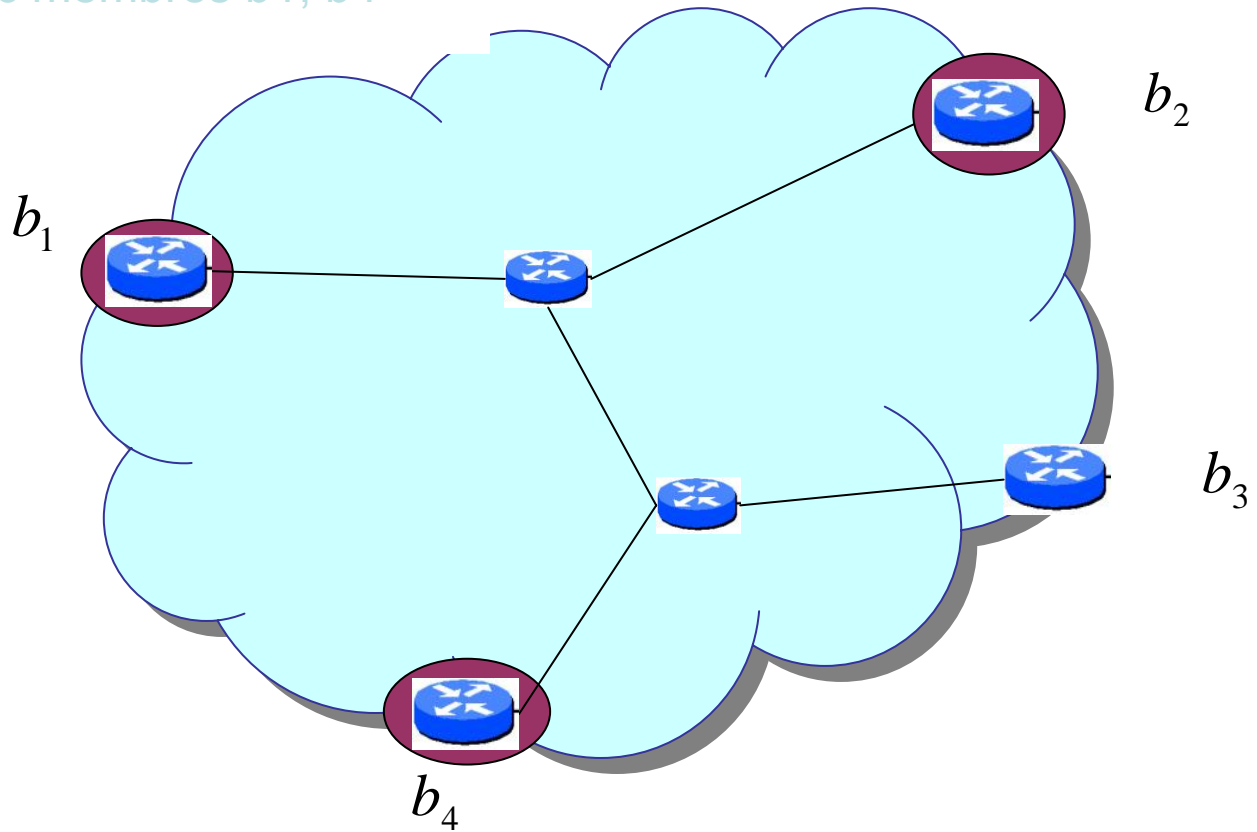


# L'agrégation d'arbres

Groupe g1 avec membres b1, b2, b4

Groupe g2 avec membres b1, b2, b4

Groupe g3 avec membres b1, b4

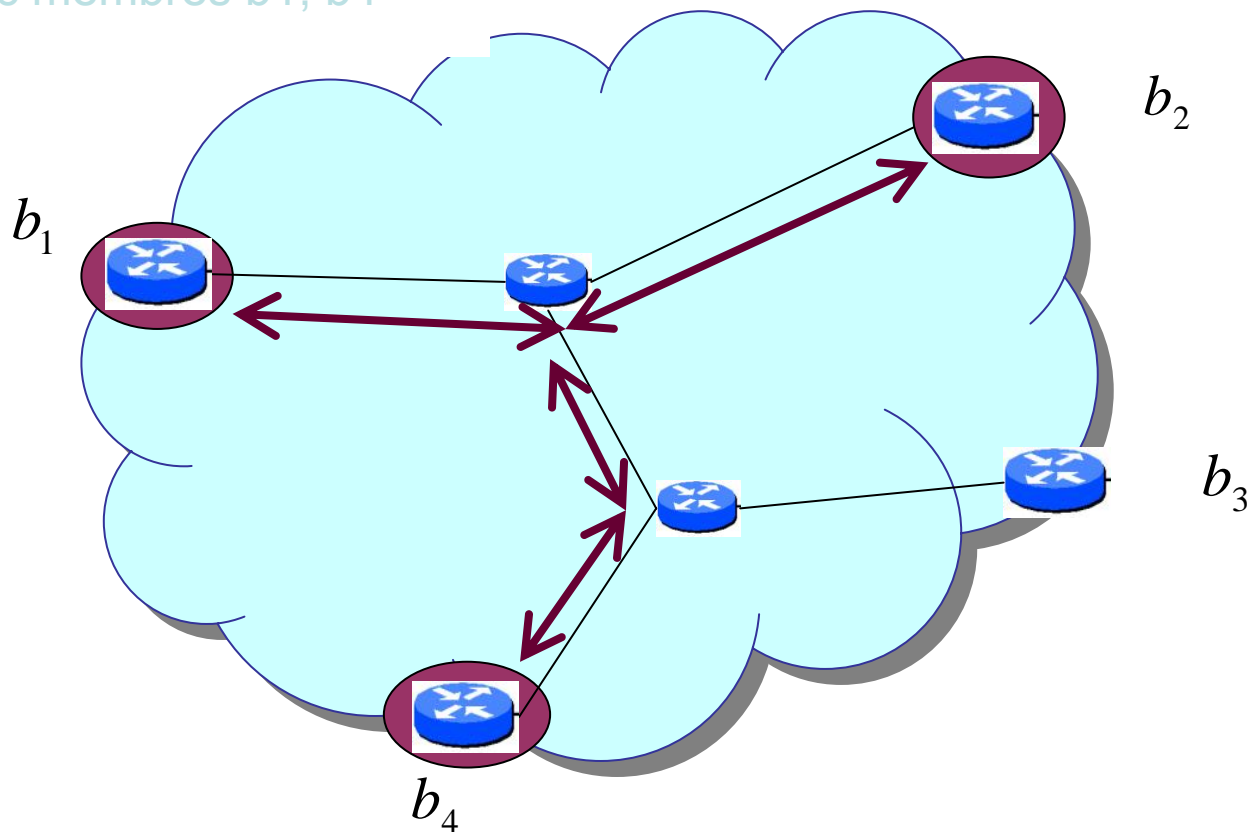


# L'agrégation d'arbres

Groupe g1 avec membres b1, b2, b4

Groupe g2 avec membres b1, b2, b4

Groupe g3 avec membres b1, b4



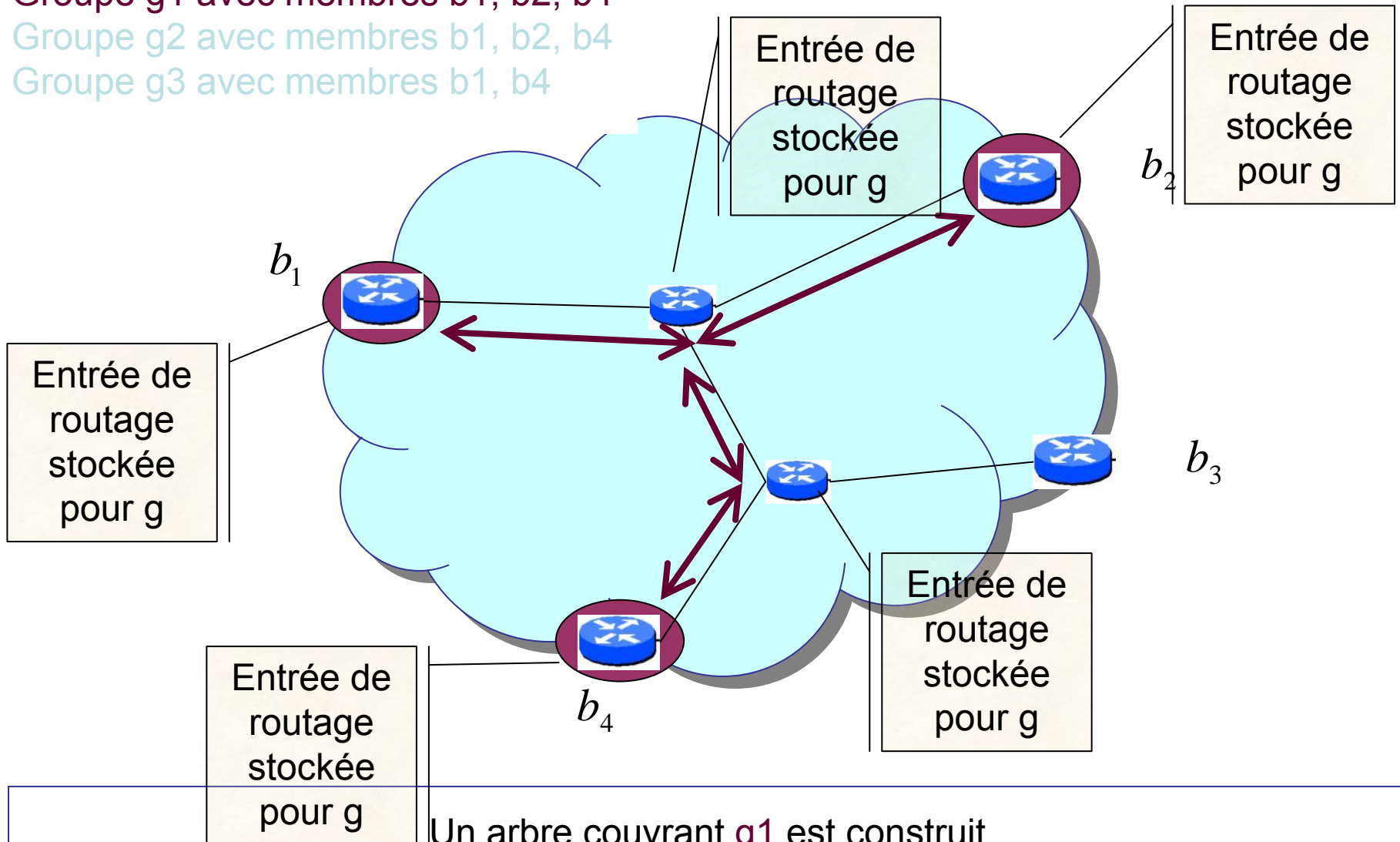
Un arbre couvrant **g1** est construit

# L'agrégation d'arbres

Groupe g1 avec membres b1, b2, b4

Groupe g2 avec membres b1, b2, b4

Groupe g3 avec membres b1, b4

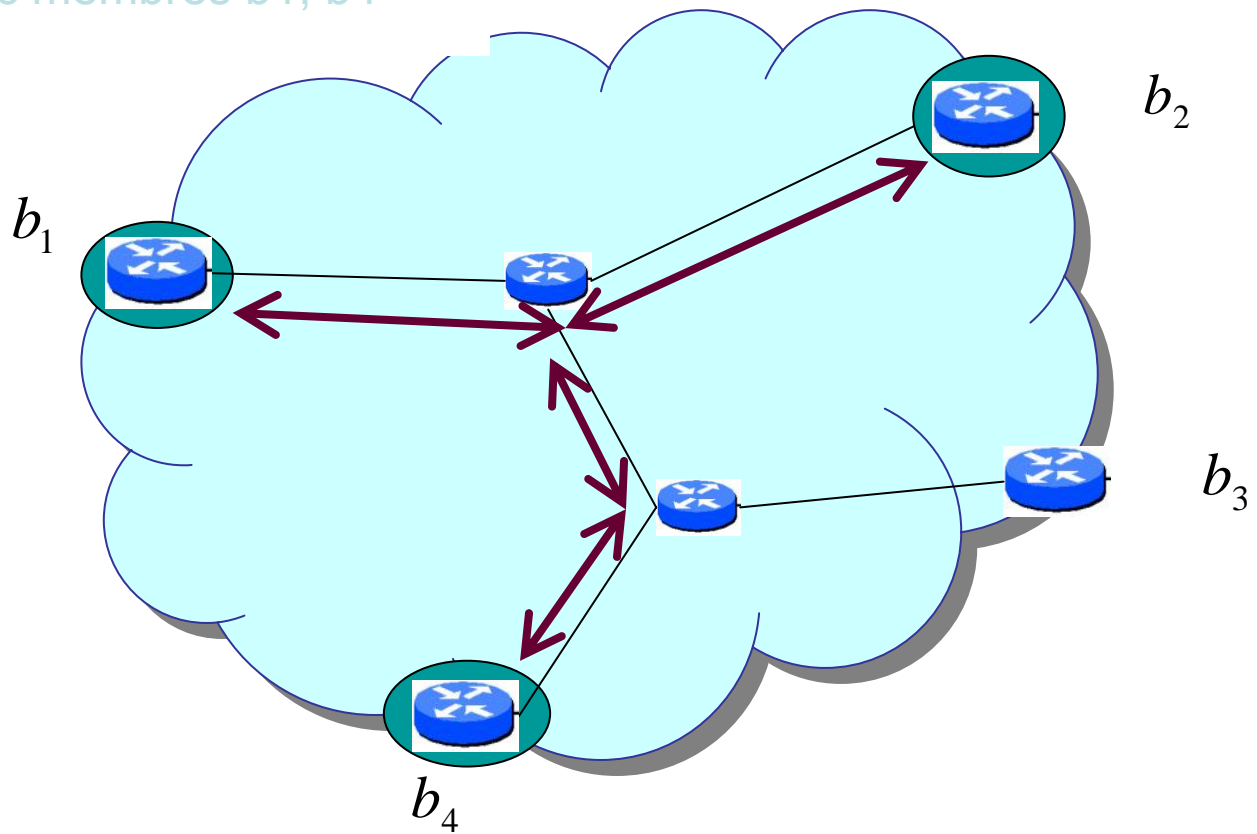


# L'agrégation d'arbres

Groupe g1 avec membres b1, b2, b4

Groupe g2 avec membres b1, b2, b4

Groupe g3 avec membres b1, b4

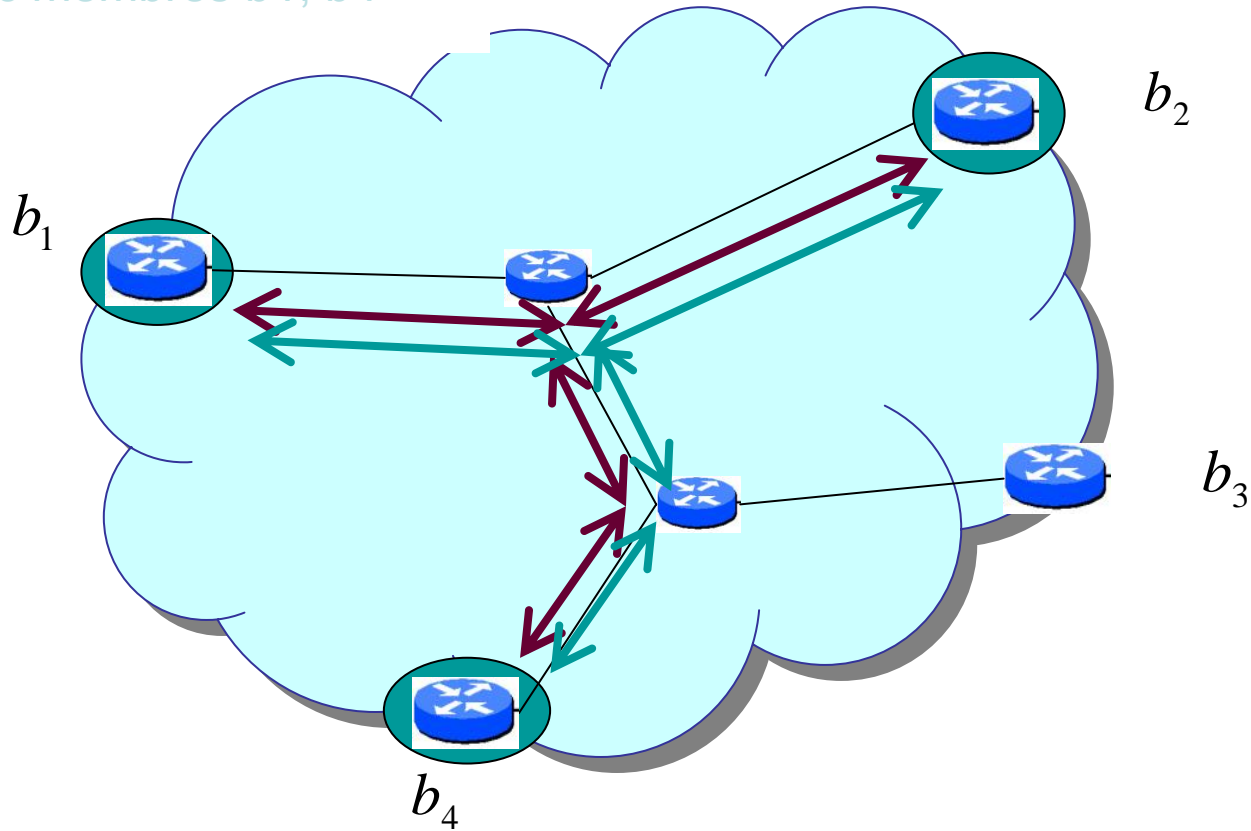


# L'agrégation d'arbres

Groupe g1 avec membres b1, b2, b4

Groupe g2 avec membres b1, b2, b4

Groupe g3 avec membres b1, b4



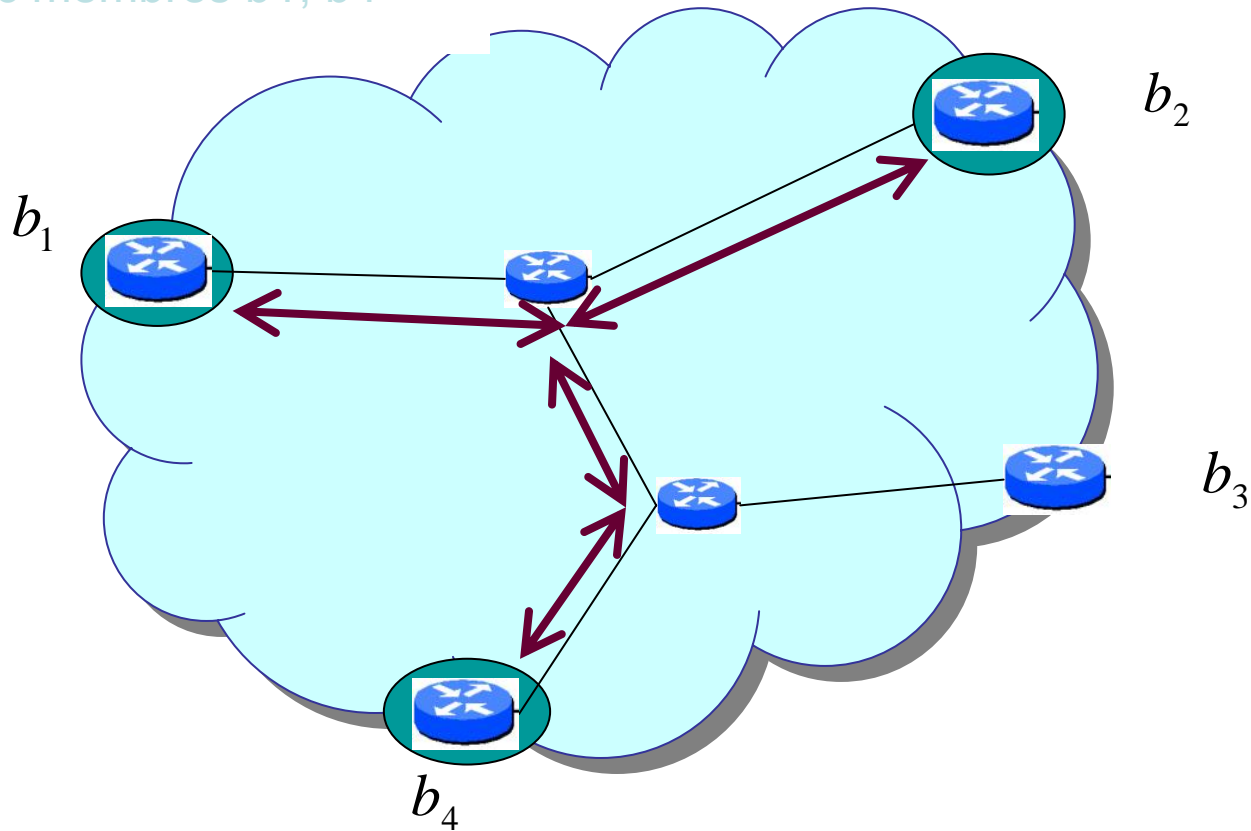
On peut construire un nouvel arbre pour le groupe g2 (comme en multicast)

# L'agrégation d'arbres

Groupe g1 avec membres b1, b2, b4

Groupe g2 avec membres b1, b2, b4

Groupe g3 avec membres b1, b4



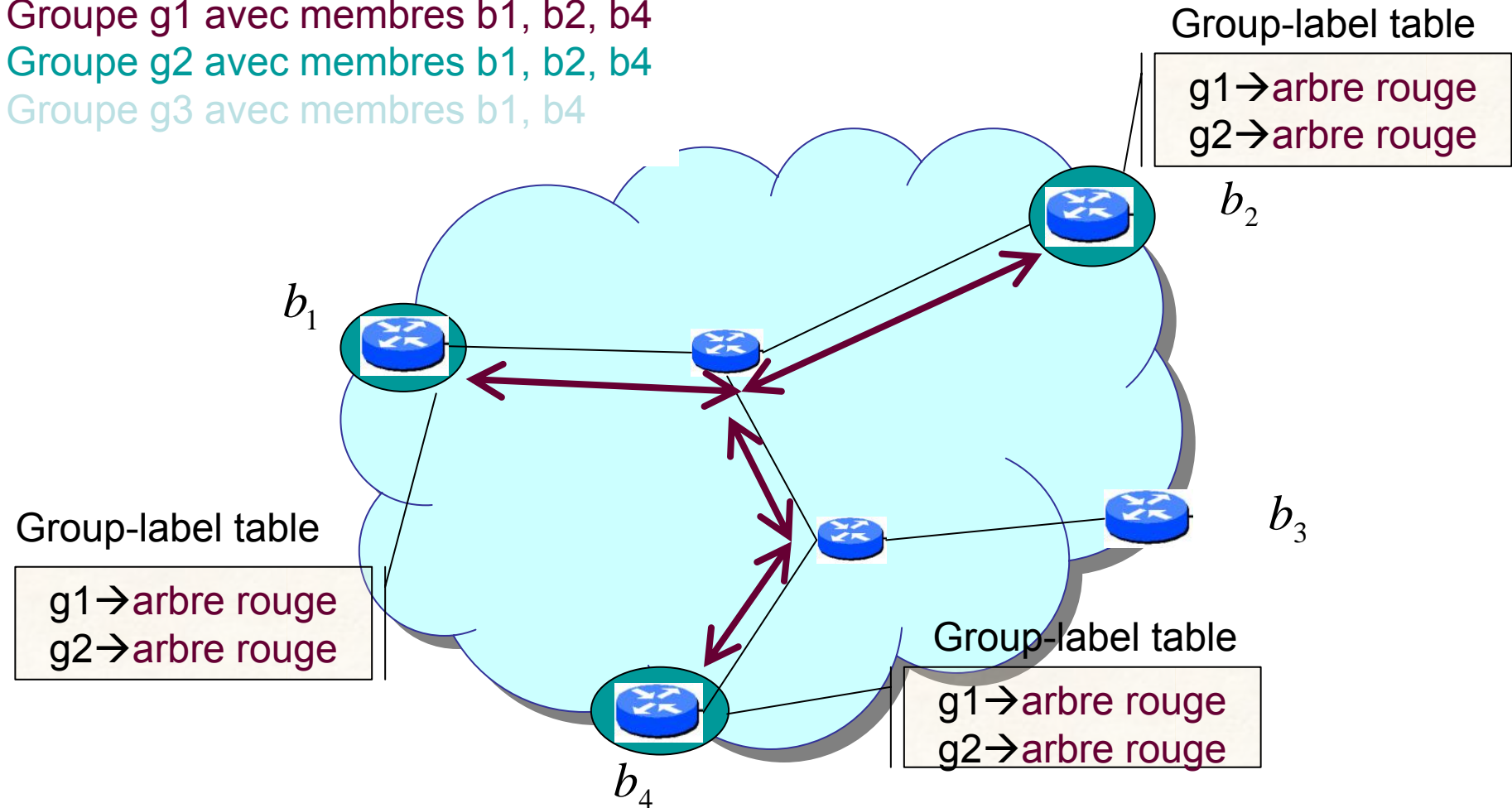
Ou g2 peut utiliser l'arbre construit pour g1

# L'agrégation d'arbres

Groupe g1 avec membres b1, b2, b4

Groupe g2 avec membres b1, b2, b4

Groupe g3 avec membres b1, b4



On rajoute alors des "entrées de groupes" dans des tables groupes-label

# L'agrégation d'arbres Leaky match

■ Si on veut réduire encore plus le nombre d'entrées de routage...

... on peut aussi agréger des groupes à des arbres plus « gros »

■ Agrégation « Leaky Match »

# L'agrégation d'arbres Leaky Match

Groupe g1 avec membres b1, b2, b4

Groupe g2 avec membres b1, b2, b4

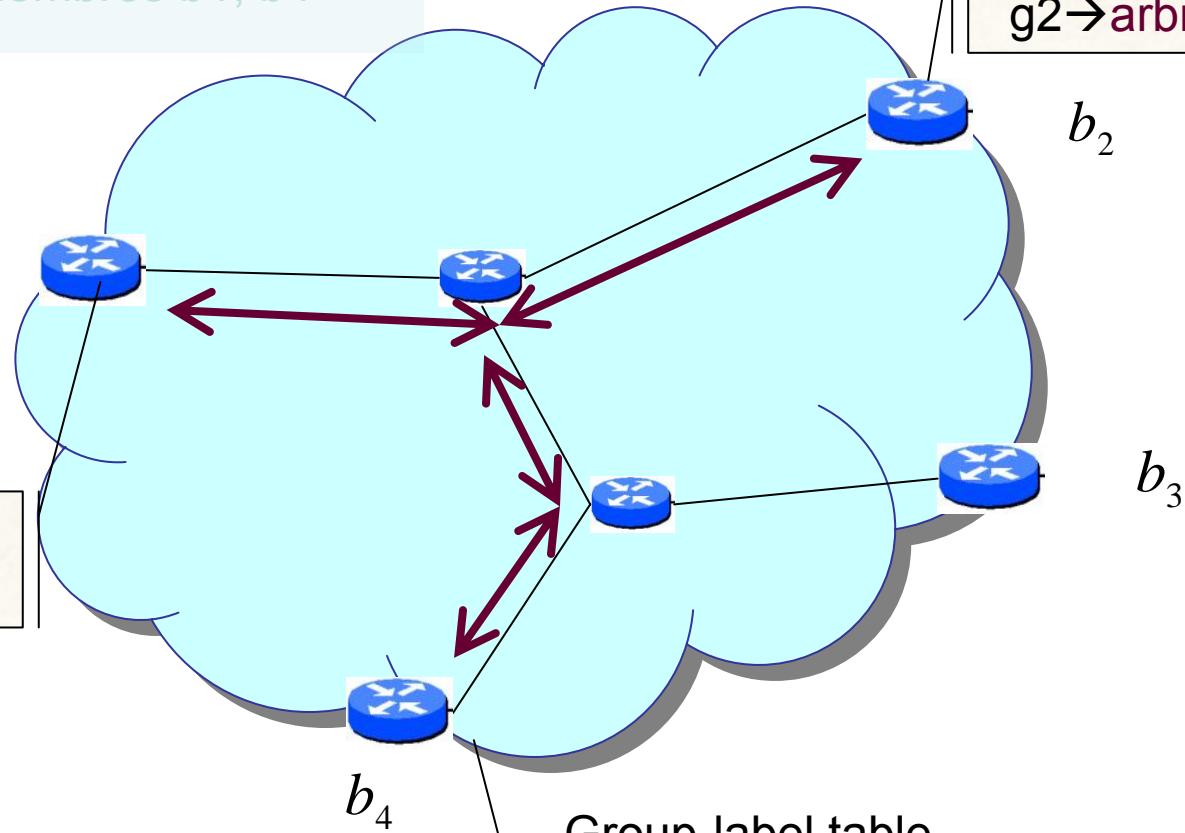
Groupe g3 avec membres b1, b4

Group-label table

g1 → arbre rouge  
g2 → arbre rouge

Group-label table

g1 → arbre rouge  
g2 → arbre rouge



Group-label table

g1 → arbre rouge  
g2 → arbre rouge

# L'agrégation d'arbres Leaky Match

Groupe g1 avec membres b1, b2, b4

Groupe g2 avec membres b1, b2, b4

Groupe g3 avec membres b1, b4

Group-label table

g1-->arbre rouge  
g2-->arbre rouge

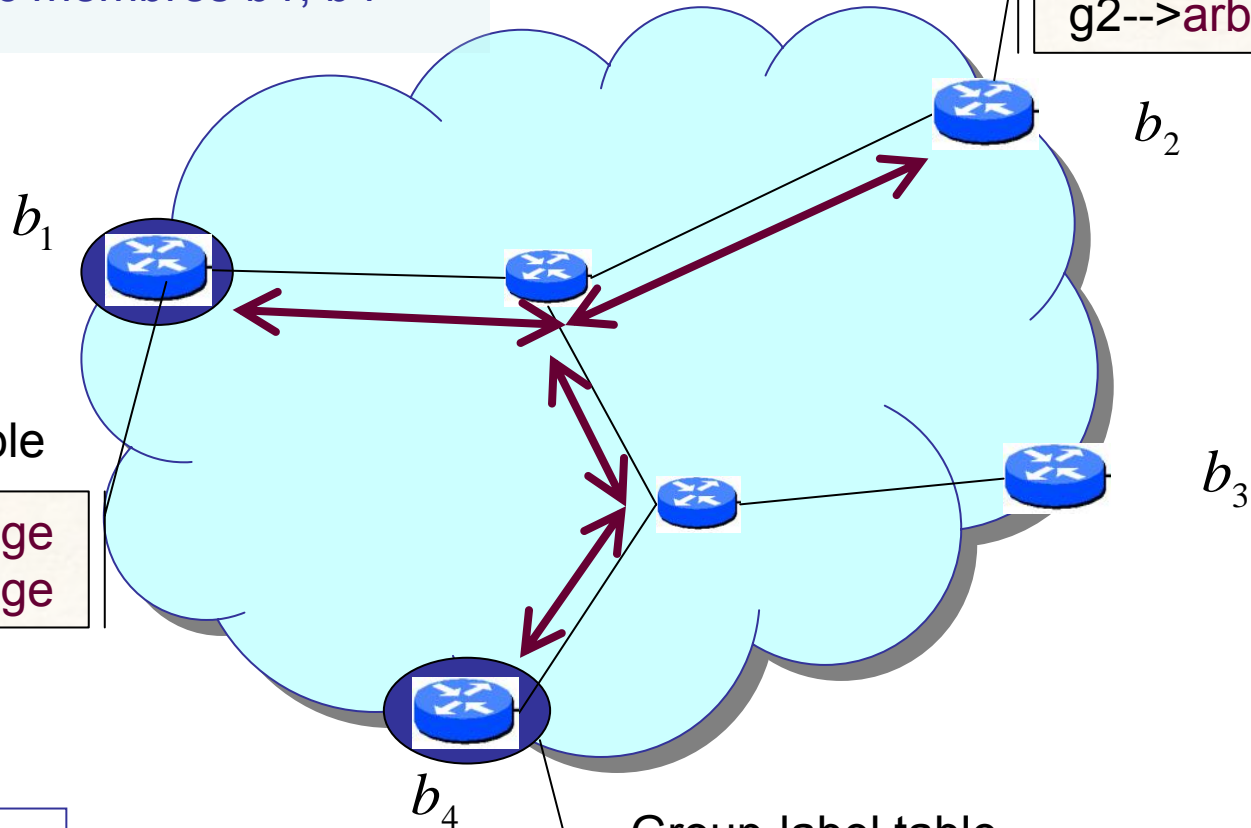
Group-label table

g1-->arbre rouge  
g2-->arbre rouge

Un nouveau  
groupe g3 arrive.

Group-label table

g1-->arbre rouge  
g2-->arbre rouge



# L'agrégation d'arbres Leaky Match

Groupe g1 avec membres b1, b2, b4

Groupe g2 avec membres b1, b2, b4

Groupe g3 avec membres b1, b4

Group-label table

g1-->arbre rouge  
g2-->arbre rouge

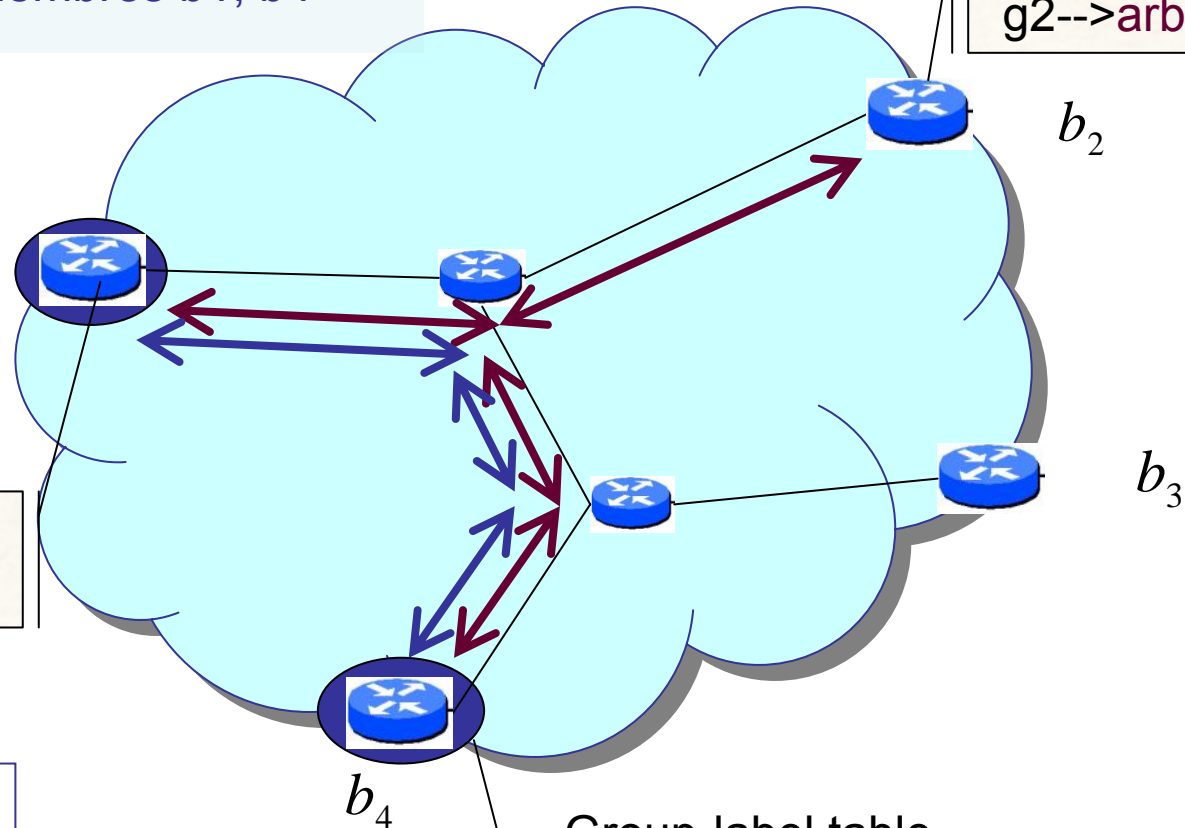
Group-label table

g1-->arbre rouge  
g2-->arbre rouge

On peut  
construire un  
arbre couvrant  
g3

Group-label table

g1-->arbre rouge  
g2-->arbre rouge



# L'agrégation d'arbres Leaky Match

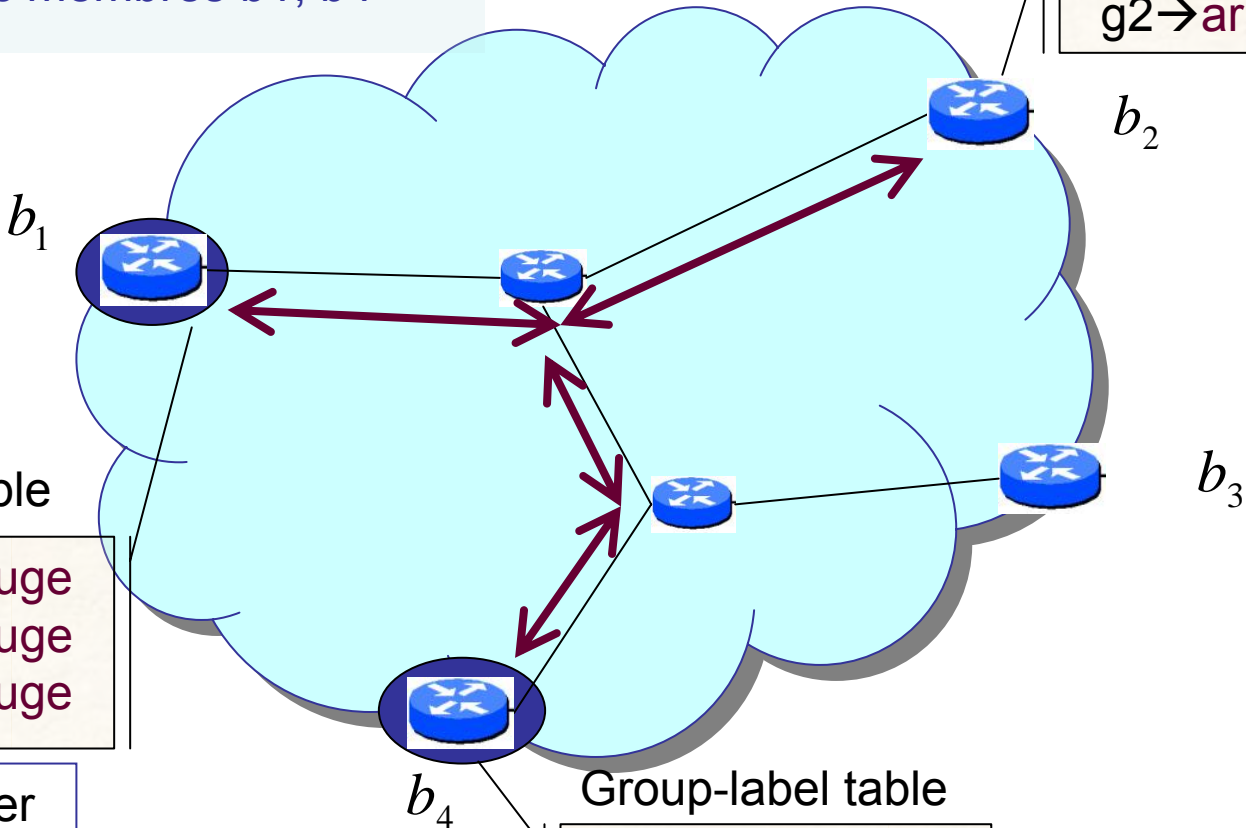
Groupe g1 avec membres b1, b2, b4

Groupe g2 avec membres b1, b2, b4

Groupe g3 avec membres b1, b4

Group-label table

g1 → arbre rouge  
g2 → arbre rouge



Group-label table

g1 → arbre rouge  
g2 → arbre rouge  
g3 → arbre rouge

Ou bien agréger  
g3 à l'arbre  
rouge, qui est  
plus gros.

Group-label table

g1 → arbre rouge  
g2 → arbre rouge  
g3 → arbre rouge

# L'agrégation d'arbres Leaky Match

- Réduction du nombre d'entrées de routage.
- Perte de bande passante car des routeurs non destinataires reçoivent des messages.

Compromis entre  
la réduction du nombre d'entrées de routage et  
la bande passante gaspillée.

# II- Notre Algorithme STA Scalable Tree Aggregation

# Notre algorithme STA

- Pour gérer 40 000 requêtes de groupes, l'algorithme AM prend plus de 2h.

→ Problèmes de passage à l'échelle

- Nous proposons un nouvel algorithme avec une agrégation plus rapide

# Notre algorithme STA

**Données** : Groupe  $g$ , seuil  $t_b$ ,  $MTS = \{MTS_2 = \{t_1, t_2, t_3, \dots\}, MTS_3 = \{t_1, t_2, t_3\}, MTS_4 = \{t_1, t_2, t_3\}, \dots\}$

**Résultat** : Un arbre pour  $g$



# Notre algorithme STA

**Données** : Groupe  $g$ , seuil  $t_b$ ,  $MTS = \{MTS_2 = \{t_1, t_2, t_3, \dots\}, MTS_3 = \{t_1, t_2, t_3\}, MTS_4 = \{t_1, t_2, t_3\}, \dots\}$

**Résultat** : Un arbre pour  $g$

For each new group  $g$ :

- ◆ Compute a native tree  $t_g$  for  $g$  of cost  $c_g$



# Notre algorithme STA

**Données** : Groupe  $g$ , seuil  $t_b$ ,  $MTS = \{MTS_2 = \{t_1, t_2, t_3, \dots\}, MTS_3 = \{t_1, t_2, t_3\}, MTS_4 = \{t_1, t_2, t_3\}, \dots\}$

**Résultat** : Un arbre pour  $g$

For each new group  $g$ :

- ◆ Compute a native tree  $t_g$  for  $g$  of cost  $c_g$
- ◆ Trees are evaluated **from** set  $MTS(c_g)$  **to** set  $MTS(c_g * t_b)$  in increasing order of their costs
  - 
  -

# Notre algorithme STA

**Données** : Groupe  $g$ , seuil  $t_b$ ,  $MTS = \{MTS_2 = \{t_1, t_2, t_3, \dots\}, MTS_3 = \{t_1, t_2, t_3\}, MTS_4 = \{t_1, t_2, t_3\}, \dots\}$

**Résultat** : Un arbre pour  $g$

For each new group  $g$ :

- ◆ Compute a native tree  $t_g$  for  $g$  of cost  $c_g$
- ◆ Trees are evaluated **from** set  $MTS(c_g)$  **to** set  $MTS(c_g * t_b)$  in increasing order of their costs
  - $t$  is evaluated
  - **If**  $t$  can cover  $g$  **then**  $t$  is chosen

# Notre algorithme STA

**Données** : Groupe  $g$ , seuil  $t_b$ ,  $MTS = \{MTS_2 = \{t_1, t_2, t_3, \dots\}, MTS_3 = \{t_1, t_2, t_3\}, MTS_4 = \{t_1, t_2, t_3\}, \dots\}$

**Résultat** : Un arbre pour  $g$

For each new group  $g$ :

- ◆ Compute a native tree  $t_g$  for  $g$  of cost  $c_g$
- ◆ Trees are evaluated **from** set  $MTS(c_g)$  **to** set  $MTS(c_g * t_b)$  in increasing order of their costs
  - $t$  is evaluated
  - **If**  $t$  can cover  $g$  **then**  $t$  is chosen

If all the trees in the subsets are evaluated **and** if no tree has been chosen **then** add  $t_g$  in  $MTS(c_g)$

Entrée  $g \rightarrow$  label( $t$ ) dans les tables groupes-label des routeurs de bordures

# Notre algorithme STA

**Données** : Groupe  $g$ , seuil  $t_b$ ,  $MTS = \{MTS_2 = \{t_1, t_2, t_3, \dots\}, MTS_3 = \{t_1, t_2, t_3\}, MTS_4 = \{t_1, t_2, t_3\}, \dots\}$

**Résultat** : Un arbre pour  $g$

For each new group  $g$ :

Seul un sous-ensemble d'arbres est évalué

- ◆ Trees are evaluated **from** set  $MTS(c_g)$  **to** set  $MTS(c_g * t_b)$  in increasing order of their costs
  - $t$  is evaluated
  - **If**  $t$  can cover  $g$  **then**  $t$  is chosen

If all the trees in the subsets are evaluated **and** if no tree has been chosen **then** add  $t_g$  in  $MTS(c_g)$

Entrée  $g \rightarrow \text{label}(t)$  dans les tables groupes-label des routeurs de bordures

# Notre algorithme STA

**Données** : Groupe  $g$ , seuil  $t_b$ ,  $MTS = \{MTS_2 = \{t_1, t_2, t_3, \dots\}, MTS_3 = \{t_1, t_2, t_3\}, MTS_4 = \{t_1, t_2, t_3\}, \dots\}$

**Résultat** : Un arbre pour  $g$

For each new group  $g$ :

Seul un sous-ensemble d'arbres est évalué

Fonction de sélection plus simple. Utilisation de bitmaps.

- If  $t$  can cover  $g$  then  $t$  is chosen

If all the trees in the subsets are evaluated and if no tree has been chosen then add  $t_g$  in  $MTS(c_g)$

Entrée  $g \rightarrow \text{label}(t)$  dans les tables groupes-label des routeurs de bordures

# Notre algorithme STA

**Données** : Groupe  $g$ , seuil  $t_b$ ,  $MTS = \{MTS_2 = \{t_1, t_2, t_3, \dots\}, MTS_3 = \{t_1, t_2, t_3\}, MTS_4 = \{t_1, t_2, t_3\}, \dots\}$

**Résultat** : Un arbre pour  $g$

For each new group  $g$ :

Seul un sous-ensemble d'arbres est évalué

Fonction de sélection plus simple. Utilisation de bitmaps.

$MTS(c_g)$  to set  $MTS(c_g * t_b)$  in increasing order of

Dès qu'un arbre est trouvé, l'algorithme s'arrête

- If  $t$  can cover  $g$  then  $t$  is chosen

If all the trees in the subsets are evaluated and if no tree has been chosen then add  $t_g$  in  $MTS(c_g)$

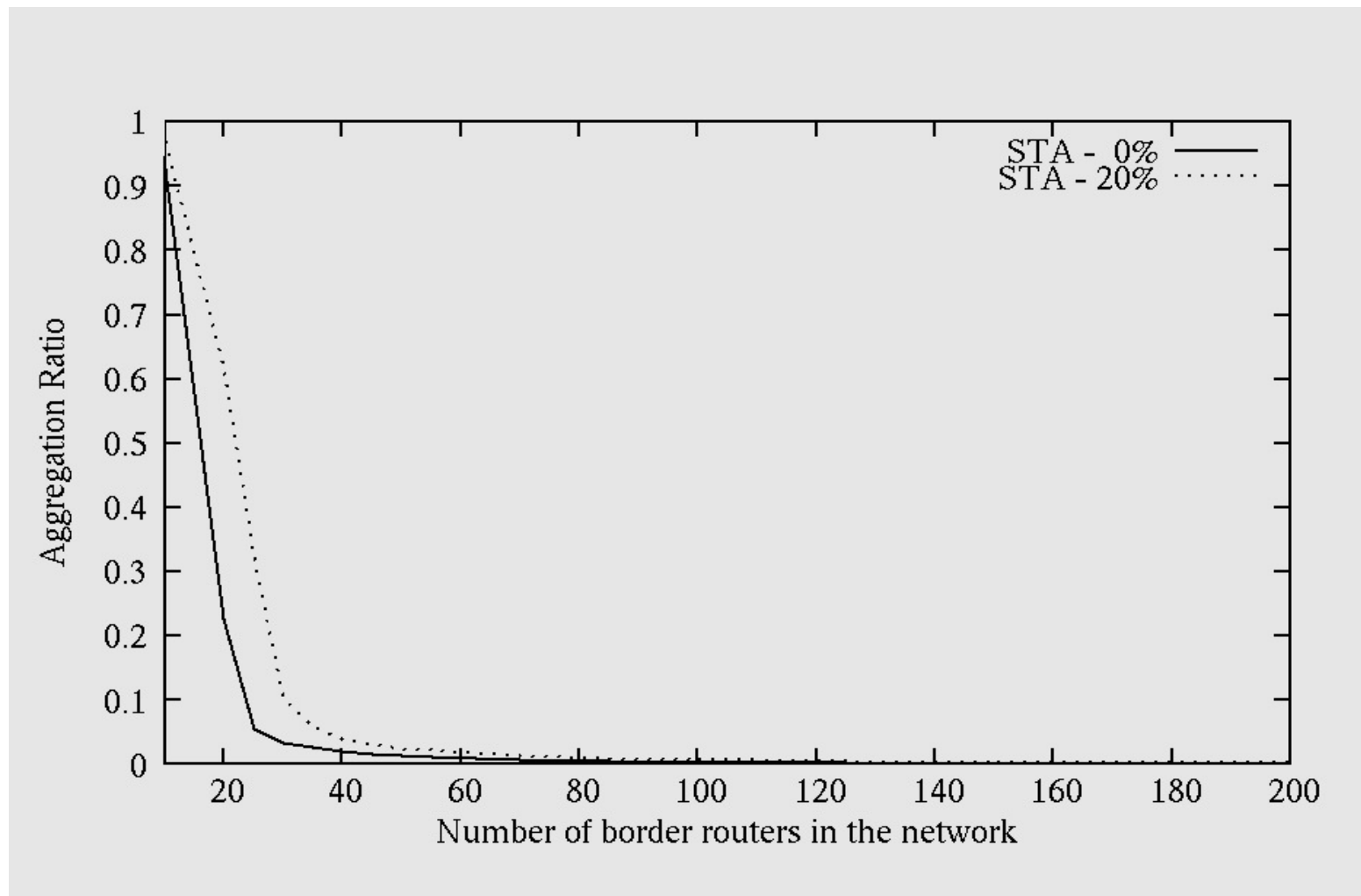
Entrée  $g \rightarrow \text{label}(t)$  dans les tables groupes-label des routeurs de bordures

# Résultats de simulation

- STA a les mêmes performances que AM
- STA est 4 fois plus rapide que AM
  - ◆ (10 ms/groupe contre 45 ms)

# III- L'agrégation d'arbres dans les grands domaines : TALD

# Taux d'agrégation



Pour 10 000 groupes concurrents

# Le contexte

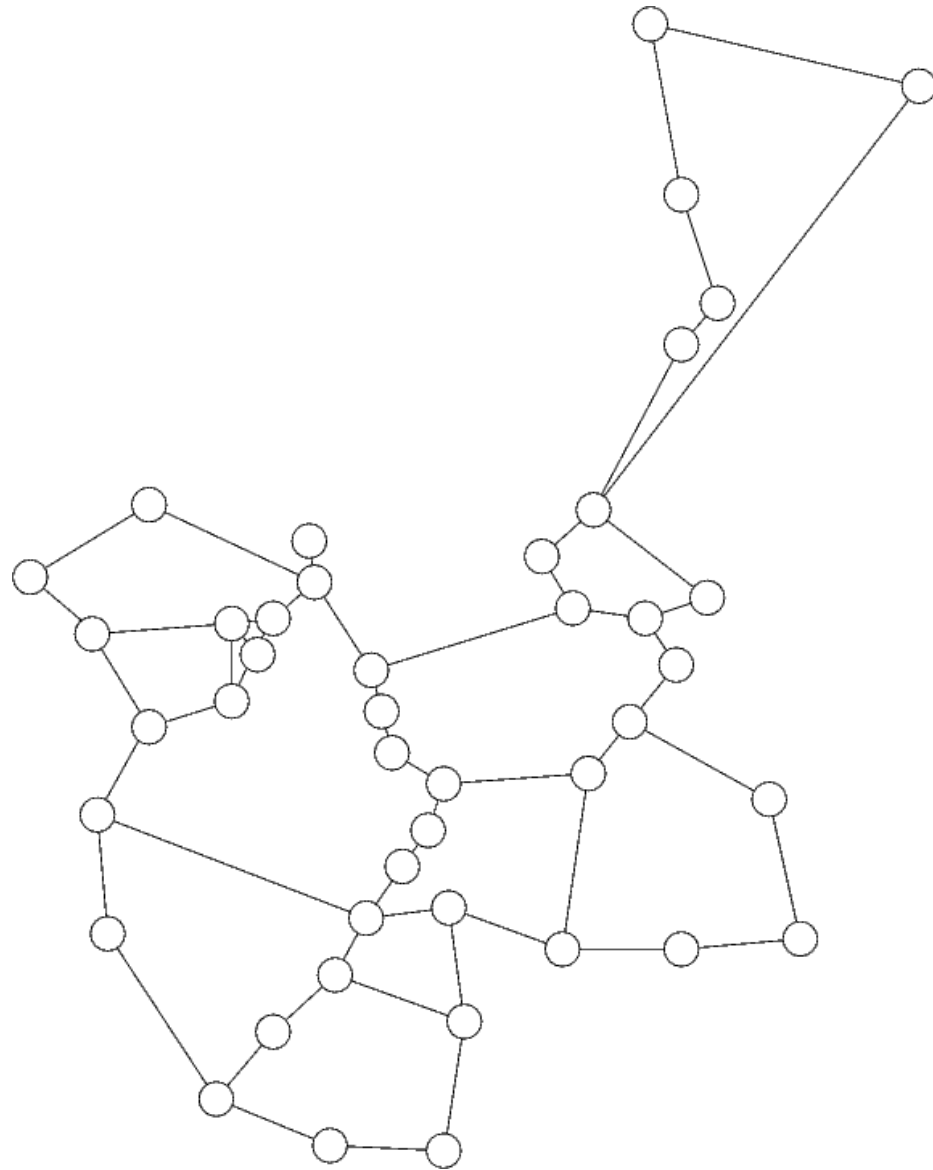
- Taux d'agrégation très faible quand le nombre de routeurs de bordure est grand
- L'agrégation d'arbres  $\approx$  Multicast dans ce cas-là

Il faut trouver un nouvel algorithme

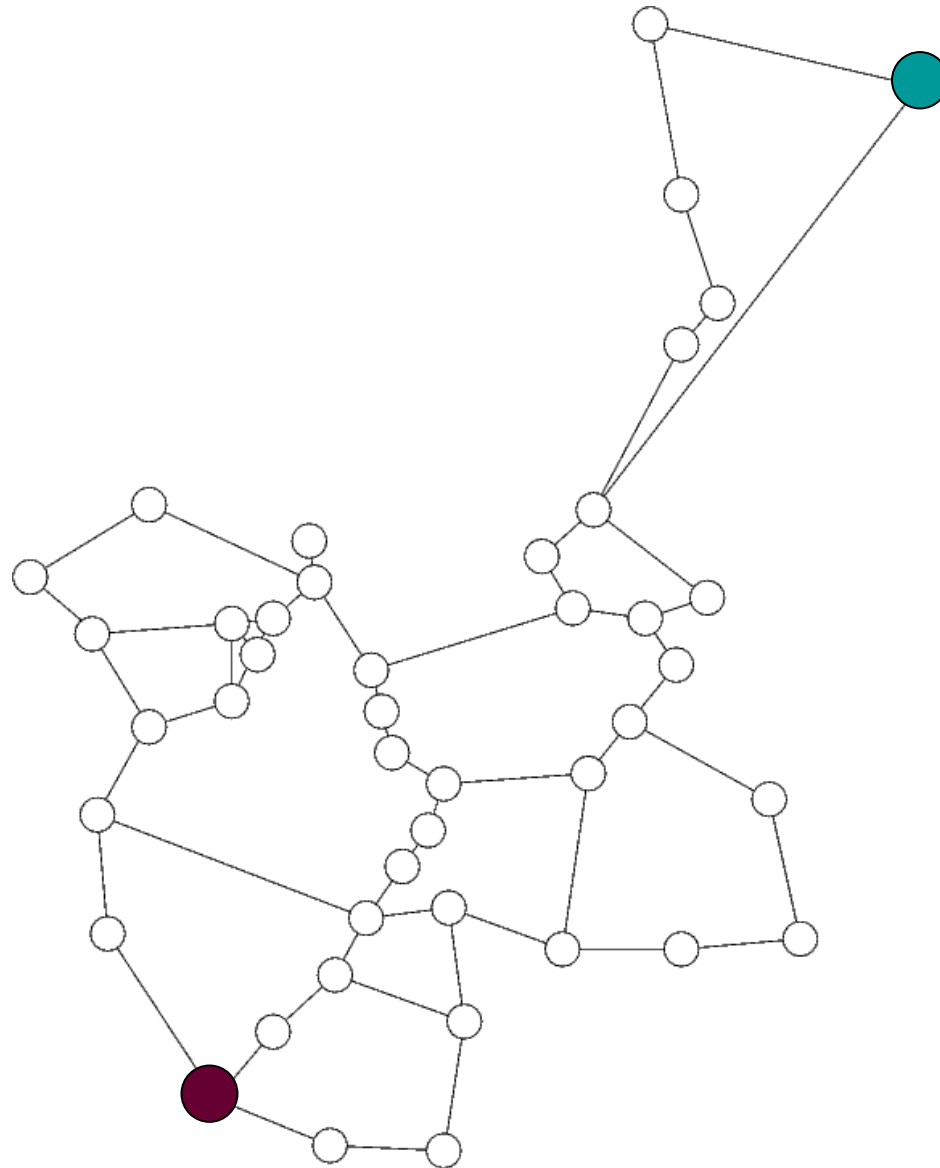
# L'agrégation dans les grands domaines

- Découper le réseau en sous-domaines
  
- Agréger les groupes séparément dans les sous-domaines.

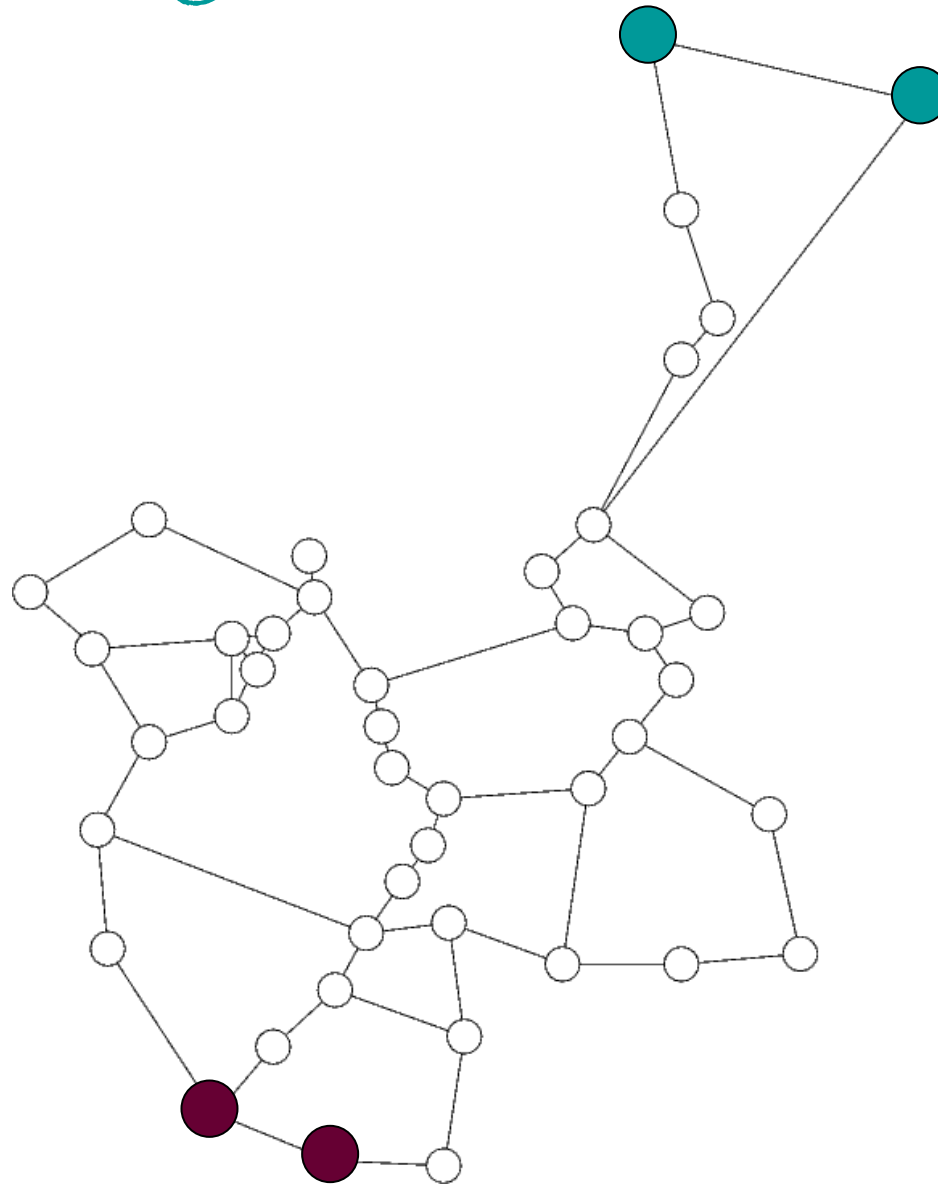
# Réseau Eurorings



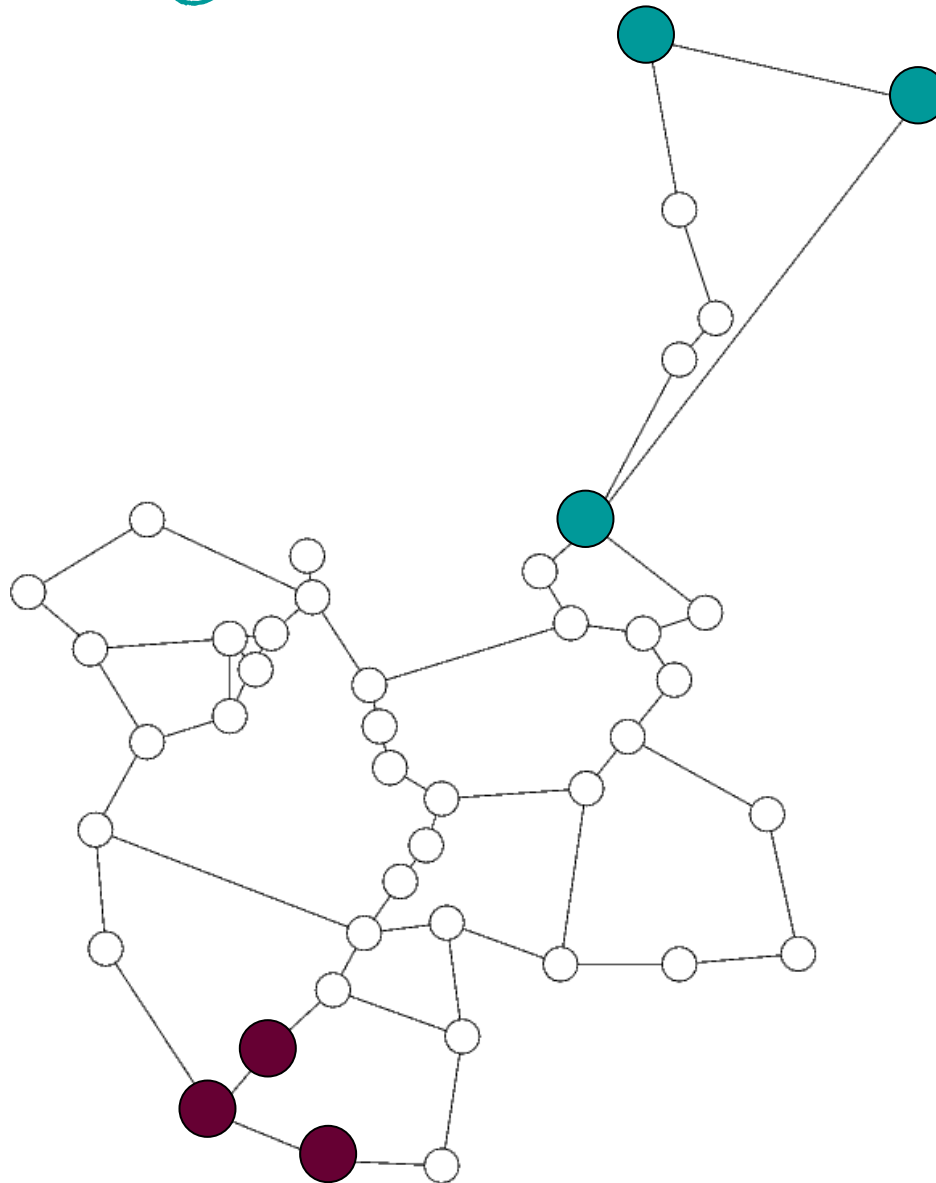
# Réseau Eurorings



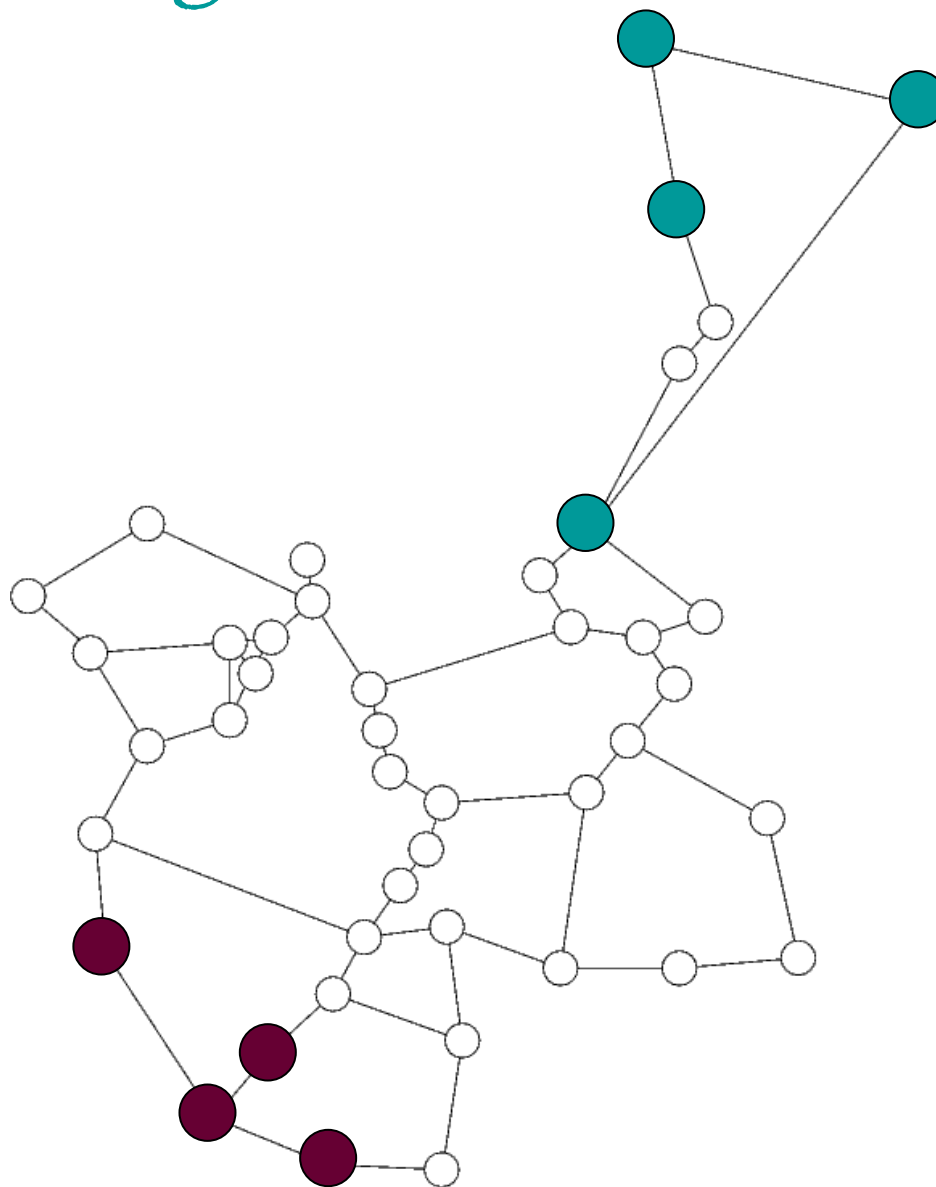
# Réseau Eurorings



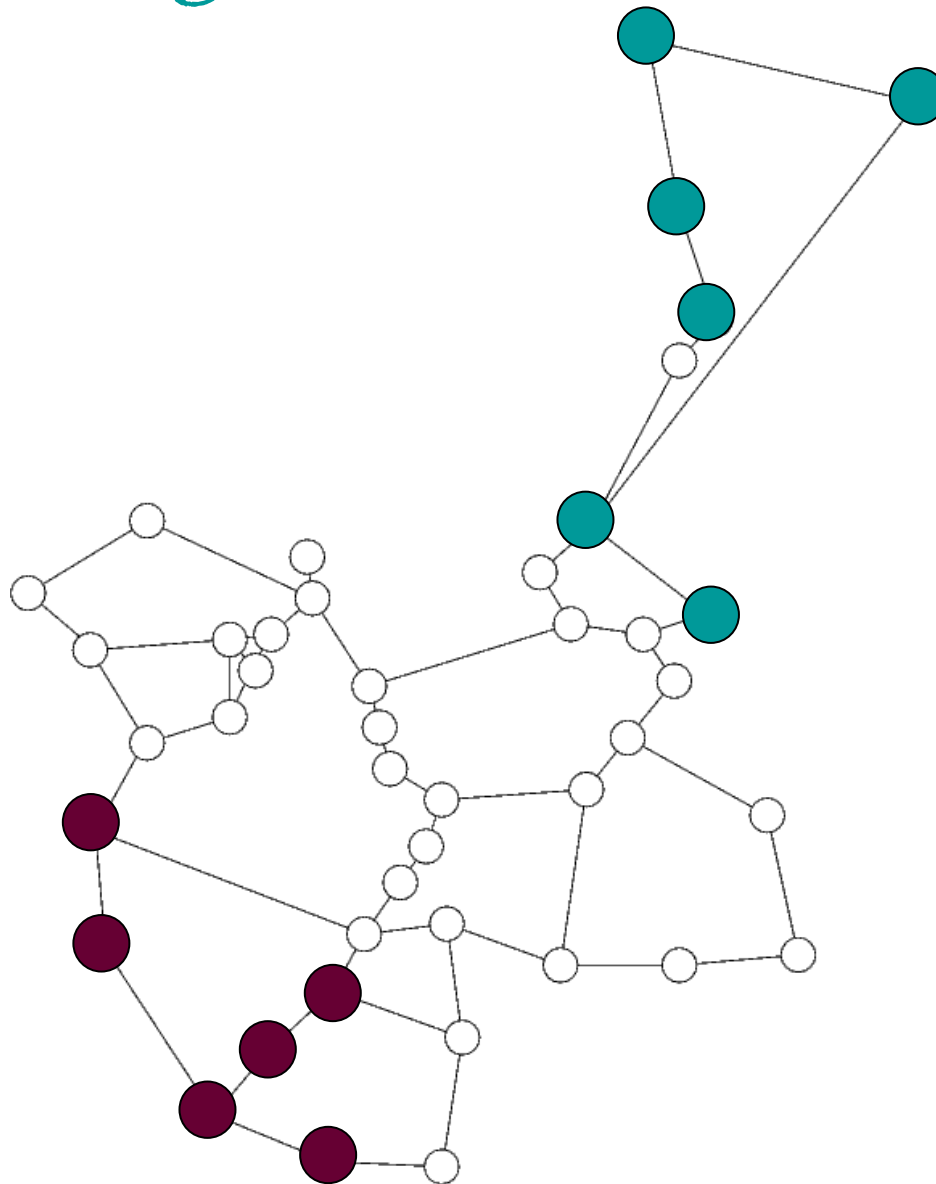
# Réseau Eurorings



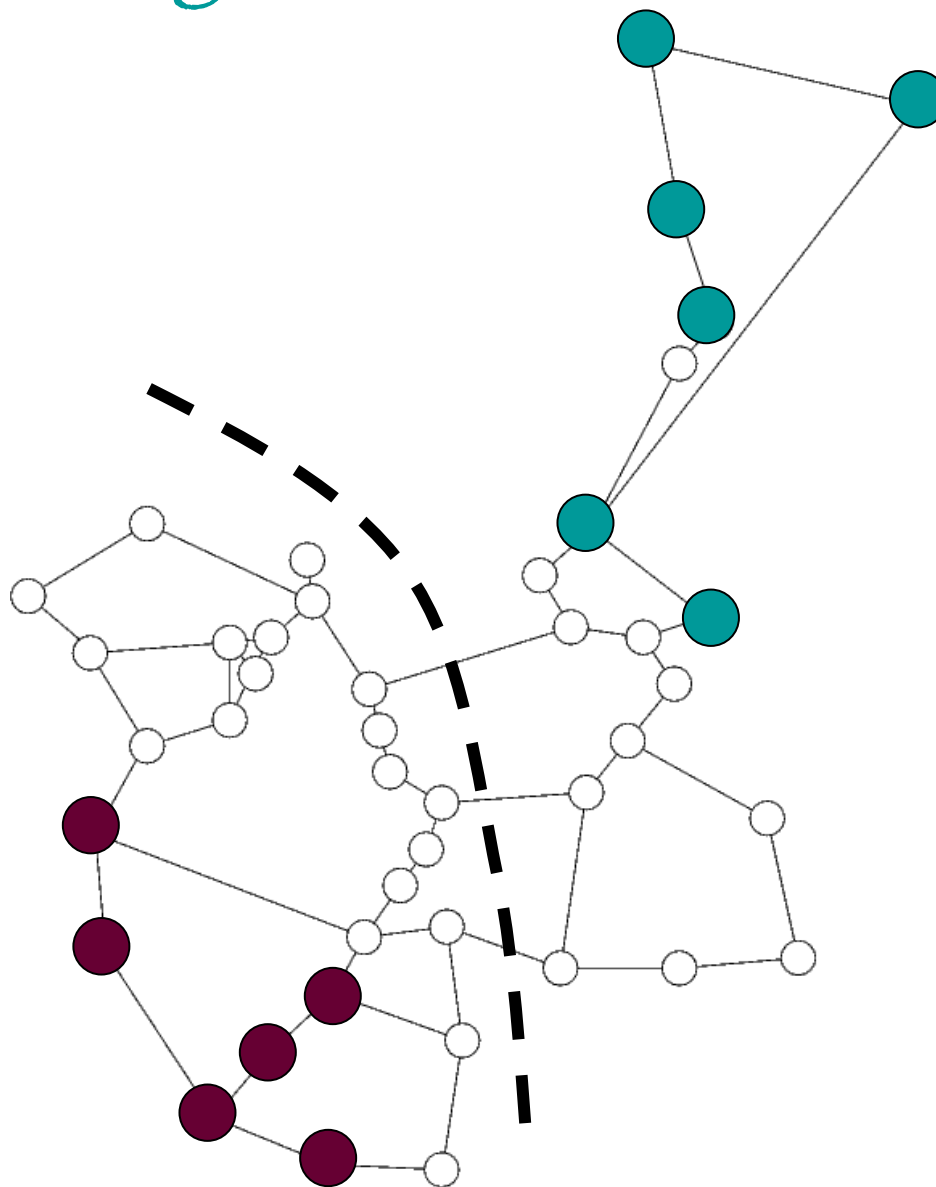
# Réseau Eurorings



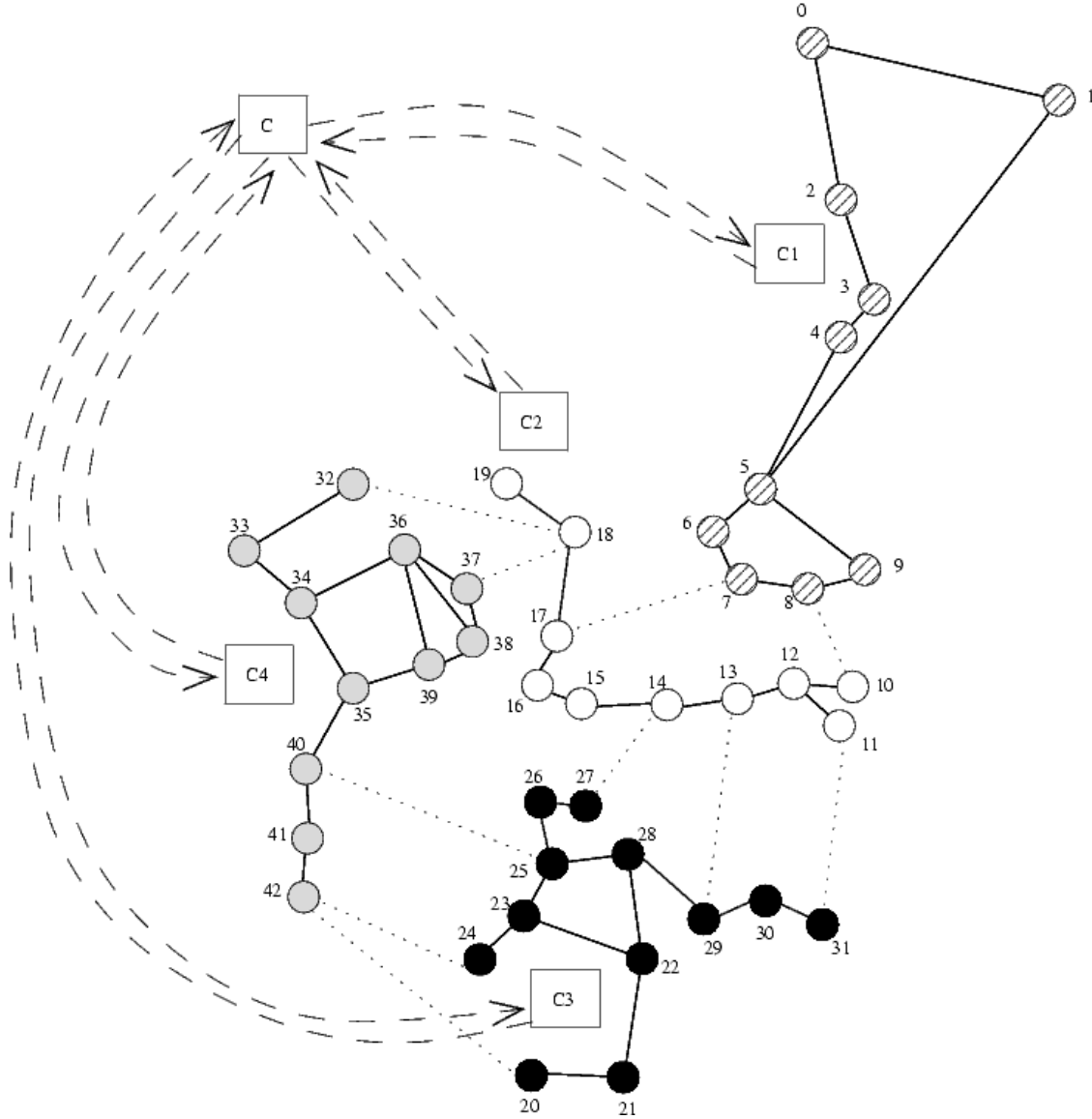
# Réseau Eurorings



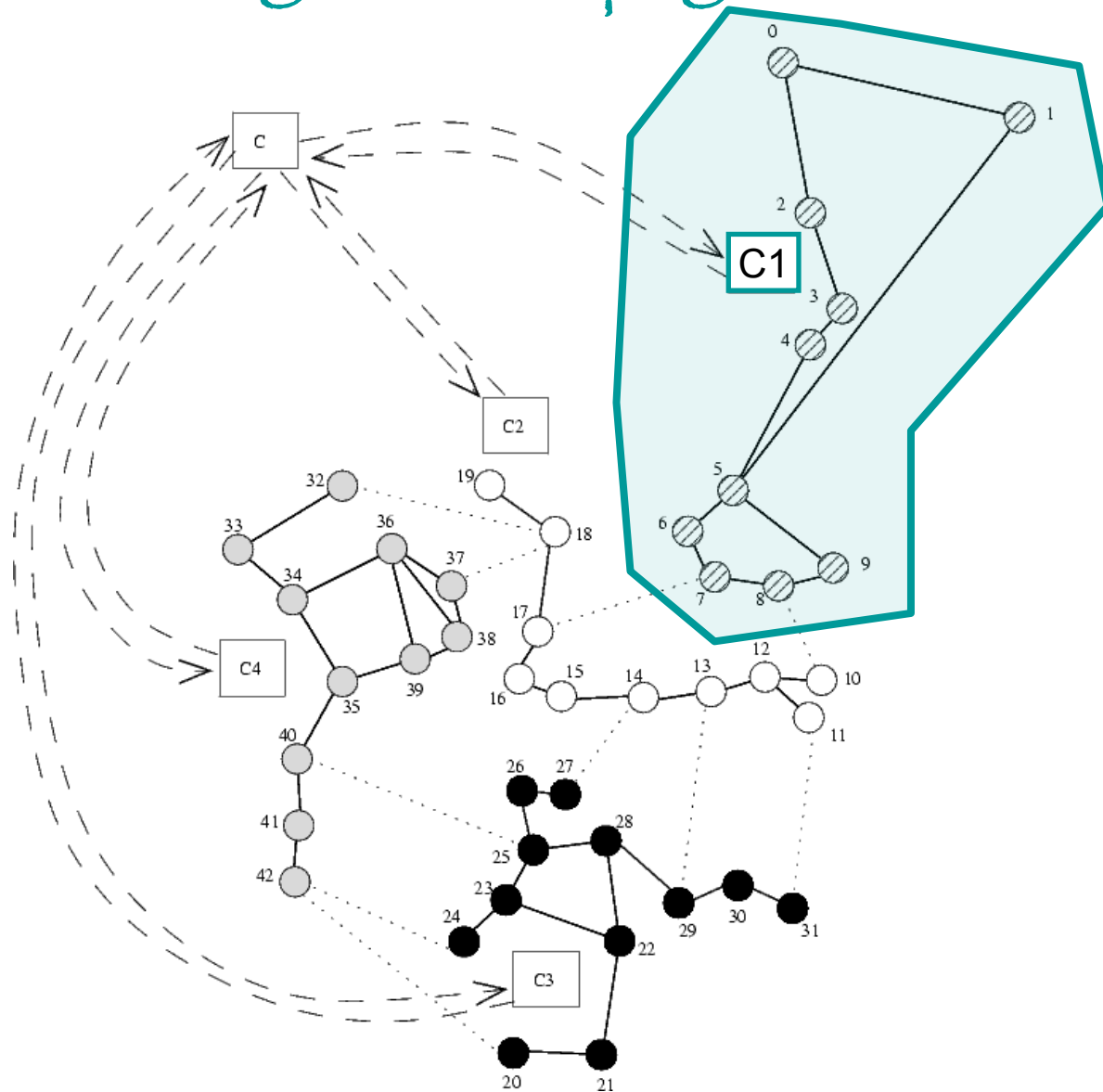
# Réseau Eurorings



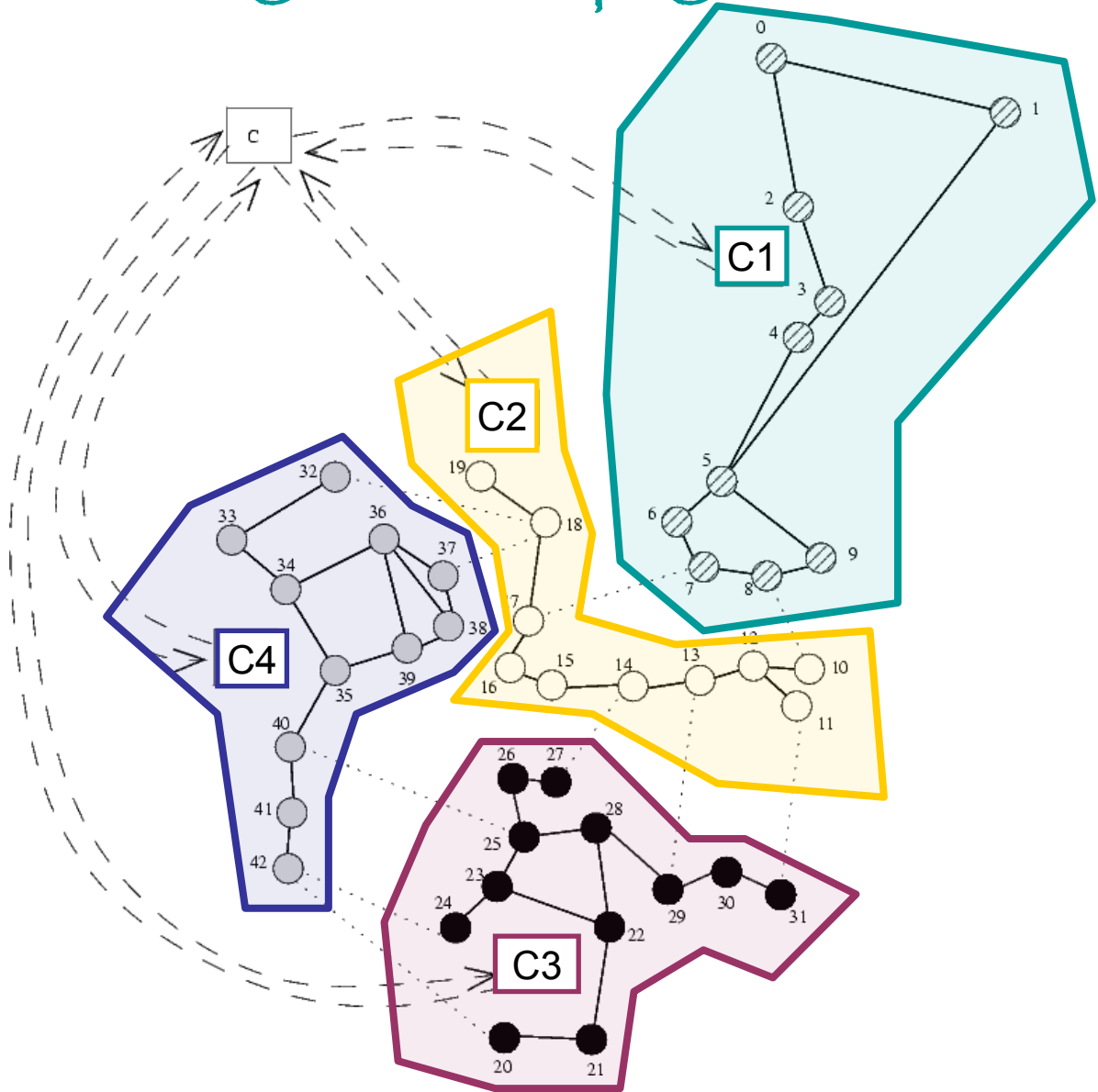
# Réseau Eurorings : découpage



# Réseau Eurorings : découpage



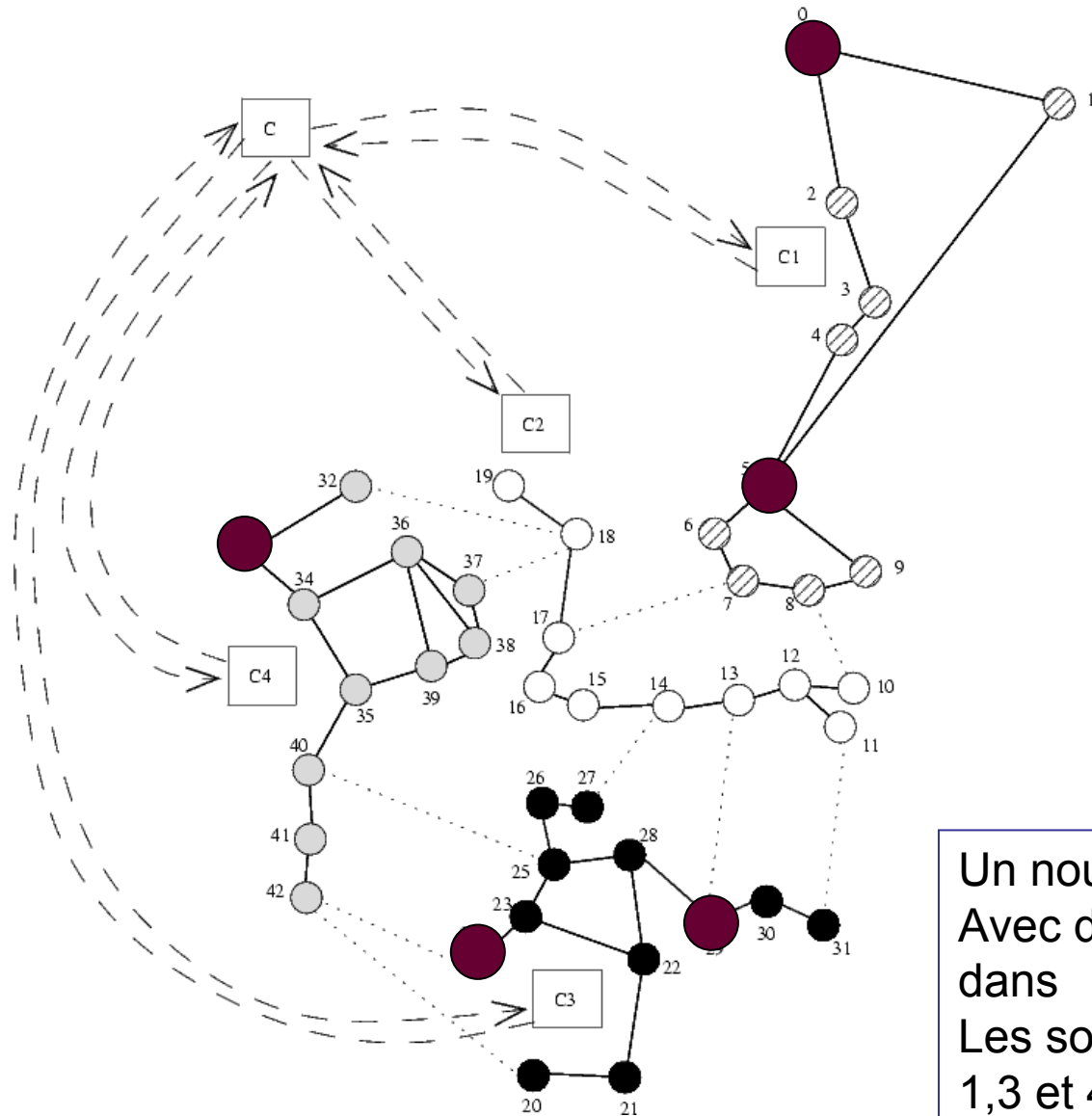
# Réseau Eurorings : découpage



# Algorithme TALD

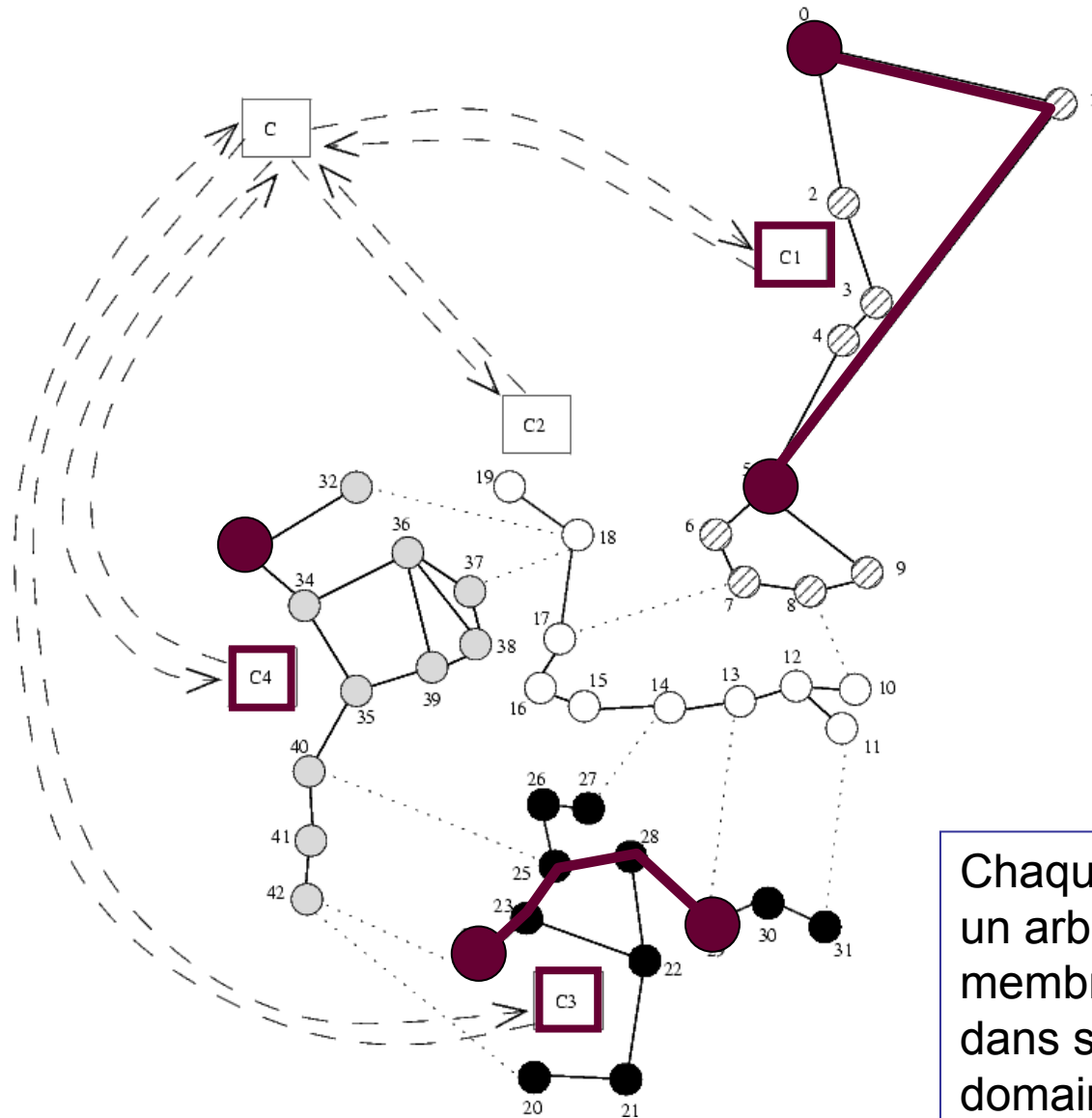
- Chaque domaine  $i$  est géré par une entité centrale  $C_i$
- L'agrégation est réalisée séparément dans chacun des sous-domaines par les  $C_i$
- L'entité centrale  $C$ , responsable du domaine entier, relie les arbres dans chacun des sous-domaines par des tunnels

# Algorithme TALD



Un nouveau groupe  $g$   
Avec des membres  
dans  
Les sous-domaines  
1,3 et 4

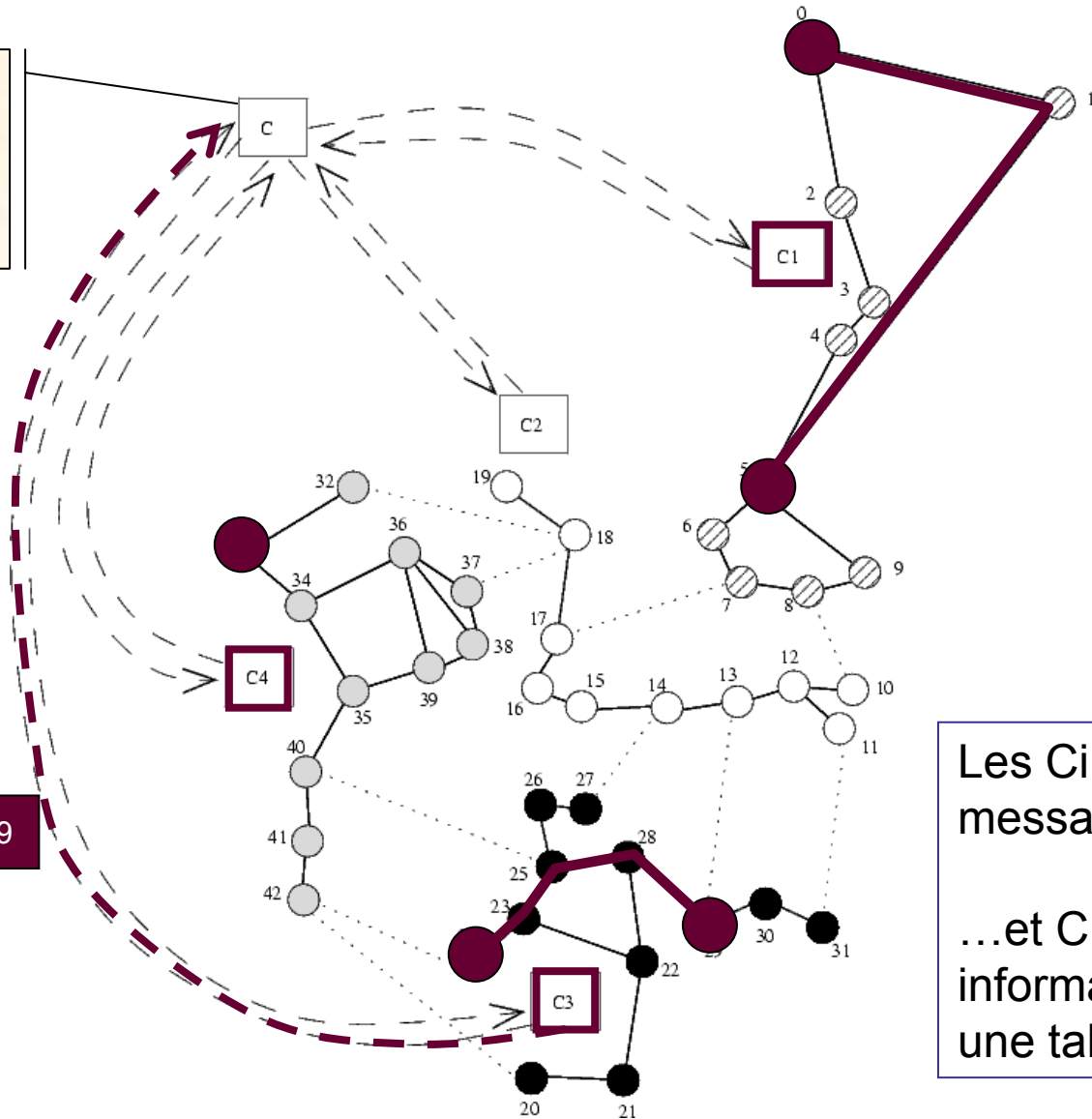
# Algorithme TALD



Chaque  $C_i$  construit un arbre couvrant les membres présents dans son sous-domaine.

# Algorithme TALD

Groupe g : C3, @IP29



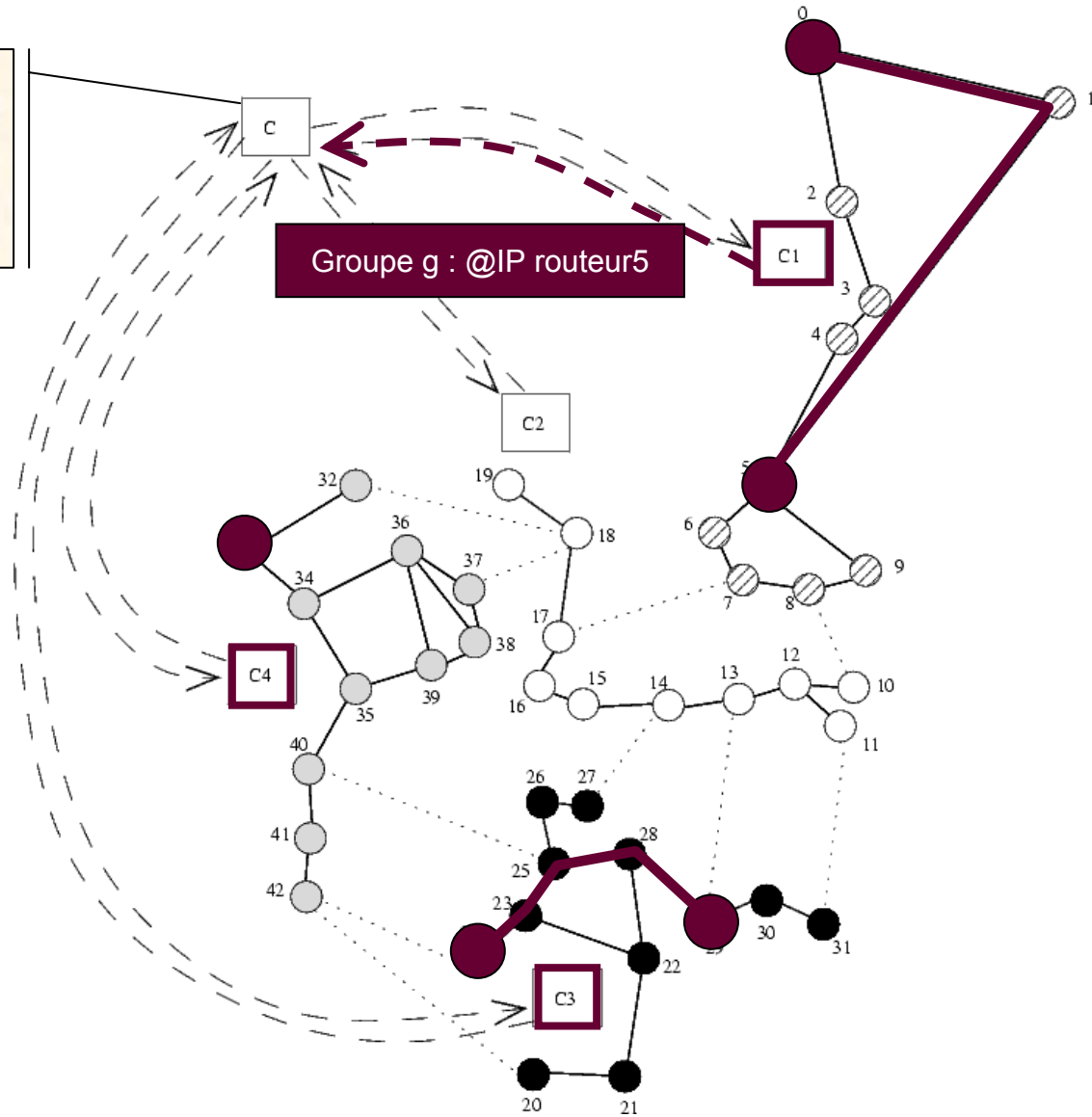
Groupe g : @IP routeur29

Les Ci envoient un message à C

...et C stocke ces informations dans une table

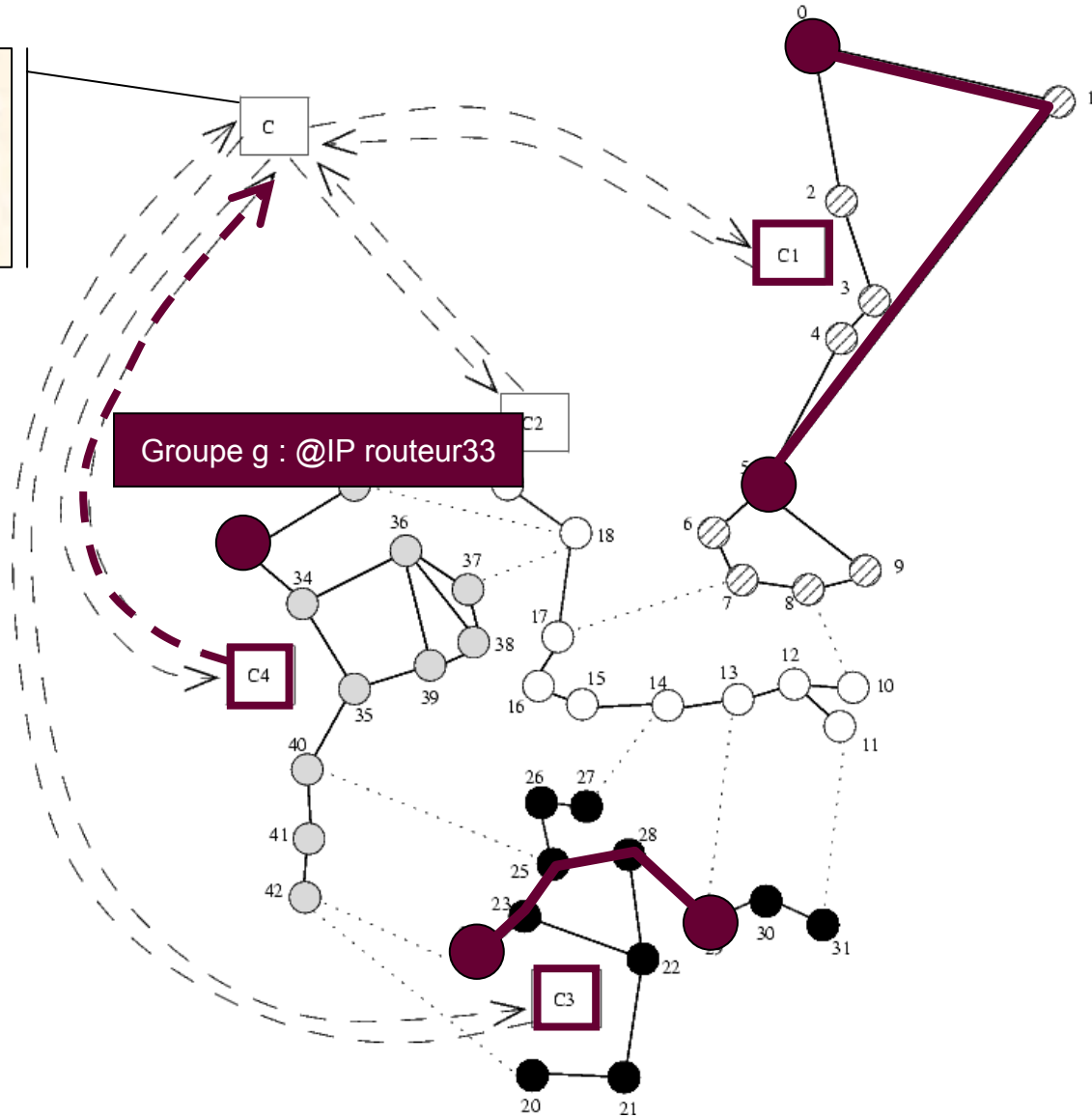
# Algorithme TALD

Groupe g : C3, @IP29  
C1, @IP5



# Algorithme TALD

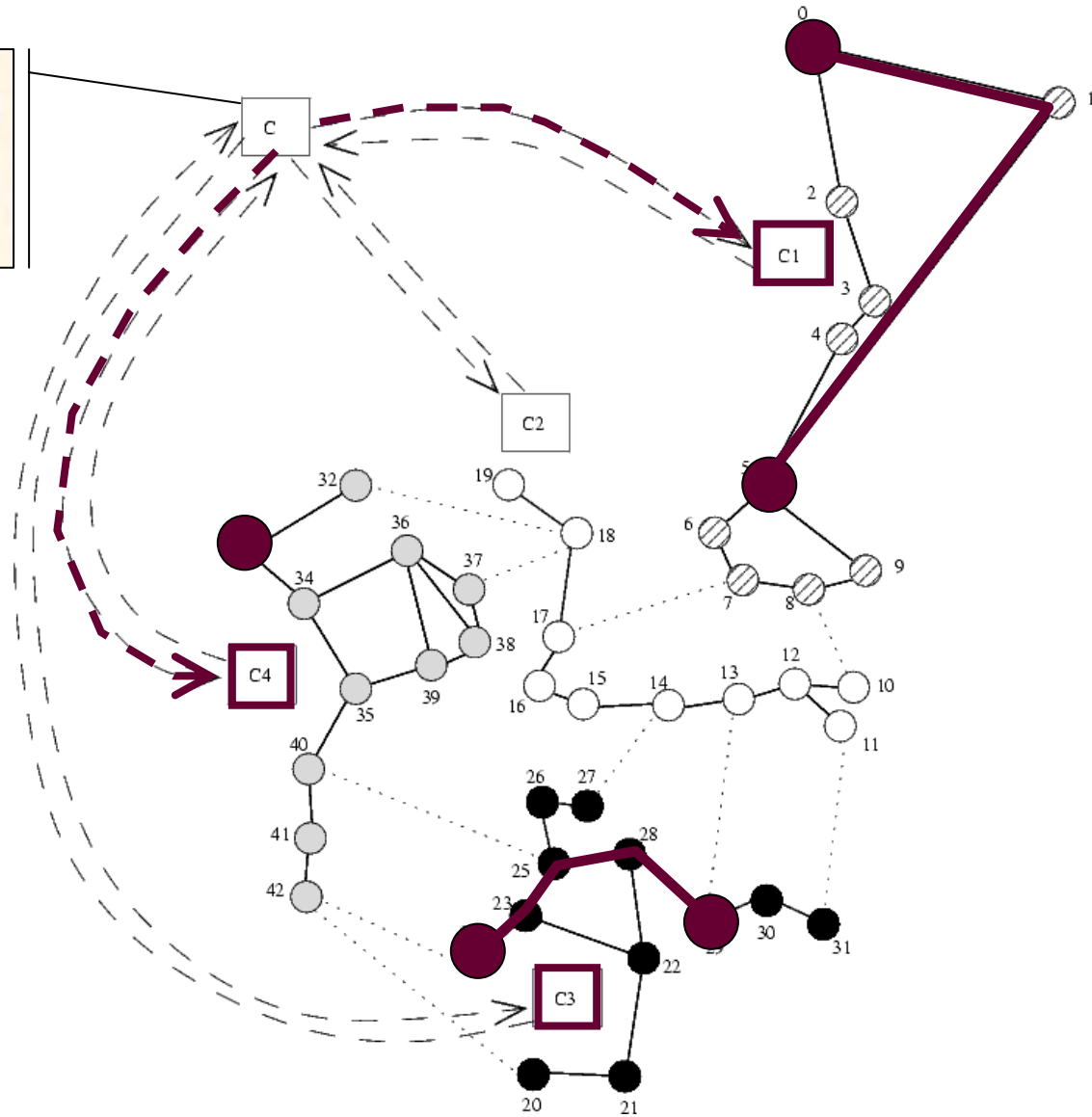
Groupe g : C3, @IP29  
C1, @IP5  
C4, @IP33





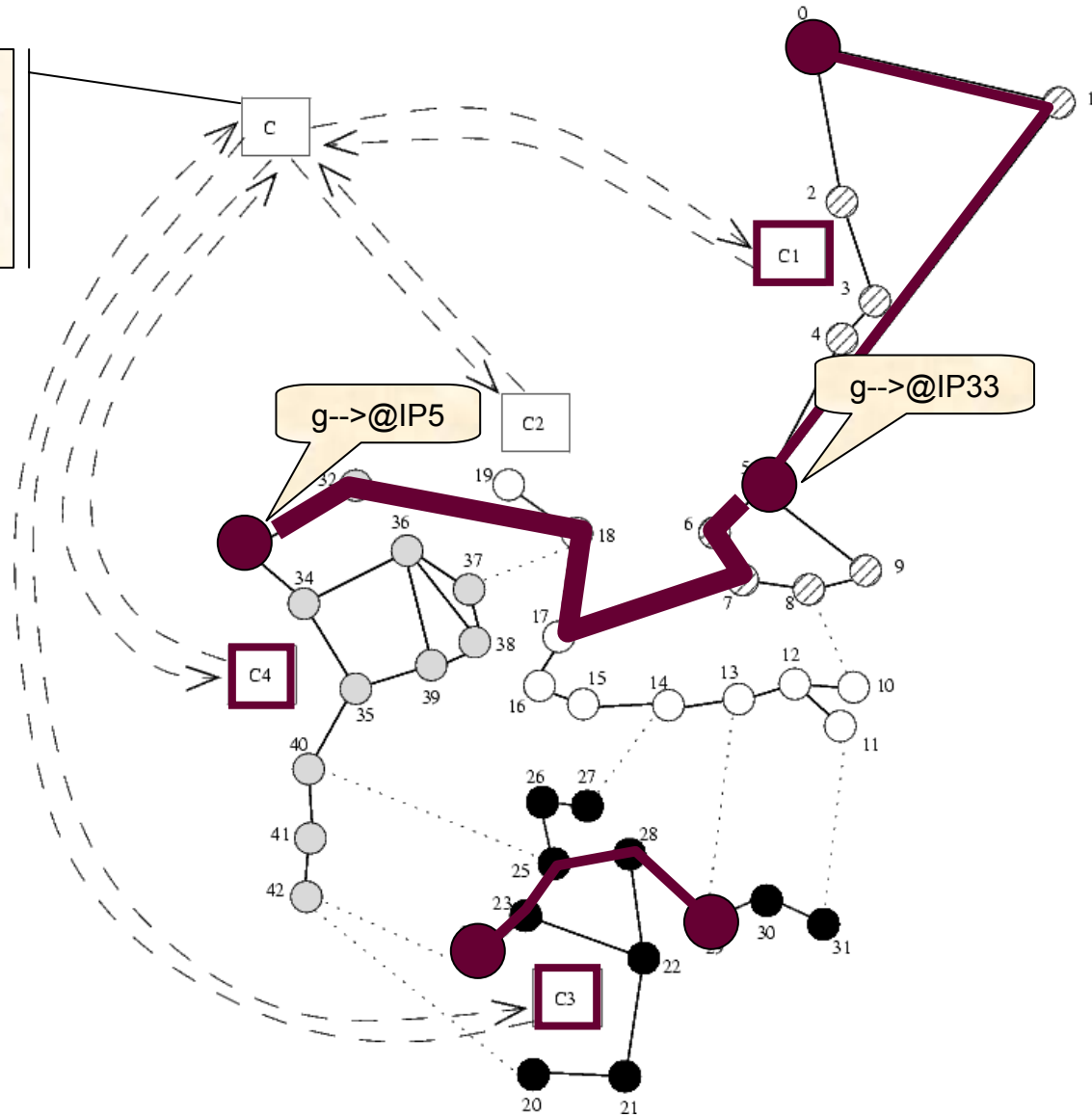
# Algorithme TALD

Groupe g : C3, @IP29  
C1, @IP5  
C4, @IP33



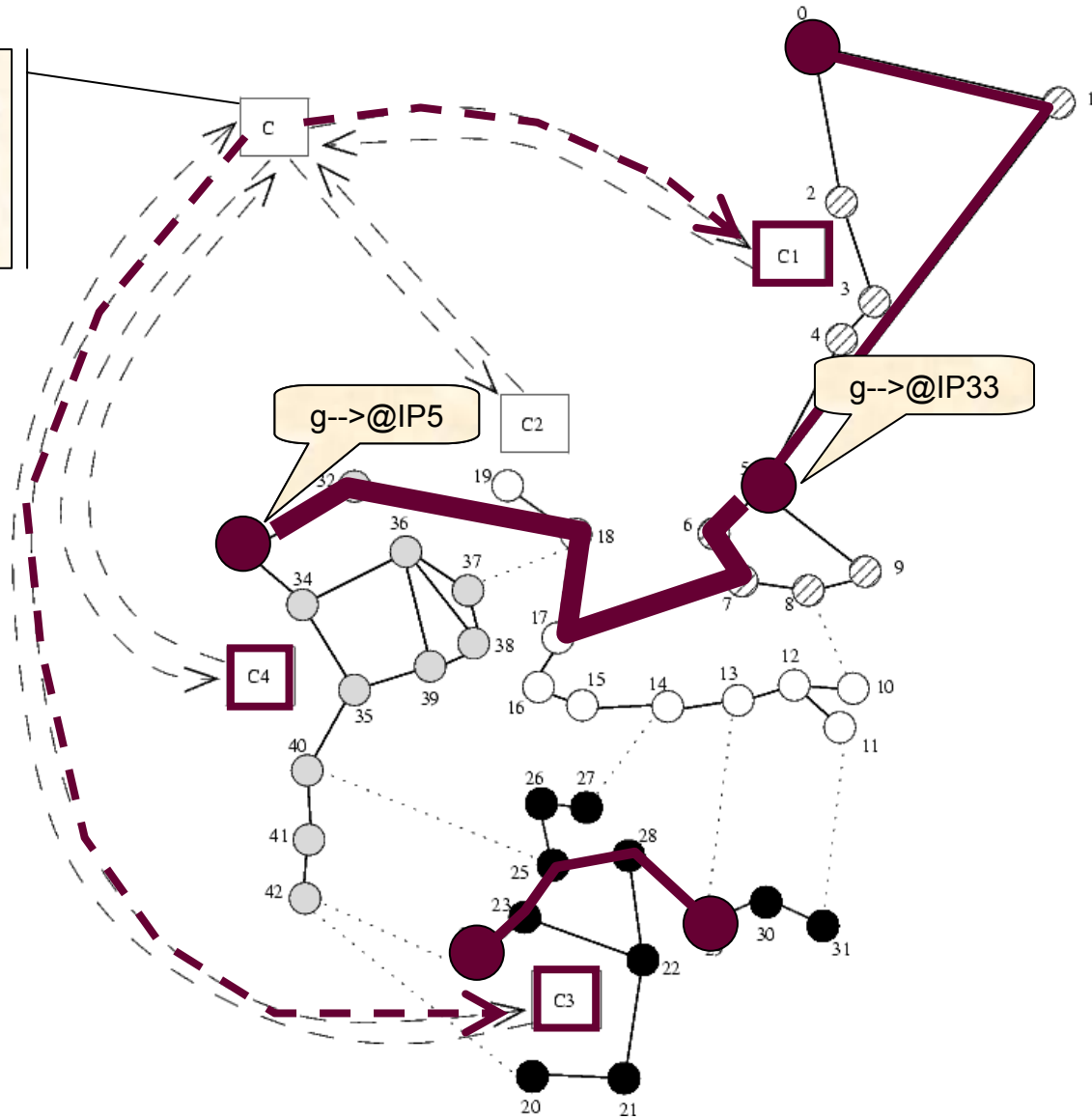
# Algorithme TALD

Groupe g : C3, @IP29  
C1, @IP5  
C4, @IP33



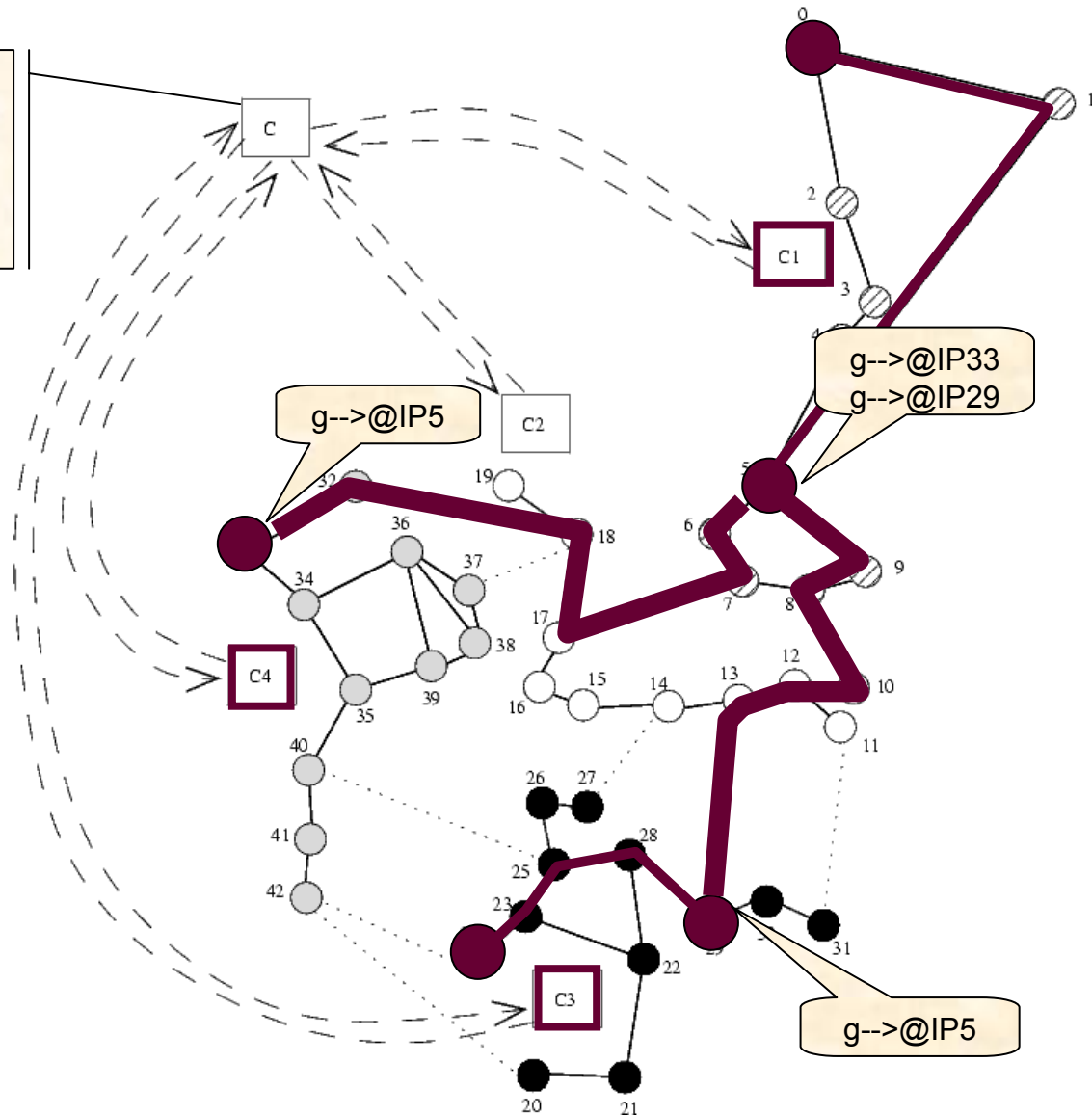
# Algorithme TALD

Groupe g : C3, @IP29  
C1, @IP5  
C4, @IP33



# Algorithme TALD

Groupe g : C3, @IP29  
C1, @IP5  
C4, @IP33



# Le protocole TALD

- Le domaine est découpé en plusieurs sous-domaines
- Ensuite, agrégation dans chacun des sous-domaines
- Finalement, routage dans le domaine entier par des tunnels :  
stockage d'entrées pour la configuration des tunnels

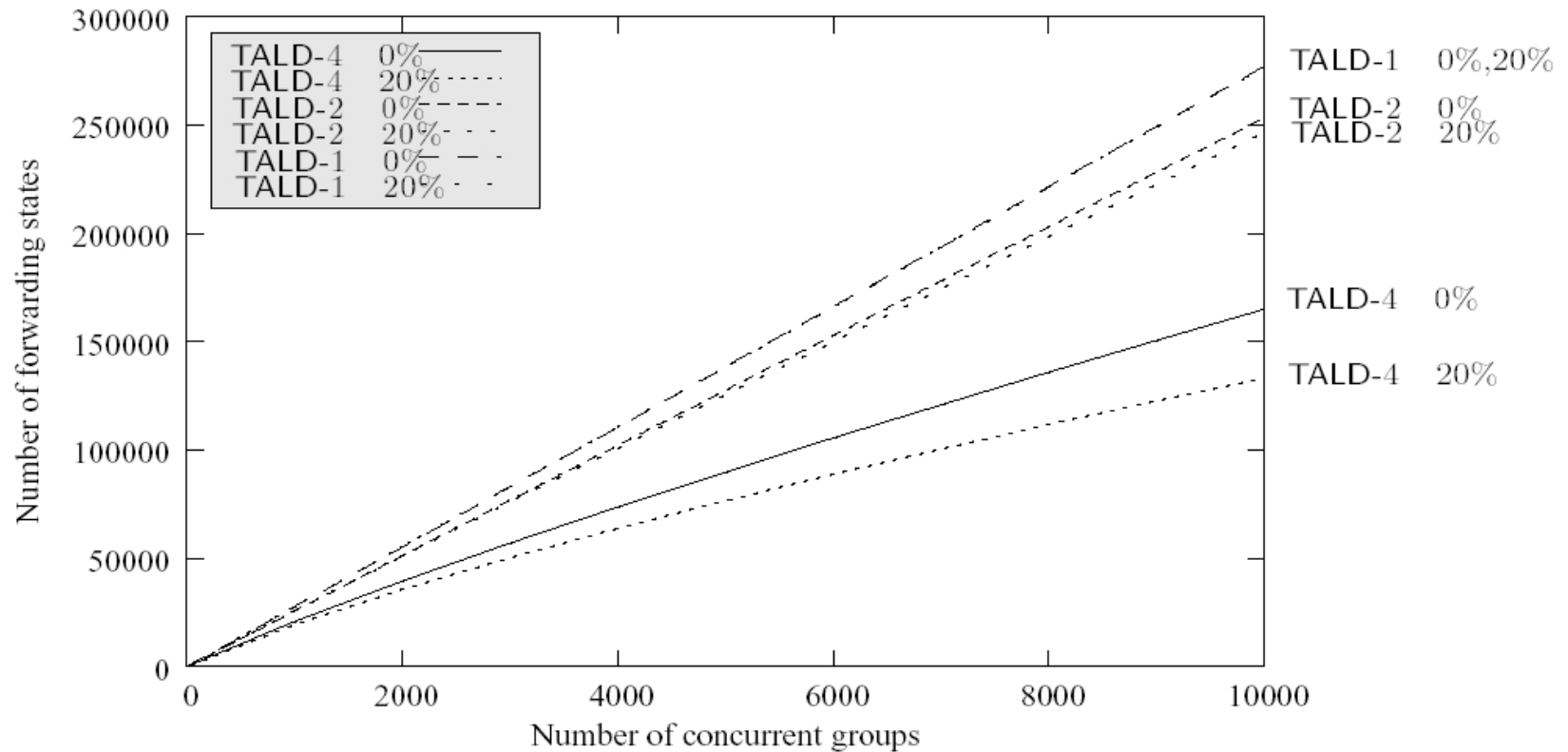
# Avantages du protocole TALD

- L'entité C n'a pas besoin de connaître la topologie du domaine entier pour configurer les tunnels
- Chaque sous-domaine est géré de manière **indépendamment** des autres
- Bon taux d'agrégation dans chacun des sous-domaines

# Simulations

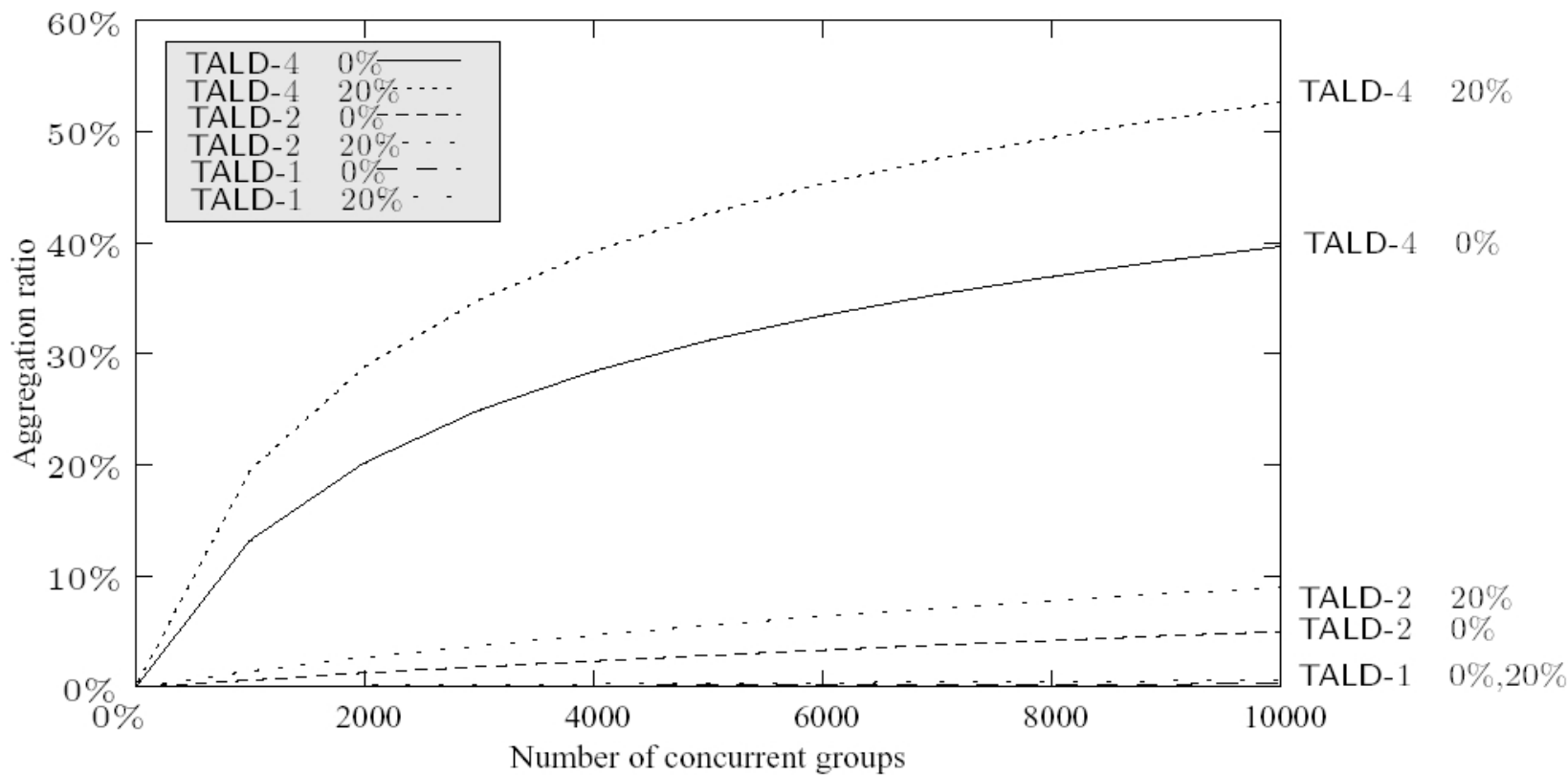
- Réseau Rocketfuel *Exodus* de 100 routeurs de bordure et 100 routeurs de cœur
  - ◆ TALD\_1 : **sans** découpe du domaine
  - ◆ TALD\_2 : domaine découpé en **2** sous-domaines
  - ◆ TALD\_4 : domaine découpé en **4** sous-domaines
  
- Membres choisis aléatoirement parmi les routeurs de bordure (entre 2 et 20 membres par groupe)

# Nombre d'entrées de routage



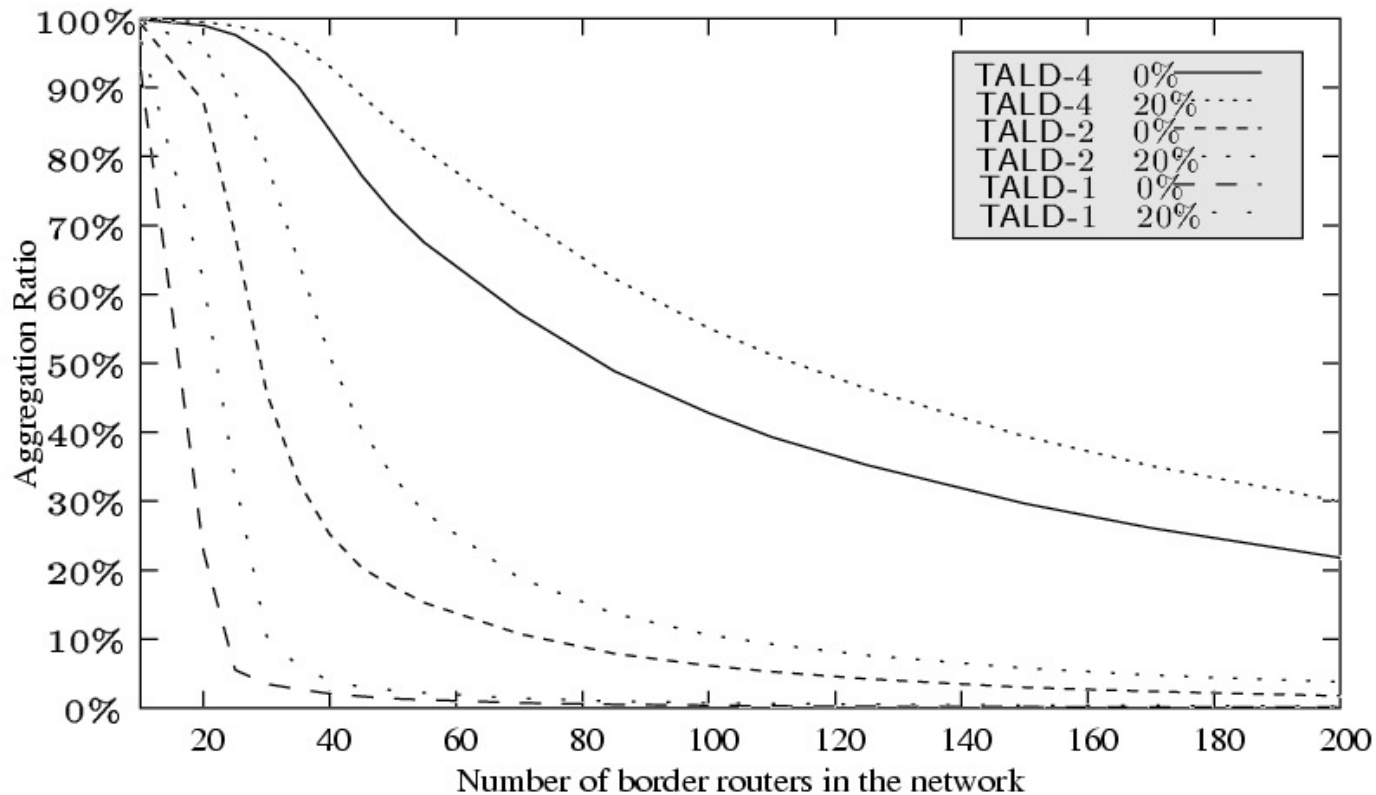
Number of forwarding states

# Taux d'agrégation



Aggregation ratio

# Taux d'agrégation



Aggregation ratio in function of the number of border routers

# Conclusion

- Agrégation d'arbres : solution pour déployer le multicast
- **STA** : proposition pour réaliser une agrégation rapide et tout aussi efficace.
- **TALD** : agrégation dans les grands domaines.

# Références

- **Aggregated Multicast --- A comparative study** Jun-Hong Cui, Jinkyu Kim, Dario Maggiorini, Khaled Boussetta, Mario Gerla  
*Special issue of cluster computing: the journal of networks, software and applications* 2003.
- **STA: Scalable Tree Aggregation** Alexandre Guitton, Joanna Moulierac  
*7th International Conference on Telecommunications ConTEL* 2005.  
(Best Student Paper Award)
- **Q-STA: QoS Scalable Tree Aggregation** Joanna Moulierac, Alexandre Guitton  
*IFIP Networking* 2005.
- **DMTA: Distributed Multicast Tree Aggregation** Joanna Moulierac, Alexandre Guitton  
*INRIA Research Report n°5636*, 2005.
- **TALD: Multicast Tree Aggregation in Large Domains** Joanna Moulierac, Alexandre Guitton, Miklós Molnár  
*IFIP Networking* 2006.
- **On the number of multicast aggregated trees in a domain** Joanna Moulierac, Miklós Molnár  
*2<sup>nd</sup> Student Workshop of IEEE INFOCOM* 2006.



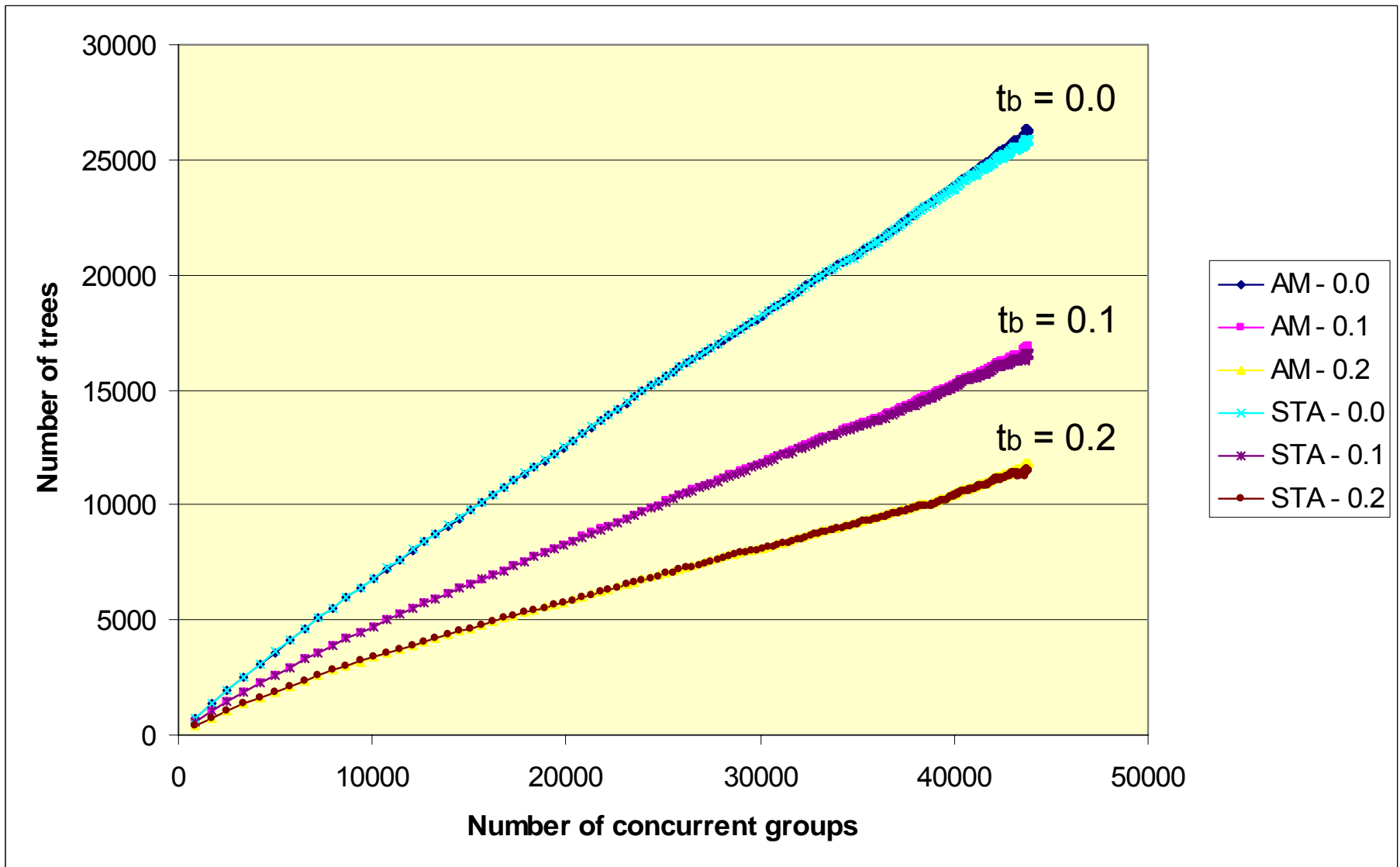
institut de recherche en informatique  
et systèmes aléatoires

# Annexes

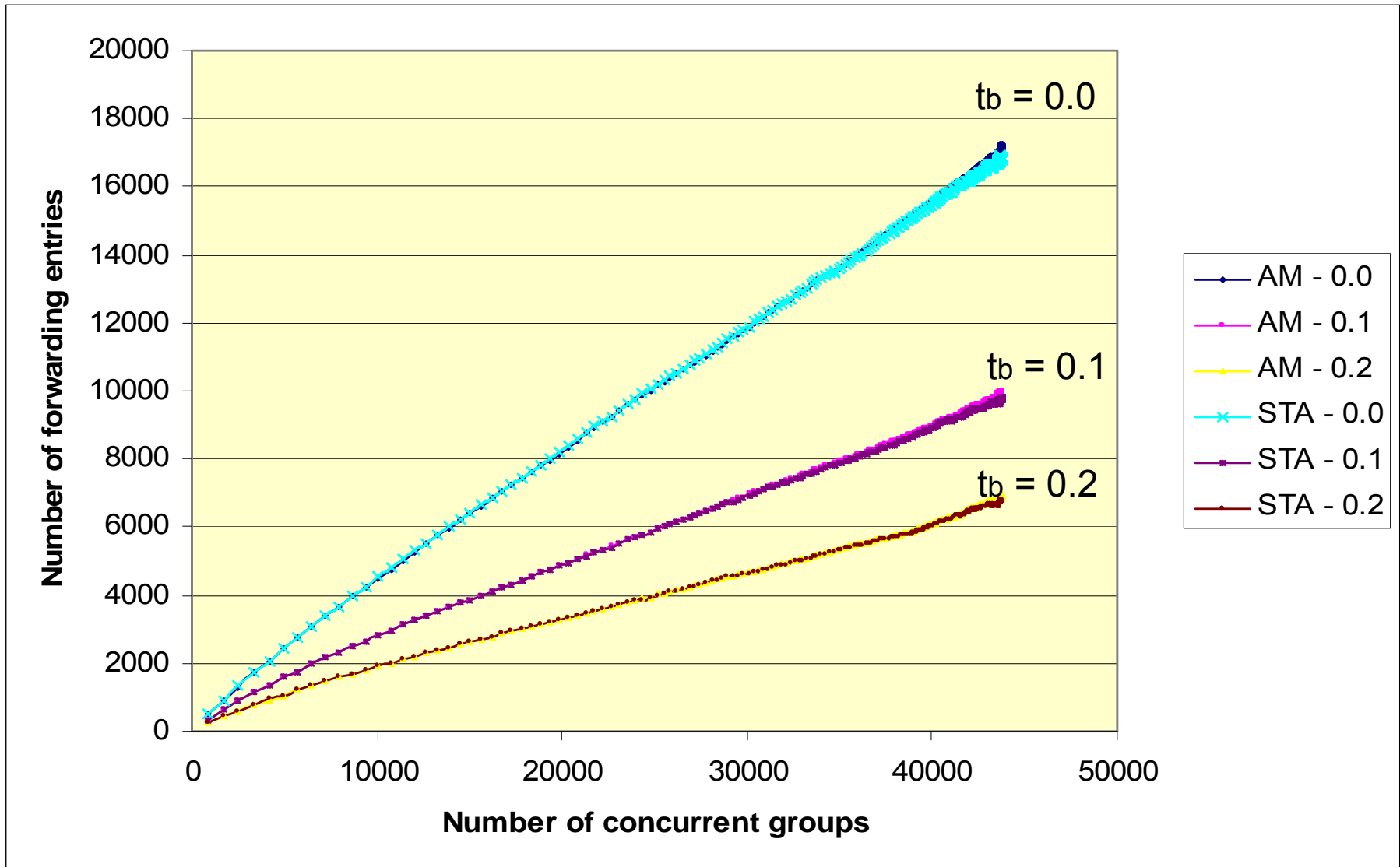
# Résultats de simulation

- Simulations sur le réseau Eurorings de 43 nœuds et 55 arêtes
- Groupes dont les membres sont répartis aléatoirement (entre 2 et 20 membres par groupe)
- Groupes dynamiques (environ 45 000 groupes concurrents)
- Simulations en fonctions de trois seuils de bande passante :
  - ◆ 0% de perte de bande passante autorisée
  - ◆ 10% de perte de bande passante autorisée
  - ◆ 20% de perte de bande passante autorisée

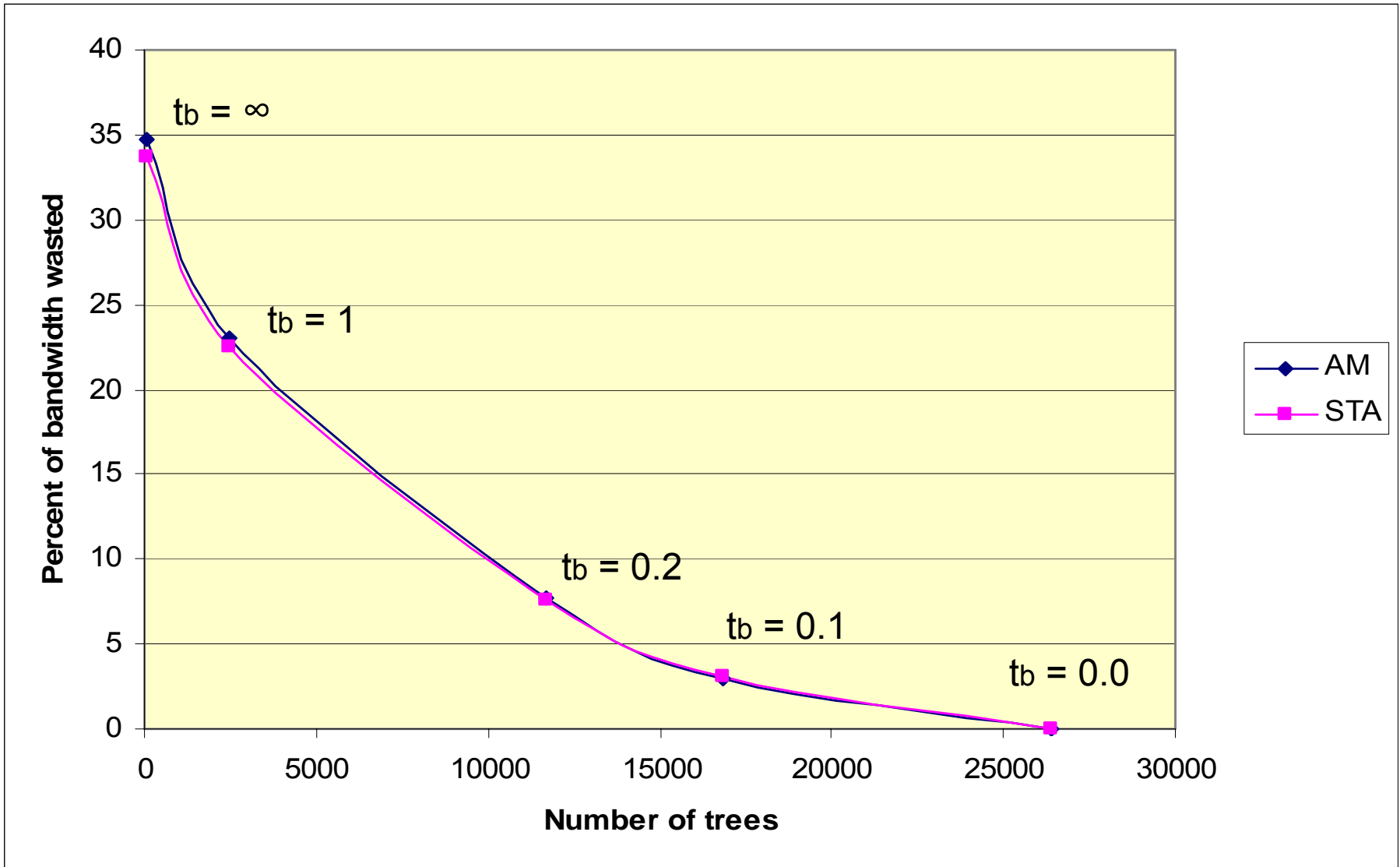
# Nombre d'arbres



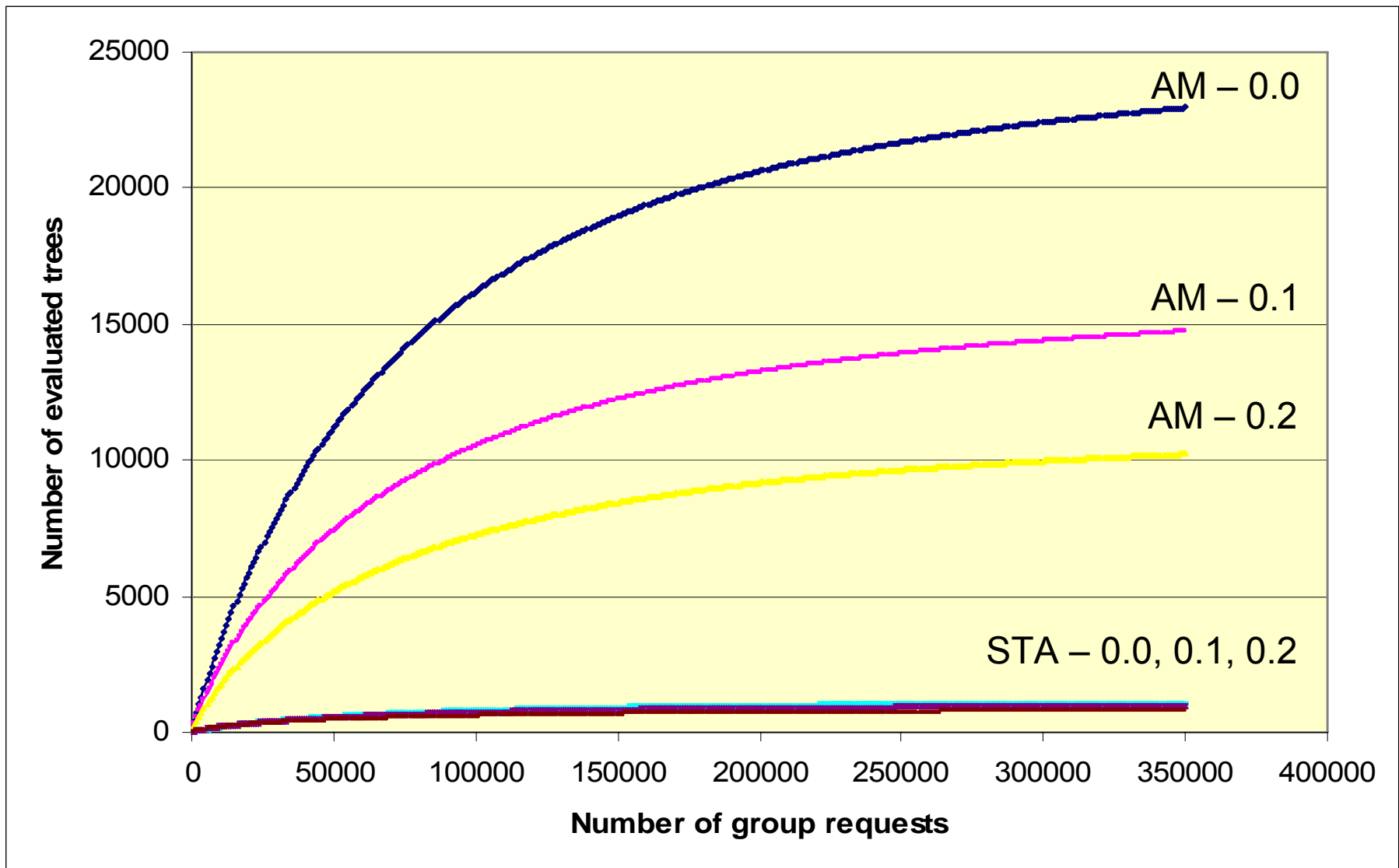
# Nombre d'entrées de routage



# Perte de bande passante



# Nombre d'arbres évalués



# Temps de calcul (en secondes)

