SDR

Battements

Algorithmes synchronisés

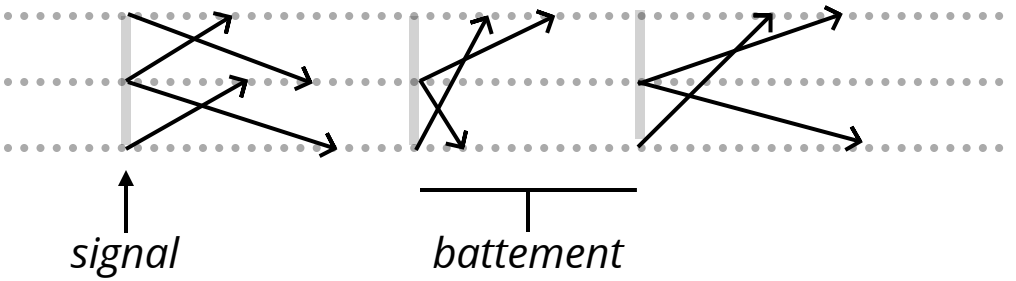
Olivier Lemer

Rappels

Algorithmes synchrones

Un **signal** est reçu pour déclencher un nouveau **battement**.

(pulse)



Dans chaque battement, un processus peut

- Faire des calculs *relatifs aux battements précédents*.
- Envoyer jusqu'à un message par voisin, *fonction des battements précédents*.
- Recevoir jusqu'à un message par voisin, *fonction des battements précédents.*

Un processus est **prêt** quand tous ses messages ont été *reçus*.

Aujourd'hui

Deux exemples d'algorithme synchronisé.

- Découverte de la topologie du réseau.
- Calcul distribué de la multiplication de deux matrices.

Découvrir la topologie

Un algorithme synchrone

Description du problème

Topologie : description du graphe *("qui est connecté à qui")*

Initialement, chaque processus connait uniquement

- ses voisins directs, et
- le nombre total de processus, `n`.

En fin d'algorithme, chaque processus doit connaitre aussi

- les voisins de chaque processus du système.

Comment faire ?

Comment faire ?

Algorithme synchrone

Comment faire ?

Algorithme synchrone

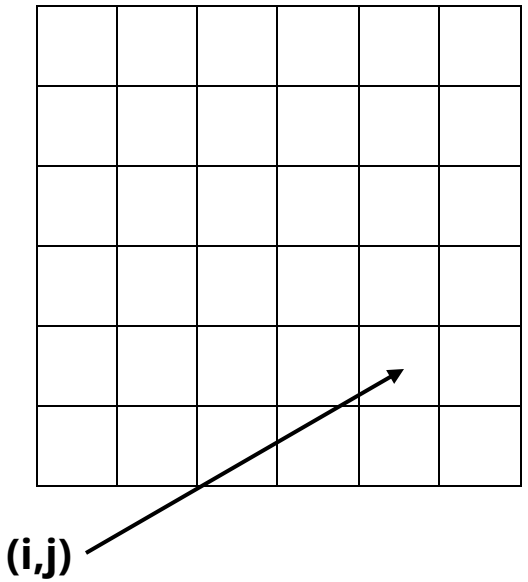
Comment représenter la topologie ?

Comment faire ?

Algorithme synchrone

Comment représenter la topologie ?

Une matrice nxn de booléens

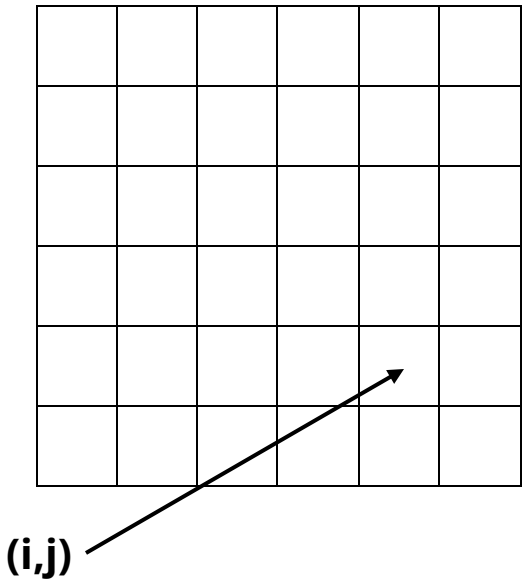


Comment faire ?

Algorithme synchrone

Comment représenter la topologie ?

Une matrice nxn de booléens



Un dictionnaire

$V \longmapsto \mathcal{P}(V)$

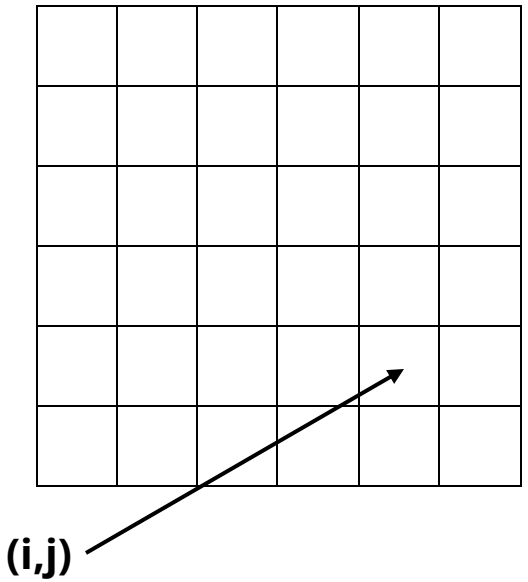
(à chaque processus, associe un sous-ensemble de processus)

Comment faire ?

Algorithme synchrone

Comment représenter la topologie ?

Une matrice nxn de booléens



Taille : $O(V^2)$

Un dictionnaire

$V \longmapsto \mathcal{P}(V)$

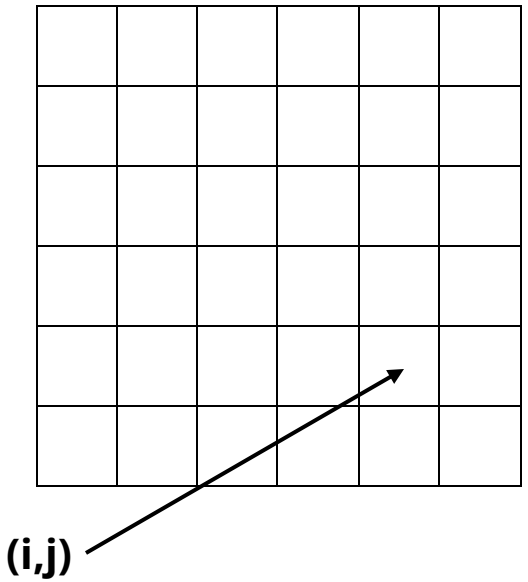
(à chaque processus, associe un sous-ensemble de processus)

Comment faire ?

Algorithme synchrone

Comment représenter la topologie ?

Une matrice nxn de booléens



Taille : $O(V^2)$

Un dictionnaire

$V \longmapsto \mathcal{P}(V)$

(à chaque processus, associe un sous-ensemble de processus)

Taille : $O(E+V)$

Comment faire ?

Algorithme synchrone

Comment représenter la topologie ?
Un dictionnaire : $V \longmapsto \mathcal{P}(V)$

Comment faire ?

Algorithme synchrone

Comment représenter la topologie ?
Un dictionnaire : $V \longmapsto \mathcal{P}(V)$

Quelle valeur initiale ?

Comment faire ?

Algorithme synchrone

Comment représenter la topologie ?
Un dictionnaire : $V \longmapsto \mathcal{P}(V)$

Quelle valeur initiale ?
Uniquement mes voisins *(c'est tout ce que je sais, de toute façon)*

Comment faire ?

Algorithme synchrone

Comment représenter la topologie ?
Un dictionnaire : $V \longmapsto \mathcal{P}(V)$

Quelle valeur initiale ?
Uniquement mes voisins *(c'est tout ce que je sais, de toute façon)*

Quoi faire à chaque battement ?

Comment faire ?

Algorithme synchrone

Comment représenter la topologie ?
Un dictionnaire : $V \longmapsto \mathcal{P}(V)$

Quelle valeur initiale ?
Uniquement mes voisins *(c'est tout ce que je sais, de toute façon)*

Quoi faire à chaque battement ?
- Mettre à jour mon dictionnaire avec ceux reçus au précédent battement.

Comment faire ?

Algorithme synchrone

Comment représenter la topologie ?
Un dictionnaire : $V \longmapsto \mathcal{P}(V)$

Quelle valeur initiale ?
Uniquement mes voisins *(c'est tout ce que je sais, de toute façon)*

Quoi faire à chaque battement ?

- Mettre à jour mon dictionnaire avec ceux reçus au précédent battement.
- Envoyer mon dictionnaire à tous mes voisins.

Comment faire ?

Algorithme synchrone

Comment représenter la topologie ?
Un dictionnaire : $V \longmapsto \mathcal{P}(V)$

Quelle valeur initiale ?
Uniquement mes voisins *(c'est tout ce que je sais, de toute façon)*

Quoi faire à chaque battement ?
- Mettre à jour mon dictionnaire avec ceux reçus au précédent battement.
- Envoyer mon dictionnaire à tous mes voisins.

Quand m'arrêter ?

Comment faire ?

Algorithme synchrone

Comment représenter la topologie ?
Un dictionnaire : $V \longmapsto \mathcal{P}(V)$

Quelle valeur initiale ?
Uniquement mes voisins *(c'est tout ce que je sais, de toute façon)*

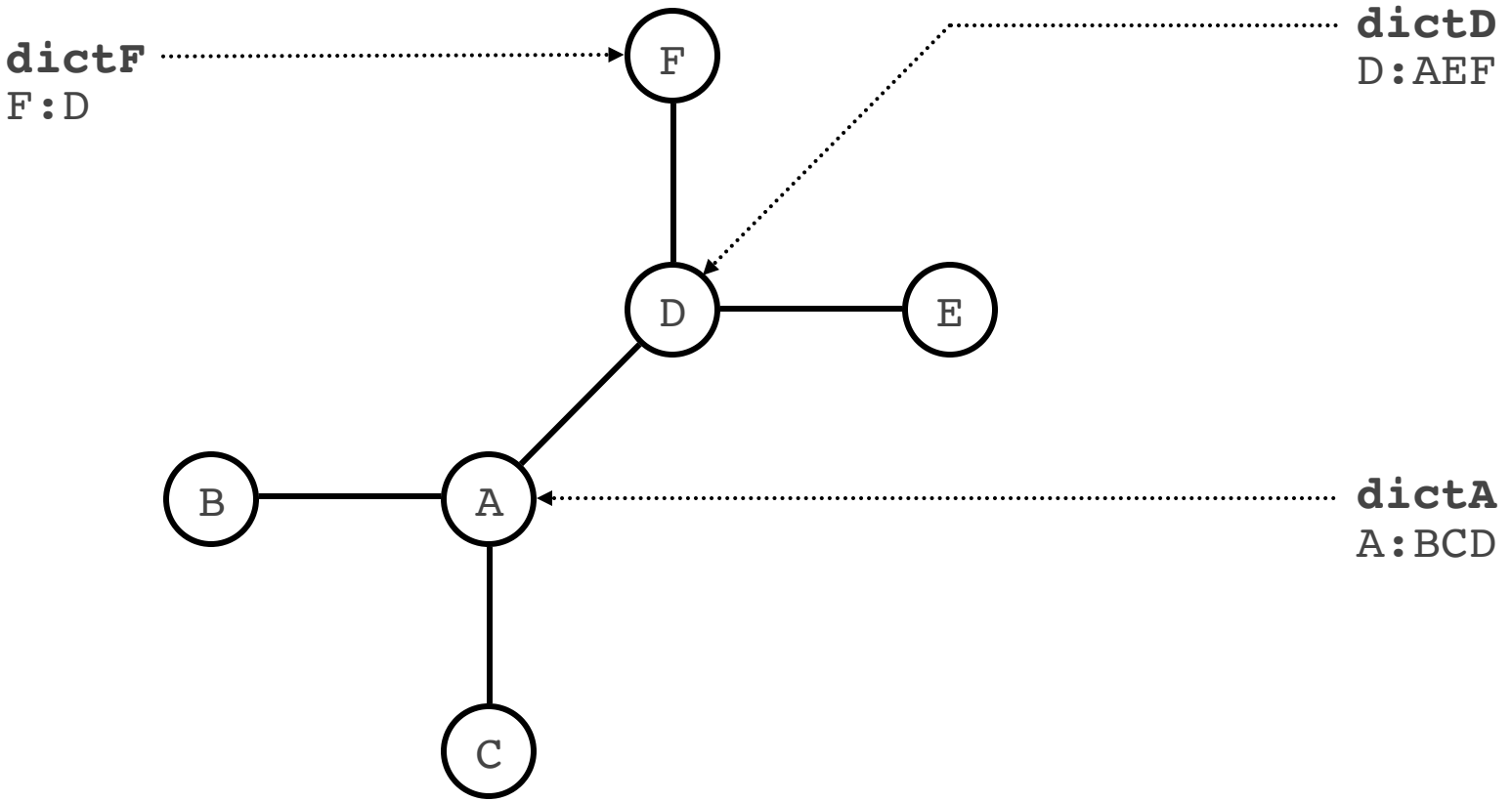
Quoi faire à chaque battement ?
- Mettre à jour mon dictionnaire avec ceux reçus au précédent battement.
- Envoyer mon dictionnaire à tous mes voisins.

Quand m'arrêter ?
Quand mon dictionnaire a une taille de $\mathtt{n}$.

Découvrir la topologie

Exemple

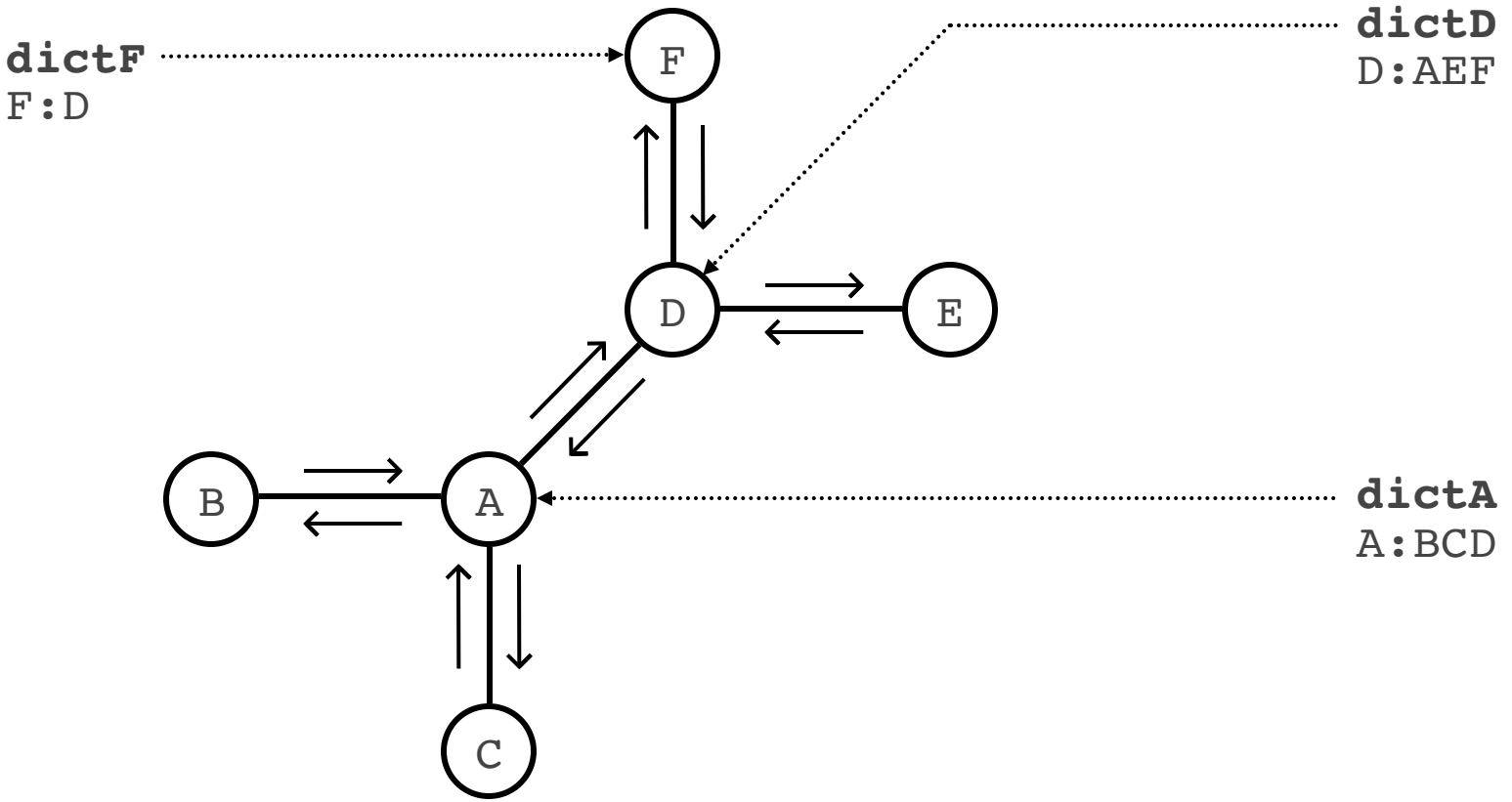
Exemple



Battement 1

Envoi des premiers messages

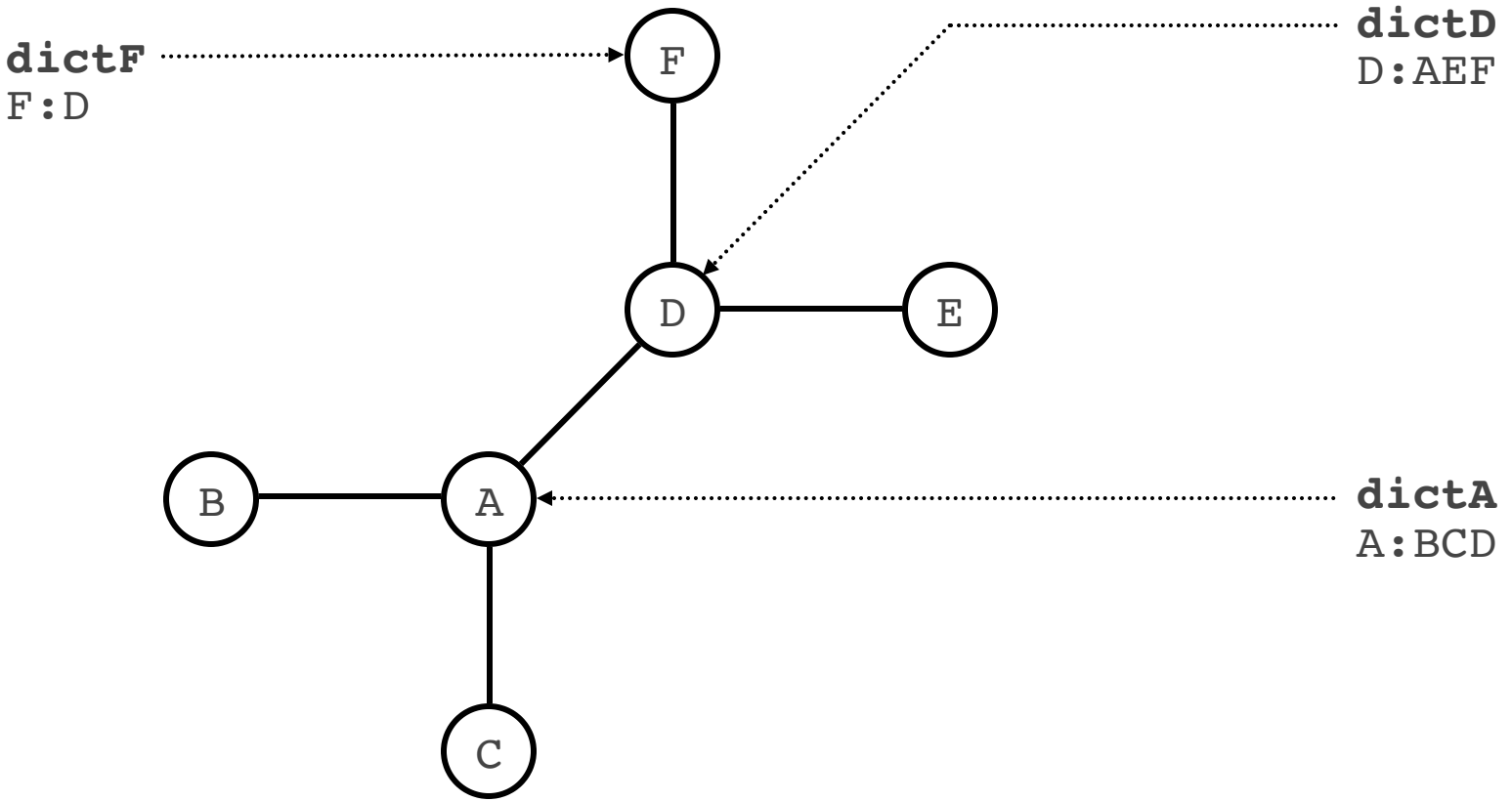
Exemple



Battement 1

Envoi des premiers messages

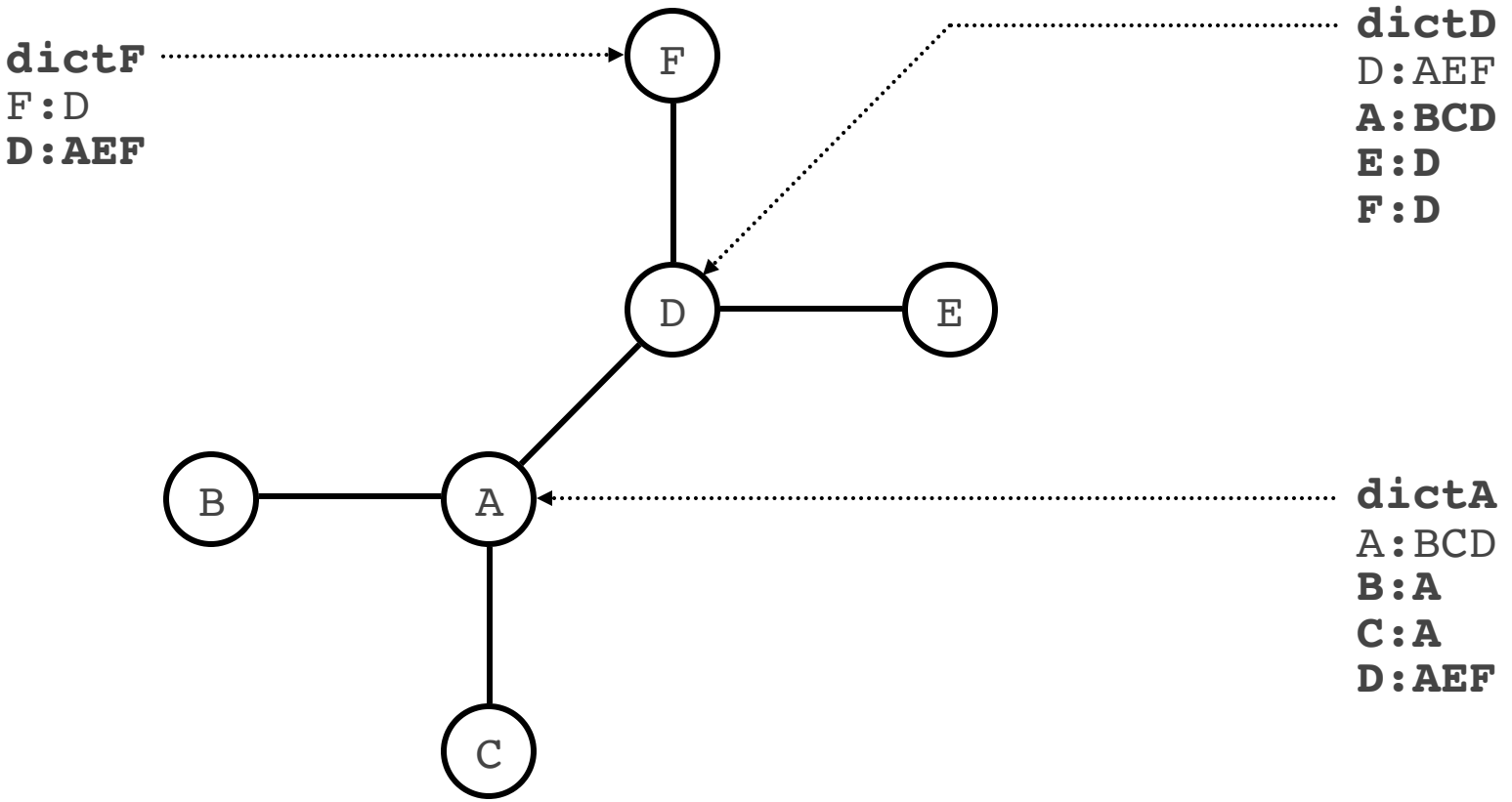
Exemple



Battement 2

Traitement des messages précédents

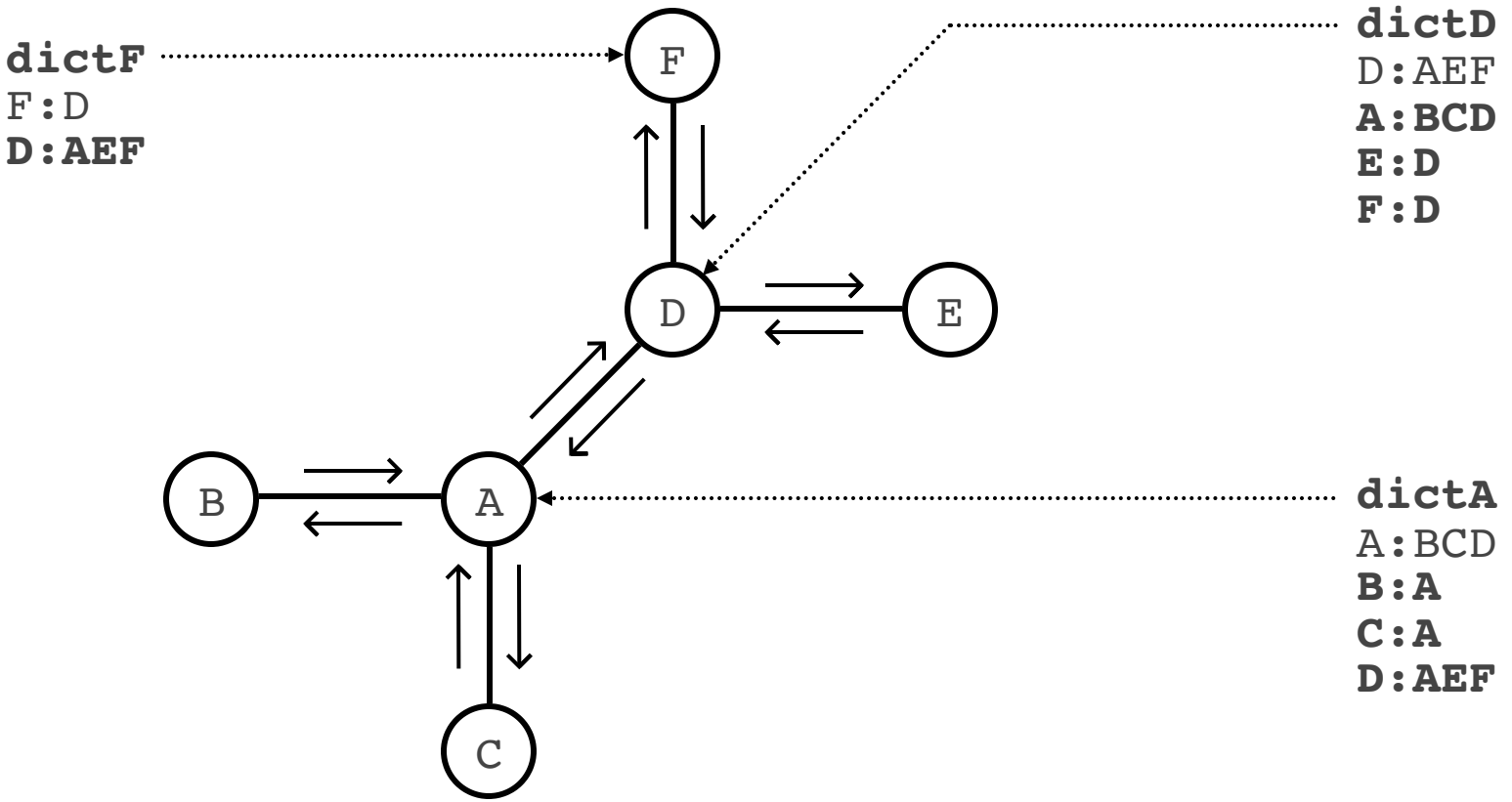
Exemple



Battement 2

Traitement des messages précédents

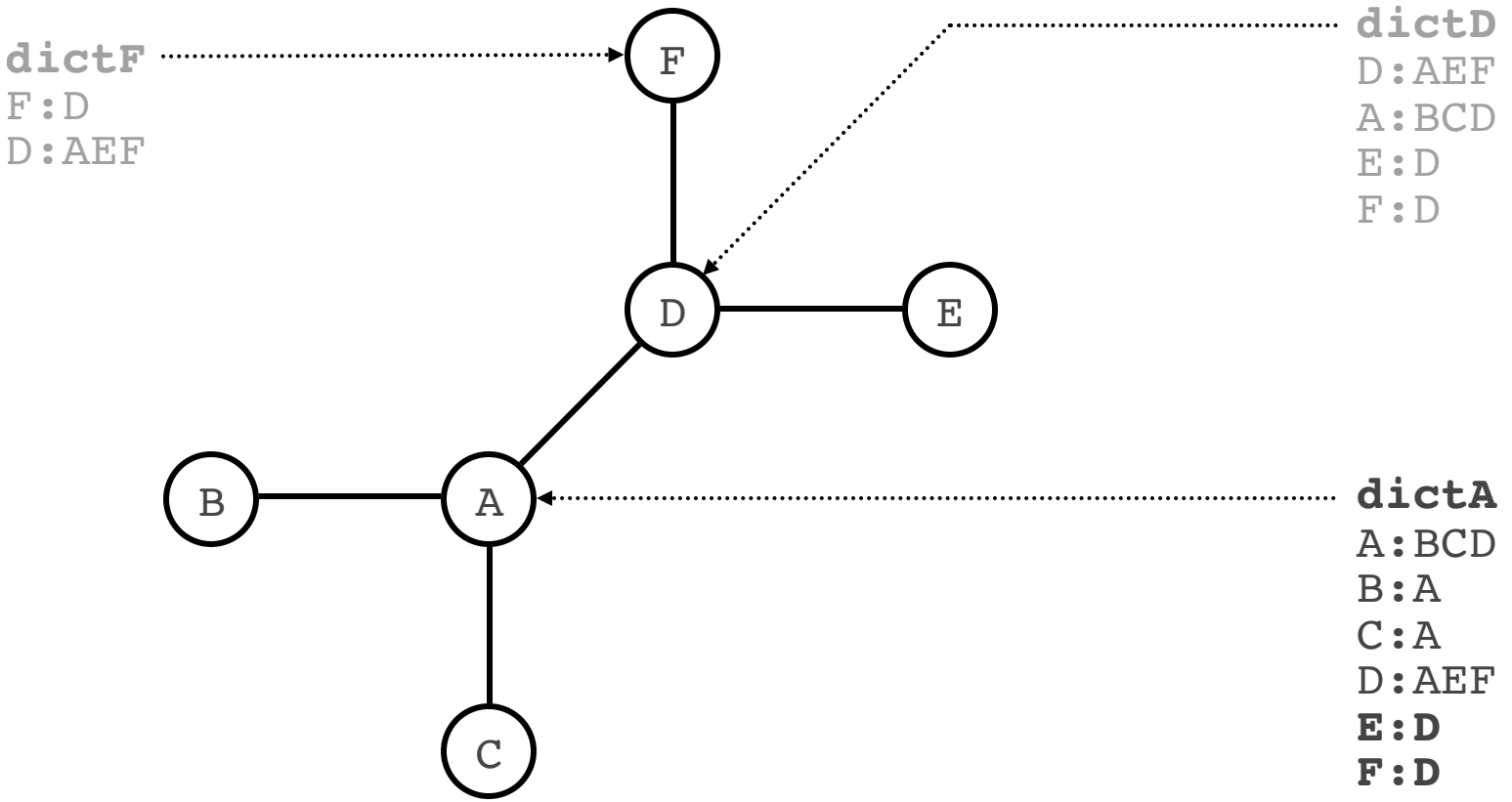
Exemple



Battement 2

Envoi des messages

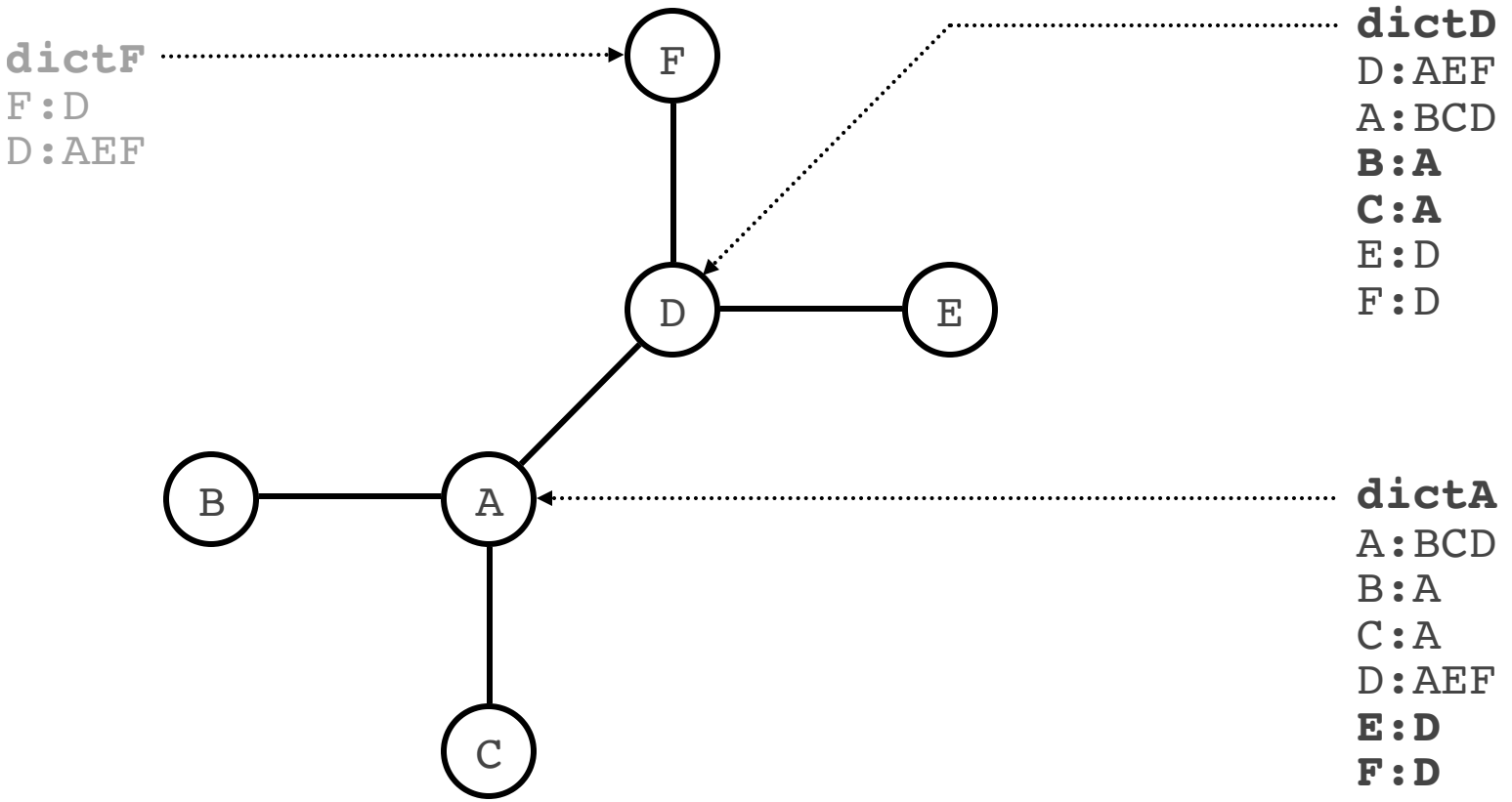
Exemple



Battement 3

Traitement des messages précédents

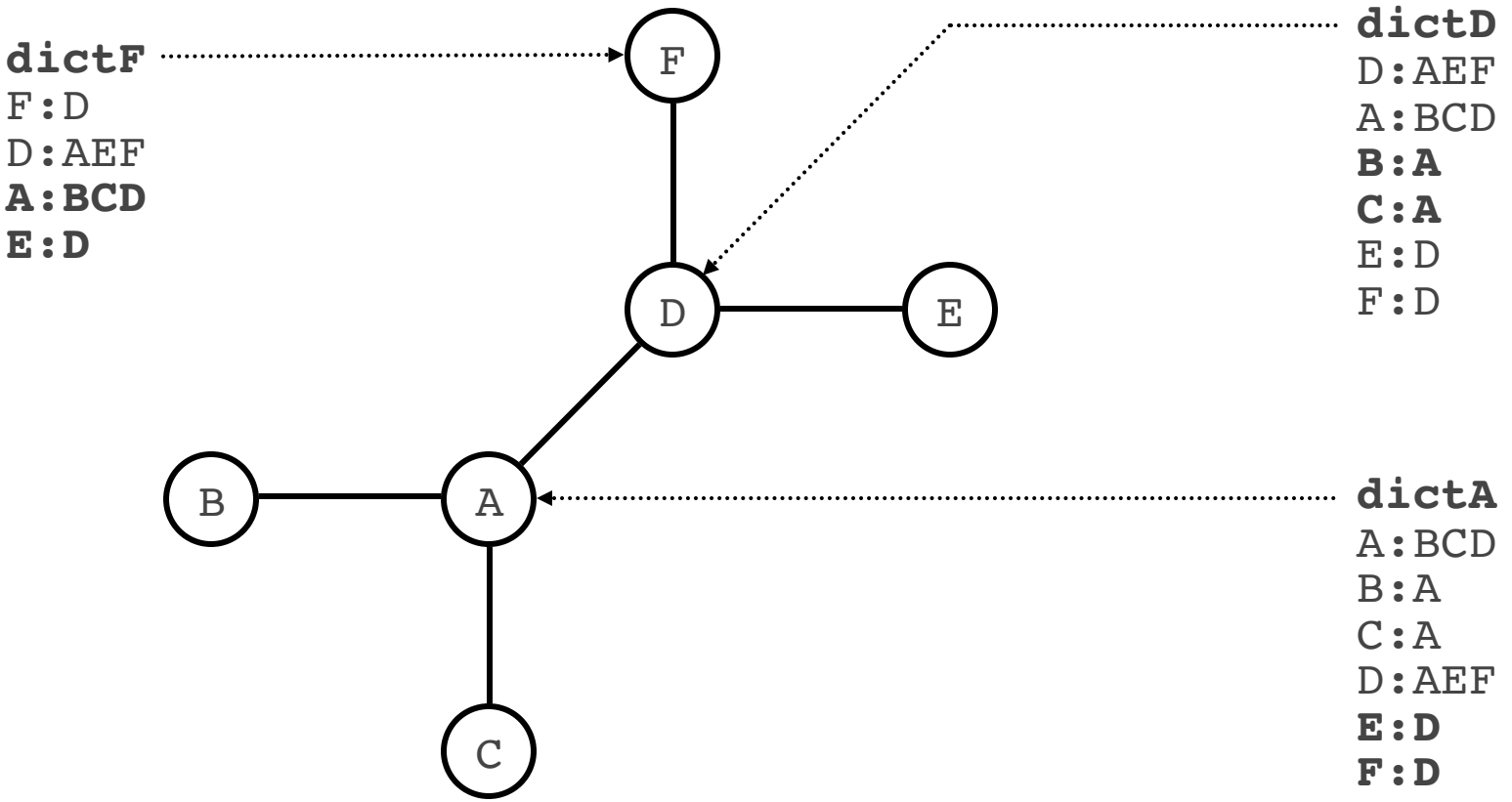
Exemple



Battement 3

Traitement des messages précédents

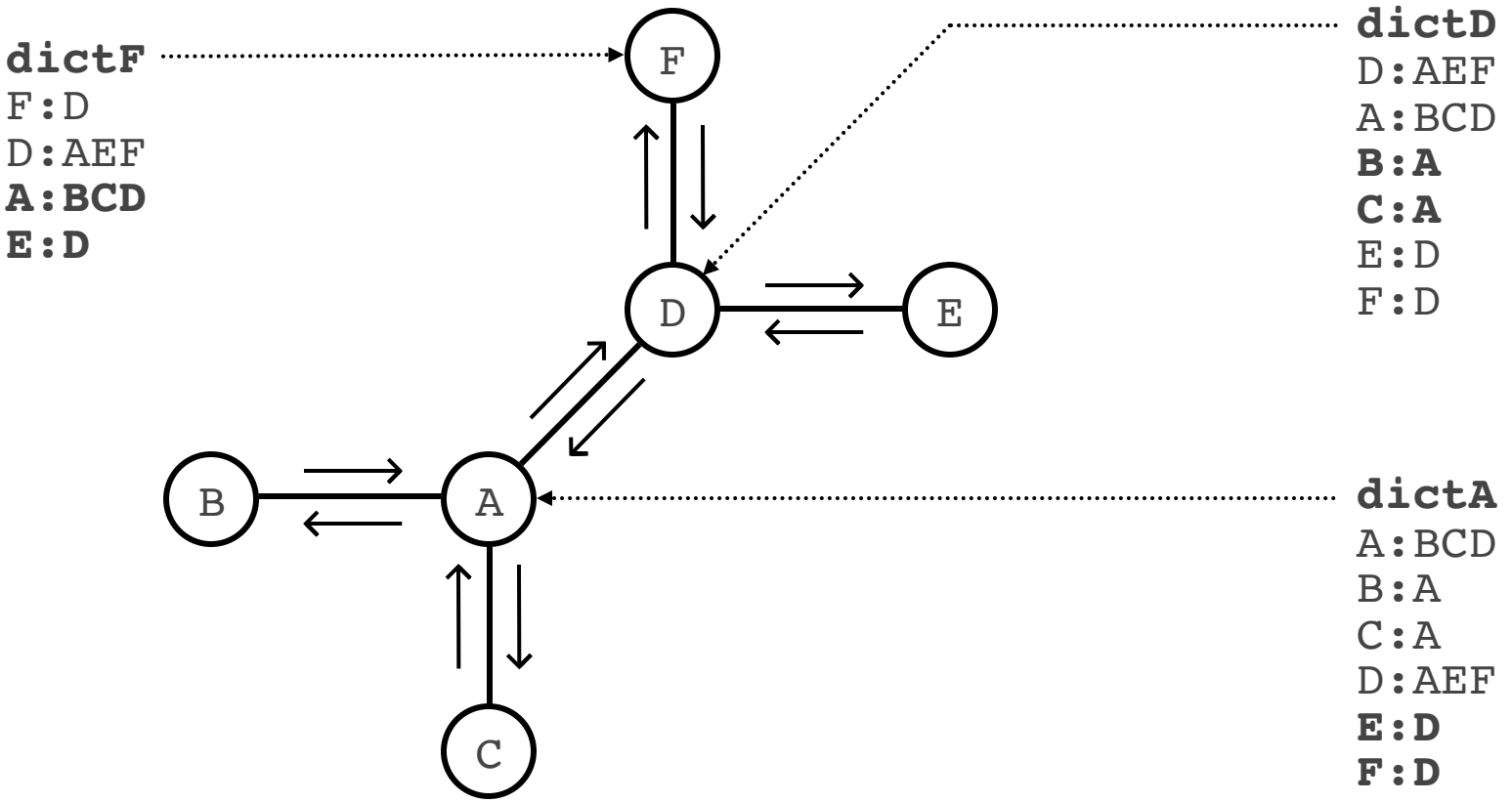
Exemple



Battement 3

Traitement des messages précédents

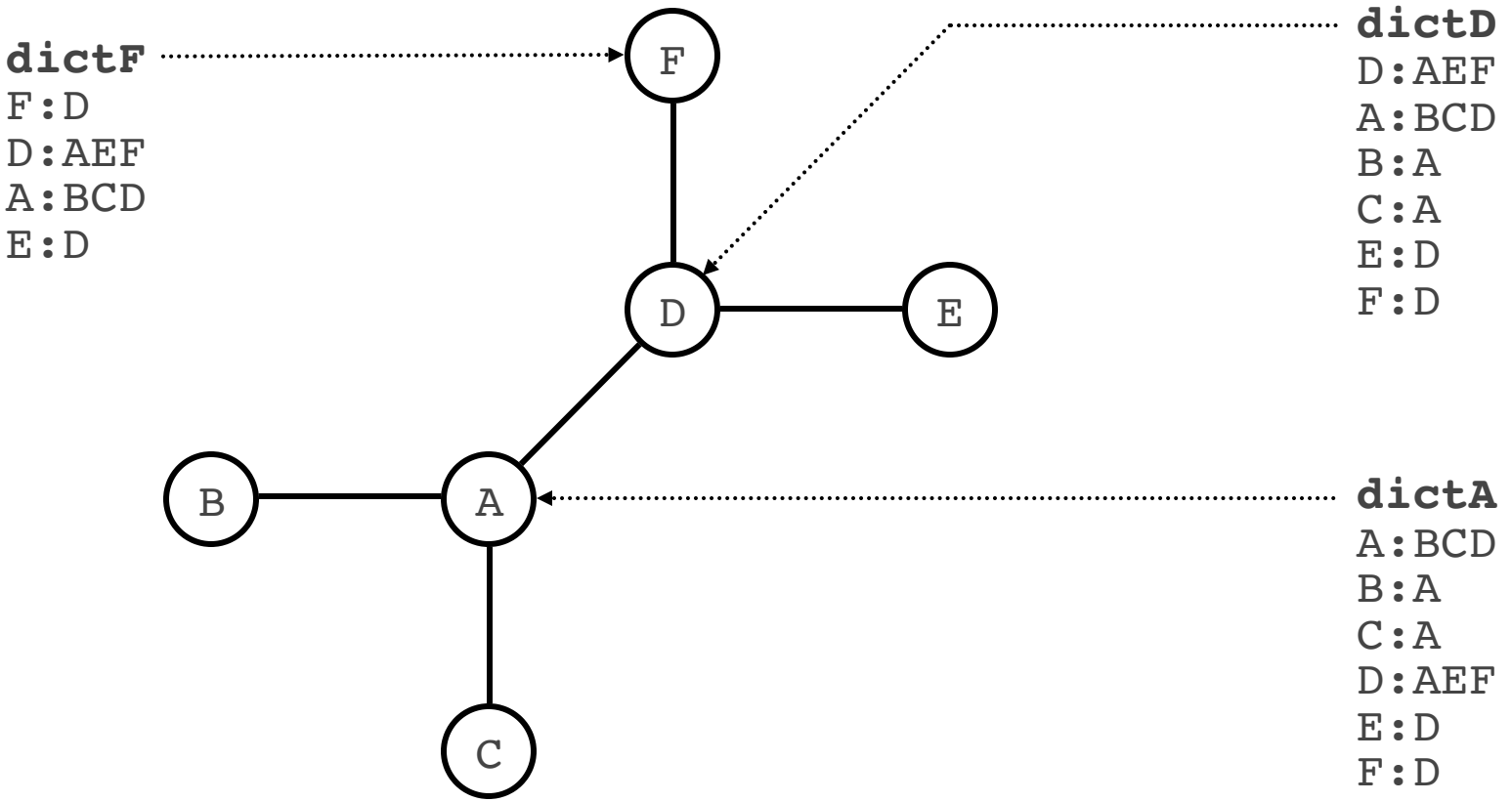
Exemple



Battement 3

Envoi des messages

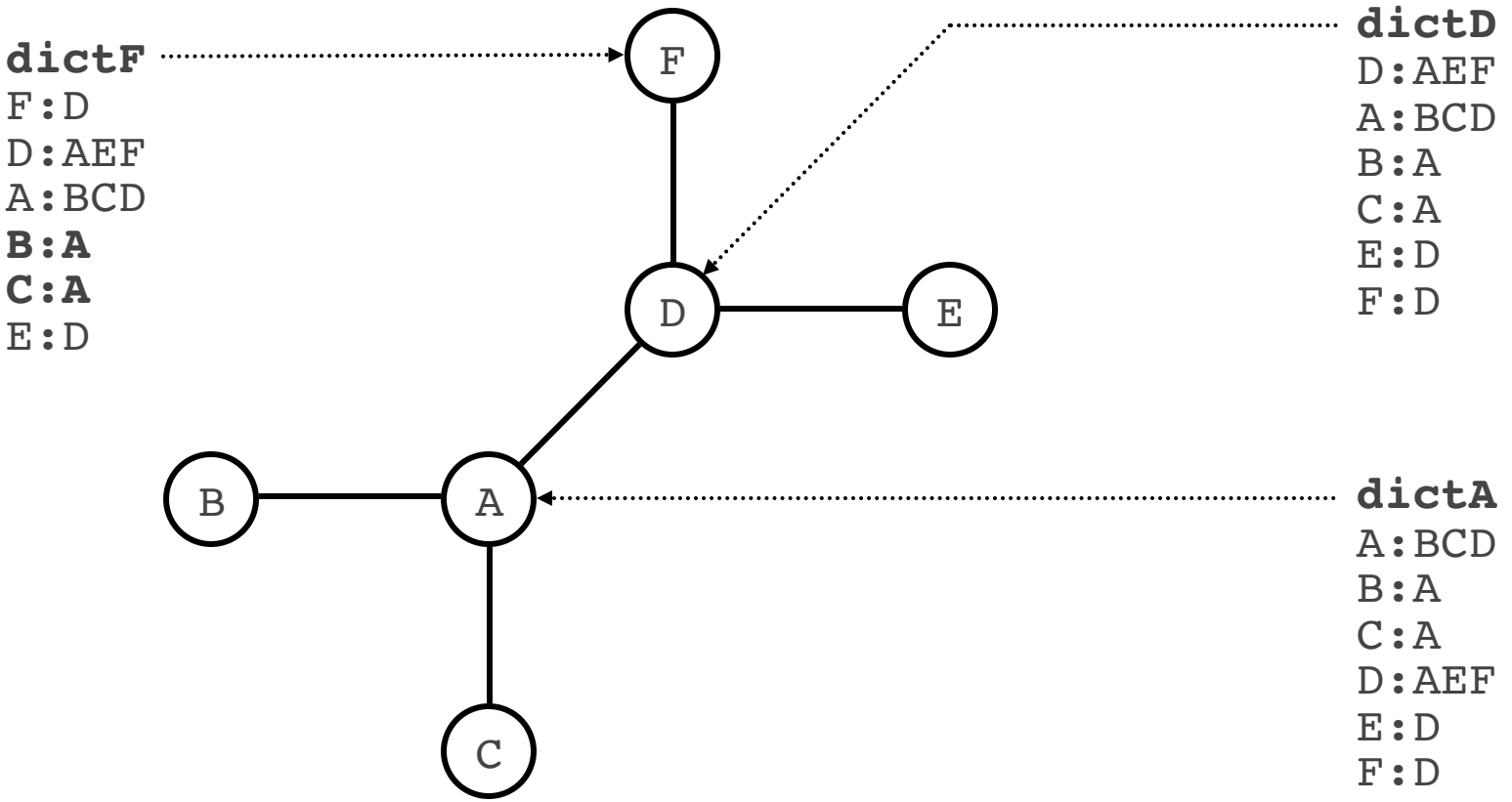
Exemple



Battement 4

Traitement des messages précédents

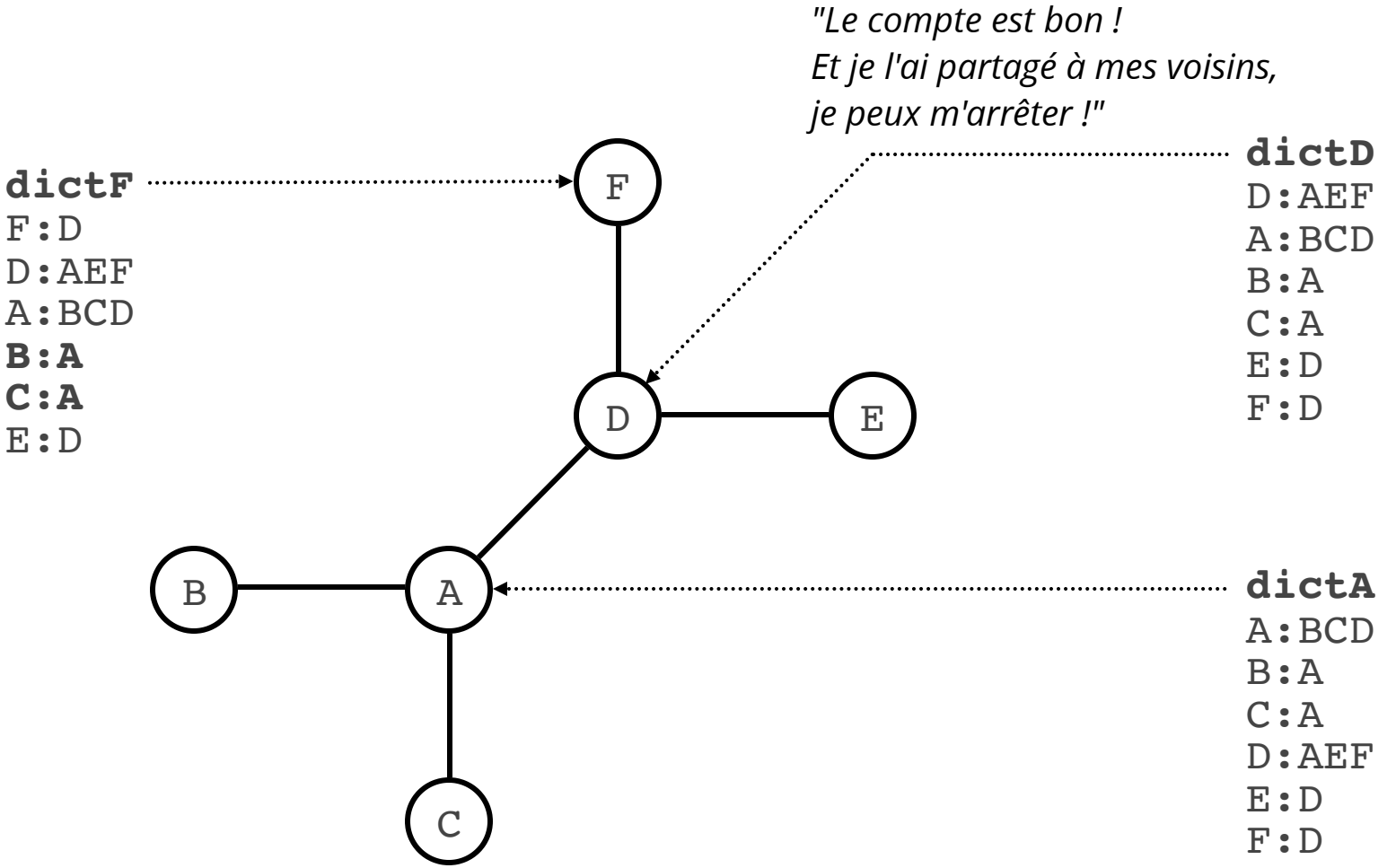
Exemple



Battement 4

Traitement des messages précédents

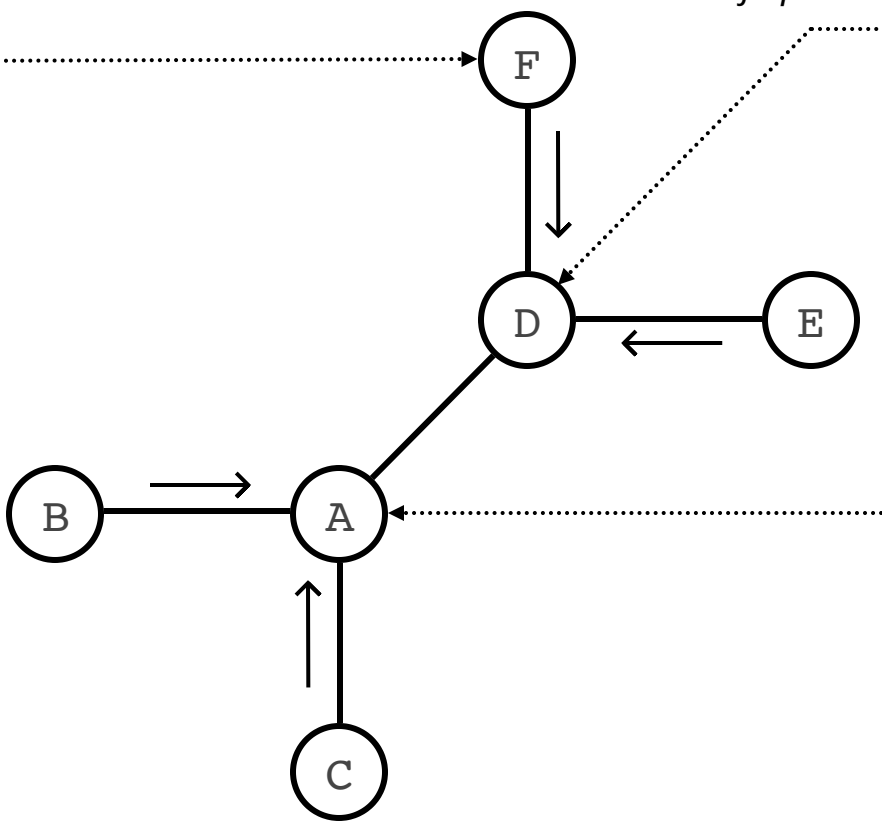
Exemple



Battement 4

Traitement des messages précédents

Exemple



dictF
F:D
D:AEF
A:BCD
B:A
C:A
E:D

"Le compte est bon !
Et je l'ai partagé à mes voisins,
je peux m'arrêter !"

dictD
D:AEF
A:BCD
B:A
C:A
E:D
F:D

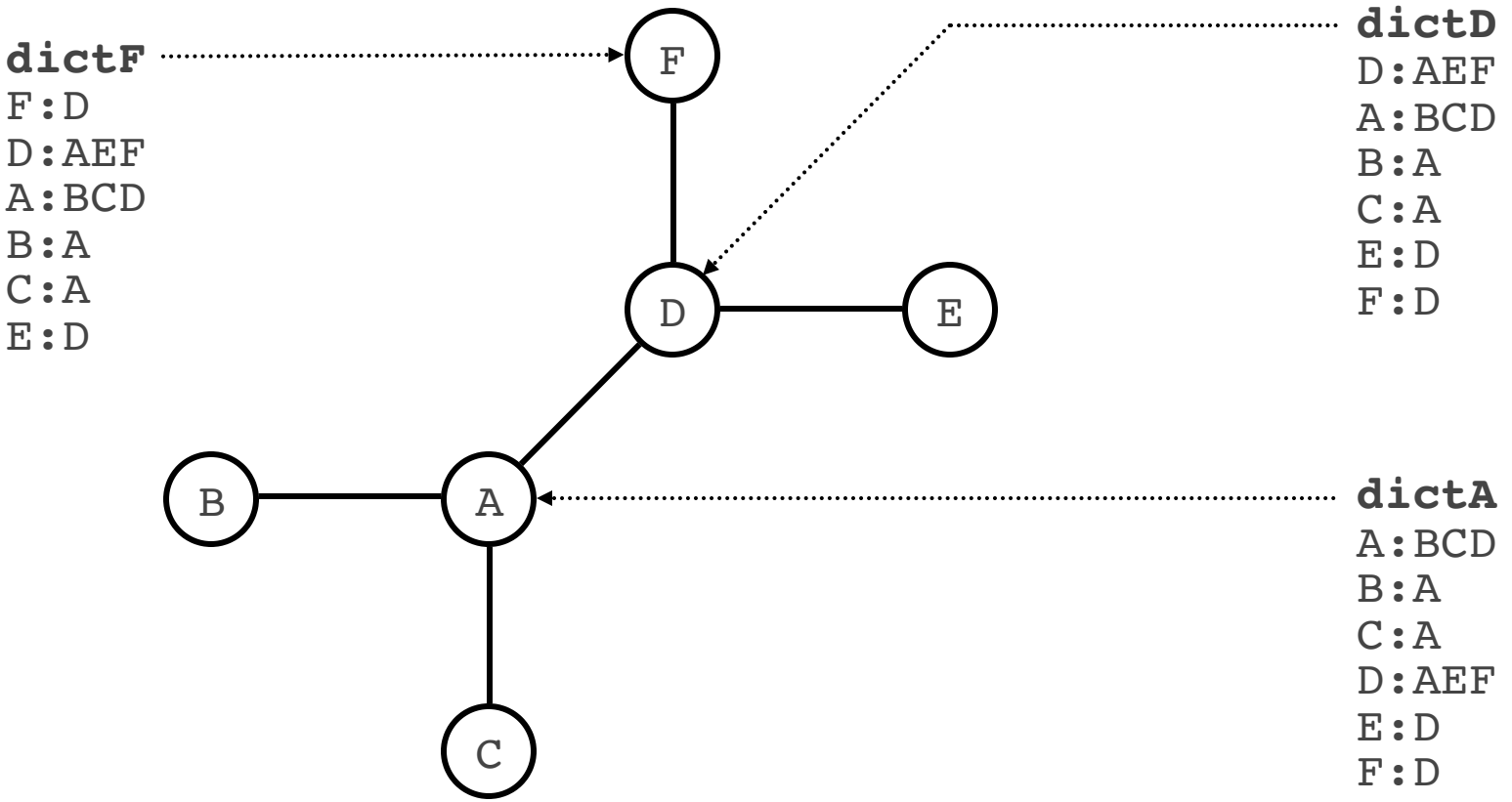
dictA
A:BCD
B:A
C:A
D:AEF
E:D
F:D

"Le compte est bon !
Et je l'ai partagé à mes voisins,
je peux m'arrêter !"

Battement 4

Envoi des messages

Exemple

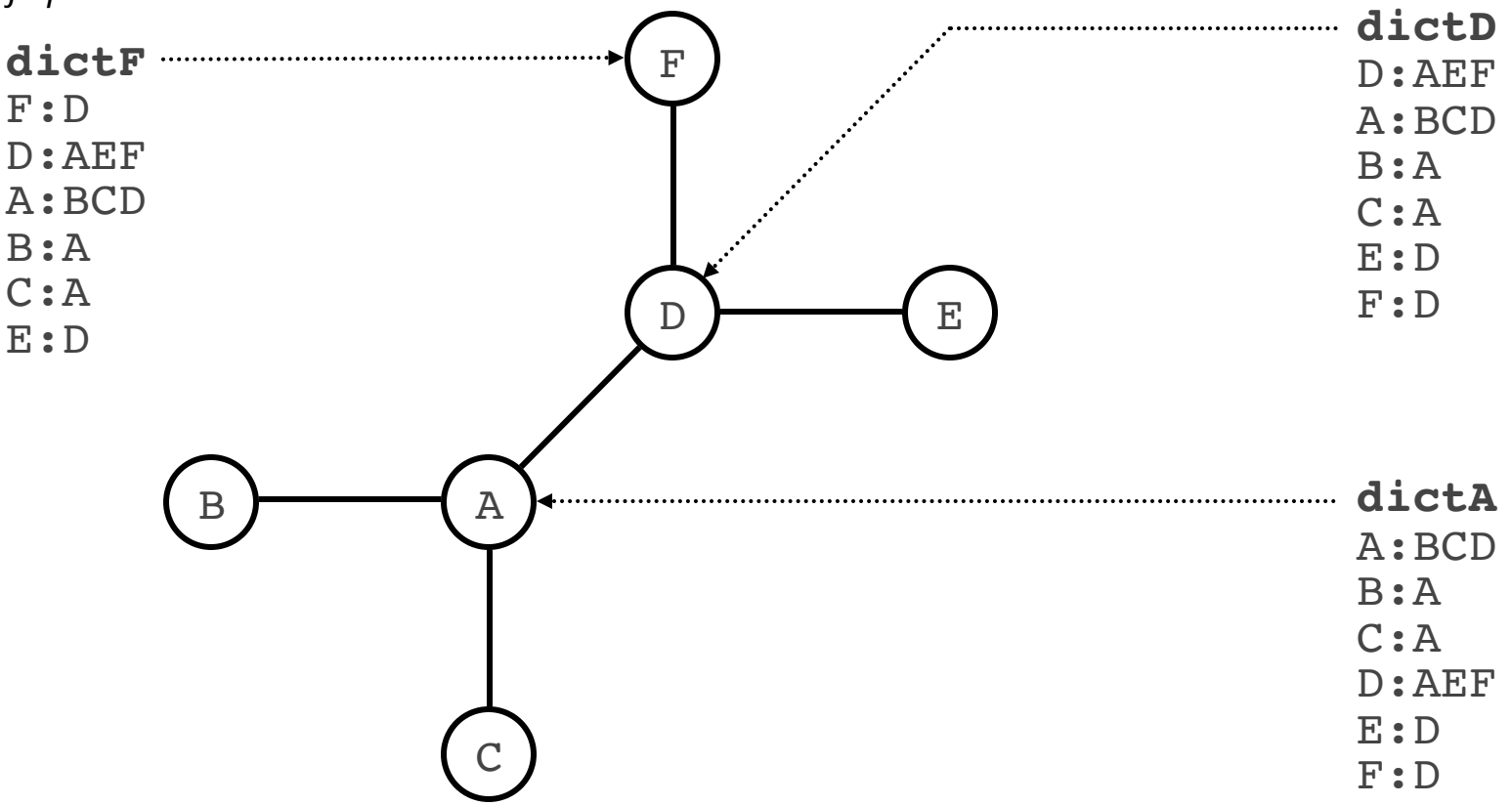


Battement 5

Traitement des messages précédents

Exemple

"Le compte est bon !
Et je l'ai partagé à mes voisins,
je peux m'arrêter !"

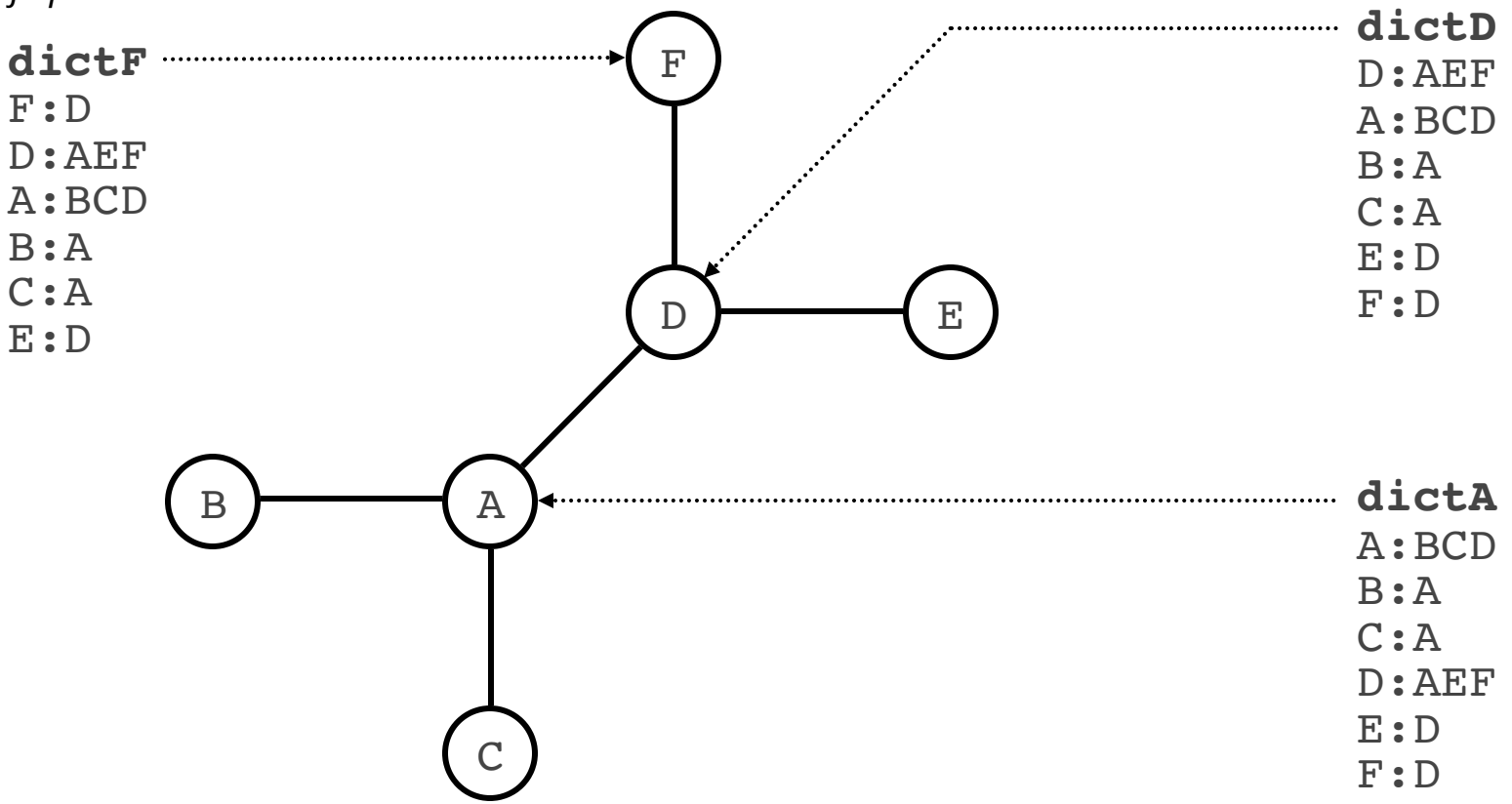


Battement 5

Traitement des messages précédents

Exemple

"Le compte est bon !
Et je l'ai partagé à mes voisins,
je peux m'arrêter !"



Battement 5

Envoi des messages

Découvrir la topologie

Pseudocode

Pseudocode

Variables

`sync`	*synchroniseur*	Utilisé pour avancer par battements
`dict`	*dictionnaire*	Topologie du réseau

Pseudocode

À chaque nouveau battement `i`

Pour chaque message du battement `i-1`

    Mise à jour de `dict` avec le dictionnaire du message

Envoi de `dict` à tous les voisins actifs

Si `dict` a une taille de `n`

    Quitter le synchroniseur

Pseudocode

À chaque nouveau battement `i`

Pour chaque message du battement `i-1`

    Mise à jour de `dict` avec le dictionnaire du message

Envoi de `dict` à tous les voisins actifs

Si `dict` a une taille de `n`

    Quitter le synchroniseur

Remarque : tout se fait à travers un synchroniseur

- Les "messages du battement `i-1`" sont obtenus du synchroniseur.
- Les "voisins actifs" sont ceux qui n'ont pas encore quitté le synchroniseur.
- "Quitter le synchroniseur" lui fera dire "j'ai fini" en fin de ce battement.
  (Souvenez-vous : quand un processus n'a plus besoin de synchronisation,
  il dit "j'ai fini" au lieu de "je suis prêt" en fin de battement)

Découvrir la topologie

Propriétés

Propriétés

Messages par battement

Taille d'un message

Mémoire par processus

Propriétés

Messages par battement

2 par lien, donc O(E)

Taille d'un message

Mémoire par processus

Propriétés

Messages par battement

2 par lien, donc O(E)

Taille d'un message

Jusqu'à O(V+E)

Donc O(E) pour tout réseau connecté

(Parce que le réseau connecté ayant le moins de liens possible est un arbre, et a E = V-1 liens, donc V devient négligeable asymptotiquement)

Mémoire par processus

Propriétés

Messages par battement

2 par lien, donc O(E)

Taille d'un message

Jusqu'à O(V+E)

Donc O(E) pour tout réseau connecté

(Parce que le réseau connecté ayant le moins de liens possible est un arbre, et a E = V-1 liens, donc V devient négligeable asymptotiquement)

Mémoire par processus

Comme les messages, O(E).

Amélioration ?

Comment réduire la taille des messages ?

Amélioration ?

Comment réduire la taille des messages ?

Chaque lien est stocké en double !

Amélioration ?

Comment réduire la taille des messages ?

Chaque lien est stocké en double !

Envoyer un set de liens connus.

Amélioration ?

Comment réduire la taille des messages ?

Chaque lien est stocké en double !

Envoyer un set de liens connus.

C'est tout ?

Amélioration ?

Comment réduire la taille des messages ?

Chaque lien est stocké en double !

Envoyer un set de liens connus.

C'est tout ?

Envoyer aussi la liste des processus dont on connait les voisins
(nécessaire pour savoir quand s'arrêter)

Amélioration ?

Comment réduire la taille des messages ?

Chaque lien est stocké en double !

Envoyer un set de liens connus.

C'est tout ?

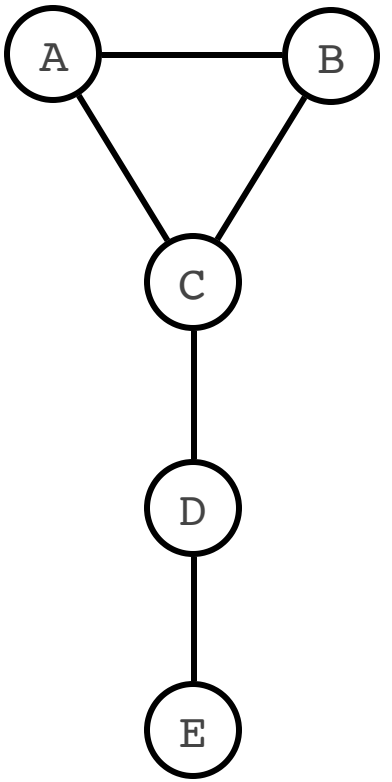
Envoyer aussi la liste des processus dont on connait les voisins
(nécessaire pour savoir quand s'arrêter)

En plus, connaitre le diamètre *d* du réseau et s'arrêter au battement *d+1*.

↓ Exercice

Exercice

Étant donné le réseau ci-contre, simuler l'algorithme synchronisé de découverte de topologie, en décrivant **l'état des dictionnaires à la fin de chaque battement**.



Multiplication de matrices

Un algorithme synchrone

Description du problème

Multiplication de matrices

Input : deux matrices nxn, A et B

Output : une matrice nxn, C = AB

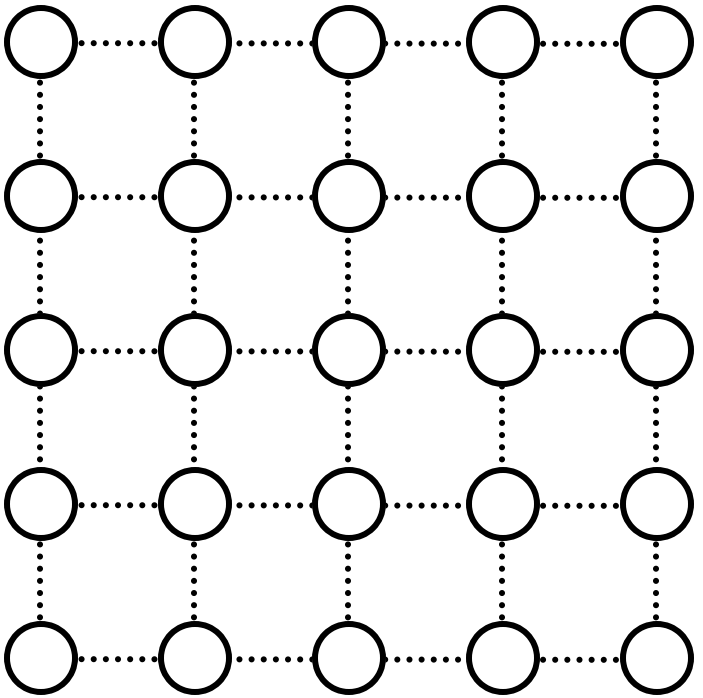
Ressources : n^2 processus

Remarque : parallélisation par distribution
Exemple parfait d'algorithme distribué dans le but d'améliorer les performances.

Arrangement des processus

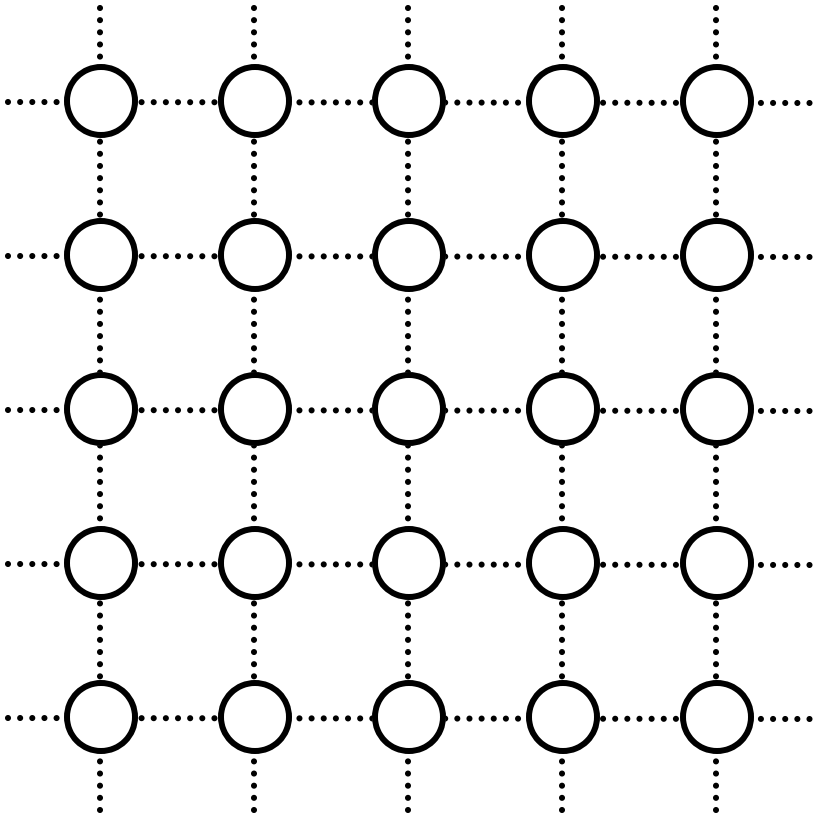
Arrangement des processus

Arrangement en grille



Arrangement des processus

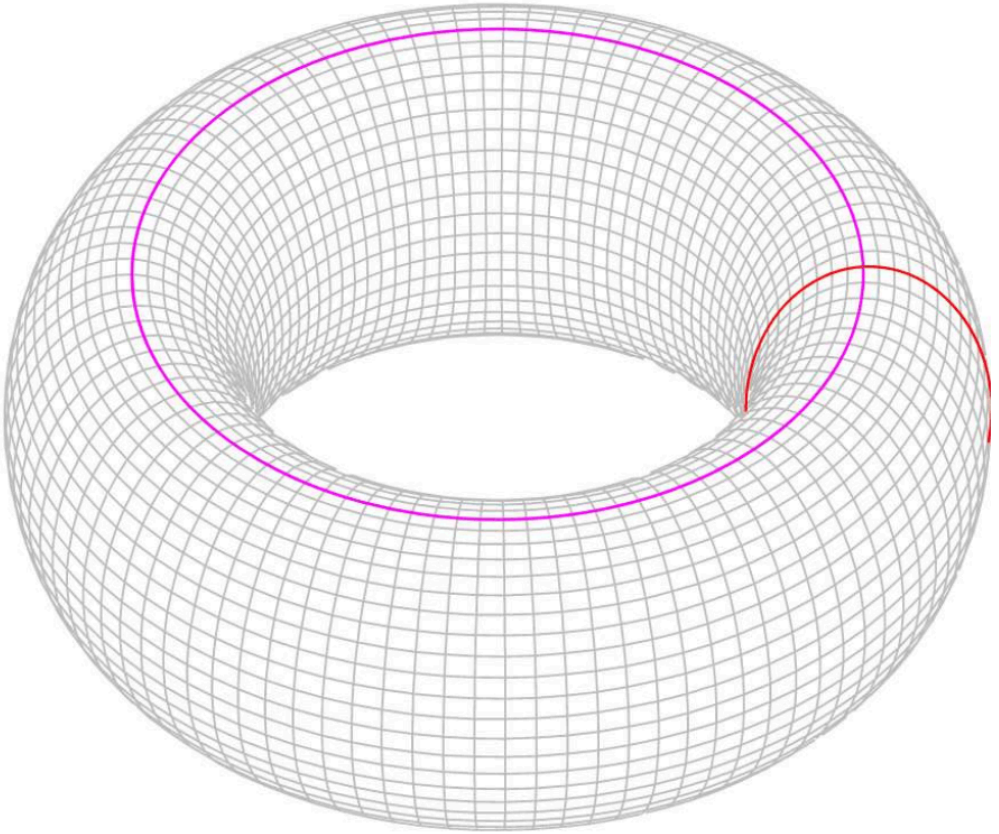
Arrangement ~~en grille~~
en tore !



Les processus d'une extrémité sont connectés à ceux de l'extrémité opposée.

Arrangement des processus

Arrangement ~~en grille~~
en tore !



Les processus d'une extrémité sont connectés à ceux de l'extrémité opposée.

Stratégie

Stratégie

Processus en (i,j) calculera $C_{ij} = \sum_{k \in [1,n]} A_{ik} \cdot B_{kj}$.

Stratégie

Processus en (i,j) calculera $C_{ij} = \sum_{k \in [1,n]} A_{ik} \cdot B_{kj}$.

À chaque battement k, (i,j)

- possède A_{ik} et B_{kj}
- les multiplie pour calculer C_{ij}
- les passe à ceux qui en ont besoin.

C_{ij} est donc calculé progressivement, en k battements.

Stratégie

Processus en (i,j) calculera $C_{ij} = \sum_{k \in [1,n]} A_{ik} \cdot B_{kj}$.

À chaque battement k, (i,j)

- possède A_{ik} et B_{kj}
- les multiplie pour calculer C_{ij}
- les passe à ceux qui en ont besoin.

C_{ij} est donc calculé progressivement, en k battements.

Quels A_{ik} et B_{kj} avoir initialement ?

Stratégie

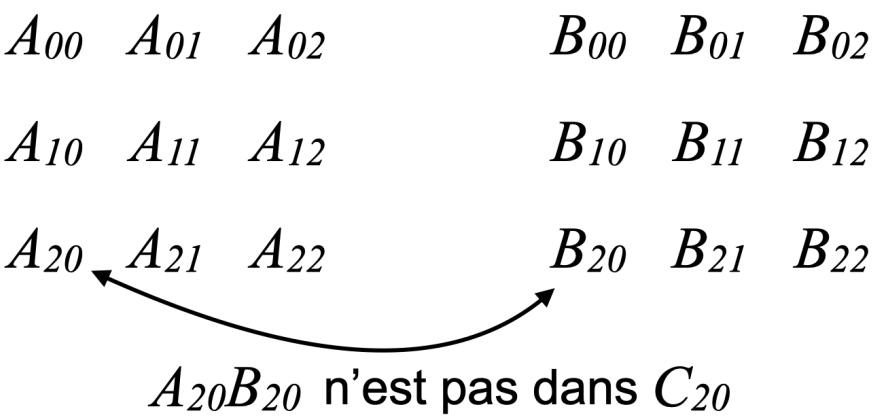
Processus en (i,j) calculera $C_{ij} = \sum_{k \in [1,n]} A_{ik} \cdot B_{kj}$.

À chaque battement k, (i,j)

- possède A_{ik} et B_{kj}
- les multiplie pour calculer C_{ij}
- les passe à ceux qui en ont besoin.

C_{ij} est donc calculé progressivement, en k battements.

Quels A_{ik} et B_{kj} avoir initialement ?
Avoir (i,j) qui possède Aij et Bij ne marche pas



Stratégie

Processus en (i,j) calculera $C_{ij} = \sum_{k \in [1,n]} A_{ik} \cdot B_{kj}$.

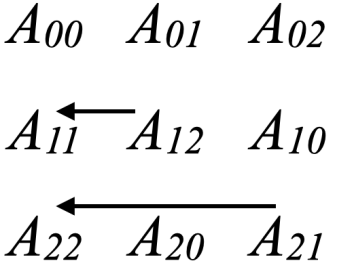
À chaque battement k, (i,j)

- possède A_{ik} et B_{kj}
- les multiplie pour calculer C_{ij}
- les passe à ceux qui en ont besoin.

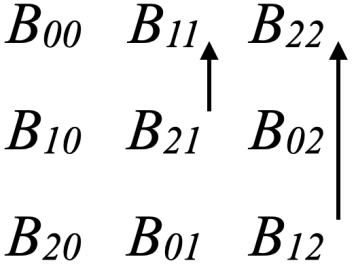
C_{ij} est donc calculé progressivement, en k battements.

Quels A_{ik} et B_{kj} avoir initialement ?

Shift A_{ij} de i vers la gauche.



Shift B_{ij} de j vers le haut.



Stratégie

Processus en (i,j) calculera $C_{ij} = \sum_{k \in [1,n]} A_{ik} \cdot B_{kj}$.

À chaque battement k, (i,j)

- possède A_{ik} et B_{kj}
- les multiplie pour calculer C_{ij}
- les passe à ceux qui en ont besoin.

C_{ij} est donc calculé progressivement, en k battements.

Quels A_{ik} et B_{kj} avoir initialement ?

Shift A_{ij} de i vers la gauche.

$$\begin{matrix} A_{00} & A_{01} & A_{02} \\ A_{11} & A_{12} & A_{10} \\ A_{22} & A_{20} & A_{21} \end{matrix}$$

Shift B_{ij} de j vers le haut.

$$\begin{matrix} B_{00} & B_{11} & B_{22} \\ B_{10} & B_{21} & B_{02} \\ B_{20} & B_{01} & B_{12} \end{matrix}$$

(i,j) a A_{ik} avec $k = i + j \mod n$

Stratégie

Processus en (i,j) calculera $C_{ij} = \sum_{k \in [1,n]} A_{ik} \cdot B_{kj}$.

À chaque battement k, (i,j)

- possède A_{ik} et B_{kj}
- les multiplie pour calculer C_{ij}
- les passe à ceux qui en ont besoin.

C_{ij} est donc calculé progressivement, en k battements.

Quels A_{ik} et B_{kj} avoir initialement ?

Shift A_{ij} de i vers la gauche.

$$\begin{matrix} A_{00} & A_{01} & A_{02} \\ A_{11} & A_{12} & A_{10} \\ A_{22} & A_{20} & A_{21} \end{matrix}$$

Shift B_{ij} de j vers le haut.

$$\begin{matrix} B_{00} & B_{11} & B_{22} \\ B_{10} & B_{21} & B_{02} \\ B_{20} & B_{01} & B_{12} \end{matrix}$$

(i,j) a A_{ik} avec $k = i + j \mod n$ et B_{kj} avec $k = i + j \mod n$

Stratégie

Processus en (i,j) calculera $C_{ij} = \sum_{k \in [1,n]} A_{ik} \cdot B_{kj}$.

À chaque battement k, (i,j)

- possède A_{ik} et B_{kj}
- les multiplie pour calculer C_{ij}
- les passe à ceux qui en ont besoin.

C_{ij} est donc calculé progressivement, en k battements.

Quels A_{ik} et B_{kj} avoir initialement ?

Shift A_{ij} de i vers la gauche.

$$\begin{matrix} A_{00} & A_{01} & A_{02} \\ A_{11} & A_{12} & A_{10} \\ A_{22} & A_{20} & A_{21} \end{matrix}$$

Shift B_{ij} de j vers le haut.

$$\begin{matrix} B_{00} & B_{11} & B_{22} \\ B_{10} & B_{21} & B_{02} \\ B_{20} & B_{01} & B_{12} \end{matrix}$$

Ainsi, (i,j) a A_{ik} et B_{kj} tels que $k = i + j \mod n$

Stratégie

Processus en (i,j) calculera $C_{ij} = \sum_{k \in [1,n]} A_{ik} \cdot B_{kj}$.

À chaque battement k, (i,j)

- possède A_{ik} et B_{kj}
- les multiplie pour calculer C_{ij}
- les passe à ceux qui en ont besoin.

C_{ij} est donc calculé progressivement, en k battements.

Quels A_{ik} et B_{kj} avoir initialement ?

(i,j) commence avec A_{ik} et B_{kj} tels que $k = i + j \mod n$

Stratégie

Processus en (i,j) calculera $C_{ij} = \sum_{k \in [1,n]} A_{ik} \cdot B_{kj}$.

À chaque battement k, (i,j)

- possède A_{ik} et B_{kj}
- les multiplie pour calculer C_{ij}
- les passe à ceux qui en ont besoin.

C_{ij} est donc calculé progressivement, en k battements.

Quels A_{ik} et B_{kj} avoir initialement ?

(i,j) commence avec A_{ik} et B_{kj} tels que $k = i + j \mod n$

Comment changer de A_{ik} et B_{kj} à chaque battement ?

Stratégie

Processus en (i,j) calculera $C_{ij} = \sum_{k \in [1,n]} A_{ik} \cdot B_{kj}$.

À chaque battement k, (i,j)

- possède A_{ik} et B_{kj}
- les multiplie pour calculer C_{ij}
- les passe à ceux qui en ont besoin.

C_{ij} est donc calculé progressivement, en k battements.

Quels A_{ik} et B_{kj} avoir initialement ?

(i,j) commence avec A_{ik} et B_{kj} tels que $k = i + j \mod n$

Comment changer de A_{ik} et B_{kj} à chaque battement ?

On veut passer à $k' = k + 1 \mod n$

Stratégie

Processus en (i,j) calculera $C_{ij} = \sum_{k \in [1,n]} A_{ik} \cdot B_{kj}$.

À chaque battement k, (i,j)

- possède A_{ik} et B_{kj}
- les multiplie pour calculer C_{ij}
- les passe à ceux qui en ont besoin.

C_{ij} est donc calculé progressivement, en k battements.

Quels A_{ik} et B_{kj} avoir initialement ?

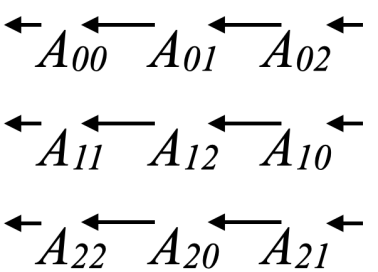
(i,j) commence avec A_{ik} et B_{kj} tels que $k = i + j \mod n$

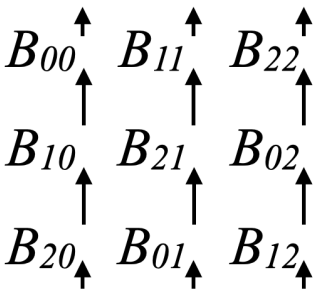
Comment changer de A_{ik} et B_{kj} à chaque battement ?

On veut passer à $k' = k + 1 \mod n$

On peut simplement envoyer

- la valeur de A vers la gauche
- la valeur de B vers le haut





Multiplication de matrices

Résumé & Pseudocode

Algorithme de Cannon

Résumé

En tant que processus (i,j)

Je suis en charge de C_{ij}, initialement 0.

Je commence avec A_{ik} et B_{kj} tels que $k = i + j \mod n$

À chaque battement

- J'ajoute $A_{ik} \cdot B_{kj}$ à C_{ij}
- J'envoie A_{ik} au voisin à gauche
- J'envoie B_{kj} au voisin au dessus
- Je reçois les A_{ik} et B_{kj} pour mon prochain battement

Algorithme de Cannon

Pseudocode

Variables

`n`	*entier, constant*	Largeur des matrices
`Aik`	*nombre*	Valeur de A en ma possession
`Bkj`	*nombre*	Valeur de B en ma possession
`Cij`	*nombre, initialement 0*	Somme intermédiaire

Avec initialement,

- `Aik = A[i][(i+j)%n]`
- `Bkj = B[(i+j)%n][j]`
- `Cij = 0`

Algorithme de Cannon

Pseudocode

À chaque nouveau battement `b` (en commençant à 0)

Si `b > 0`

    `Aik` ← message reçu de la droite au battement précédent

    `Bkj` ← message reçu d'en dessous au battement précédent

`Cij ← Cij + Aik * Bkj`

Envoi de `Aik` à ma gauche

Envoi de `Bkj` au dessus

Si `b` vaut `n-1`

    Quitter le synchroniseur

Multiplication de matrices

Propriétés

Propriétés

Messages par battement

Taille d'un message

Mémoire par processus

Propriétés

Messages par battement

Un par lien.

Donc 2V

Donc 2n^2

Taille d'un message

Mémoire par processus

Propriétés

Messages par battement

Un par lien.

Donc 2V

Donc 2n^2

Taille d'un message

Constante

Mémoire par processus

Propriétés

Messages par battement

Un par lien.

Donc 2V

Donc 2n^2

Taille d'un message

Constante

Mémoire par processus

Constante

↓ Exercice

Exercice

À chaque battement, quelles valeurs intermédiaires ont chaque processus lorsqu'ils exécutent l'algorithme de Cannon pour effectuer la multiplication suivante.

$$C = A \cdot B = \begin{bmatrix} 2 & 1 & 0 \\ 0 & 1 & 2 \\ 3 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 & 1 & 0 \\ 0 & 2 & 1 \\ 0 & 3 & 2 \end{bmatrix}$$