Correction Exercice 3

Échange total

Notre troisième (et dernière) procédure est l'échange total, ou *all-to-all*. Chaque processeur P_k veut envoyer un message à tous les autres : il s'agit donc de p diffusions simultanées. Supposons encore tous les messages de même longueur L. Au départ, chaque processeur dispose d'un message stocké à l'adresse my_adr. À la fin, tous les processeurs auront en mémoire le même tableau, contenant à la case k le message diffusé par le processeur numéro k. L'algorithme est évident : on fait tourner les messages en $p-1$ étapes de communication. On a la procédure :

```
ALL_TO_ALL(my_adr,adr,L)
1:  q ← MY_NUM()
2:  p ← TOT_PROC_NUM()
3:  adr[q] ← my_adr
4:  Pour i = 1 à p − 1 :
5:      SEND(adr[q − i + 1],L) || RECEIVE(adr[q − i],L)
```

Les commentaires au sujet de la sémantique sont les mêmes que pour la diffusion personnalisée. Le temps d'exécution est encore $(p-1)(\beta + L\tau)$, et cette fois tous les liens de communication sont utilisés à plein régime.

Correction Exercice 4

Bleu et Rouge => N et P

Procédure mystère (Sélection dans une liste)

► **Question 1.** On utilise la technique de saut de pointeur vu précédemment pour concevoir l'algorithme 1.6. Chaque processeur détermine le processeur bleu qui le suit dans la liste en temps $O(\log n)$.

Cet algorithme est bien EREW, mais cela demande une petite explication. Il fonctionne comme l'algorithme du saut de pointeur en parallèle sur plusieurs listes dont le nœud terminal est bleu ou *NIL*, chaque processeur conservant un pointeur sur la fin de sa liste, donc sur le prochain élément bleu. Chaque saut de pointeur se fait sur des listes indépendantes et n'interfère pas avec les autres. À la fin de l'algorithme, la liste des éléments bleus commence avec le premier élément de la liste initiale si ce dernier est bleu et avec son successeur bleu sinon.

```
EXTRAIT-BLEUS()
 1:  Pour tout i en parallèle :
 2:      Si suivant(i) = NIL Ou couleur(suivant(i)) = bleu Alors
 3:          fini(i) ←Vrai
 4:          bleu(i) ←suivant(i)
 5:  Tant que il existe un nœud i tel que fini(i) = Faux :
 6:      Pour tout i en parallèle :
 7:          Si fini(i) = Faux Alors
 8:              fini(i) ← fini(suivant(i))
 9:              Si fini(i) = TRUE Alors bleu(i) ← bleu(suivant(i))
10:              suivant(i) ← suivant(suivant(i))
```

Algorithme 1.6 Algorithme de séparation de listes.