République Algérienne Démocratique et Populaire

Rapport du TP algorithmique distribuée

*Auteur :*IKHLEF Ali
Filière : Génie Informatique
Spécialité : RSI

Année Universitaire
2020/2021

Table des matières

Table des figures

Liste des tableaux

Lab 1

Diffusion de l'information

la complexité du problème de diffusion en term de nombre de message et de temps est :
$M[\text{diffuse}/\mathbf{RI+}] = O(m)$ et$T[\text{diffuse}/\mathbf{RI+}] = O(d(G))$

Restrictions

— l'ensemble des restrictions standard $\mathbf{R} = \{$BL, CN, TR$\}$, [Liens bidirectionnels (BL),Connectivité (CN) et Fiabilité(TR)].
— L'initiateur unique est celui qui détient l'information initiale **UI+**,

Diffuse_v0

.

1) initiator $\times S_p \to$ { send(I) to $N(x)$;}
2) idle $\times R_{msg} \to$ { Process(I) ; send(I) to $N(x)$; }
3) initiator$\times R_{msg} \to$ **nil**
4) idle $\times S_p \to$ **nil**

On constate que l'algorithme **diffuse_v0** ne s'arrêtera jamais car lorsqu'un nœud déjà informé reçoit un message ,il va le reenvoyé.

Diffuse_v1

Afin de résoudre le probleme vu en **diffuse_v0** on introduit un nouveau état **done** , donc un noeud ne reevoie pas de messages à ces voisins que lorsque il reçcoit un message pour la premier fois.

1) initiator $\times S_p \to$ { send(I) to $N(x)$; **become done**}
2) idle $\times R_{msg} \to$ { Process(I) ; send(I) to $N(x)$; **become done**}
3) initiator$\times R_{msg} \to$ **nil**
4) idle $\times S_p \to$ **nil**
5)**done**$\times R_{msg} \to$ **nil**
6)**done**$\times S_p \to$ **nil**

Diffuse_v2

Afin d'améliorer **diffuse_v1** on ne doit pas envoyer le message au **sender** l'algorithme devient :

1) initiator $\times S_p \rightarrow$ { send(I) to $N(x)$; **become done**}
2) idle $\times R_{msg} \rightarrow$ { Process(I) ; **become done**; $send(I)to$ N(x)-sender ;}
3) initiator$\times R_{msg} \rightarrow$ **nil**
4) idle $\times S_p \rightarrow$ **nil**
5)**done**$\times R_{msg} \rightarrow$ **nil**
6)**done**$\times S_p \rightarrow$ **nil**

topologie	n	m	$M_{diffuse\_v1}$	$M_{diffuse\_v2}$
$FullGraph_4$	4	$\frac{n(n-1)}{2}$	24	21
$FullGraph_5$	5		40	36
$FullGraph_6$	6		60	55
$FullGraph_{100}$	100		19800	19701
$HyperCube_{k=2}$	2^2	$\frac{n\times k}{2}=4$	16	13
$HyperCube_{k=3}$	2^3	12	48	41
$HyperCube_{k=4}$	2^4	32	128	113
$HyperCube_{k=10}$	2^{10}	5120	20480	19475
$BTree_{h=1}$	$2^{h+1}-1$	$n-1$	4	2
$BTree_{h=4}$			32	16
$BTree_{h=8}$			512	256
$BTree_{h=10}$			2048	1024
$Ring_4$	4	4	8	5
$Ring_{10}$	10	10	20	11
$Ring_{30}$	30	30	30	31
$Ring_{100}$	100	100	200	101
$Chain_3$	3	2	4	2
$Chain_{10}$	10	9	18	9
$Chain_{100}$	100	99	198	99
$Chain_{1000}$	1000	999	1998	999
$Random_{n=10,u(0,1)<\frac{1}{2}}$	10	29	82	73
$Random_{n=99,u(0,1)<\frac{1}{2}}$	99	7350	9708	9610

TABLE 1.1: Résultats de la simulation.

— Dans la topologie Arbre **diffuse_v2** améliore bien la complexité par un facteur de $\frac{1}{2}$ .Donc il est préférable d'utiliser cette version dans un tel arbre.
— Dans des topologie Random l'améloiration n'apparait pas.
— Dans d'autre topologie **diffuse_v2** dimunie le nombre de messages avec $n-1$

Diffuse_v3

.

1) initiator $\times S_p \rightarrow \{$ send$(I, N(x))$ to $N(x)$; **become done**$\}$
2) idle $\times R_{msg}(I, Z) \rightarrow \{$
Process(I) ; **become done** ; send(I) to $N(x) - sender$;
if$(Y = (N(x) - Z) \neq \varnothing)$ send$(I, Y \cup Z + x)$ to Y ;
$\}$
3) initiator$\times R_{msg}(I, Z) \rightarrow$ **nil**
4) idle $\times S_p \rightarrow$ **nil**
5)**done**$\times R_{msg}(I, Z) \rightarrow$ **nil**
6)**done**$\times S_p \rightarrow$ **nil**

L'algorithme **diffuse_v3** résoudre correctement le problème de diffusion de l'information .

Dans cette partie les résultas des simulations sont stockées dans le fichier **diffuse_v3_res.csv** pour la suite on va les utilisées pour determiner la complexité.

Remarques

— **diffuse_v3** souffre du problème que un noeud reçoie plusieurs messages, supposons que l'initiateur a n voisins qui ne sont pas reliées entre eux , ces derniers ont un voisin en commun. ce noeud reçoie plus d'un message du fait qu'aucune des noeud connue q'il est déja informé.
— l'itilisation de **diffuse_v3** sera donc préfirable dans des topologie spécifique .

Complexité de diffuse_v3

l'initiateur dans un graphe de n sommets envoie $|N(I)|$ messages à ses voisins et après chacun d'eux (x) refait la même chose en envoyant des messages à ses voisins sauf ceux qui sont dans le champs Z (l'initiateur et ses voisins $N(I) - x$).

D'où, on peut dire que ce dernier noeud exécute un algorithme de diffusion sur un graphe de n - N(I) sommets.

Donc $M_n = |N(I)| \times (1 + M_{n-|N(I)|})$

M_n : la comlplexié de l'algorithm dans un graph de n sommets, $N(I)$: l'ensemble de voisins de l'initiateur I.

De plus, on peut admettre que la complexité de cet algorithme ne dépend pas de n et m uniquement mais aussi de la disposition des liens.

L'etude est faite dans un graphe aleatoire dans les liens sont générés à l'aide d'une loi de probabilité Uniforme sur l'intervalles $[0, 1]$.
telle que si $U[0, 1] < alpha \Rightarrow (x, y) \in E(G)$, $alpha$ est un hyperparametre

Résultas et interprétations

FIGURE 1.1: résultats de la simulation avec des différents valeurs de *alpha*

$M = (2 \times m \times slope) \times n + intercept$

alpha	slope	intercept	score
0.2	0.5466	83.044	0.9986
0.4	0.3487	196.5706	0.9964
0.6	0.1953	309.5794	0.9869
0.8	0.0640	271.7635	0.9223

TABLE 1.2: résultats de la regression

- le nombre de messages **M** est en fonction de *alpha*, n .
- le nombres de messages **M** est polynomial en terme de nombre de liens **m**.
- lorsque la connectivitée augmente , le nombre de message diminue d'une manière sinificative , c'est l'inverse lorsque elle diminue .
- il est préférable donc d'utiliser **diffuse_v3** lorsqu'il s'agit des topologie fotement connectées.

Lab 2

Wake up

la complexité du problème de flooding en term de nombre de message et de temps est :

$2m \geq M[\text{Wflood}] \geq 2m - n - 1$
$M[\text{Wflood}/\mathbf{R}] = O(m)$ et $T[\text{Wflood}/\mathbf{R}] = O(d(G))$

Restrictions

— l'ensemble des restrictions standard $\mathbf{R}$ = {BL, CN, TR}, [Liens bidirectionnels (BL),Connectivité (CN) et Fiabilité(TR)].

Résultats

topologie	n	m	M_{wflood}
$Random_5$	5	5	13
$Random_6$	6	11	25
$Random_7$	7	12	30
$HyperCube_{k=3}$			46
$HyperCube_{k=4}$			122
$HyperCube_{k=5}$			306
$BTree_{h=3}$			14
$BTree_{h=4}$			25
$BTree_{h=5}$			50
$FullGraph_7$			82
$FullGraph_8$			110
$FullGraph_12$			261

TABLE 2.1: Résultats de la simulation Wflood.

Lab 3

Parcours

Restrictions

— l'ensemble des restrictions standard $\mathbf{R} = \{\text{BL, CN, TR}\}$, [Liens bidirectionnels (BL),Connectivité (CN) et Fiabilité(TR)].
— L'initiateur unique est celui qui détient l'information initiale **UI+**,

Algorithme de Terry

— la compléxité de cet algorithme est $M[\text{TerryTraversal}/\mathbf{R}] = O(2m)$, du fait qu'on a 2 messages exactement sur chaque lien.

Parcours par profondeur (DFT)

— $T[\text{DFT\_v0}] = M[\text{DFT\_v0}] = 2m$
— $T[\text{DFT\_v1}] = 4n - 2$ et $M[\text{DFT\_v1}] = 4m$
— $T[\text{DFT\_v2}] = 4n - 2$ et $M[\text{DFT\_v2}] \leq 4m - n + 1$
— $T[\text{DFT\_v3}] = 2n - 2$ et $M[\text{DFT\_v3}] \leq 4m - 2n + f^* + 1$, f^* disigne le nombre de feuilles dans le graphe

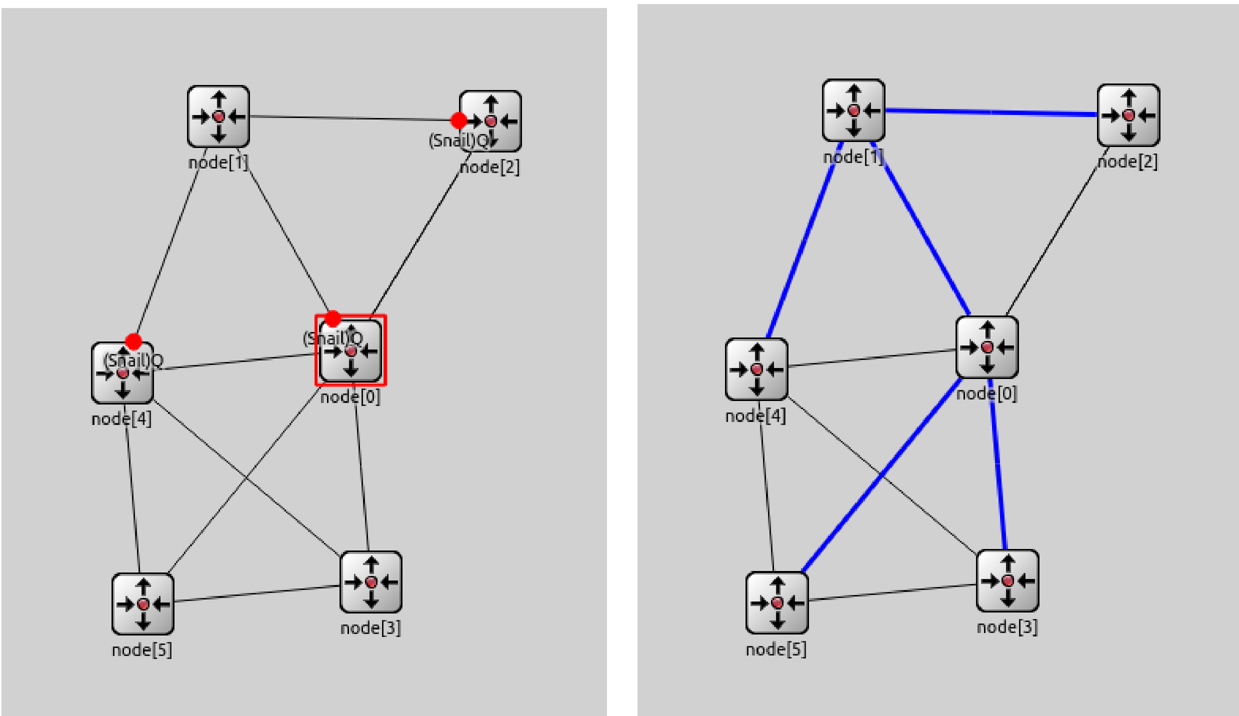
Lab 4

Construction d'un ACM

restrictions

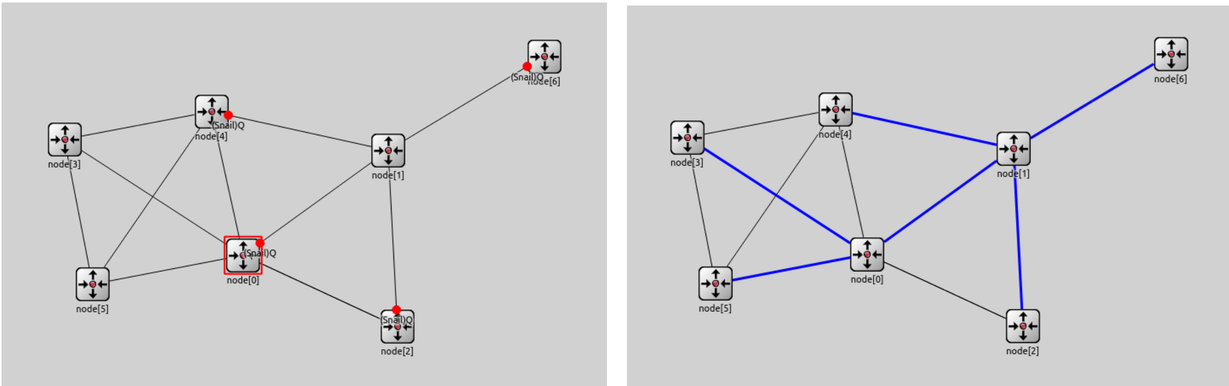
— l'ensemble des restrictions standard **R** = {BL, CN, TR}, [Liens bidirectionnels (BL),Connectivité (CN) et Fiabilité(TR)].
— racine unique(L'initiateur unique est celui qui lance le protocole) **UI+**.

Resultats



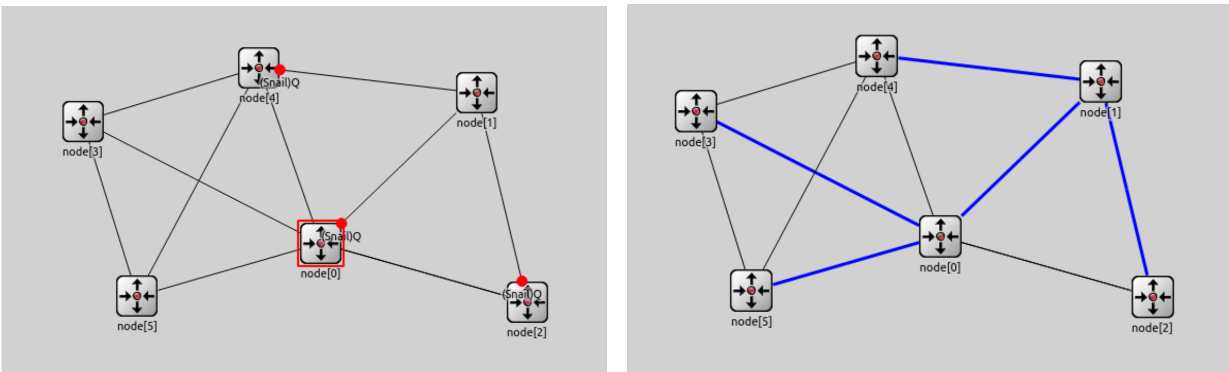
(a) topologie (b) resultats

FIGURE 4.1: exécution de ST_v0



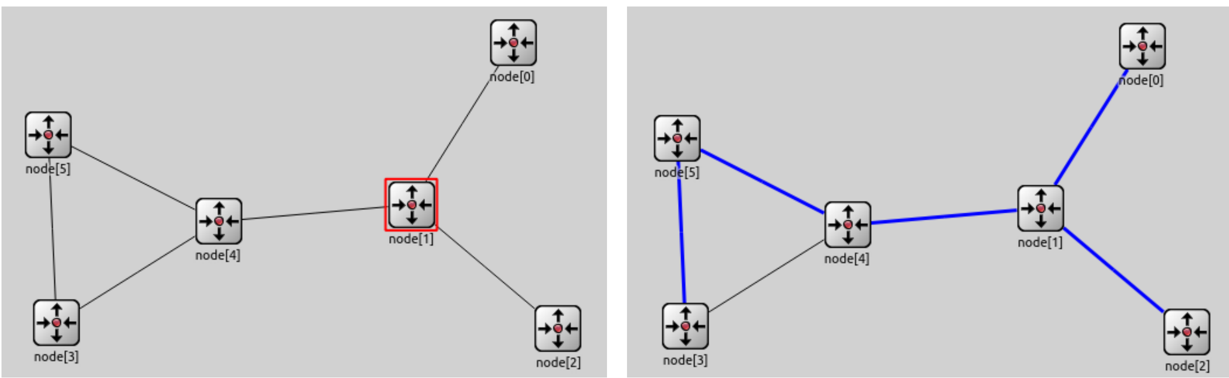
(a) topologie (b) resultats

FIGURE 4.2: exécution de ST_v1(Avec terminaison global)



(a) topologie (b) resultat

FIGURE 4.3: exécution de ST_v2(implicit negative acknowledgment)



(a) topologie (b) resultat

FIGURE 4.4: exécution de ST_DFT(ACM avec parcours DFT)

Lab 5
Routage

restrictions

- l'ensemble des restrictions standard **R** = {BL, CN, TR}, [Liens bidirectionnels (BL),Connectivité (CN) et Fiabilité(TR)].
- unique ID

topologie utilisée

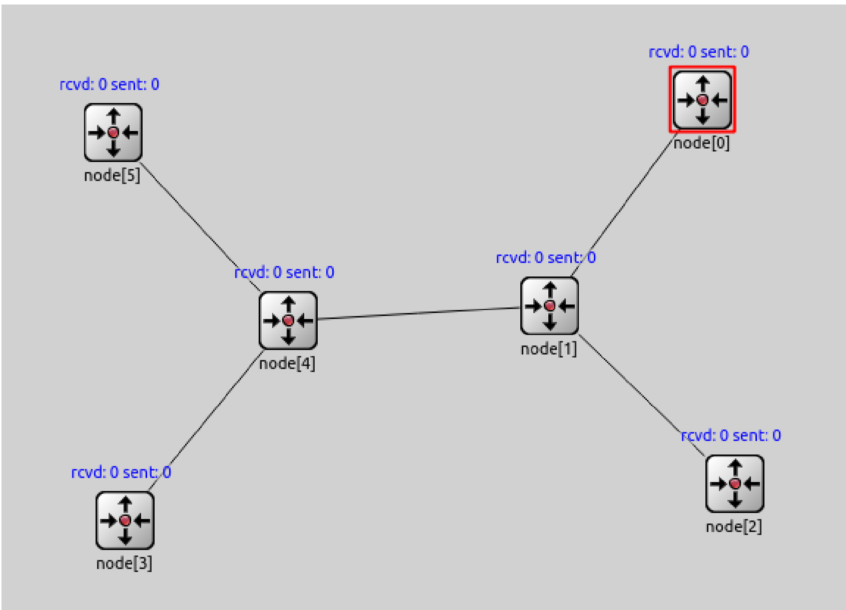
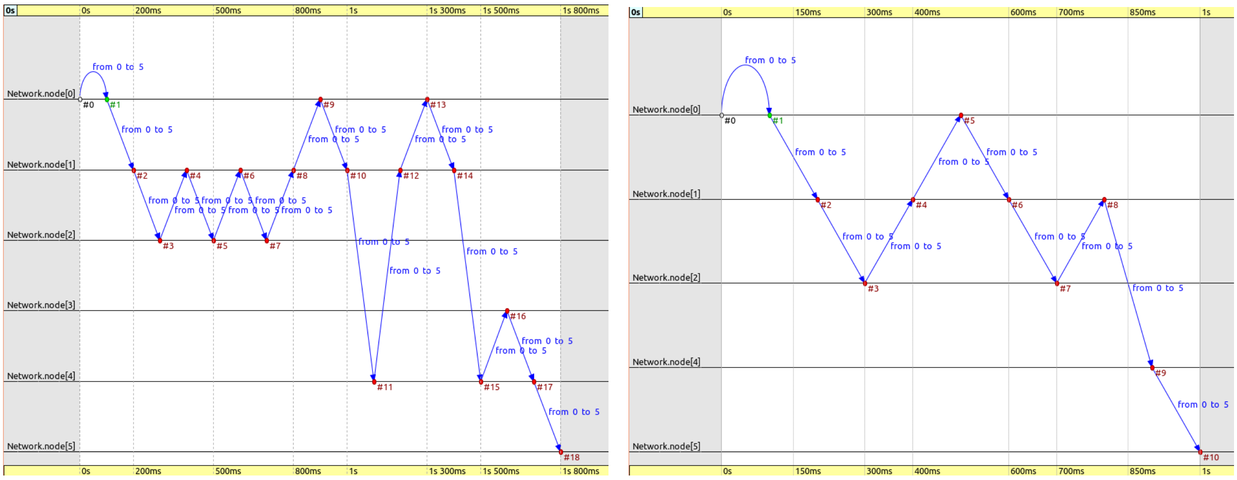


FIGURE 5.1: Arbre minimal de Steiner

Resultats



(a) RR_v0 (b) RR_v1

FIGURE 5.2: diagrammes espace-temps

noeud	hope_min	hope_max	hope_moyenne	hope_stddev
0	1	173	12.6541	12.366
1	1	46	3.79496	4.44399
2	1	117	12.4876	12.1094
3	1	134	12.7032	12.4133
4	1	43	3.87455	4.47895
5	1	127	12.5029	12.0173

TABLE 5.1: statistiques de RR_v0 .

noeud	hope_min	hope_max	hope_moyenne	hope_stddev
0	1	77	7.52897	6.44112
1	1	32	2.64143	2.59375
2	1	67	7.48133	6.38516
3	1	74	7.55505	6.50084
4	1	32	2.73866	2.64254
5	1	74	7.45265	6.40722

TABLE 5.2: statistiques de RR_v1 .

noeud	hope_min	hope_max	hope_moyenne	hope_stddev
0	0	71	3.60351	4.60219
1	0	1	0.397725	0.489433
2	0	46	3.58665	4.5868
3	0	54	3.5775	4.55982
4	0	1	0.40022	0.489908
5	0	49	3.58947	4.56117

Table 5.3: statistiques de RT_v0 .

noeud	hope_min	hope_max	hope_moyenne	hope_stddev
0	0	1	0.399993	0.489898
1	0	0	0	0
2	0	1	0.402886	0.49048
3	0	1	0.39784	0.489337
4	0	0	0	0
5	0	1	0.400211	0.489943

Table 5.4: statistiques de RT_v1 .

Interprétation et Comparaison

— Selon les diagrammes espace-temps (fig 5.2), on constate que le nombre de messages issues lors de l'exécution de **RR_v0** et nettememt supérieur à ceux de **RR_v1**. De plus on remarque l'existance de certaines pairs de noeuds interchangeants des messages(pour **RR_v0** noeuds 1 et 2). Or cet effet est reduit dans **RR_v1** car un noeud ne renvoie pas un message a son expéditeur.

— D'après la table 5.1 et 5.2, il est clair que le nombre moyen de hope est diminue avec l'utilisation du protocole de routage **RR_v1** par rapport à **RR_v0**.

— Le nombre moyen de hope du protocole **RT** est meilleur vu que l'utilisation d'une table de routage.

Lab 6

Arbre de diffusion

restrictions

— l'ensemble des restrictions standard $\mathbf{R}$ = {BL, CN, TR}, [Liens bidirectionnels (BL),Connectivité (CN) et Fiabilité(TR)].
— unique ID

Algorithme

Pour construire un arbre de diffusion, on a choisi de procéder comme suit :

1. Construire un ACM, on utilisant ST_1.
2. Lance l'algorithme de saturation modifié pour la determination du centre.
3. Commancer un parcours BFT depuis le centre.

Resultats

Saturation

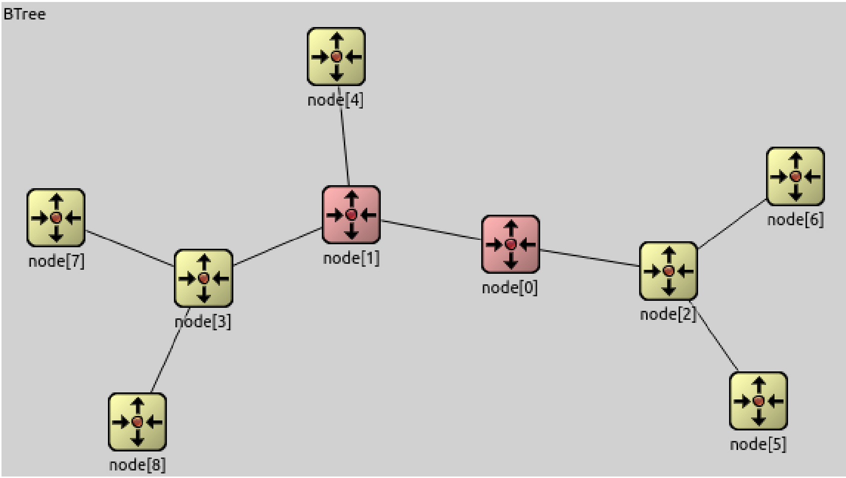
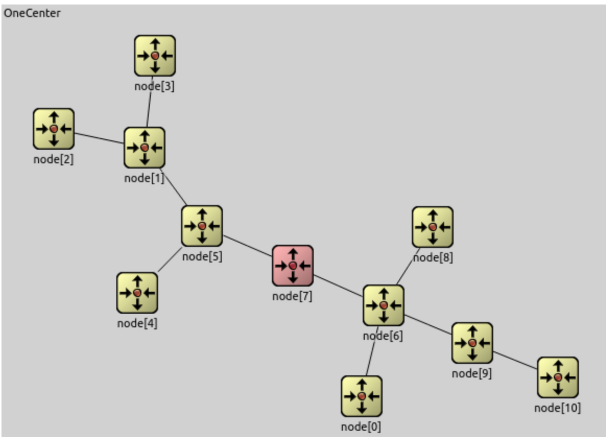
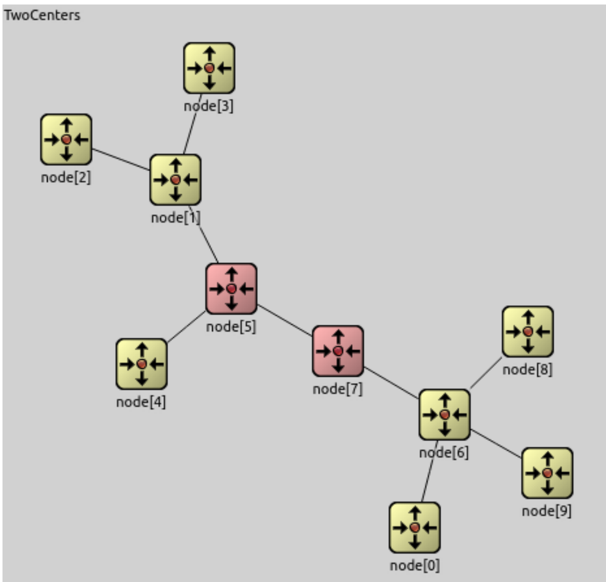


FIGURE 6.1: La saturation dans un arbre.

Determination du centre



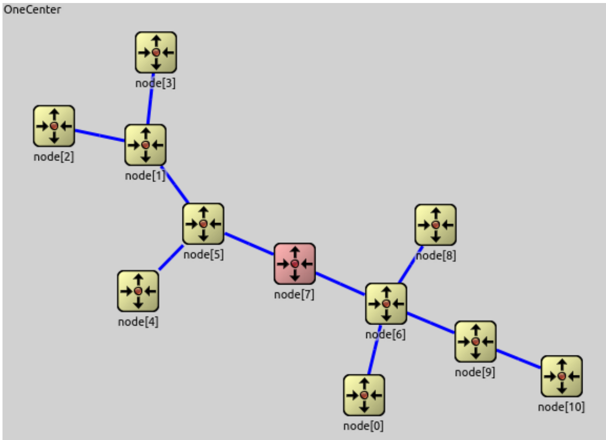
(a) un seul centre



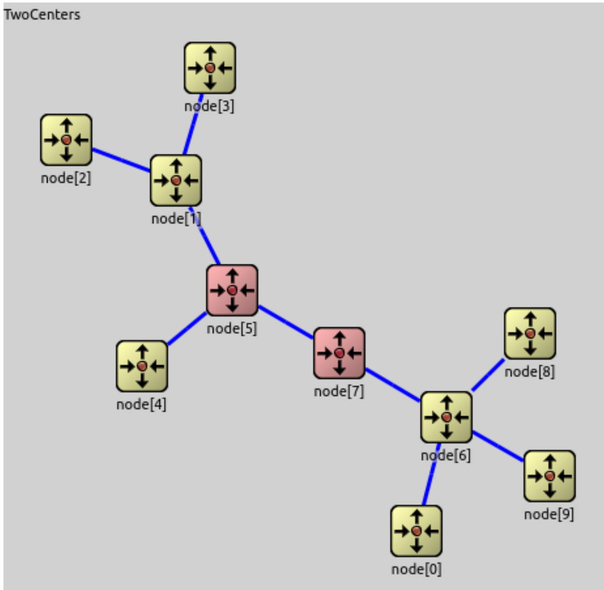
(b) deux centres voisins

FIGURE 6.2: Cas possible du centres d'un graphe

BFT à partir du centre

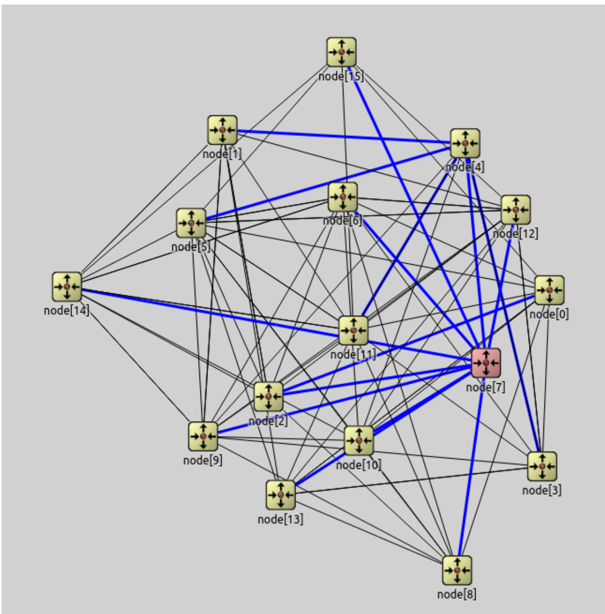


(a) Arbre avec un seul centre

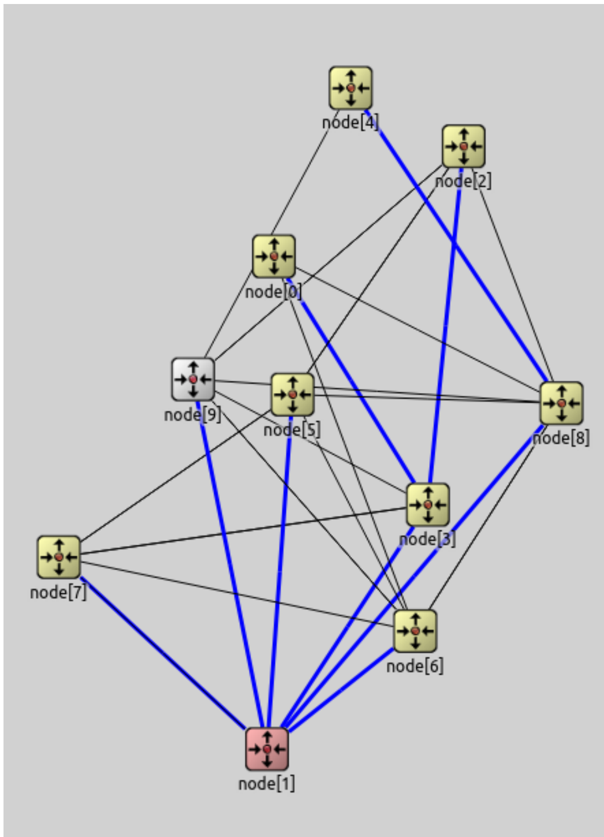


(b) Arbre avec deux centres voisins

FIGURE 6.3: Parcours BFT à partir du centre dans un arbre



(a) topologie 01



(b) topologie 02

FIGURE 6.4: Parcours BFT à partir du centre dans un graphe quelconque

Lab 7

Capture d'Etat global

Restrictions

— l'ensemble des restrictions standard **R** = {BL, CN, TR}, [Liens bidirectionnels (BL),Connectivité (CN) et Fiabilité(TR)].
— unique ID

Résultats

Pour la démonstration de l'exécution de l'algorithme de Chendy Lamport avec le code de couleurs, on a choisi l'algorithme de TikTok. le log (Fig 7.1) montre que **Token-1** est rentré dans l'état du canal ainsi que le diagramme d'espace temps correspondant (Fig 7.2).

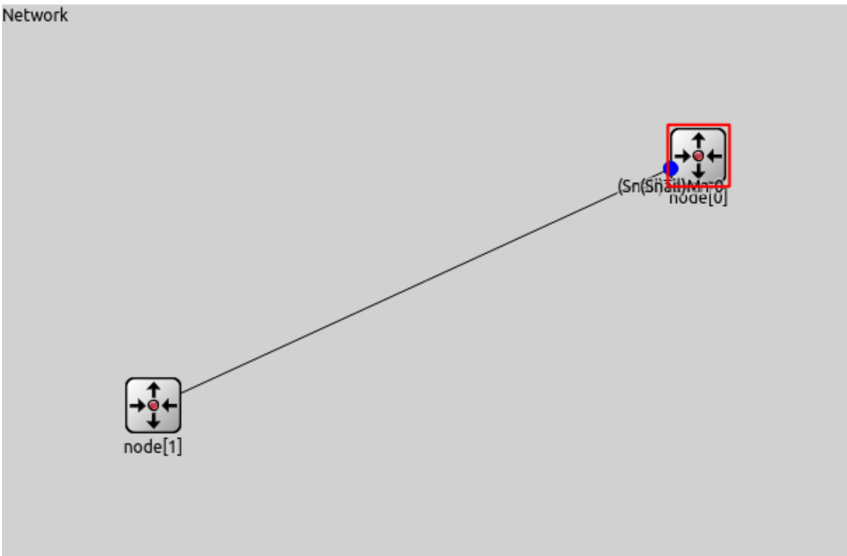


FIGURE 7.1: Topologie utilisée.

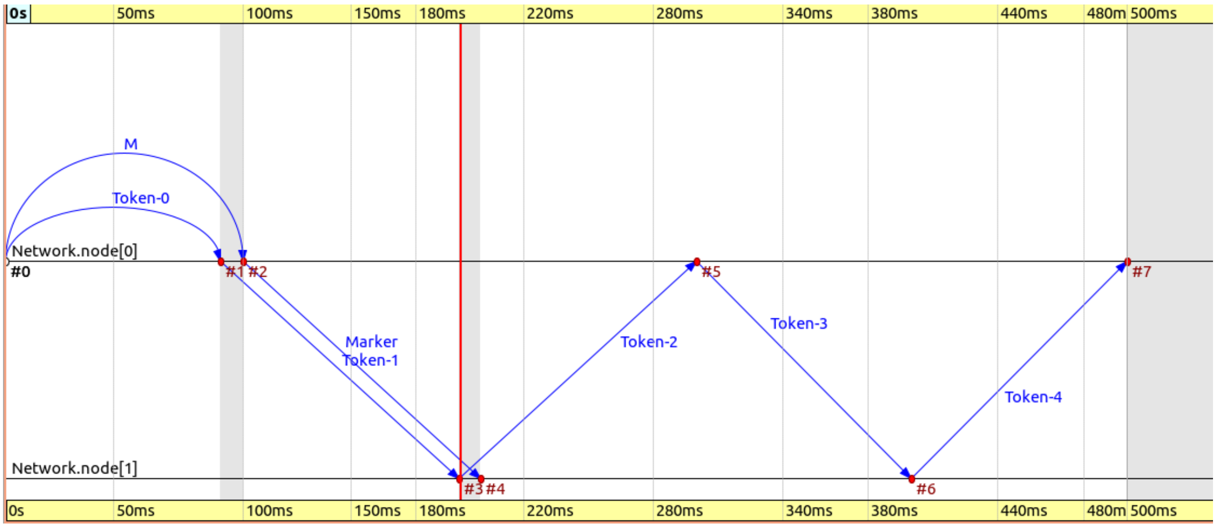


FIGURE 7.2: Diagrame espase temps de l'exécution.

```
** Initializing network
Initializing channel Network.node[0].gate$o[0].channel, stage 0
Initializing channel Network.node[1].gate$o[0].channel, stage 0
Initializing module Network, stage 0
Initializing module Network.node[0], stage 0
Initializing module Network.node[1], stage 0
** Event #1  t=0.1  Network.node[0] (Node, id=2)  on Token-0 (Snail, id=0)
** Event #2  t=0.1  Network.node[0] (Node, id=2)  on M (Snail, id=1)
INFO:turned Red
** Event #3  t=0.2  Network.node[1] (Node, id=3)  on Token-1 (Snail, id=2)
INFO:Token-1
** Event #4  t=0.2  Network.node[1] (Node, id=3)  on Marker (Snail, id=3)
** Event #5  t=0.3  Network.node[0] (Node, id=2)  on Token-2 (Snail, id=4)
** Event #6  t=0.4  Network.node[1] (Node, id=3)  on Token-3 (Snail, id=5)
** Event #7  t=0.5  Network.node[0] (Node, id=2)  on Token-4 (Snail, id=6)
<!> No more events, simulation completed -- at t=0.5s, event #7
** Calling finish() methods of modules
```

FIGURE 7.3: Log.

Lab 8

Problème de processus noirs

Restrictions

— l'ensemble des restrictions standard **R** = {BL, CN, TR}, [Liens bidirectionnels (BL),Connectivité (CN) et Fiabilité(TR)].
— Les identifiants des entités existent et ils sont uniques

Enoncé

Protocole utilisé

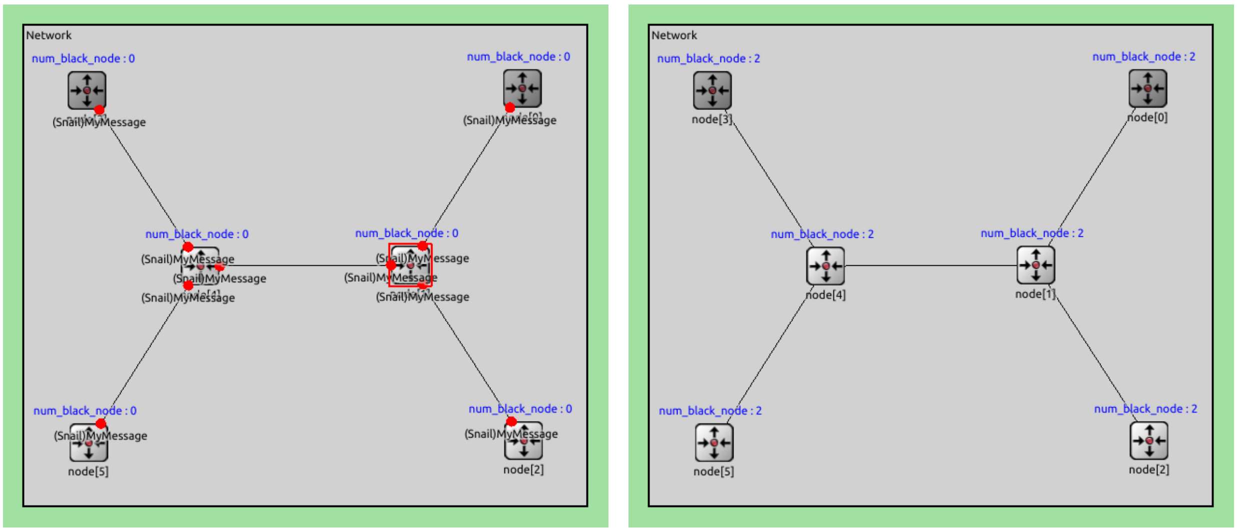
pour résoudre ce probleme un algorithme basé sur la diffusion est nécessaire.
$S_{init} = \{IDLE\}, S_{finale} = \{DONE\}$

1) $IDLE \times S_p \rightarrow \{$
$Unknown \leftarrow N(x);$
$if(BLACK)\{$
$Black\_nodes \leftarrow \{x\}$
$\}else\{$
$White\_nodes \leftarrow \{x\}$
$\}$
diffuse() ;
$\}$
2) idle $\times Recieve_{(B,W,U)} \rightarrow \{$
$Black\_nodes \leftarrow Black\_nodes \cup B;$
$White\_nodes \leftarrow White\_nodes \cup W;$
$Unknown \leftarrow Unknown \cup U - Black\_nodes - White\_nodes;$

$if(Unknown == \varnothing)\{$
become $DONE; // \#black\_nodes \leftarrow \mid Black\_nodes \mid$
$\}$
diffuse() ;
$\}$

procedude $diffuse()\{send(Black\_nodes, White\_nodes, Unknown)$ to $N(x)\,;\}$

Résultas



(a) RR_v0 (b) RR_v1

FIGURE 8.1: Etat avant et après l'exécution du protocole

All implementation are in my Github account.

Bibliographie

[1] Michel Raynal, *Distributed Algorithms for Message-Passing Systems.* Springer, 2013.

[2] Nicola Santoro, *Design and Analysis of Distributed Algorithms.* Wiley-Interscience, 2007

[3] Fokkink, Wan *Distributed algorithms an intuitive approach.* MIT Press, 2014

[4] Bang Ye Wu, Kun-Mao Chao *Spanning Trees and Optimization Problems.* Chapman & Hall_CRC, 2004.