Université de PIERRE et MARIE CURIE
MASTER 2 RES / RECH
Rapport de l'UE NRES

Protection de BitTorrent: conception et évaluation de deux solutions contre les attaques DoS

Réalisé par:

- **Abdelkrim HADJIDJ**
- **Djamel MOUCHENE**
- **kamelia KESSAL**

Introduction

Protecting BitTorrent: design and evaluation of effective countermeasures against DoS attacks

Marinho P. Barcellos, Daniel Bauermann, Henrique Sant'anna

L'architecture BitTorrent

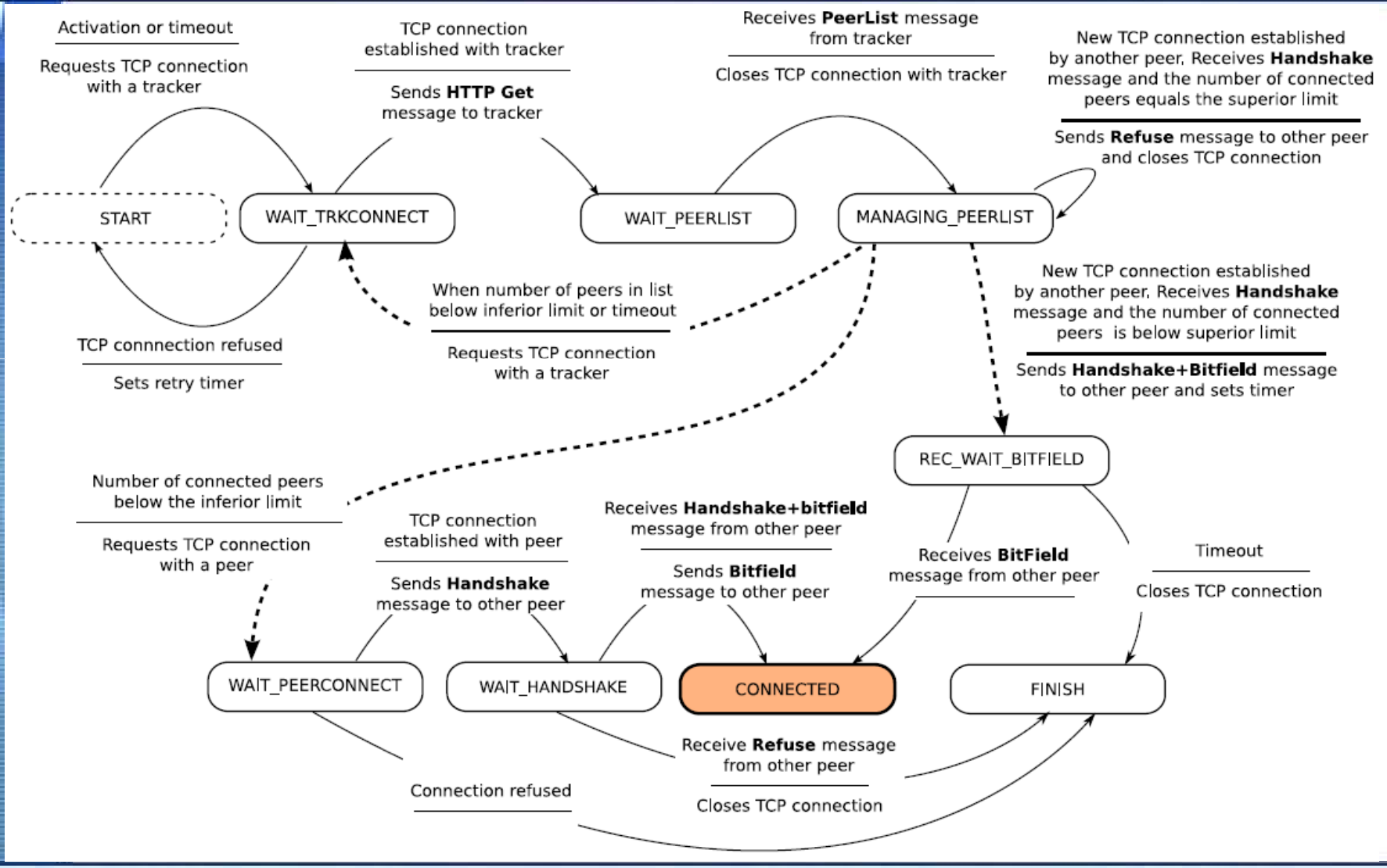
- Peers et Trackers
- Un contenu est un ensemble de fichiers

  Pièces de taille fixe : b_x ou $b_{x,*}$

  Blocs de 16 ko : $b_{x,y}$

- Seeds et Leechers
- Swarm : ensemble des peers qui s'intéressent au même contenu

Le fonctionnement de BitTorrent



Le fonctionnement de BitTorrent

- La liste des peers connu par p_i est appelé P_i
- Reconnexion périodique au Trackers

  Typiquement chaque 30 minutes

  À chaque connexion P_i <- P_i U L

- La liste des peers connectés à p_i est appelé A_i
- $|A_i| < A_{min}$: connexion sortantes et entrantes
- $A_{min} < |A_i| < A_{max}$: connexion entrantes

Diagramme d'état (connexion)

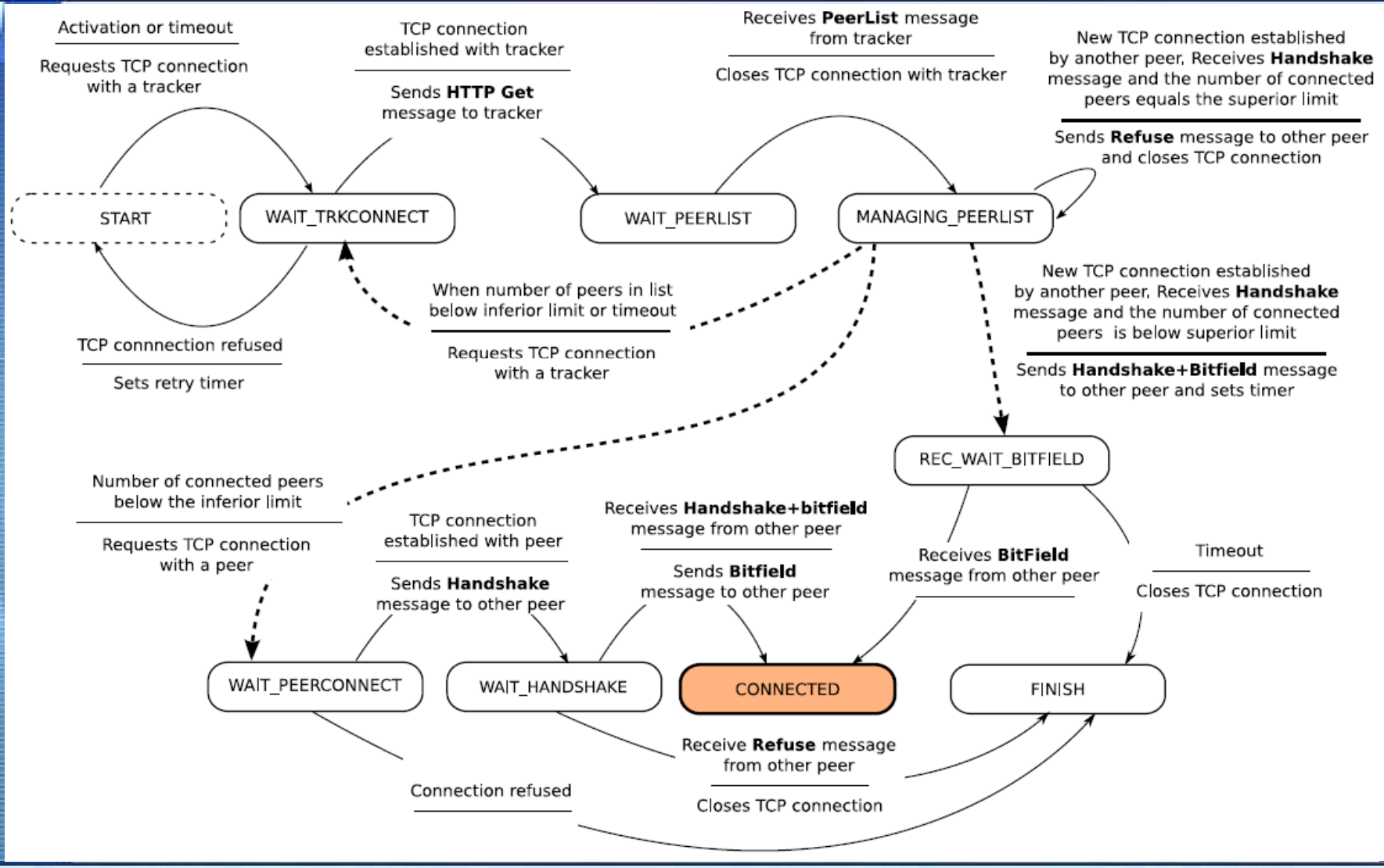
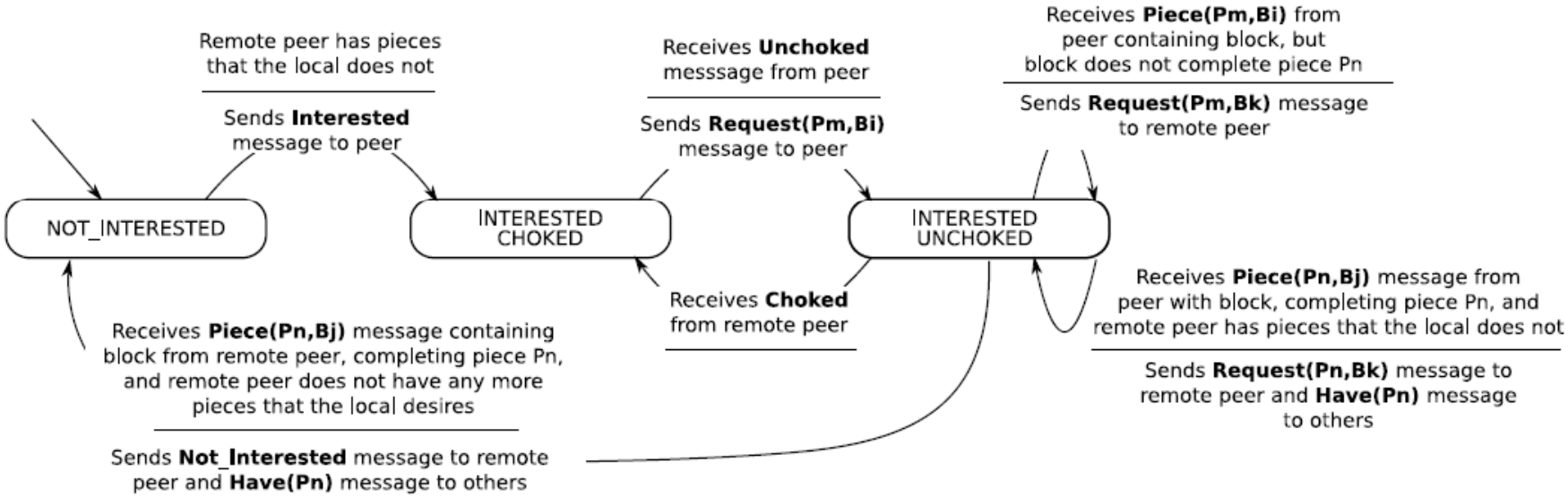


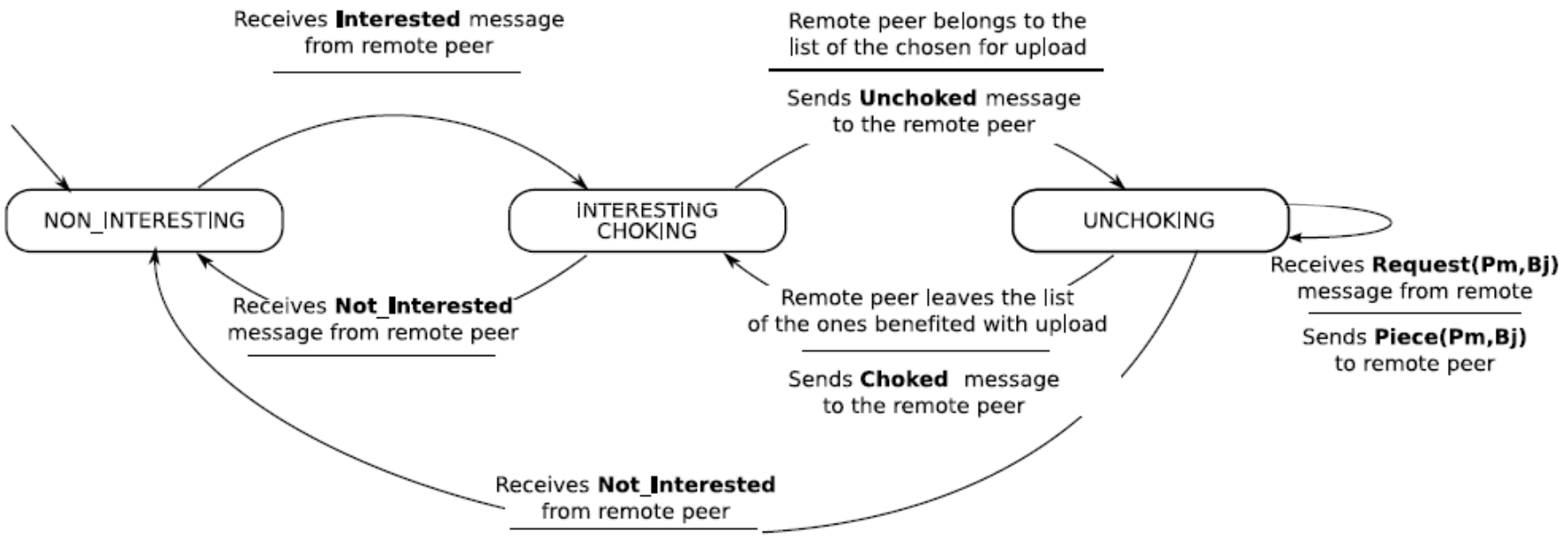
Diagramme d'états (download)



Incentive mechanisme

- Motiver les utilisateurs à uploader
- S'exécute en round de 10 secondes
- Choisit les trois meilleurs peers
- Optimistic Unchoking

Diagramme d'états (Upload)



Les attaques (1)

- L'attaque vise à profiter du réseau au maximum
  - Selfishness
  - Free - Riders
- Dénis de service
  - Sybils
  - Eclipsing correct peers
  - Piece Lying ou Massive Lying
  - Piece corruption

Les attaques (2)

Eclipsing correct peers

- Grand nombre de peers malicieux
- Un peer honnête ne se connecte qu'aux peers malicieux

$$\text{Si } |M| \gg |H|, \text{ alors } \frac{|P_i \cup M|}{|P_i|} \to 1$$

- p_i est éclipsé si : $\frac{|A_i \cup M|}{|A_i|} \to 1$

Les attaques (3)

Piece laying ou massive lying

- LRF Local Rarest First
- Attaque menée par un grand nombre de Sybils
- Diminution de la disponibilité des pièces
- Echec du Swarm si une pièce disparait

Les attaques (4)

Piece corruption

- Re-téléchargement de toute la pièce
- Pas de moyen de détecter les blocs corrompus
- Envoi d'un bloc corrompu et déconnexion
- Economie de la bande passante de l'attaquant

Solutions proposée (1)

- **Algorithme « PeerRotation »**

  ↳ **Contre les attaques « Massive Laying »**

- **Algorithme « Anti Corruption»**

  ↳ **Contre les attaques « Piece Corruption»**

Solutions proposée (2)

Objectif de la simulation

Répondre à quatres questions

Q1 : quelle est l'impact des attaques « massive lying » et « corruption attaque » sur les performances du réseau ?

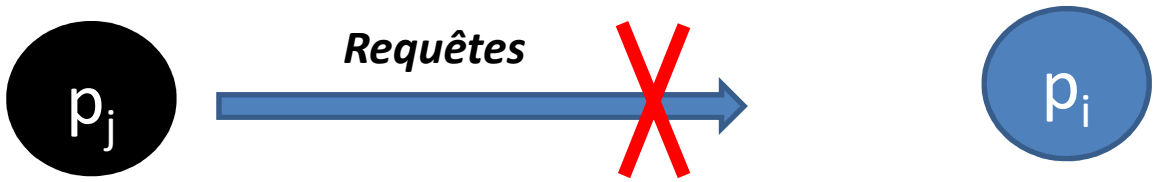
Q2 : quelle est l'efficacité des algorithmes proposé contre les attaques mentionnées dans Q1 ?
Et quel est l'impact de la contre attaque s'il ya des attaques par apport au cas optimal (le cas ou il n'y a ni Attaque, ni contre attaque) ?

Q3 : quel est le surcout en efficacité introduit par l'utilisation des algorithmes lorsqu'il n'y pas d'attaques.

Q4 : est ce que les algorithmes proposés identifient correctement les peers malicieux.

L'algorithme 1 : PeerRotation (1)

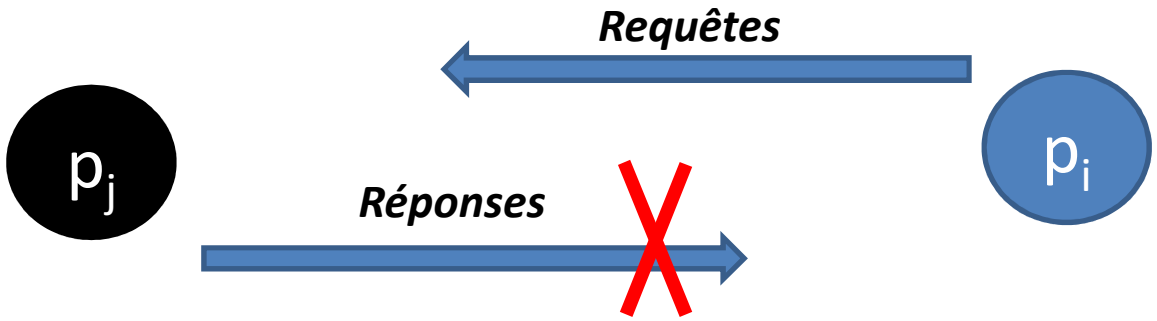
- **Le but est de détecter les peers (pairs) inactifs**



L'algorithme 1 : PeerRotation (2)

Principe

✓**Le but est de détecter les peers (pairs) inactifs**



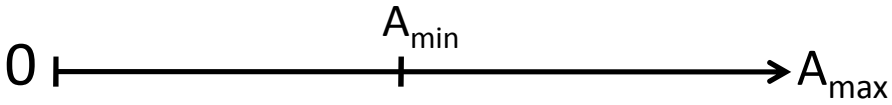
- p_j **est placé en quarantaine (l'ensemble Q_i).**
- $p_k \in (P_i \setminus (A_i \cup Q_i))$ **remplace** p_j.

Rotation entre p_j et p_k.

L'algorithme 1 : PeerRotation (3)

fonctionnement de l'algorithme :

- **Chaque pair pi exécute a son niveau le même algorithme.**
  - P_i : Ensemble de peers distant de p_i.
  - A_i : Ensemble de peers connectés à p_i.



  - Q_i : Ensemble des peers en quarantaines.
  - cq_j : La durée que passera p_j dans Qi, prochainement.

➡ **$(A_i \cup Q_i) \cap P_i$.**

L'algorithme 1 : PeerRotation (4)

1: **for all** $p_i \in Q$ **do**
2: $q_i \leftarrow q_i - 1$
3: **if** $q_i = 0$ **then**
4: $Q \leftarrow Q \setminus \{p_i\}$
5: **end if**
6: **end for**

Bloc 1

7: $a \leftarrow |P \setminus (A \cup Q)|$
8: $A' \leftarrow \{p_i, p_k, p_l \ldots \in A | \frac{d_j}{t_j} \leq \frac{d_k}{t_k} \leq \frac{d_l}{t_l} \ldots\}$
9: **for all** $p_j \in A'$ **do**
10: **if** $(t_j \geq t_{min} \wedge \frac{d_j}{t_j} < r_{min}) \wedge (|A| > A_{min} \vee (|A| \geq \lfloor \frac{3}{4} A_{min} \rfloor \wedge a > 0))$ **then**
11: $A \leftarrow A \setminus \{p_j\}$
12: $Q \leftarrow Q \cup \{p_j\}$
13: $q_j \leftarrow \lfloor c q_j \rfloor$
14: $c q_j \leftarrow c q_j \times f$
15: **if** $|A| < A_{min}$ **then**
16: $a \leftarrow a - 1$
17: **end if**
18: **end if**
19: **end for**

Bloc 2

20: **while** $|A| < A_{min} \wedge P \setminus (A \cup Q) \neq \emptyset$ **do**
21: $p_k \leftarrow \forall p_j \in P \setminus (A \cup Q)$
22: $A \leftarrow A \cup \{p_k\}$
23: $t_k \leftarrow 0$
24: $d_k \leftarrow 0$
25: **end while**

Bloc 3

L'algorithme 1 : PeerRotation (5)

fonctionnement de l'algorithme :

- **Bloc 1:**

```
Pour chaque (pj ∈ Qi ) faire
        Si (- - qj = 0)  Alors      // temps cqj écoulé.
                Qi ← Qi \ { pi }   // pi quitte la quarantaine
        Finsi
Fait
```

Bloc 1

NB: Initialement Qi est vide (évident).

L'algorithme 1 : PeerRotation (6)

Bloc 2 : Evaluation des peers connectés à p_i ,un par un.

P_j est suspecté

1. S'il n'atteint pas le taux r_{min} ($rap(d_j, t_j)$) au bout de t_{min}

ET

2.Si : Soit ($|A_i| \geq |A_{min}|$), **OU**

$$(|A_i| \geq |3/4\ A_{min}|) \wedge (P_i \setminus (A_i \cup Q_i) > 0)$$

Dès lors ;

Bloc 2;

L'algorithme 1 : PeerRotation (7)

Bloc 2 : Evaluation des peers connectés à p_i ,un par un.

Quand P_j est suspecté :

1. $A_i \leftarrow A_i \setminus \{p_j\}$ // *pj est déconnecté de Ai.*
2. $Q_i \leftarrow Q_i \cup \{p_j\}$ // *pj est inséré dans Qi.*
   $q_j \leftarrow cq_j$ // *pour une* durée cq_j.
3. $cq_j \leftarrow cq_j \times f$ // *pour la prochaine fois.*
4. Si $|A_i| < A_{min}$ alors // Droit : connexion sortante.
   $|(P_i \setminus (A_i \cup Q_i)|$ - - // le peer qui remplace p_j.

Bloc 2 (suite et fin).

L'algorithme 1 : PeerRotation (8)

Bloc 3 : Remplacer tout les peers mis en quarantaine.

Tant que ($|A_i| < A_{min}$) $\wedge$ ($P_i \setminus (A_i \cup Q_i) = 0$) faire

Choisir un peer $p_k \in P_i \setminus (A_i \cup Q_i)$ // Aléatoire.

$A_i \leftarrow A_i \cup \{p_k\}$ *// nouvelle connexion pour p_i*

$t_k \leftarrow 0$; // démarrer le compteur de connexion.

$d_k \leftarrow 0$; // initialisé le compteur de donnée.

Fait

Bloc 3

L'algorithme 1 : PeerRotation (9)

Résumé :

En résumé, l'intérêt de cet algorithme réside, dans le fait que les peers malicieux sont remplacés par peers honnêtes, ce qui évite qu'un peer honnête soit entouré de peer malicieux.

Simulation et évaluation(1)

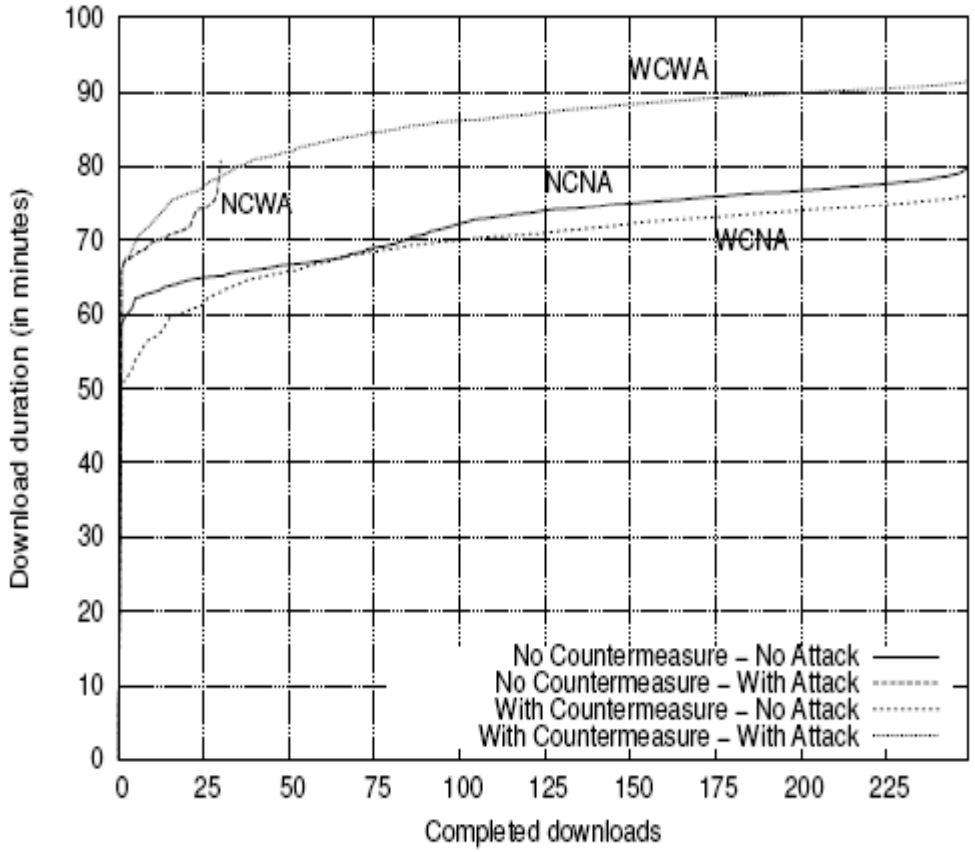
Scénarios de simulation :

	Sans Attaque	Avec Attaque
Sans Contre mesure	NCNA	NCWA
Avec Contre mesure	WCNA	WCWA

L'algorithme 1 : PeerRotation (3)

Simulation et évaluation

nombre de peers qui ont complété leurs téléchargements

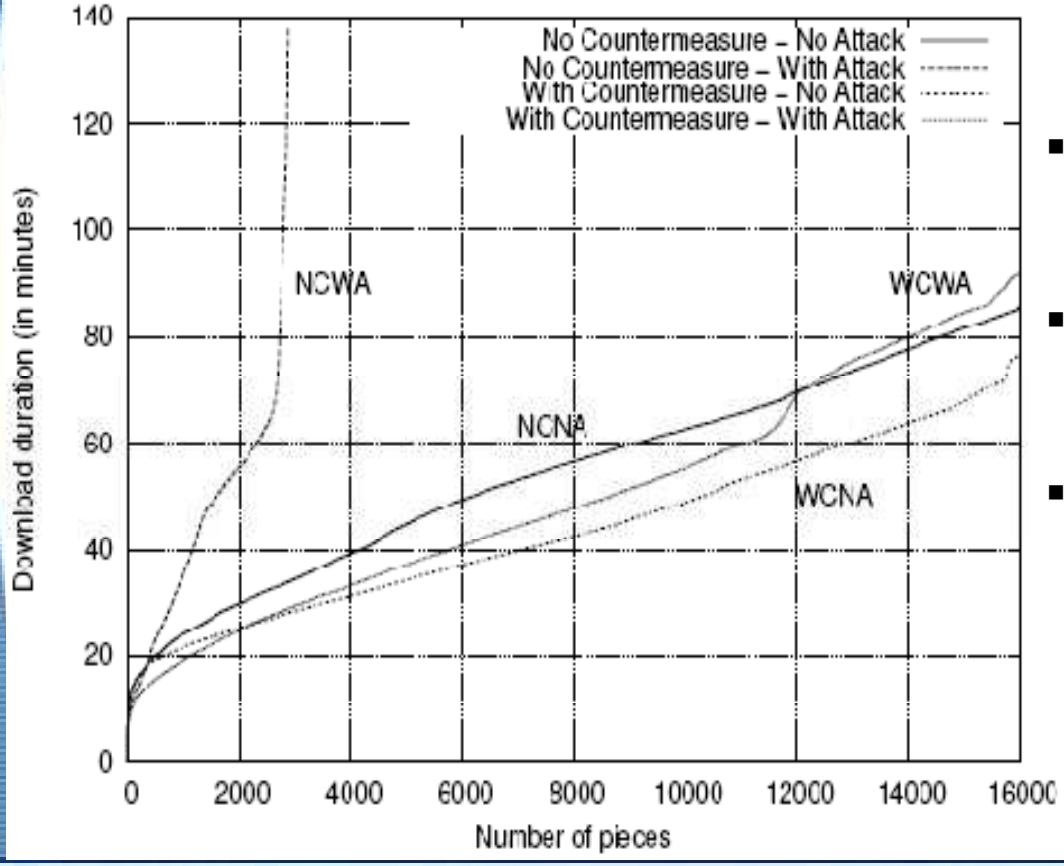


- **Question 1** ➡ **NCNA & NCWA**
- **Question 2** ➡ **- NCWA & WCWA**
  -WCWA & NCNA
- **Question 3** ➡ **NCNA & WCNA**

L'algorithme 1 : PeerRotation (4)

Simulation et évaluation

nombre de pièces correctement téléchargés



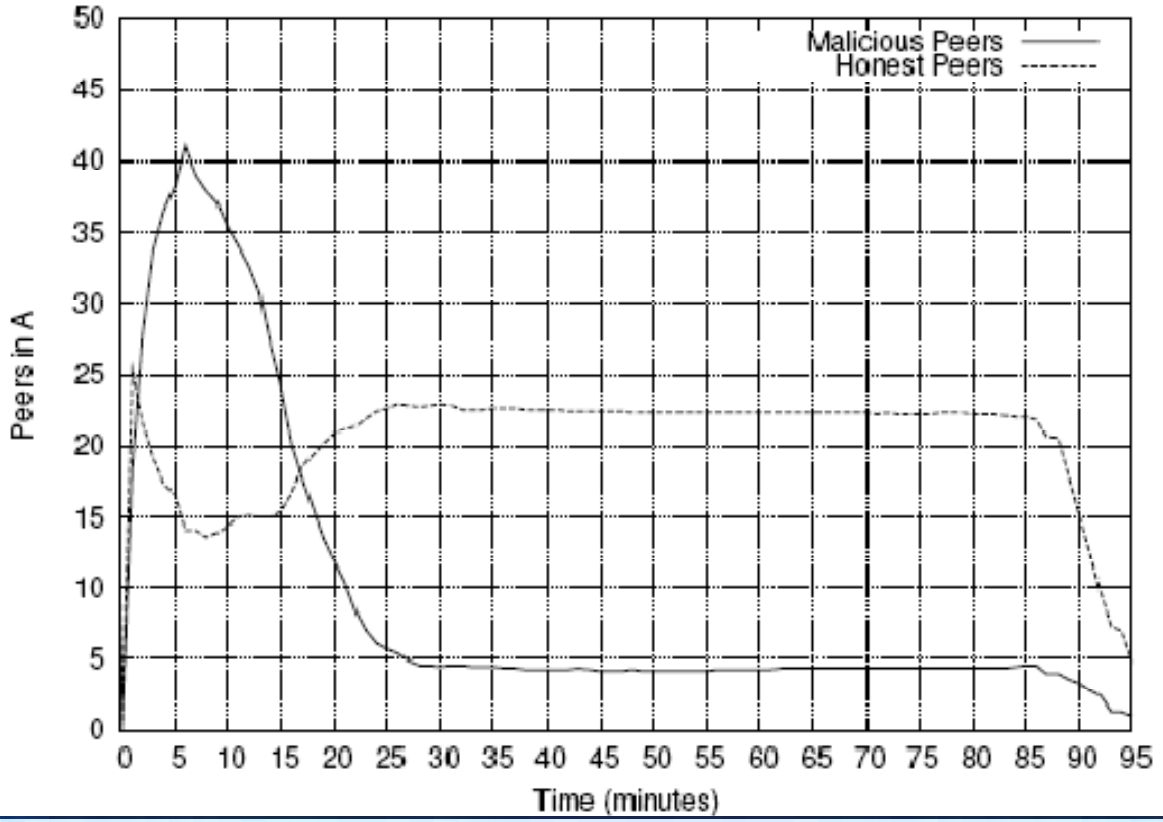
- **Question 1** ⇒ **NCNA & NCWA**
- **Question 2** ⇒ **- NCWA & WCWA**
  -WCWA & NCNA
- **Question 3** ⇒ **NCNA & WCNA**

L'algorithme 1 : PeerRotation (5)

Simulation et évaluation

Précision de l'algorithme

nombre de peers malicieux et honnête



- **Question 4**

Peer malicieux
&
Peer Honnête

L'algorithme 1 : PeerRotation (4)

Critiques

- solution non distribué.
  Une solution : Communiquer les comportements.

- Possibilité aux malicieux d'attaquer à nouveau.
  Une solution : fixer un seuil.

L'algorithme 2 : Anti Corruption (1)

Objectif

↳ **Arrêter les attaques « Piece Corruption »**

- Ces attaques nuisent aux performances du swarm

Permet de

↳ **- Identifier les peers malicieux**
- reconstruire la pièce corrompue

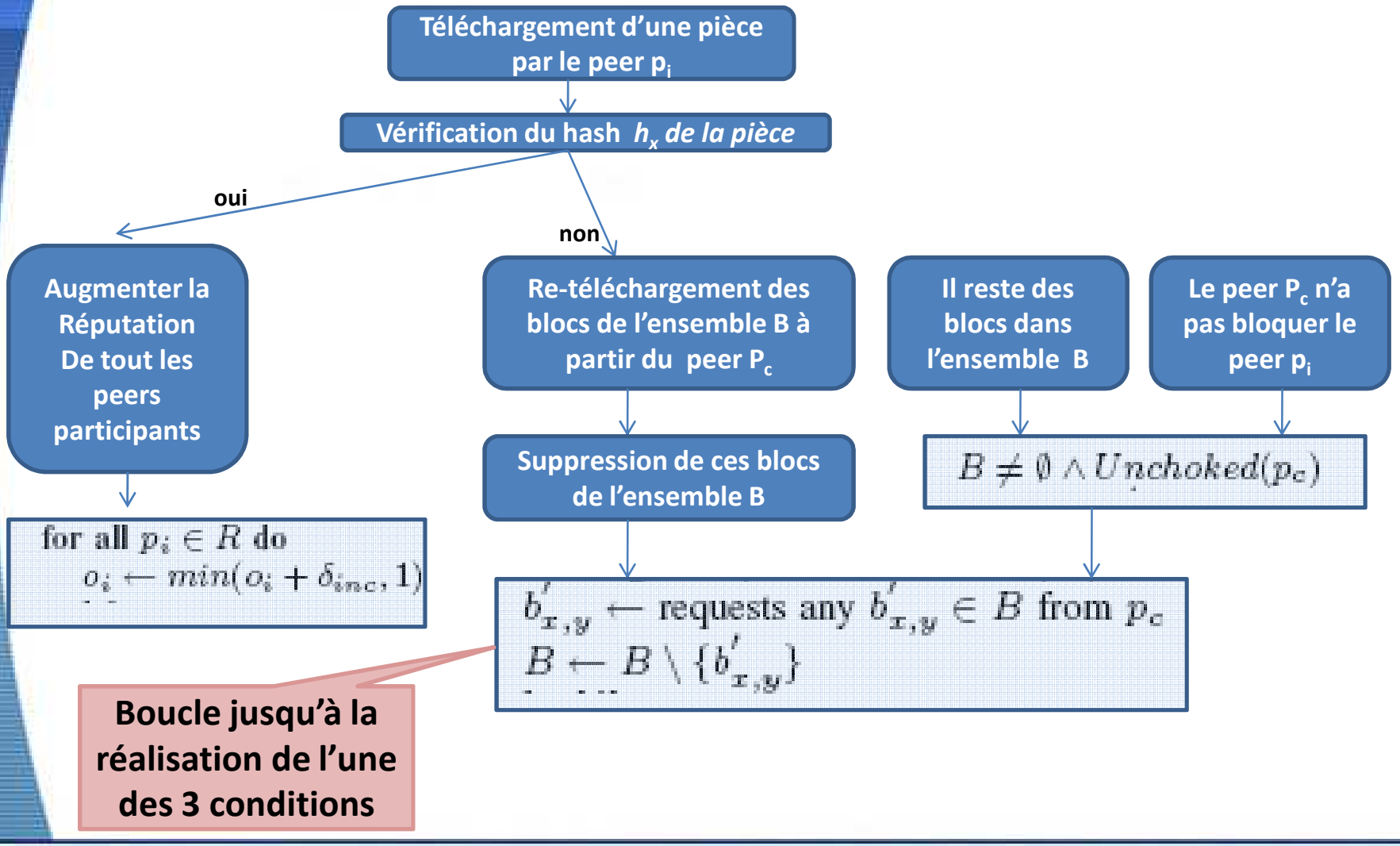
Principe

↳ **- Algorithme basé sur la « Réputation »**

- Réputation (P_i)= 0 ⟹ P_i rentre an quarantaine et n'en sort jamais

- Algorithme est « event-driven »

L'algorithme 2 : Anti Corruption(2)



L'algorithme 2 : Anti Corruption(3)

La nouvelle pièce devient correcte

1

if h'_x then

Augmenter la Réputation du peer p_c

$o_{c_i} \leftarrow min(o_c + \delta_{inc}, 1)$

Détecter et mettre les peers malicieux dans R`, puis diminuer leurs Réputations

$R' \leftarrow \{u_{x,y} | b_{x,y} \neq b'_{x,y}\}$
for all $p_i \in R'$ do
$o_i \leftarrow max(o_i - \delta_{dec}, 0)$

Le peer P_c se déconnecte ou bien bloque le peer P_i

2

if $p_c \notin A \vee \neg Unchoked(p_c)$

Diminuer la Réputation de P_c

$o_c \leftarrow max(o_c - \frac{\delta_{dec}}{2}, 0)$

B = { }
Et la pièce est toujours corrompue

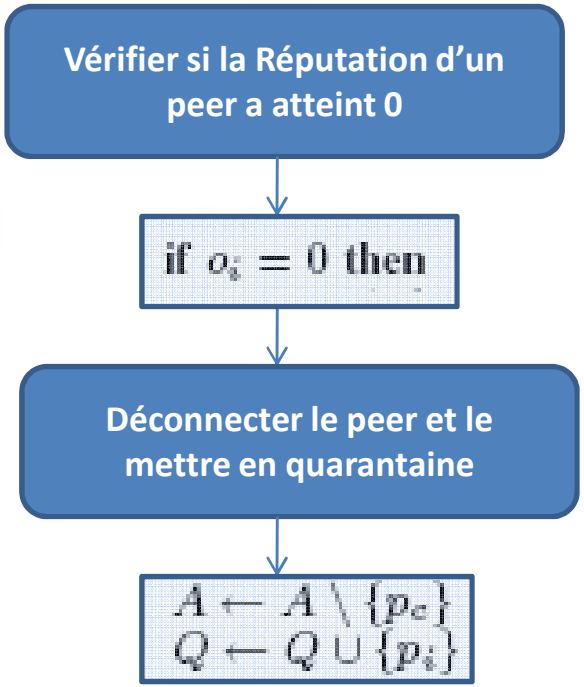
3

P_c est un peer malicieux

Diminuer la Réputation de P_c

$o_c \leftarrow max(o_c - 2 \times \delta_{dec}, 0)$

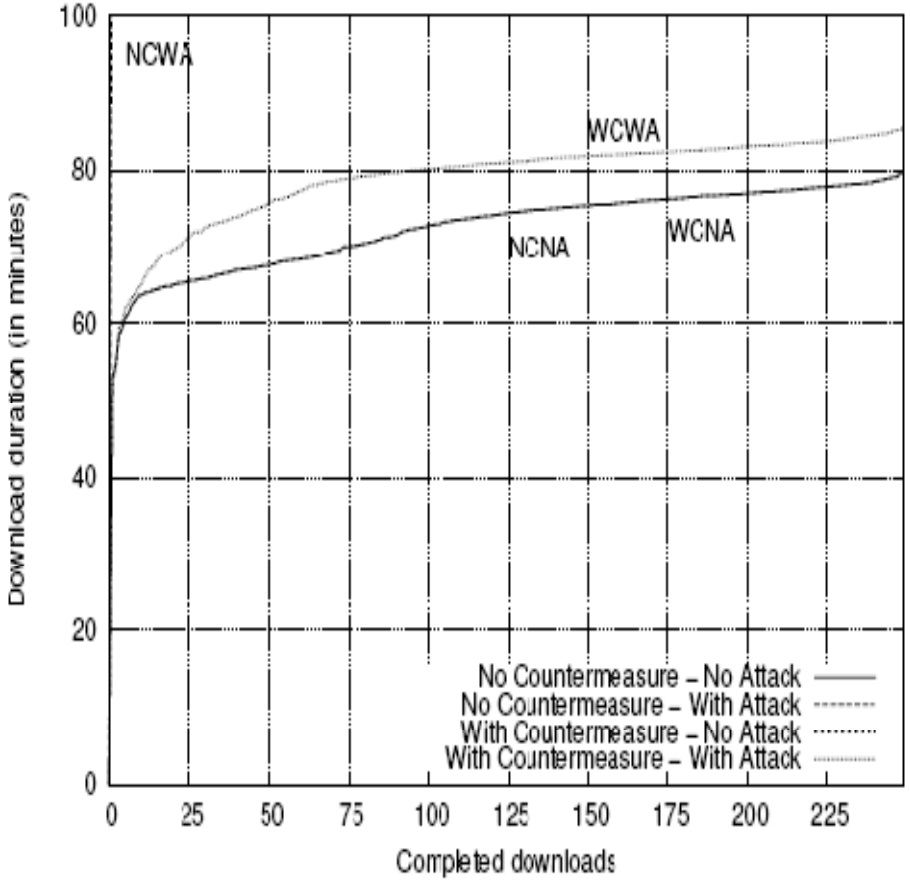
L'algorithme 2 : Anti Corruption(4)



L'algorithme 2 : Anti Corruption (5)

Simulation et évaluation

nombre de peers qui ont complété leurs téléchargements

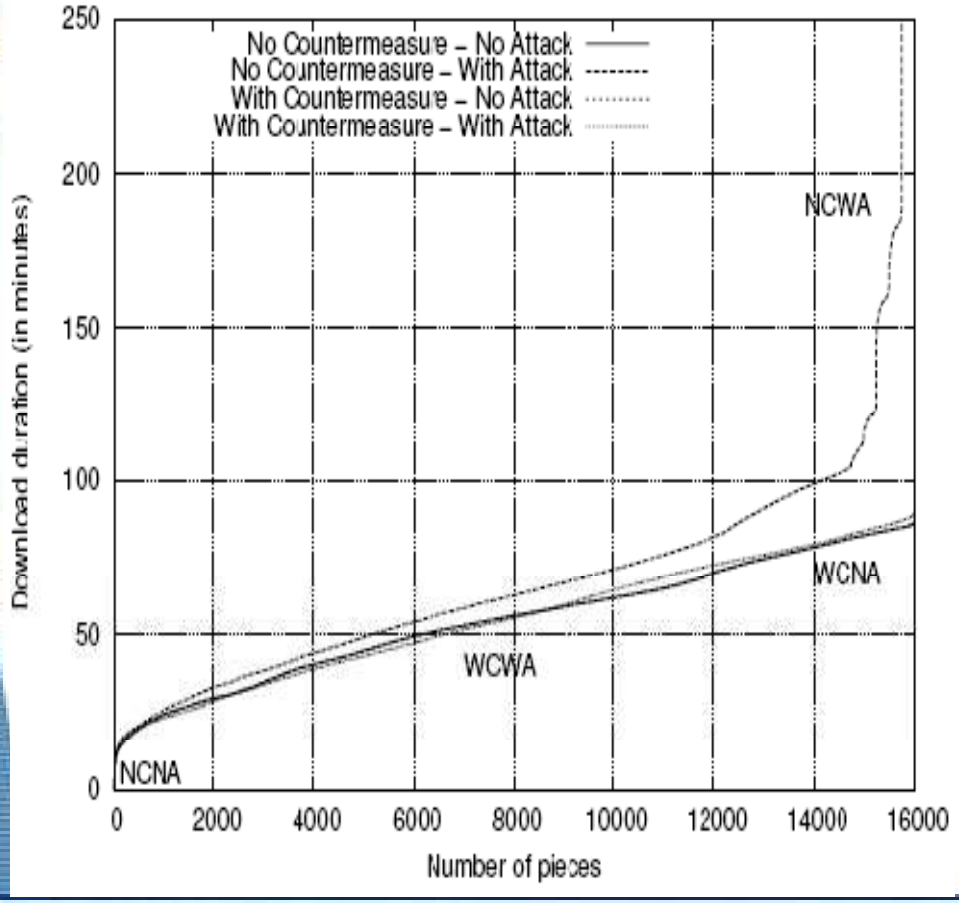


- **Question 1** ⇒ **NCNA & NCWA**
- **Question 2** ⇒ **- NCWA & WCWA**
  - WCWA & NCNA
- **Question 3** ⇒ **NCNA & WCNA**

L'algorithme 2 : Anti Corruption (6)

Simulation et évaluation

nombre de pièces correctement téléchargés



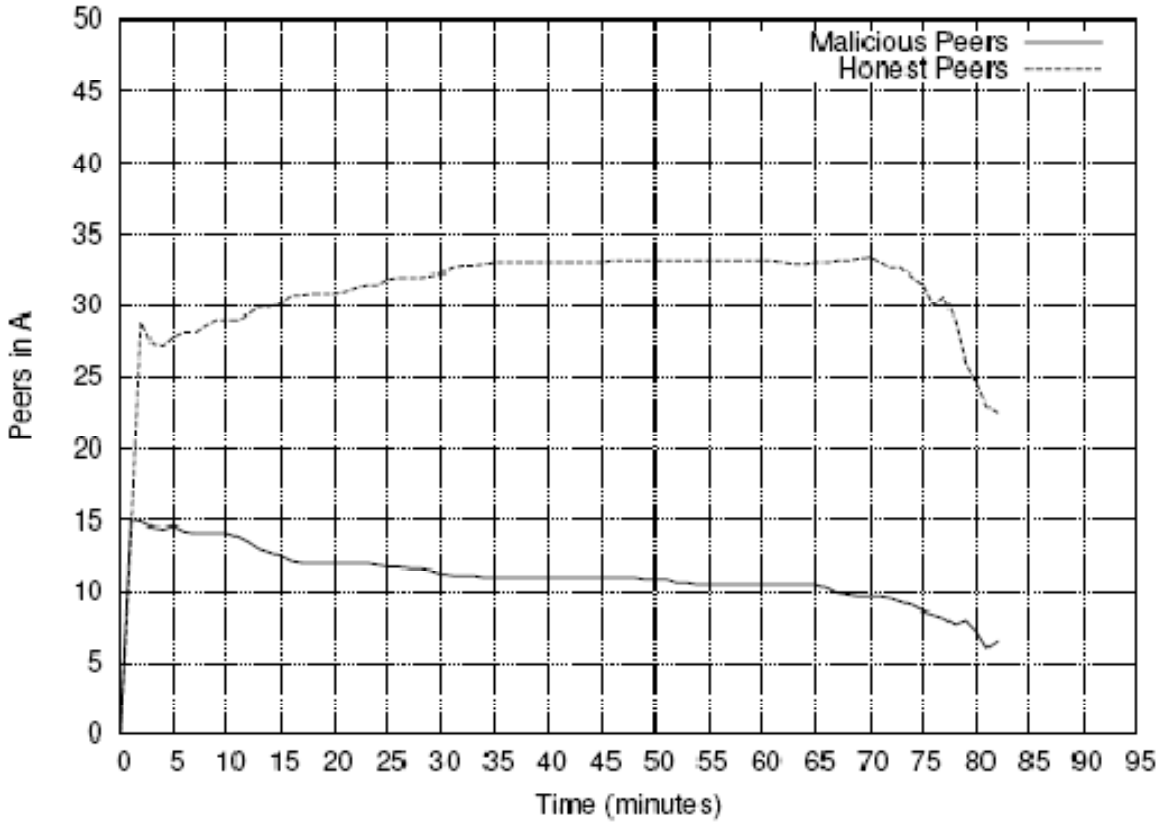
- **Question 1** ⇒ **NCNA & NCWA**
- **Question 2** ⇒ **- NCWA & WCWA**
  -WCWA & NCNA
- **Question 3** ⇒ **NCNA & WCNA**

L'algorithme 2 : Anti Corruption (7)

Simulation et évaluation

Précision de l'algorithme

nombre de peers malicieux et honnêtes



- **Question 4**

Peer malicieux
&
Peer Honnête

L’algorithme 2 : Anti Corruption (8)

Critique

Un peer malicieux peut déjouer le mécanisme de « Réputation »

Conclusion

Merci de votre attention