Stratégie de réplication résistante au churn pour les tables de hachage distribuées en pair-à-pair

Sergey Legtchenko

Université Pierre et Marie Curie
LIP6-INRIA

Stockage de données en réparti

Contexte :

- Stockage à large échelle.
- Réplication de données critiques (tolérance aux pannes).

Problème :

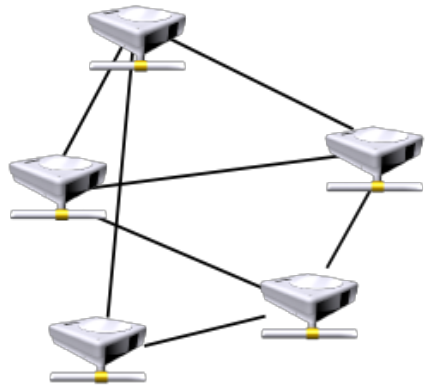
Connexions/deconnexions intempestives (*churn*).

Conséquence :

Perte de données non répliquées.

Objectif :

Améliorer la tolérance au churn.



Plan

1 Stratégies de réplication de données en réparti

2 Présentation de RelaxDHT

3 Évaluation

Principe du stockage de données en pair à pair

On veut stocker une donnée dans un réseau logique (*overlay*) pair à pair.

Contrainte :

La disponibilité des nœuds est incertaine.

Solution :

Répliquer les données.

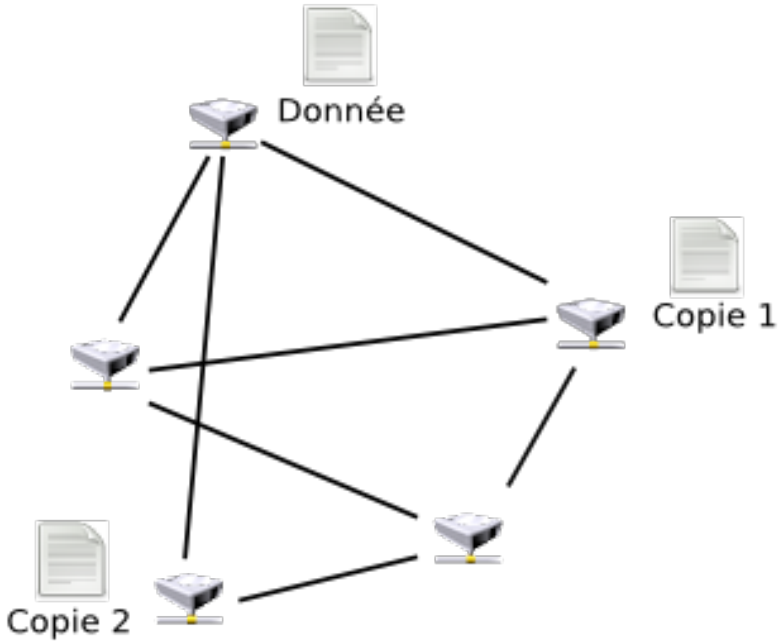


FIG.: Réplication de données sur réseau logique pair à pair (*overlay*)

Principe du stockage de données en pair à pair

On veut stocker une donnée dans un réseau logique (*overlay*) pair à pair.

Contrainte :

La disponibilité des nœuds est incertaine.

Solution :

Répliquer les données.

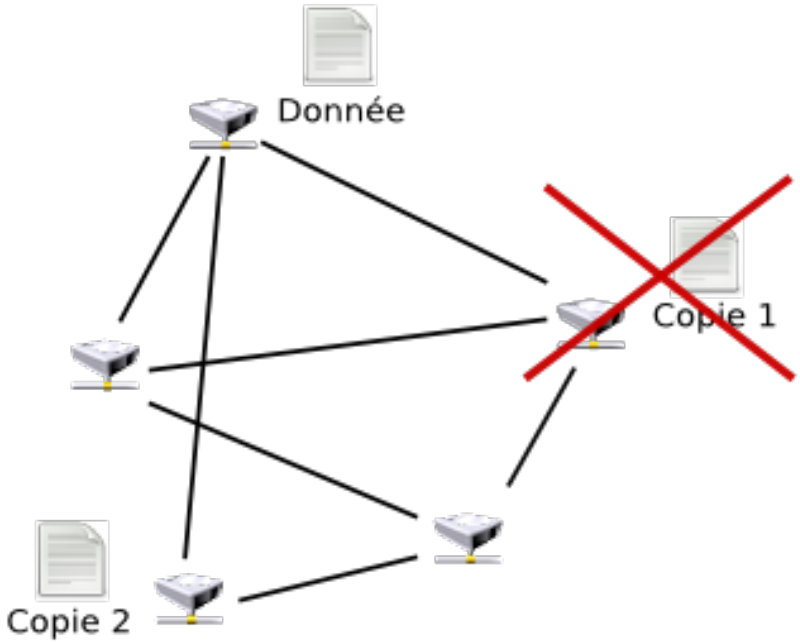


FIG.: Réplication de données sur réseau logique pair à pair (*overlay*)

Principe du stockage de données en pair à pair

On veut stocker une donnée dans un réseau logique (*overlay*) pair à pair.

Contrainte :

La disponibilité des nœuds est incertaine.

Solution :

Répliquer les données.

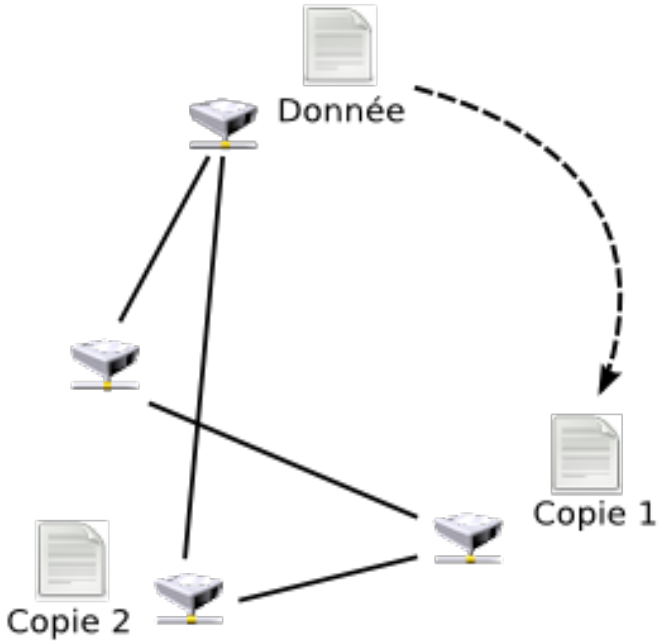


FIG.: Réplication de données sur réseau logique pair à pair (*overlay*)

Une organisation en couches

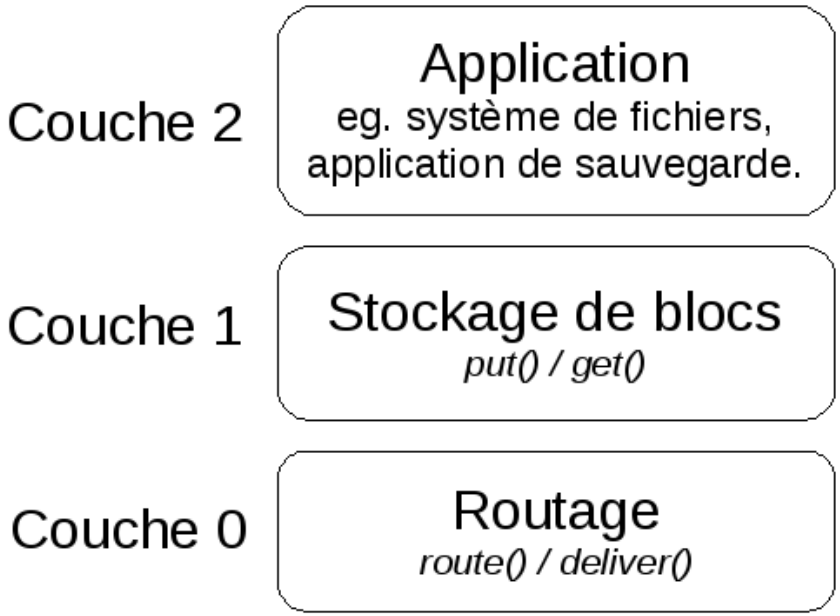


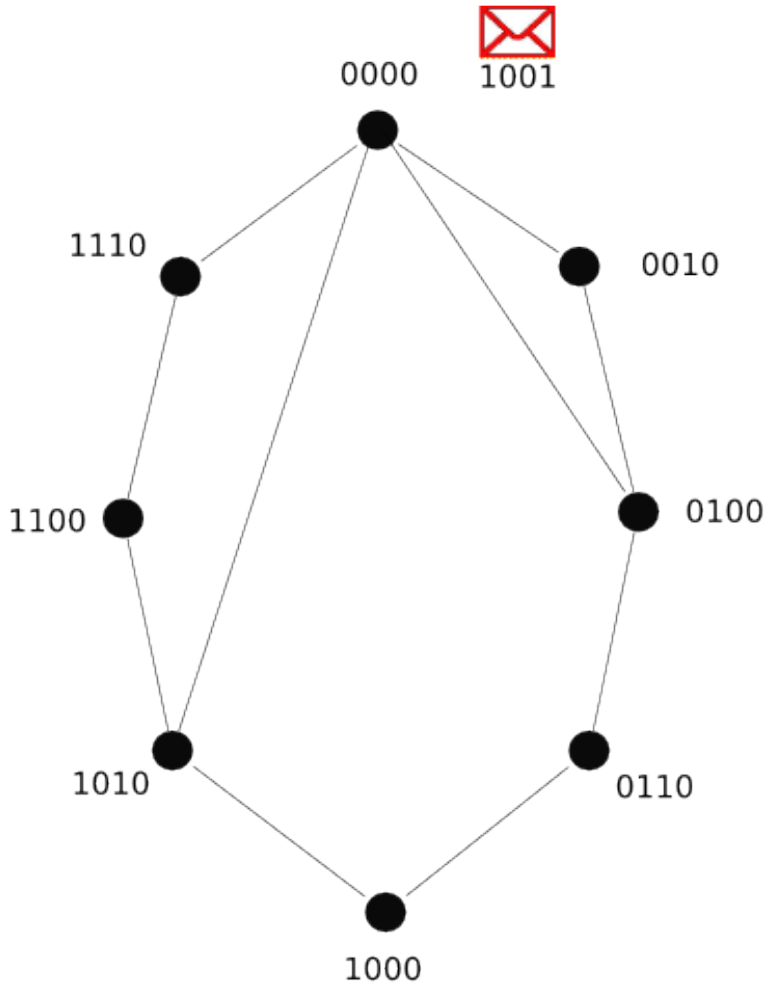
FIG.: *Spécialisation des couches du système de stockage pair à pair. Les algorithmes de réplication se situent dans la couche de stockage.*

Le routage par clés

Couche KBR (*Key-Based Routing*)

Fonctionnement :

- Valeurs identifiant chaque nœud de la topologie.
- Chaque message est routé selon une clé.
- Le message arrive sur le nœud dont la clé est la plus proche de la sienne.

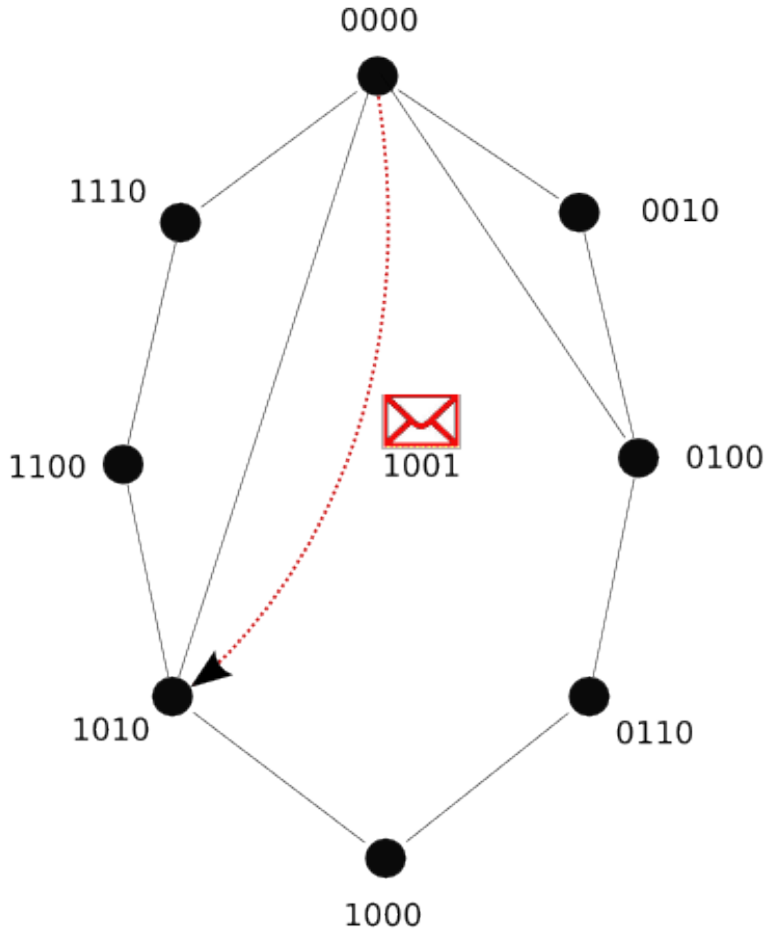


Le routage par clés

Couche KBR (*Key-Based Routing*)

Fonctionnement :

- Valeurs identifiant chaque nœud de la topologie.
- Chaque message est routé selon une clé.
- Le message arrive sur le nœud dont la clé est la plus proche de la sienne.

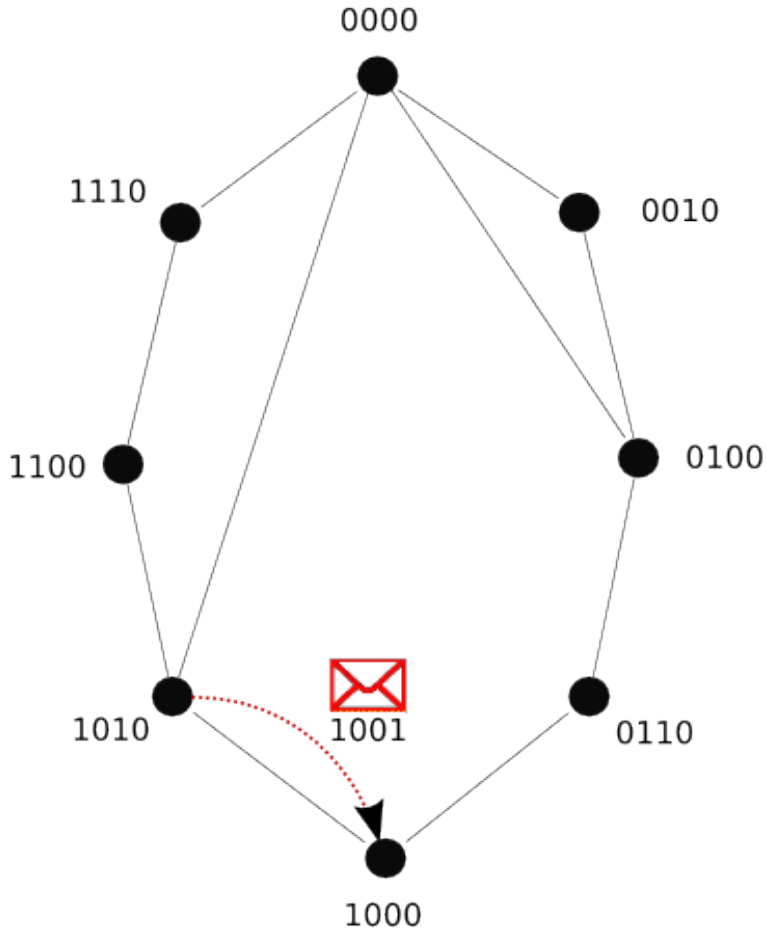


Le routage par clés

Couche KBR (*Key-Based Routing*)

Fonctionnement :

- Valeurs identifiant chaque nœud de la topologie.
- Chaque message est routé selon une clé.
- Le message arrive sur le nœud dont la clé est la plus proche de la sienne.

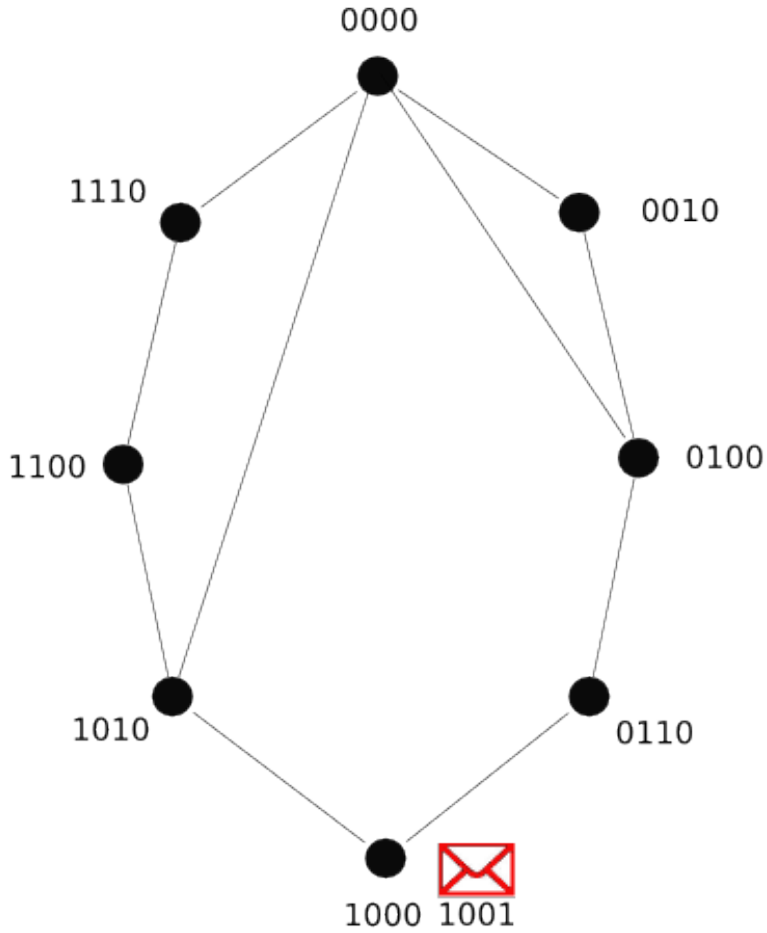


Le routage par clés

Couche KBR (*Key-Based Routing*)

Fonctionnement :

- Valeurs identifiant chaque nœud de la topologie.
- Chaque message est routé selon une clé.
- Le message arrive sur le nœud dont la clé est la plus proche de la sienne.



Stockage décentralisé de données

Couche DHT (*Distributed Hash Table*)

Utilisation de la couche KBR

- Données divisées en blocs.
- Création d'une clé pour chaque bloc via une fonction de hachage.
- Routage des blocs par la couche KBR en fonction de leur clé vers leur nœud *racine*.
- Maintien de k copies de la donnée.

Stratégies de réplication classiques

Structure logique en anneau

Pastry, Chord.

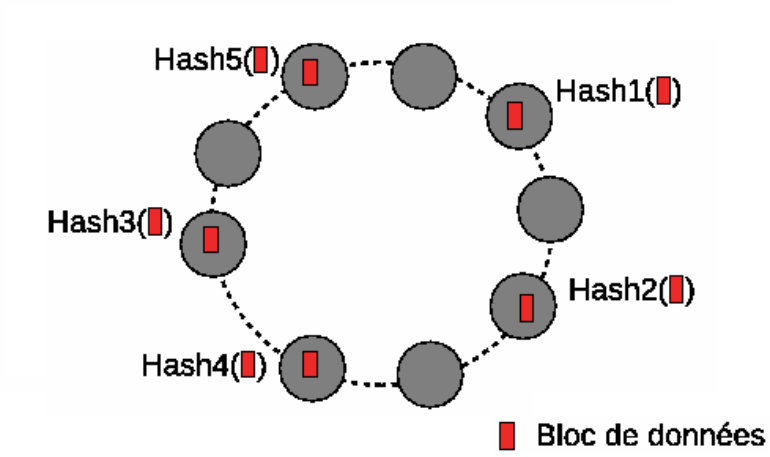


FIG.: *Réplication à base de clés multiples*

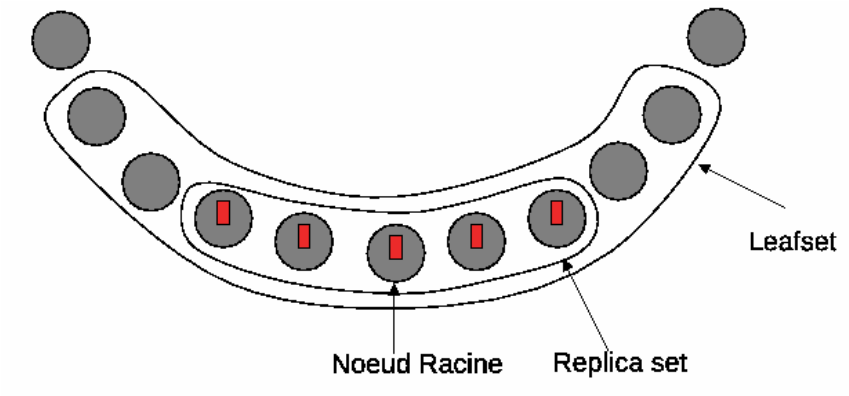
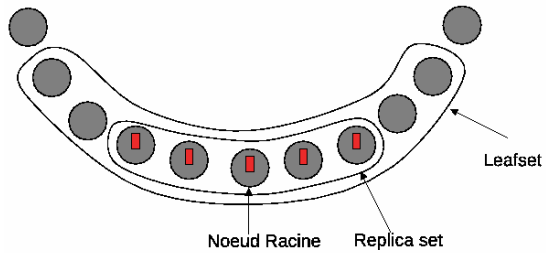


FIG.: *Réplication à base de leafset*

Défauts des stratégies classiques

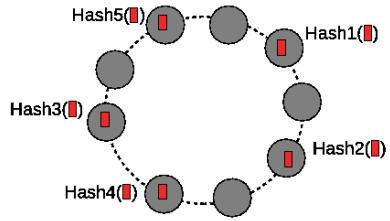
À base de leafset :

- Fortes contraintes de placement des réplicats...
- ...donc gaspillage de bande passante en cas de churn.



À base de clés multiples :

- Coût élevé en nombre de messages si beaucoup de blocs par nœud...
- ...donc difficultés de passage à l'échelle.



Plan

1 Stratégies de réplication de données en réparti

2 **Présentation de RelaxDHT**

3 Évaluation

Relâchement des contraintes de placement

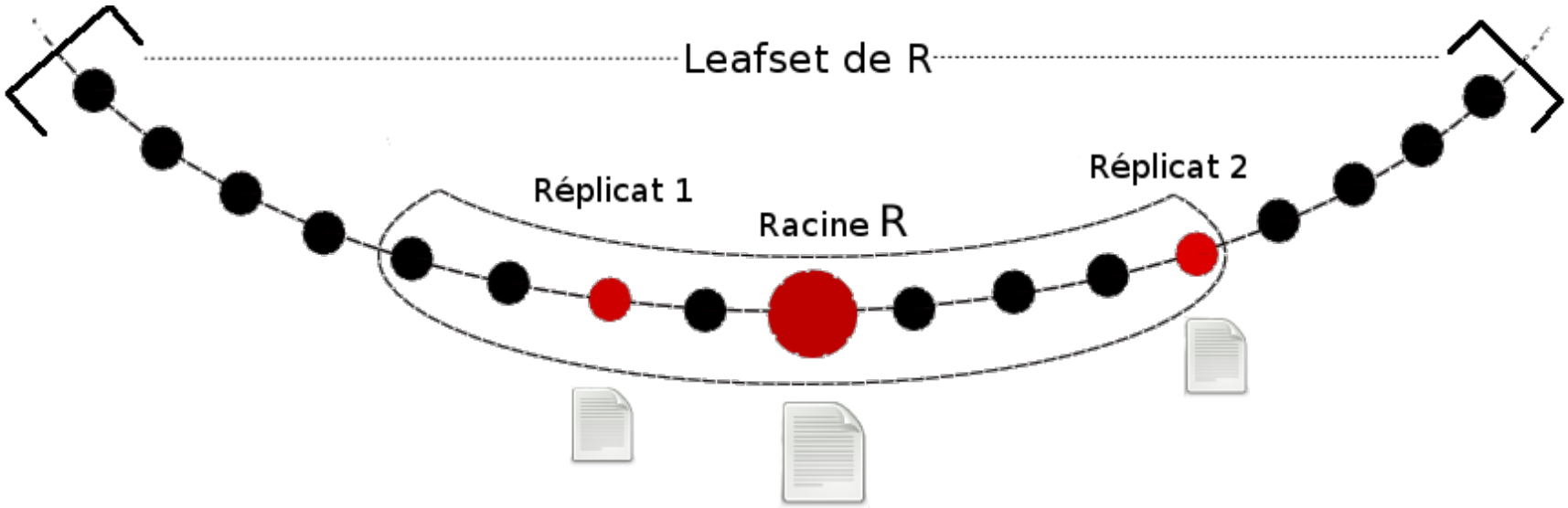


FIG.: *Principe de RelaxDHT : Réplication à base de leafset avec contraintes de placement relâchées.*

Procédure d'insertion de bloc

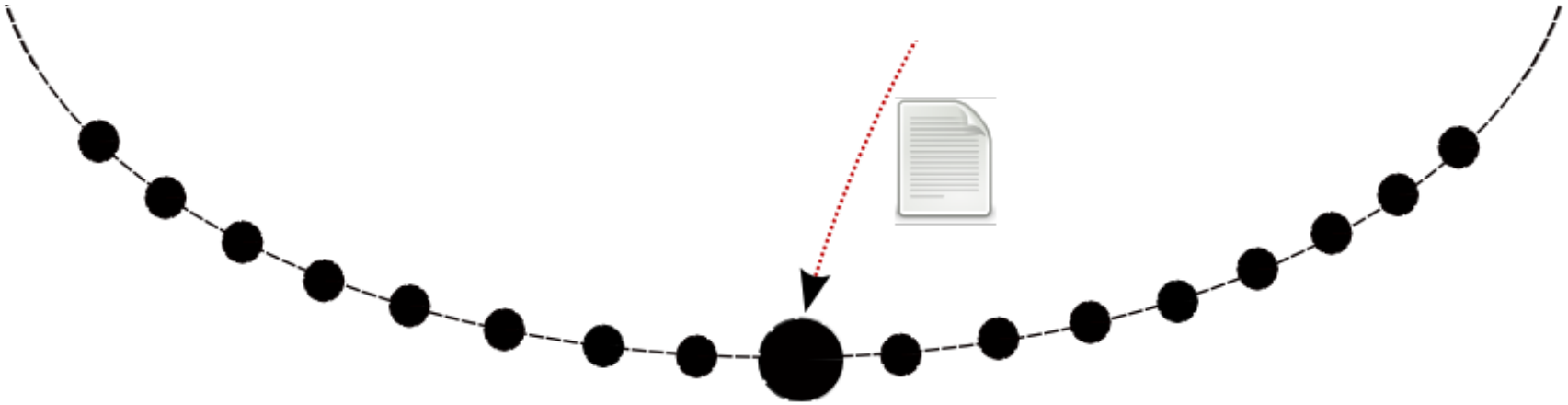


FIG.: *Choix des réplicats par la racine lors de l'insertion d'un nouveau bloc.*

Procédure d'insertion de bloc

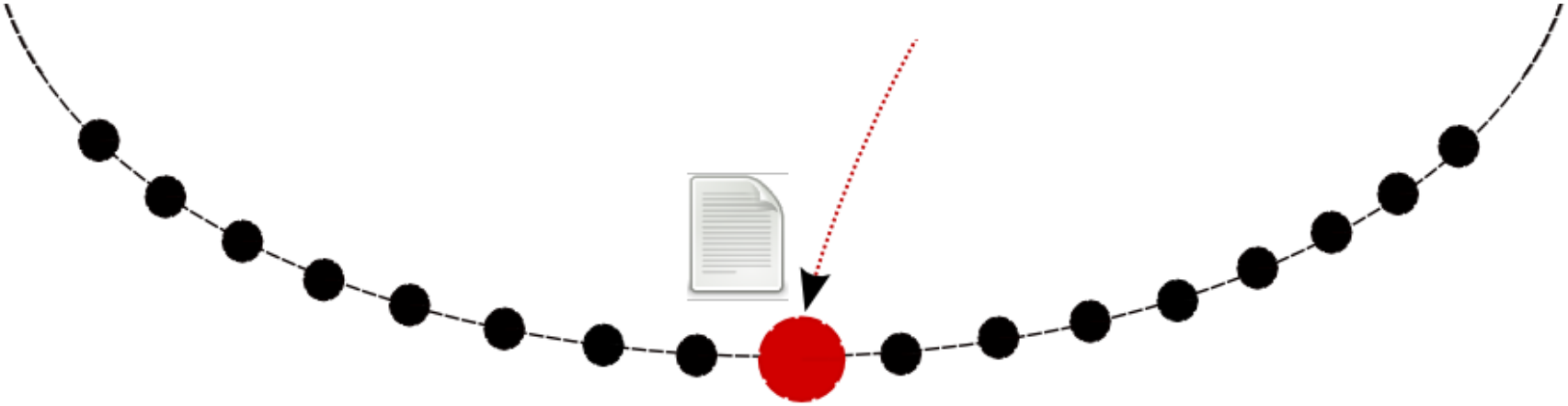


FIG.: *Choix des réplicats par la racine lors de l'insertion d'un nouveau bloc.*

Procédure d'insertion de bloc

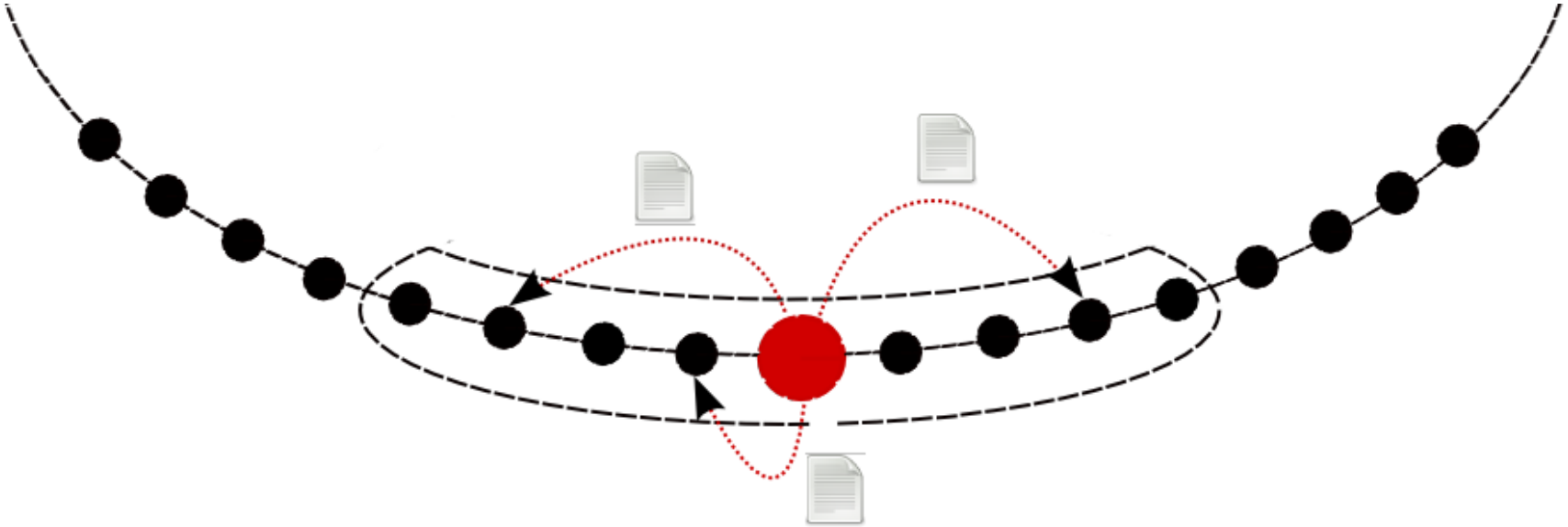


FIG.: *Choix des réplicats par la racine lors de l'insertion d'un nouveau bloc.*

Procédure d'insertion de bloc

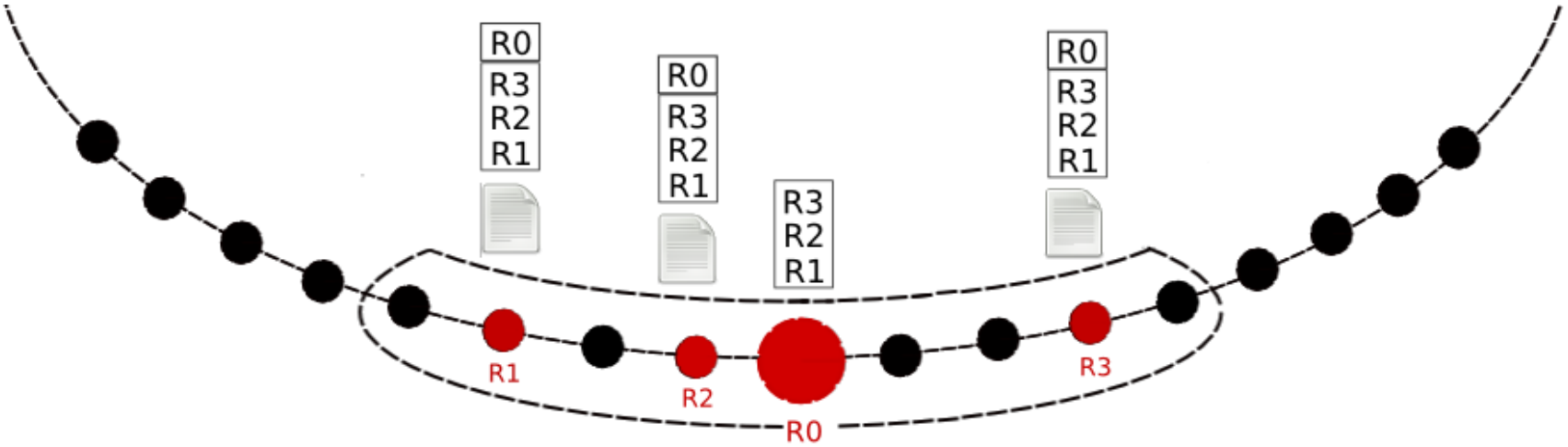


FIG.: *Choix des réplicats par la racine lors de l'insertion d'un nouveau bloc.*

Protocole de maintenance

Maintien des réplicats

- Un réplicat de bloc se voit associer un bail sur le nœud-réplicat.
- À chaque periode de maintenance, la racine envoie un message de renouvellement de bail.

Si le bail expire, le réplicat est supprimé.

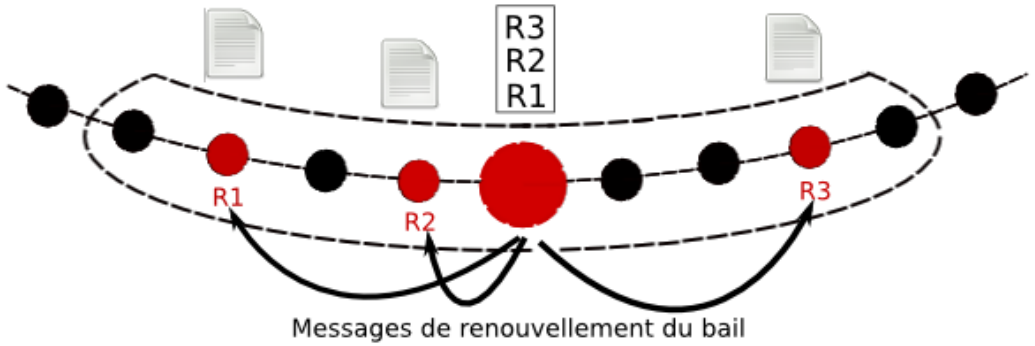


FIG.: *Envoi de messages de maintenance aux réplicats*

En cas de perte ou d'éloignement excessif d'un nœud-réplicat

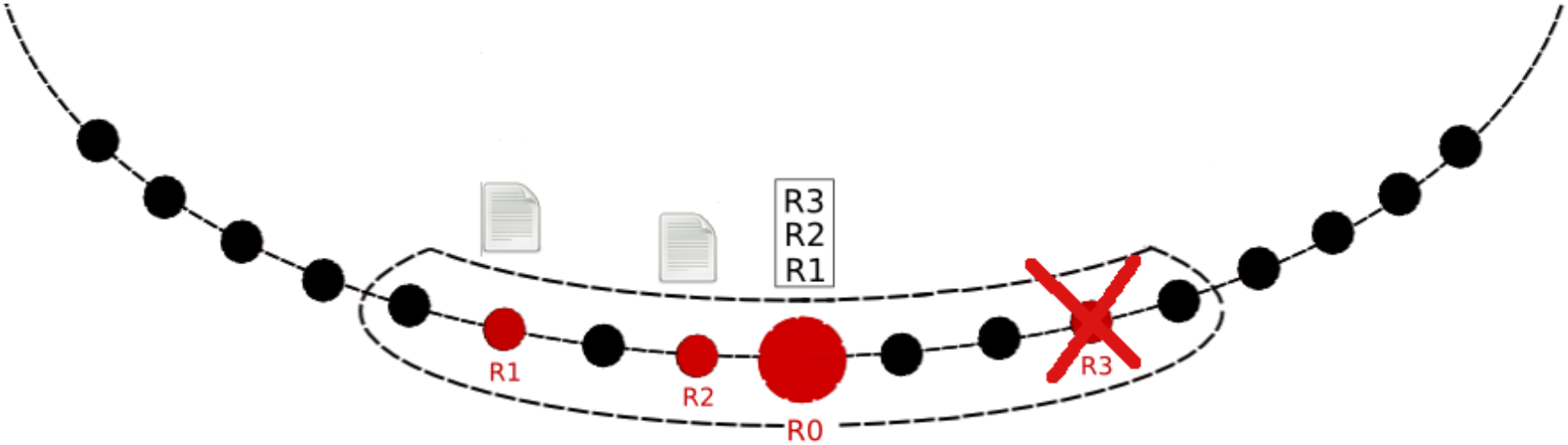


FIG.: *Lors de la maintenance, un nouveau nœud réplicat est choisi par la racine.*

En cas de perte ou d'éloignement excessif d'un nœud-réplicat

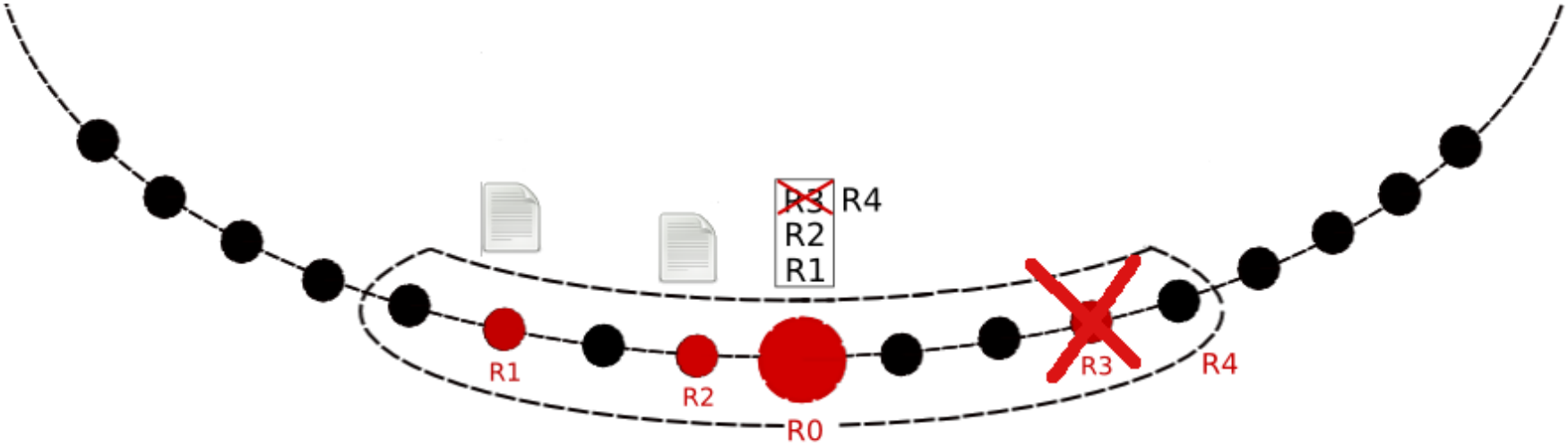


FIG.: *Lors de la maintenance, un nouveau nœud réplicat est choisi par la racine.*

En cas de perte ou d'éloignement excessif d'un nœud-réplicat

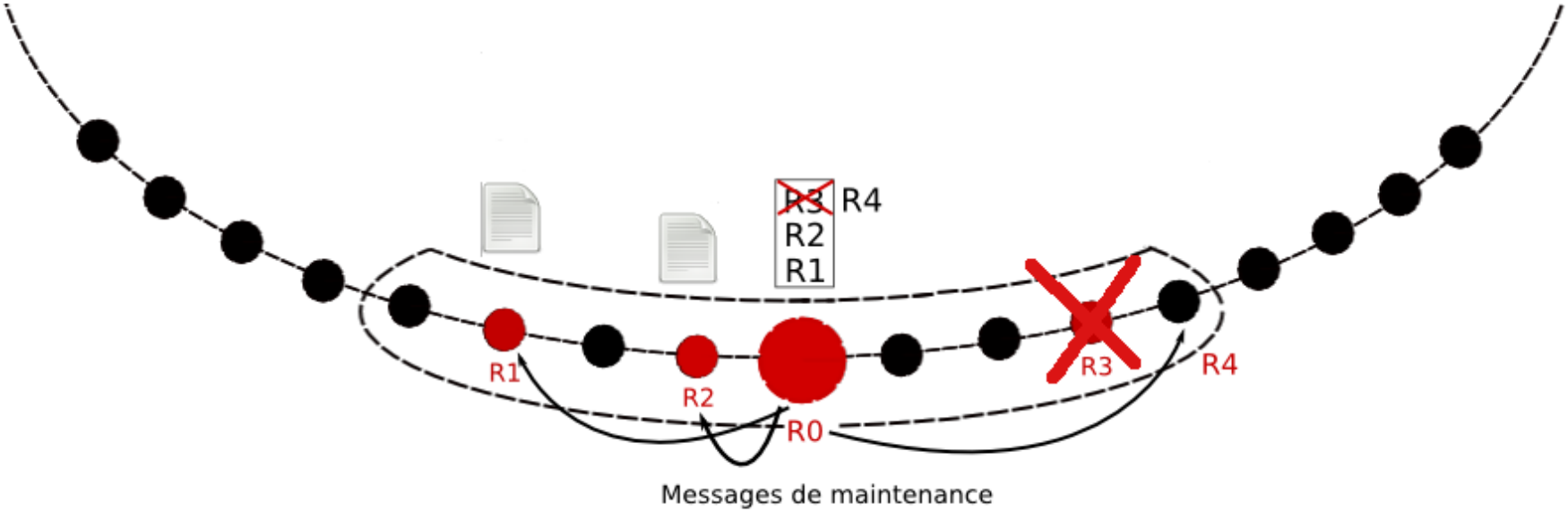


FIG.: *Lors de la maintenance, un nouveau nœud réplicat est choisi par la racine.*

En cas de perte ou d'éloignement excessif d'un nœud-réplicat

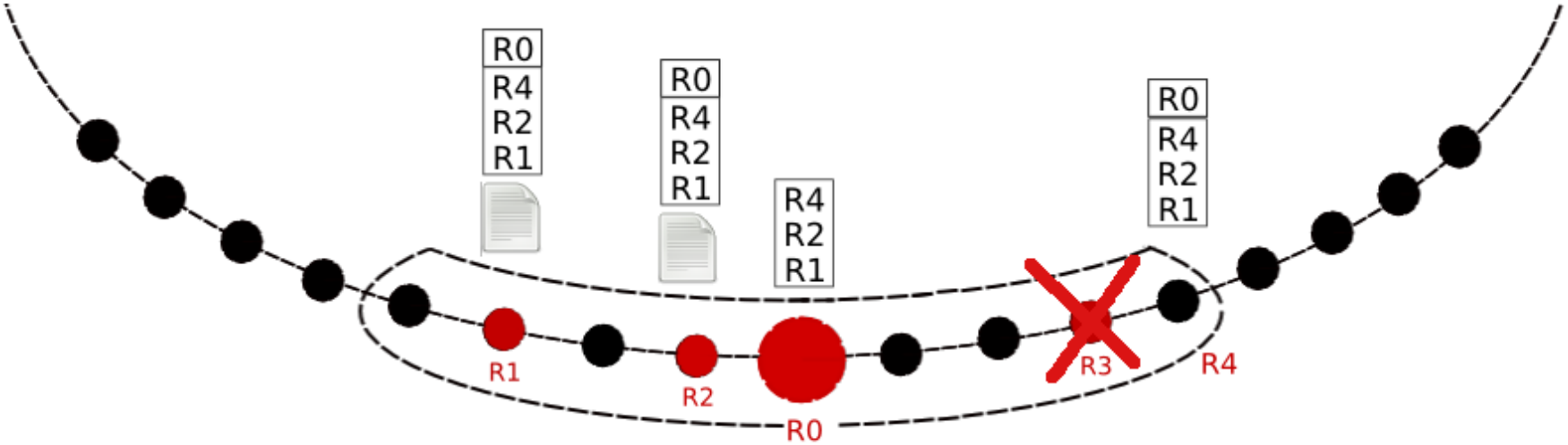


FIG.: *Lors de la maintenance, un nouveau nœud réplicat est choisi par la racine.*

En cas de perte ou d'éloignement excessif d'un nœud-réplicat

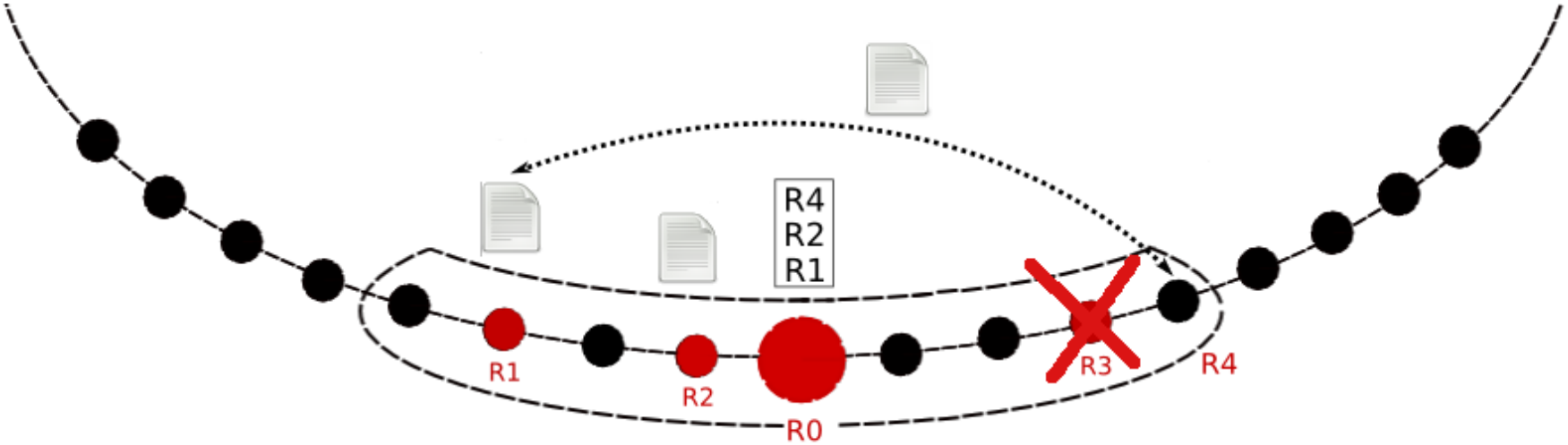


FIG.: *Lors de la maintenance, un nouveau nœud réplicat est choisi par la racine.*

En cas de perte ou d'éloignement excessif d'un nœud-réplicat

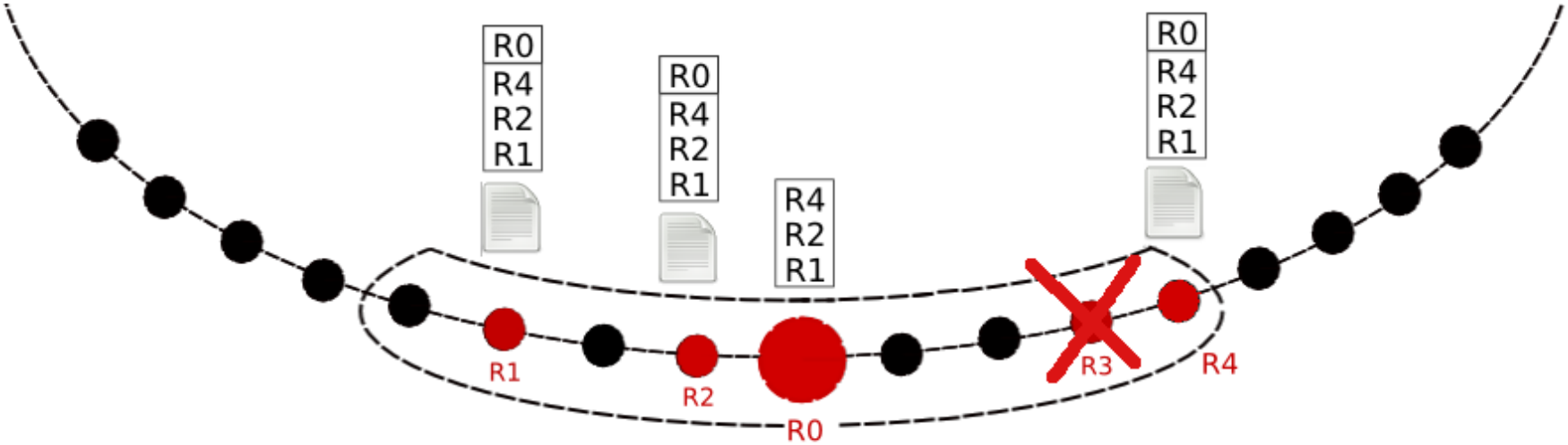


FIG.: *Lors de la maintenance, un nouveau nœud réplicat est choisi par la racine.*

En cas de perte de la racine

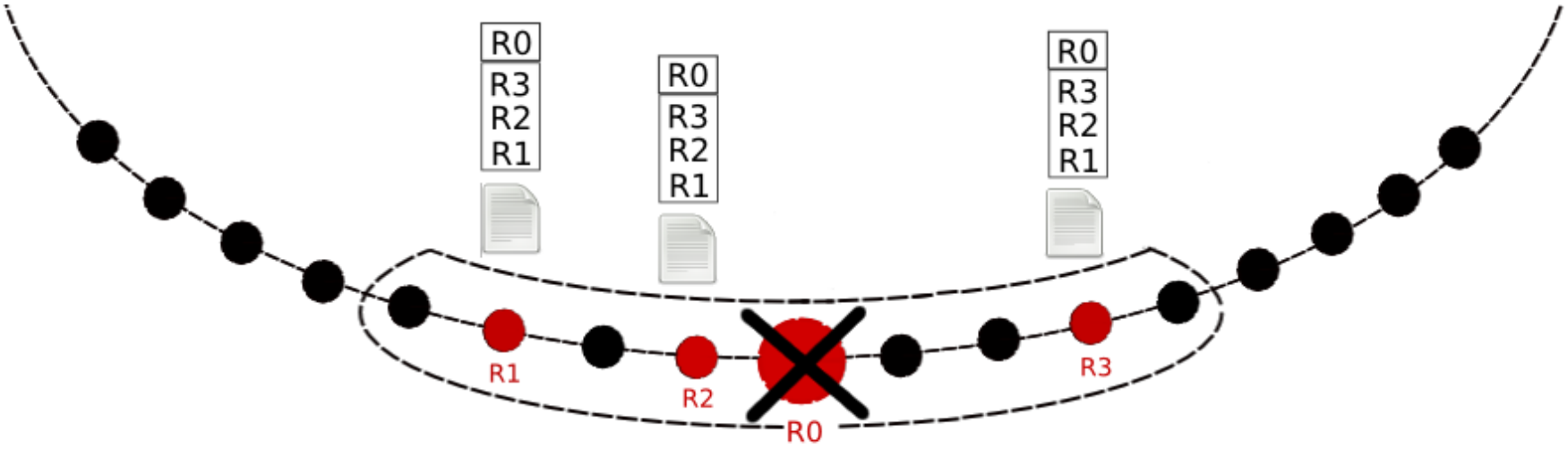


FIG.: *Lors de la maintenance, une nouvelle racine est désignée par un ou plusieurs nœud réplicats.*

En cas de perte de la racine

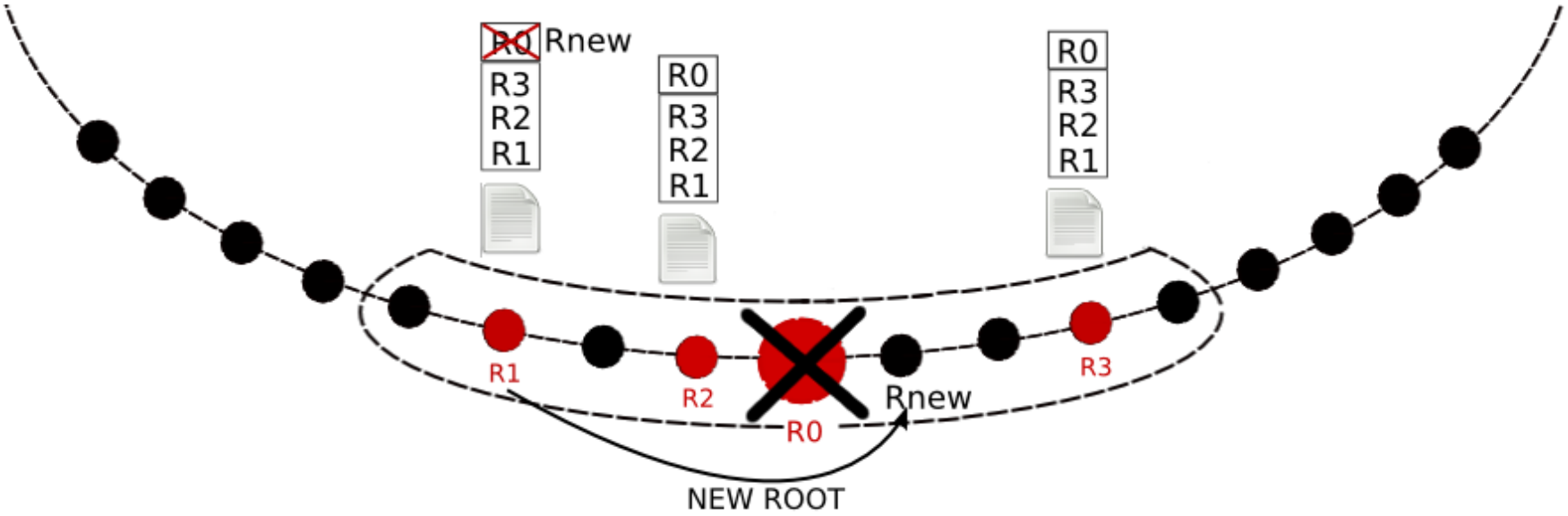


FIG.: *Lors de la maintenance, une nouvelle racine est désignée par un ou plusieurs nœud réplicats.*

En cas de perte de la racine

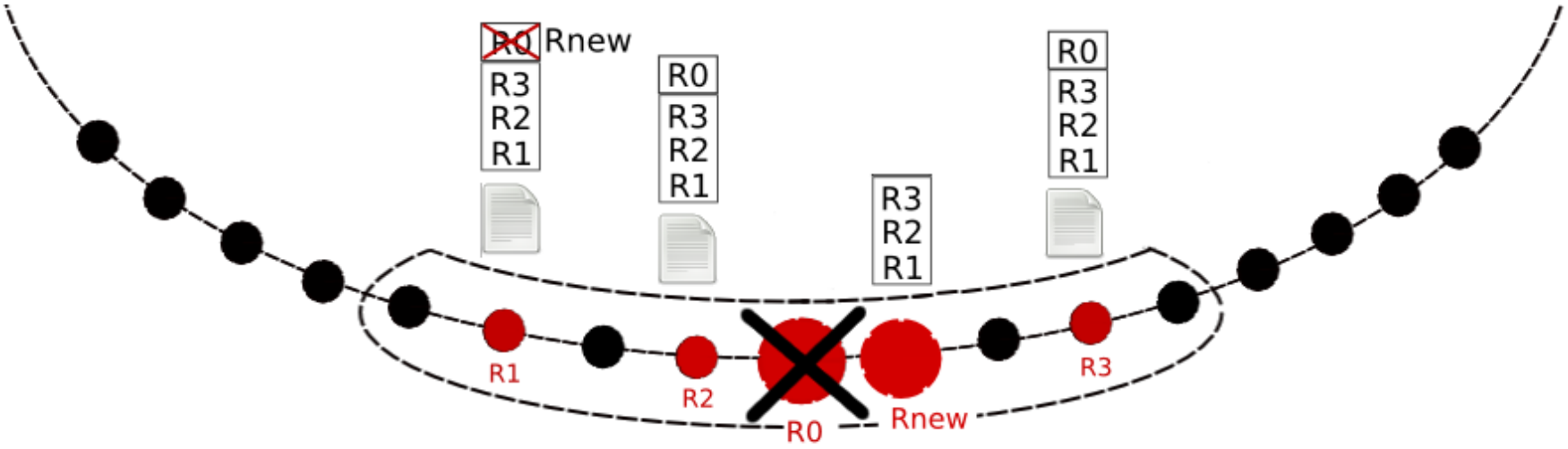


FIG.: *Lors de la maintenance, une nouvelle racine est désignée par un ou plusieurs nœud réplicats.*

En cas de perte de la racine

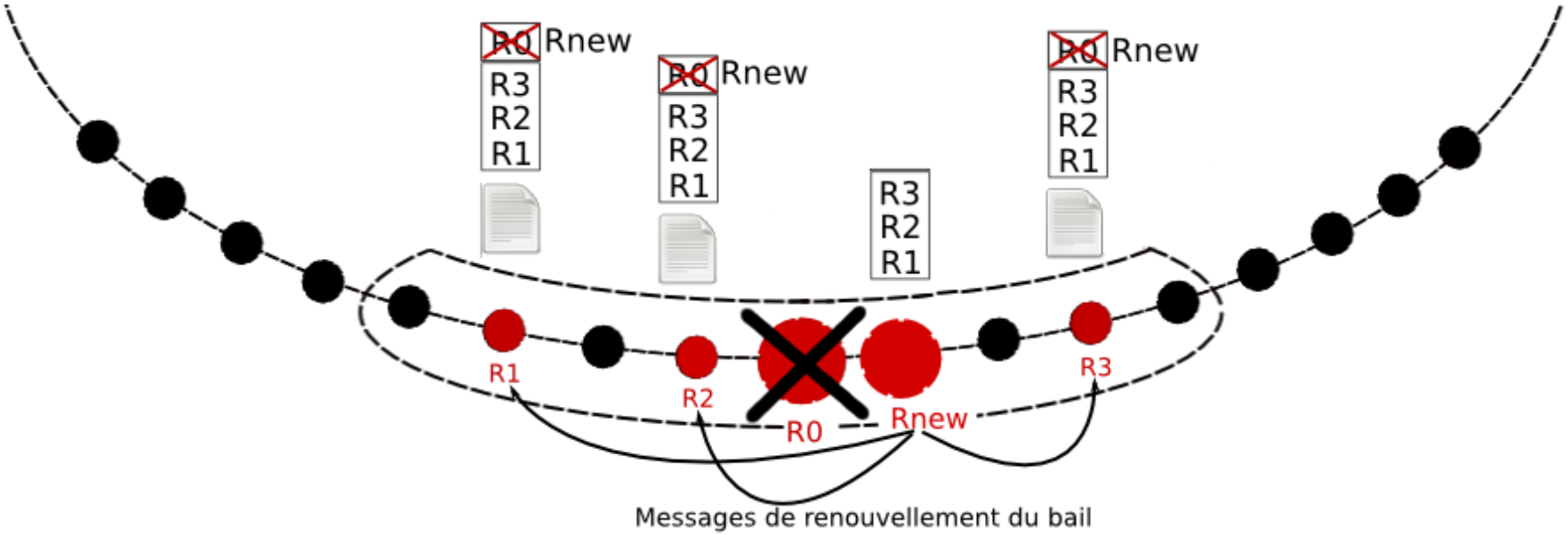


FIG.: *Lors de la maintenance, une nouvelle racine est désignée par un ou plusieurs nœud réplicats.*

Analyse de RelaxDHT

Avantages :

- **Tolérance aux pannes accrue :**
  - Diminution du nombre de blocs à transferer en cas de churn grâce au relâchement des contraintes de placement.
  - Diversification des sources pour le téléchargement d'un bloc lors de la maintenance.
- **Meilleure scalabilité :**
  - Le nombre de messages est constant (dépend de la taille du leafset).
  - Possibilité de bien compresser les messages.

Limitations :

En cas de perte de la racine d'un bloc, augmentation *temporaire* du routage d'une requête lookup sur ce bloc.

Plan

1 Stratégies de réplication de données en réparti

2 Présentation de RelaxDHT

3 Évaluation

Conditions d'évaluation

Implémentation de RelaxDHT dans le simulateur Peersim.

- Couche KBR : Pastry
- Couche DHT de référence : PAST

Paramètres :

- 100 nœuds.
- 10000 blocs de 10 Mo répliqués 3 fois.
- Modèle de churn uniforme.
- Taille du leafset : 24
- 1 Mbit/s en flux montant et 10 Mbit/s en flux descendant.

Quelques essais de simulation à plus grande échelle (1000 nœuds et 100 000 blocs) donnent des résultats comparables.

Scénarios de test

Évaluation selon deux scénarios différents :

1. **Période de churn d'une heure :**

   Une heure (temps simulateur) de perturbations intenses. La période sans churn qui suit permet de mesurer les caractéristiques de convergence du système vers son état normal.

2. **Churn continu :**

   Permet d'évaluer le comportement du système soumis en continu aux connexions/déconnexions de nœuds.

Métriques employées

Critères d'évaluation importants :

- Nombre de blocs perdus (ie. tous réplicats perdus).
- Nombre de blocs échangés.
- Temps de stabilisation post-churn (*Scénario 1*).

Évaluation du premier scénario (1)

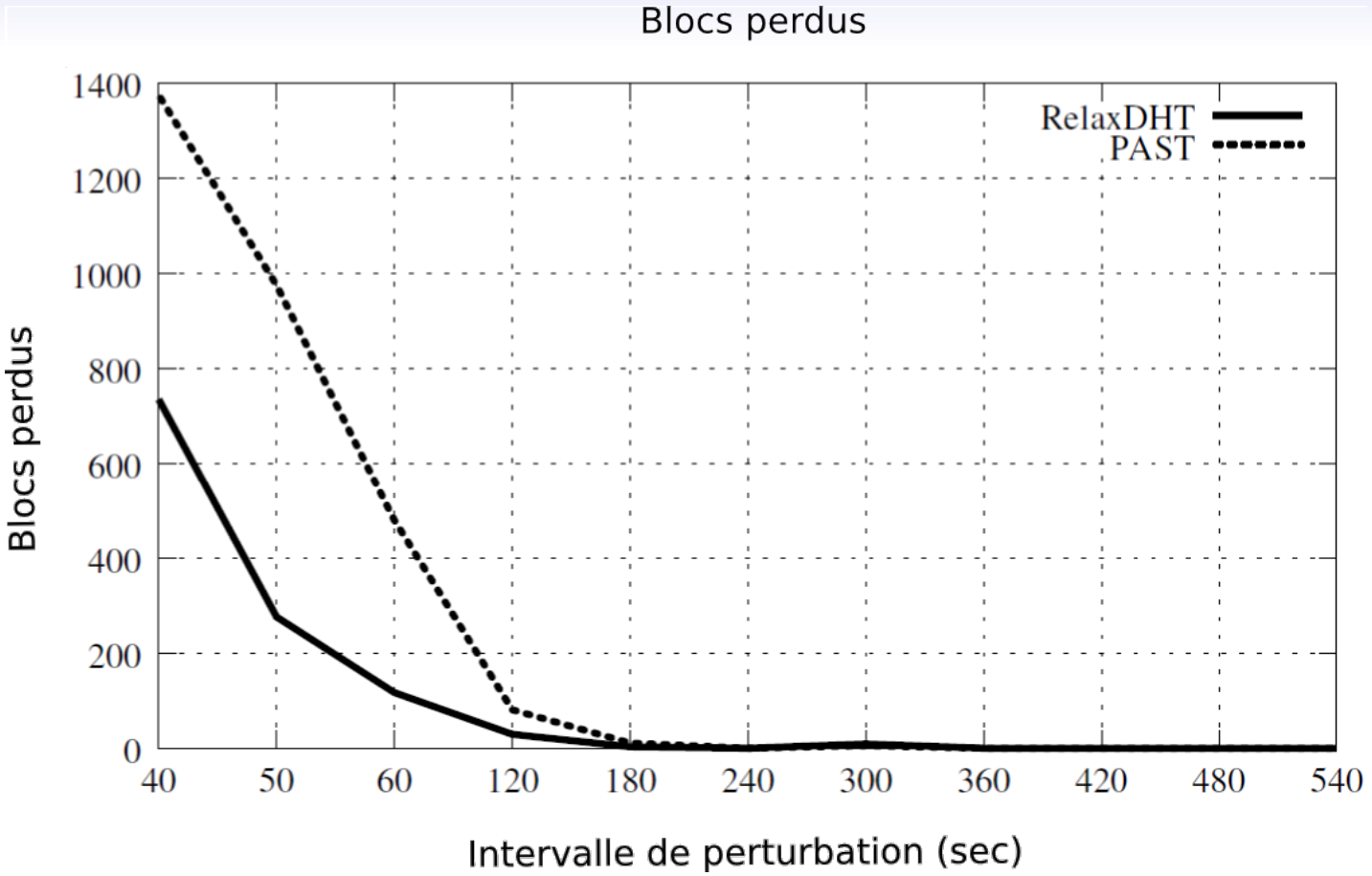
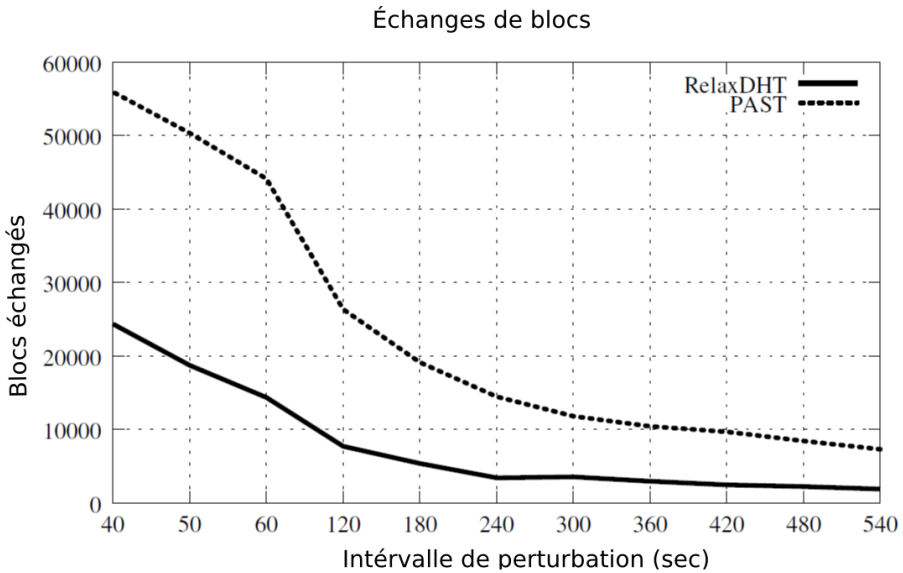


FIG.: *Nombre de blocs perdus (les 3 réplicats perdus) après la période de perturbation.*

Évaluation du premier scénario (2)



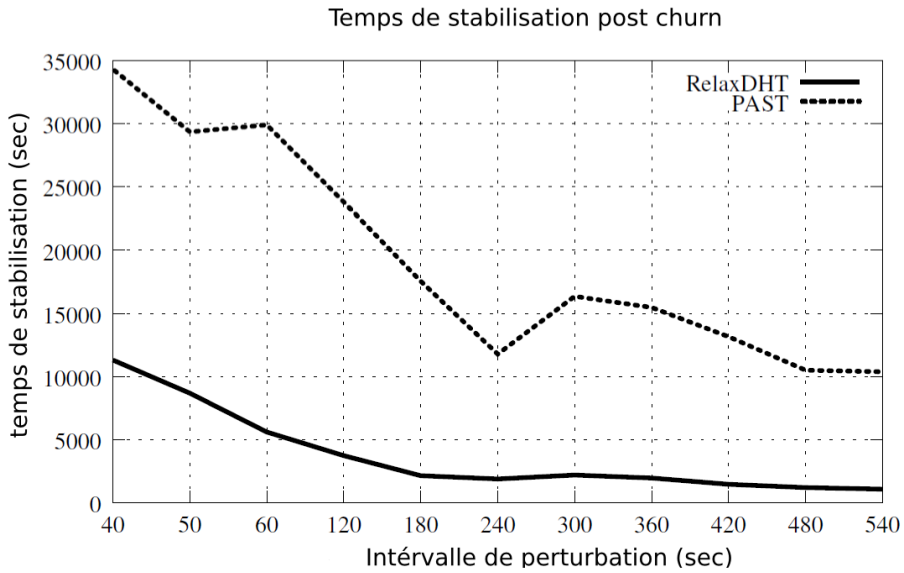
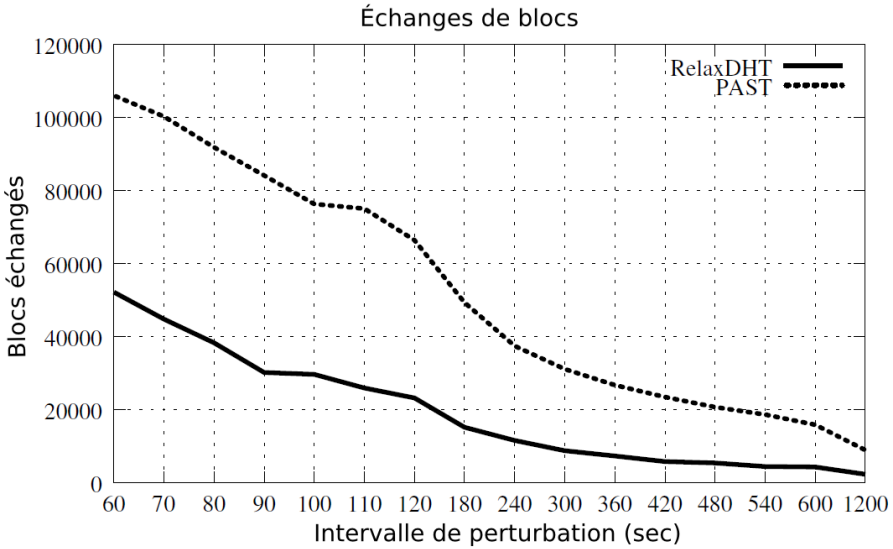


FIG.: *Caractéristiques de la convergence du système à un état stable (ie. tous les blocs restants sont à nouveau répliqués 3 fois) après la période de perturbation :* **quantité de blocs** *échangés et* **temps de convergence**.

Évaluation du deuxième scénario



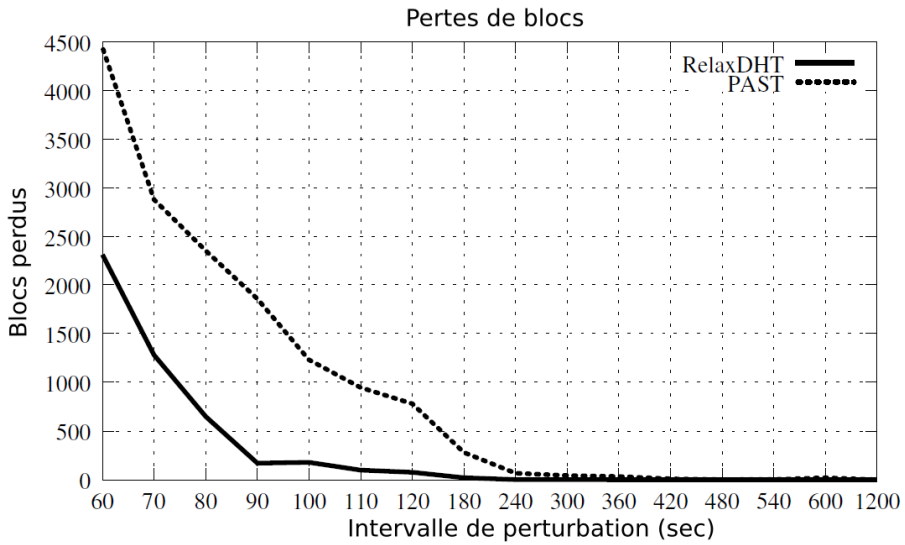


FIG.: *Propriétés du système soumis à un churn continu :* **nombre total de blocs perdus** *et* **nombre total de blocs échangés**.

Conclusion

Gains :

- **Temps de convergence plus court :** Meilleure répartition des sources lors de la récuperation de blocs.
- **Charge réseau plus faible :** moins de transferts de données liés au churn.
- **Tolérance au churn accrue** (50% de gains en moyenne).

Travaux en cours :

- Expérimentations plus poussées : à plus large échelle, avec un modèle de churn plus évolué.
- Quantification de l'impact sur les requêtes de lookup.
- Stratégies hybrides de réplication.

Merci pour votre attention !

1. Stratégies de réplication de données en réparti
2. Présentation de RelaxDHT
3. Évaluation

Questions.