Estimation dynamique du nombre de processus vivants dans un système asynchrone soumis aux crashs

A. Mostefaoui, M. Raynal and G. Trédan

1er février 2008

IRISA aSAp

Objectifs de cet exposé

- Proposer un nouveau modèle de défaillances
- Présenter un protocole utilisant ce modèle pour prédire le nombre de processus présents

Un système asynchrone

- Un ensemble Π de n processus : $\{p_1, \ldots, p_n\}$
- Connectés par un réseau fiable asynchrone
- Quelques processus peuvent crasher: ils deviennent muets *(failstop)*.

De lourdes conséquences

- Crashs + Asynchronie $\Rightarrow$ Dilemme :
- p_i veut communiquer avec p_j, il a deux possibilités :
- Soit p_i attend $\Rightarrow$ Bloquage
- Soit p_i n'attend pas $\Rightarrow$ Perte d'information

IRISA aSAp

Un système asynchrone

- Un ensemble Π de n processus : $\{p_1, \ldots, p_n\}$
- Connectés par un réseau fiable asynchrone
- Quelques processus peuvent crasher: ils deviennent muets *(failstop)*.

De lourdes conséquences

- Crashs + Asynchronie $\Rightarrow$ Dilemme :
- p_i veut communiquer avec p_j, il a deux possibilités :
- Soit p_i attend $\Rightarrow$ Bloquage
- Soit p_i n'attend pas $\Rightarrow$ Perte d'information

besoin de plus d'hypothèses

IRISA aSAp

Ancien Modèle de défaillances

Dans l'ancien modèle, un paramètre : t

- Borne haute du nombre de processus pouvant crasher
- Pour toute exécution !
- $\Rightarrow$ qualité de réponse constante

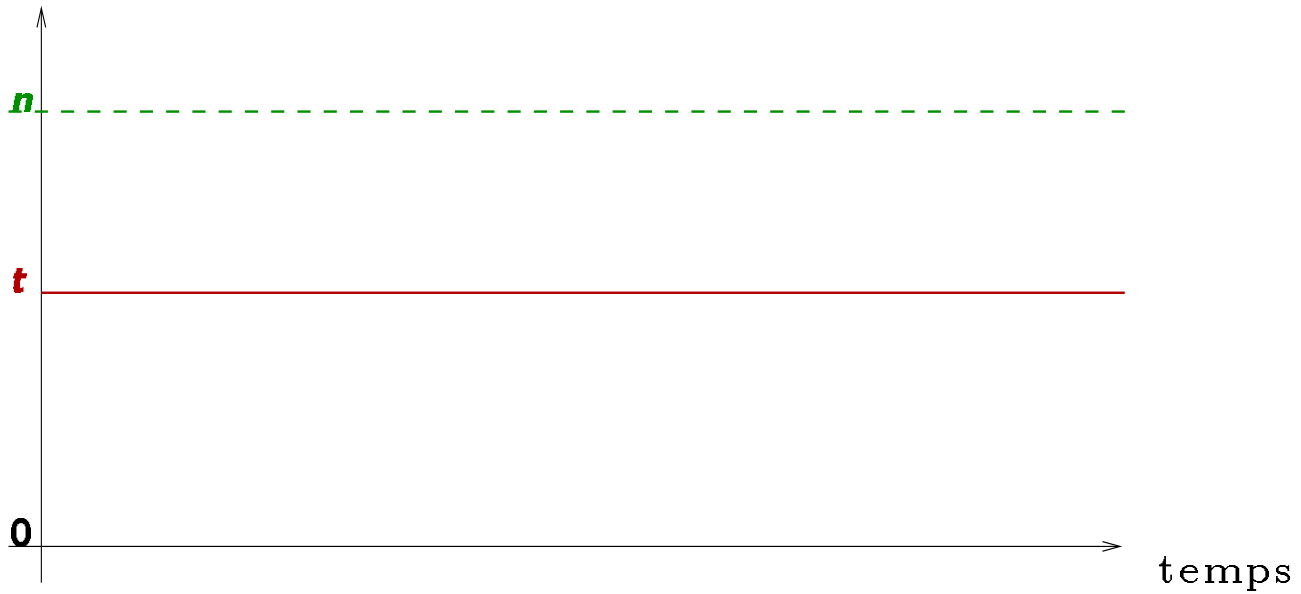


Fig.: modèle classique

Ancien Modèle de défaillances

Dans l'ancien modèle, un paramètre : t

- Borne haute du nombre de processus pouvant crasher
- Pour toute exécution !
- $\Rightarrow$ qualité de réponse constante

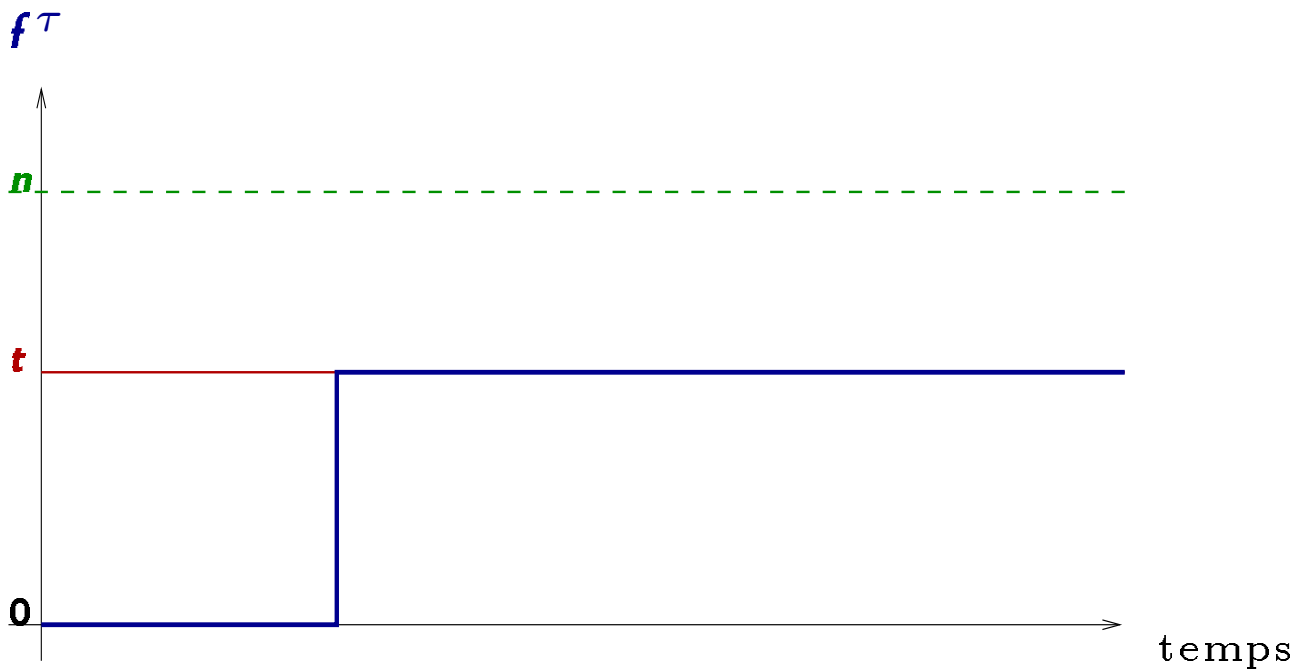


Fig.: modèle classique

Ancien Modèle de défaillances

Dans l'ancien modèle, un paramètre : t

- Borne haute du nombre de processus pouvant crasher
- Pour toute exécution !
- $\Rightarrow$ qualité de réponse constante

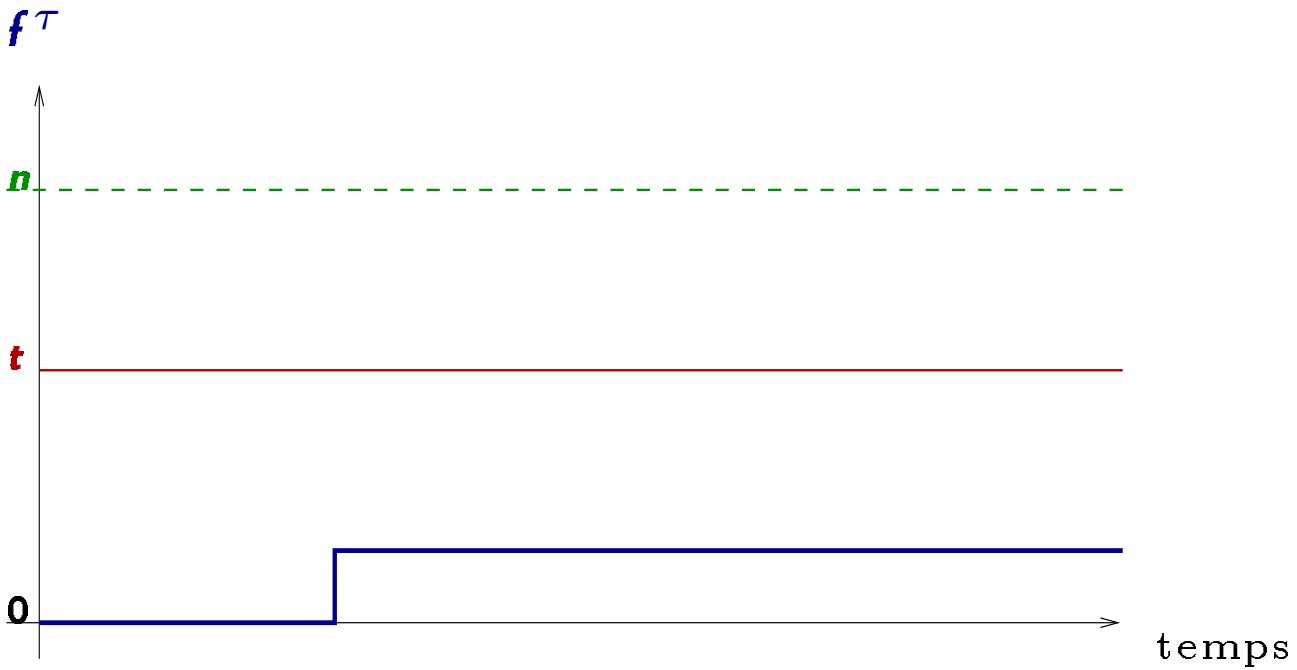


Fig.: modèle classique

Ancien Modèle de défaillances

Dans l'ancien modèle, un paramètre : t

- Borne haute du nombre de processus pouvant crasher
- Pour toute exécution !
- $\Rightarrow$ qualité de réponse constante

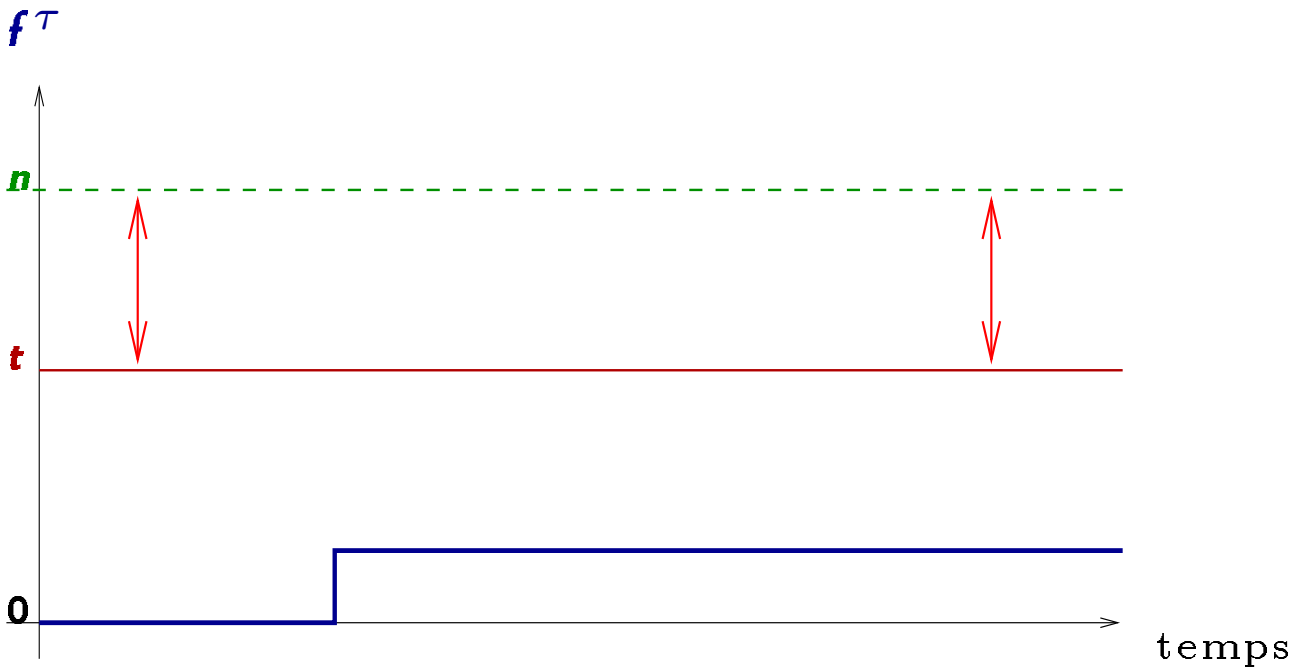


Fig.: modèle classique

Nouveau modèle de défaillances

On remplace t par une fonction $\alpha()$ qui ...

- prend une durée Δ en paramètre
- rend une (sur)estimation du nombre de processus pouvant crasher durant Δ unités de temps

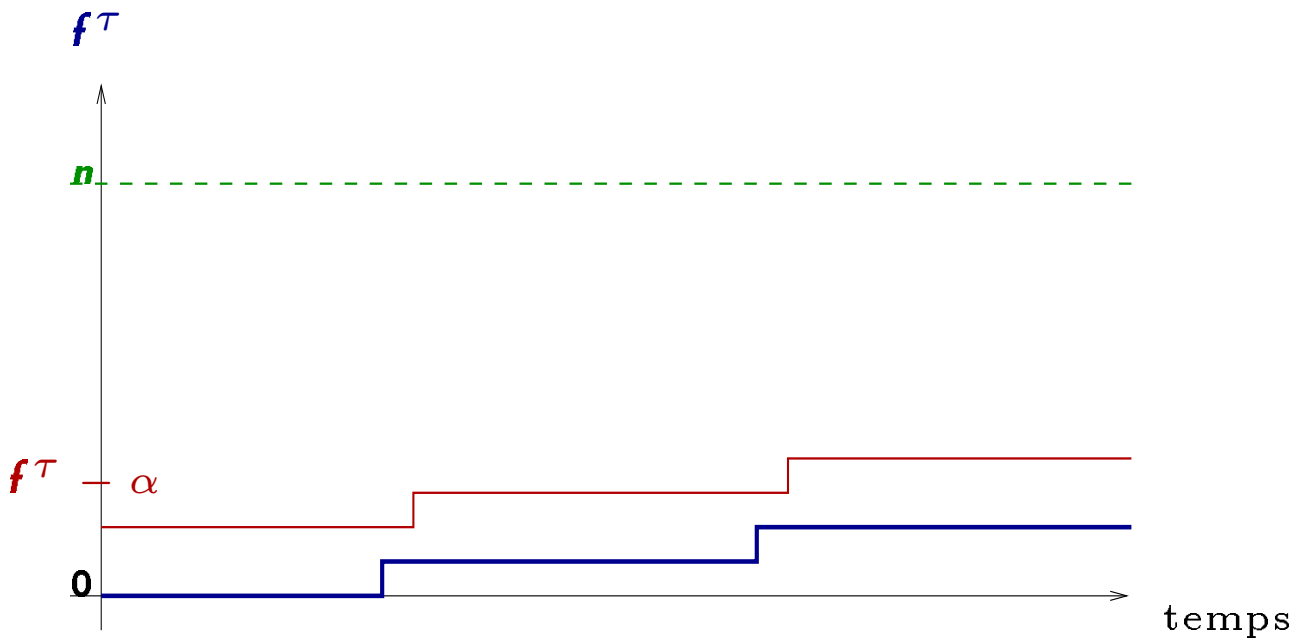


Fig.: nouveau modèle

Nouveau modèle de défaillances

On remplace t par une fonction $\alpha()$ qui ...

- prend une durée Δ en paramètre
- rend une (sur)estimation du nombre de processus pouvant crasher durant Δ unités de temps

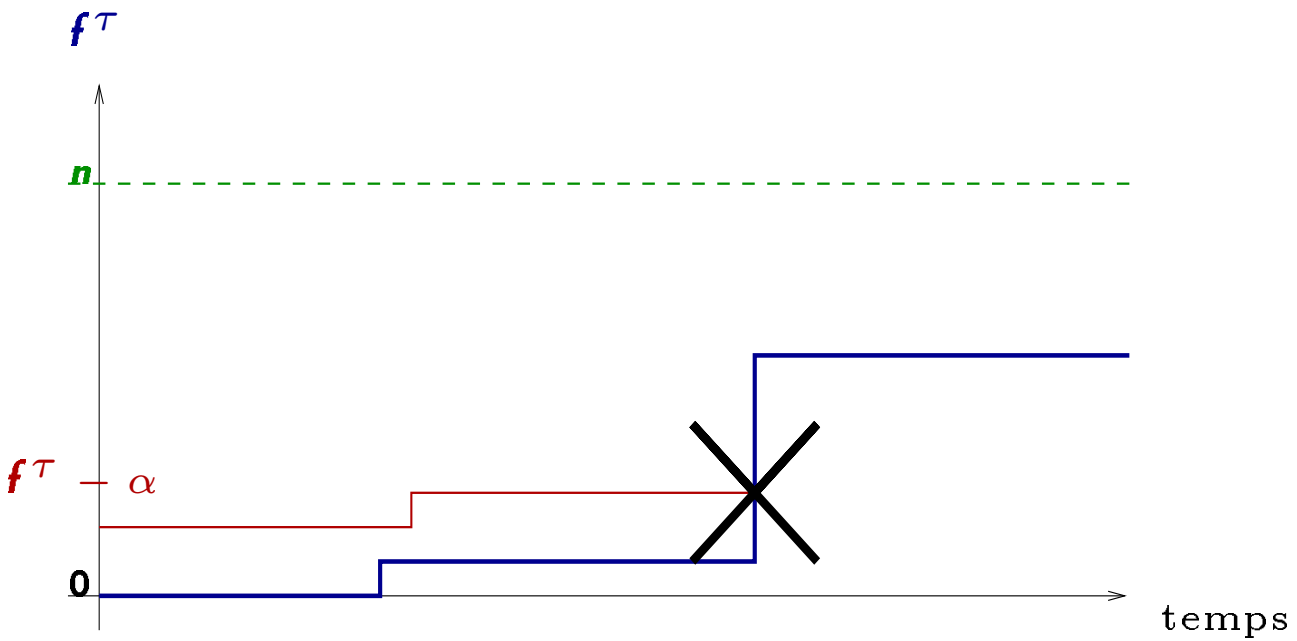


Fig.: nouveau modèle

Nouveau modèle de défaillances

On remplace t par une fonction $\alpha()$ qui ...

- prend une durée Δ en paramètre
- rend une (sur)estimation du nombre de processus pouvant crasher durant Δ unités de temps

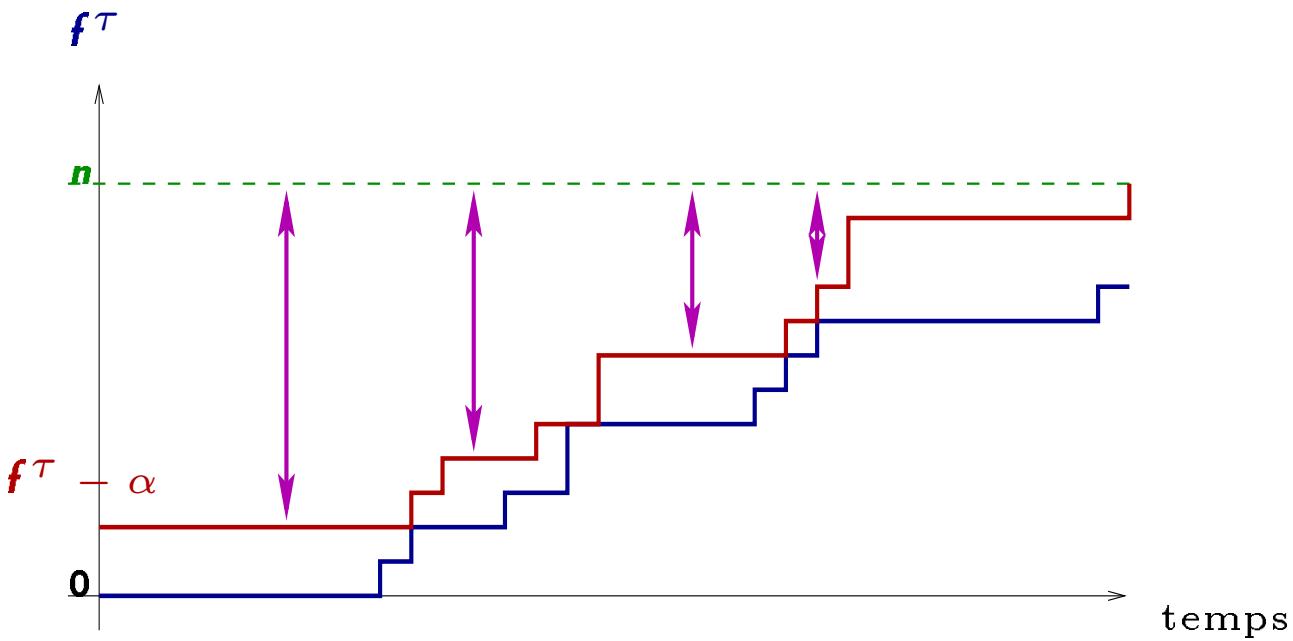


Fig.: nouveau modèle

Estimer le nombre de crashs

objectif Estimer dynamiquement le nombre de processus vivants/crashés dans un système

principe

- Mécanisme de requêtes-réponses
- Système des *réponses gagnantes*.
- Datation de l'information reçue
- Utilisation de la fonction $\alpha()$

comment ? Deux algorithmes

- En supposant que le système possède une horloge globale
- L'équivalent avec des horloges locales

IRISA aSAp

Avec des horloges globales

- Une horloge commune: global_clock().
- Chaque processus exécute régulièrement l'opération estimate()
- Cette opération retourne un nombre de processus supposés vivants

Informellement à chaque appel, chaque processus p_i:

- est **sûr** de ce qu'il voit
- **croît** ce qu'il entend

Algorithme

operation estimate():

(1) $start\_time_i \leftarrow$ global_clock();
(2) broadcast query();
(3) **wait until** $(|est_i.set| - \beta))$ corresponding response(rec_from_j) have been received
(4) **where** $\beta = \alpha(\text{global\_clock}() - est_i.date)$ is continuously evaluated;
% If any, when they arrive, the other corresponding response messages are discarded
(5) $rec\_from_i.date \leftarrow start\_time_i$;
(6) $rec\_from_i.set \leftarrow$ the set processes from which p_i has received response() at line 3;
(7) $est_i.date \leftarrow$ min over the $rec\_from_j.date$ received at line 3;
(8) $est_i.set \leftarrow \bigcup$ of the $rec\_from_j.set$ received at line 3;
(9) return($est_i.set.count$)

background task T:
when query() **is received from** p_j **do** send response(rec_from_i) to p_j **end_do**

Fig.: L'opération estimate() (version basée sur une horloge globale)

Simulation

Un simulateur discret

- simulation discrète
- quanta de temps = une ronde = un appel à estimate() pour chaque processus

Nous cherchons à étudier:

- La **précision**: Quelle chances avons nous d'avoir une bonne estimation ?
- La **vitesse de convergence**: Á quelle vitesse a-t-on une bonne estimation ?

Protocole expérimental:

- 100 processus
- On introduit de fausses suspicions à la ronde 0
- R rondes pour se rétablir.

IRISA aSAp

Simuler les phases critiques

En situation normale:

- bonne estimation $\Rightarrow$ beaucoup de messages
  $\Rightarrow$ beaucoup d'informations $\Rightarrow$ bonne estimation

Après une période très asynchrone:

- mauvaise estimation $\Rightarrow$ peu de messages
  $\Rightarrow$ peu d'informations $\Rightarrow$ peu de chances de se corriger

Ce sont les pires cas que nous voulons simuler

- $\Rightarrow$ aucun crash réel
- $\Rightarrow$ on introduit de fausses suspicions
- $\Rightarrow$ on crée de l'asymétrie

Simuler l'asymétrie

Temps de trajet d'un message = Partie aléatoire + Partie fixe

Partie fixe :

- r routeurs répartis régulièrement
- n processus éparpillés aléatoirement
- chaque processus est rattaché au routeur le plus proche
- Partie fixe(trajet p_i-p_j) = $d(N_i - R_a) + k * d(R_a - R_b) + d(N_j - R_b)$
- k permet de jouer sur l'asymétrie

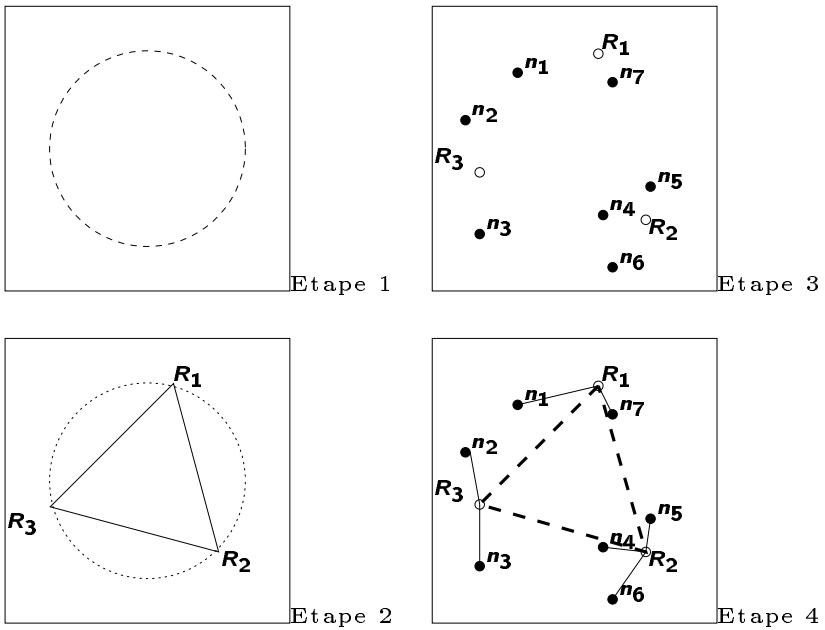
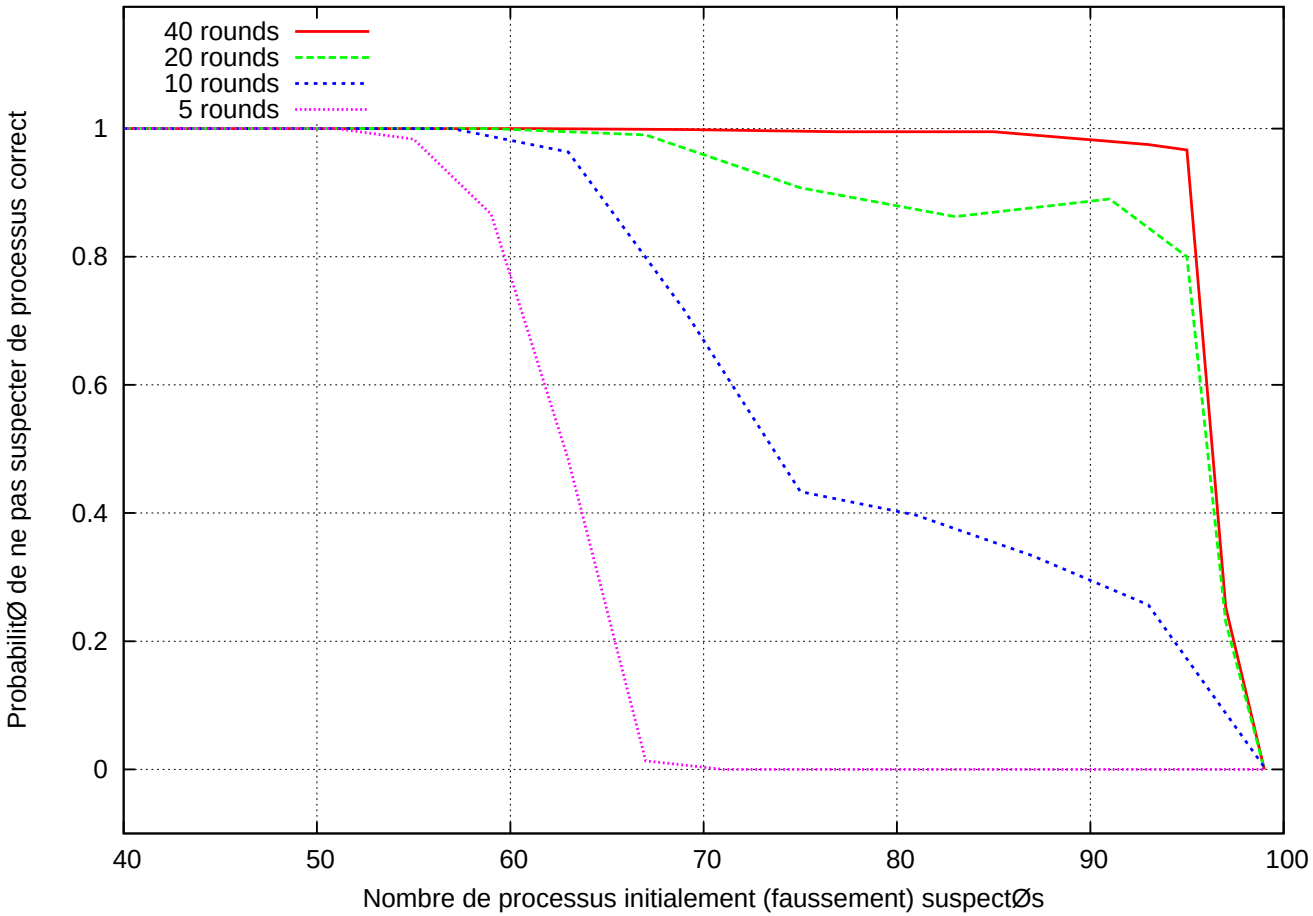


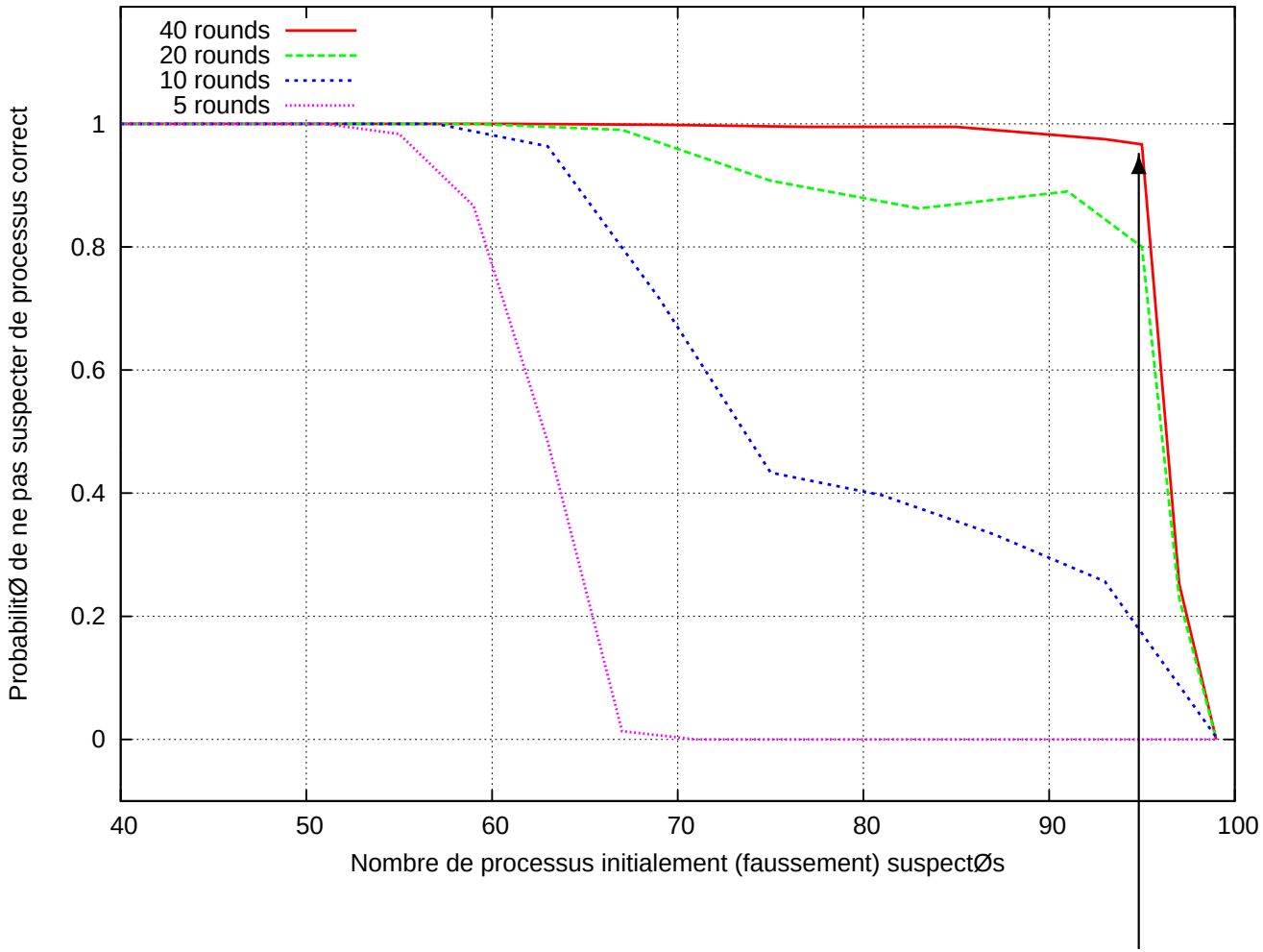
Fig.: Génération d'un topologie

Précision de l'estimation



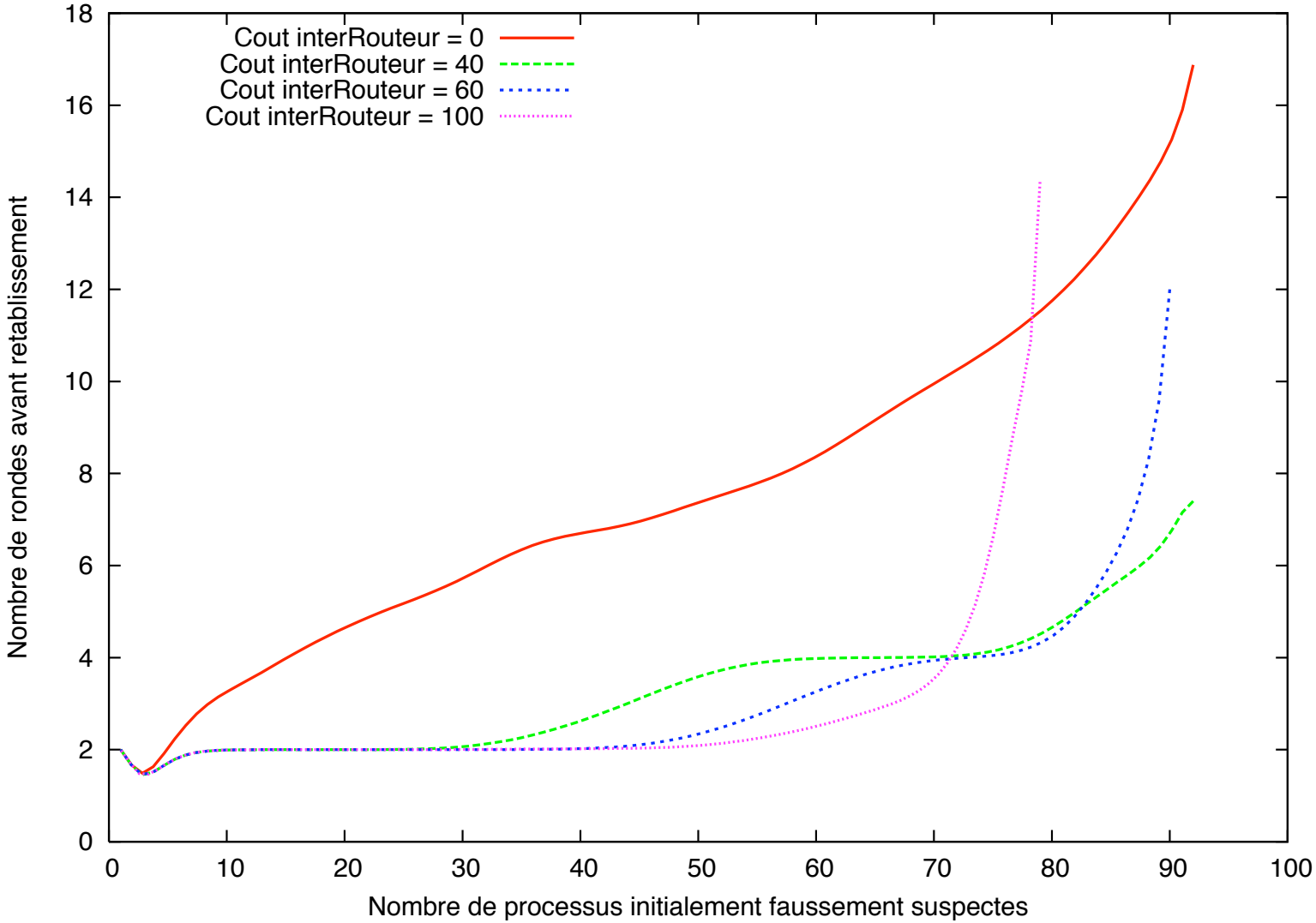
1. La précision est bonne !

Précision de l'estimation



2. Point de chute du système autour de 95% de processus faussement suspectés

Vitesse de convergence



Questions soulevées

- Comprendre...
  - Un protocole simple...
  - aux effets étonnants
  - analyse statistique éprouvante !
- De retour vers un modèle de défaillances avec un t ?
  - choisi par l'utilisateur
  - compromis entre performances et sûreté
- Vers des simulations de réseaux ouverts

Conclusion

Nous avons réalisé

- un protocole qui estime le nombre de processus vivants dans un système
- à l'aide d'un nouveau modèle de défaillances
- stable et précis en situation symétrique