Algorithmes efficaces d'estimation de densité et de diffusion dans les nanoréseaux denses

Thierry Arrabal, Dominique Dhoutaut, **Eugen Dedu**

FEMTO-ST institute, Univ. Bourgogne Franche-Comté, CNRS
Montbéliard, France

http://eugen.dedu.free.fr

Journées du GDR RSD
Toulouse, 18–19 01 2018

Plan

- Introduction aux nanoréseaux
- Simulateur
- Algorithmes :
  - d'estimation de densité
  - de diffusion (broadcast)
- Conclusion

Course vers miniaturisation

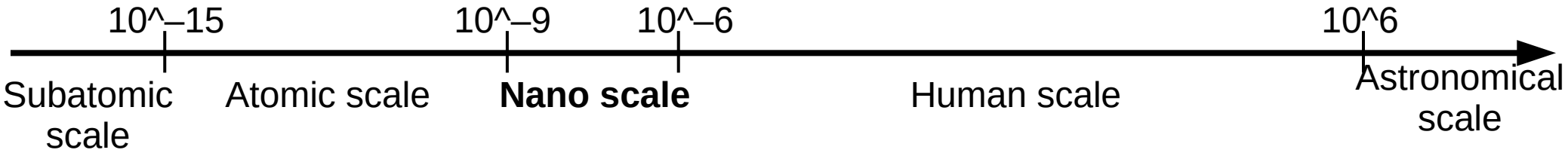
Mainframe IBM 704 (1964)

Arduino

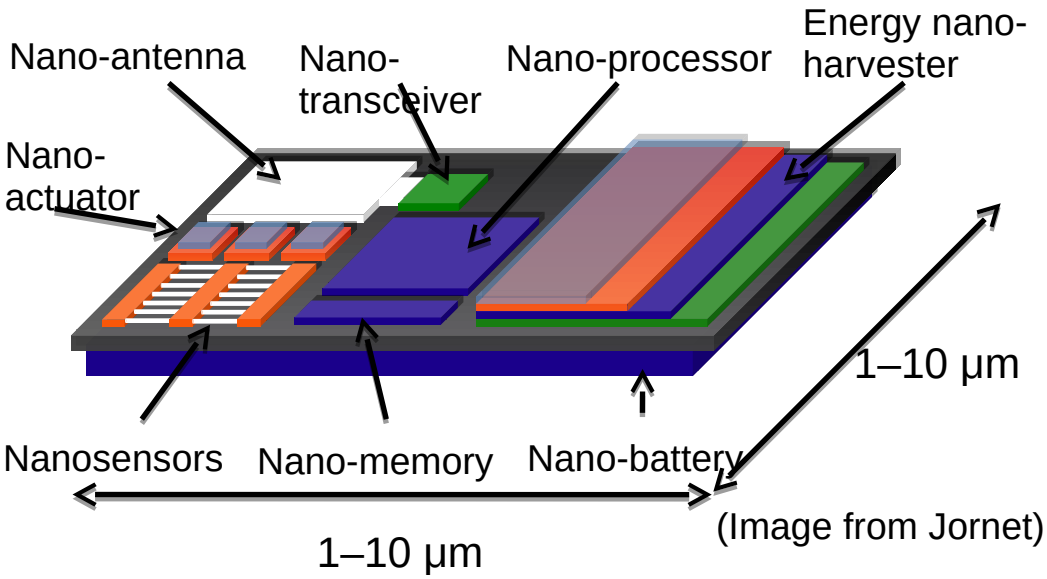
?

Prochaine étape dans la miniaturisation des composants : nano

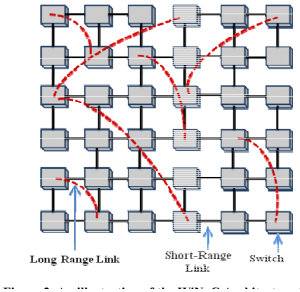
Nanomachine

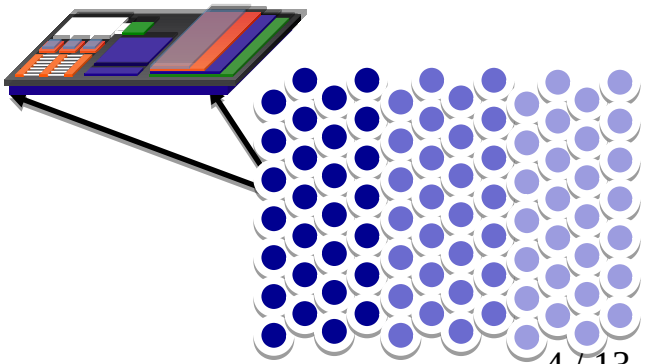


- Exemples de nanothings / applications :
  - biologie : diamètre hélice DNA 2 nm, forme de vie la plus petite 200 nm
  - circuits électriques : µP technologie courante 14 nm, matière programmable
- Domaine de recherche très jeune



(Image from Jornet)





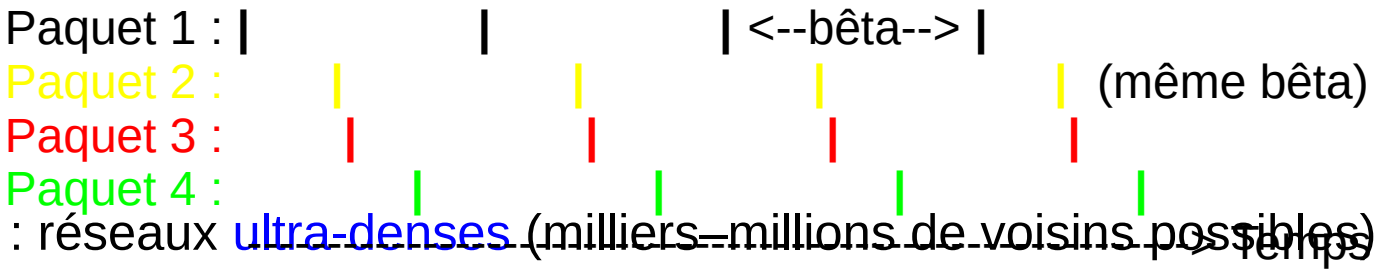
Nanoréseaux électromagnétiques

- Pas encore de nanomachine fabriquée, vu sa petite taille => simulation
- Nanoréseau = réseau sans fil de nanothings, multi-hop possible
- Caractéristiques particulières :
  - peu d'énergie : pas de porteuse, mais pulses et silence => modulation TS-OOK

On sender:
Signal: /____/____.____/____
Bit sent: 1 1 0 1

Signal on receiver:
Expected: /____/____.____/____

  - très petits : attente entre pulses => entrelacement des paquets (multiplexage temporel)

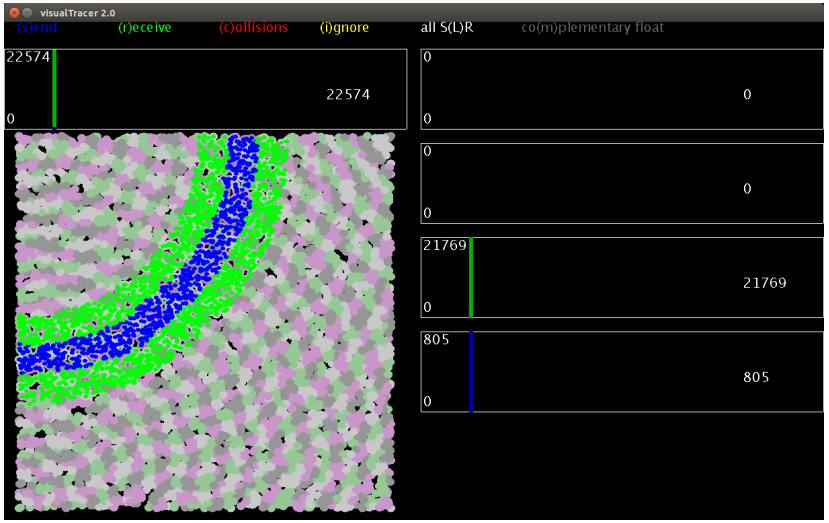


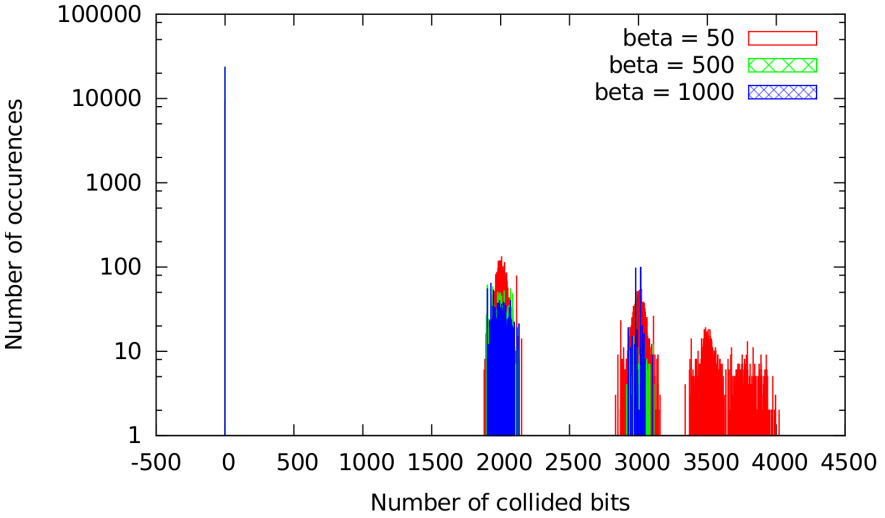
  - très petits : réseaux ultra-denses (milliers–millions de voisins possibles)
  - très petits : antennes petites, grande fréquence (0.1–10 THz), grande atténuation du signal, Tb/s
  - petite capacité de calcul : pas de flottants

BitSimulator

- Ciblé nanoréseaux, rapide en exécution
- À événements discrets (comme NS2/NS3), C++
- Position 3D des nœuds
- TS-OOK (Time Spread On-Off Keying) comme modulation
- Simule les collisions des bits et le temps de propagation
- Flooding, routage SLR (+/- le plus court chemin)
- CBR au niveau application
- Entrée : fichier XML (scénario)
- Sortie : fichiers de log
- Nous a permis de découvrir des propriétés intéressantes qu'on a pu démontrer par la suite
- http://eugen.dedu.free.fr/bitsimulator

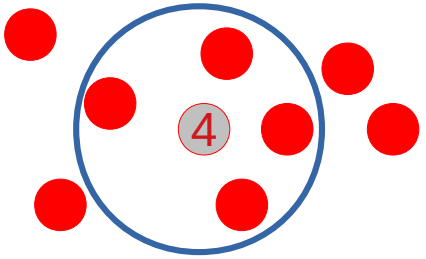
Fenêtre de VisualTracer





Validation des collisions

Estimation du nombre de voisins



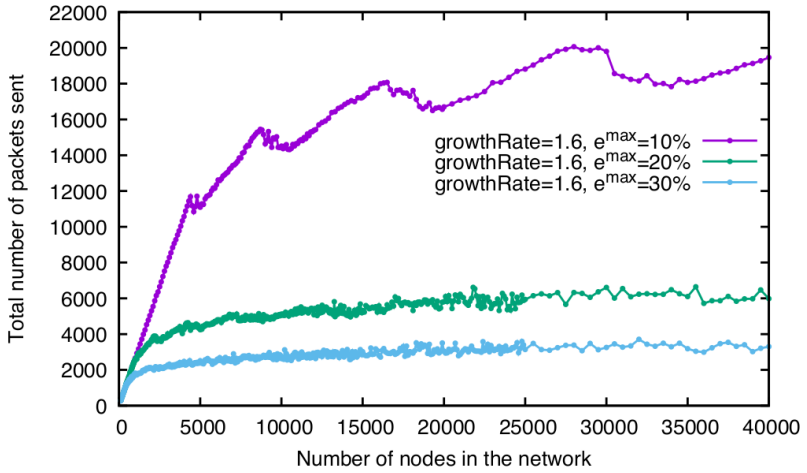
- Intérêt : flooding (voir la suite)
- Hello, solution classique, surcharge trop
- Exécuté au besoin
- Propriétés :
  - mémoire négligeable
  - petite surcharge en nb de paquets échangés
  - communications en background non gênantes
  - confiance et erreur paramétrables (voir la suite)

Fonctionnement :

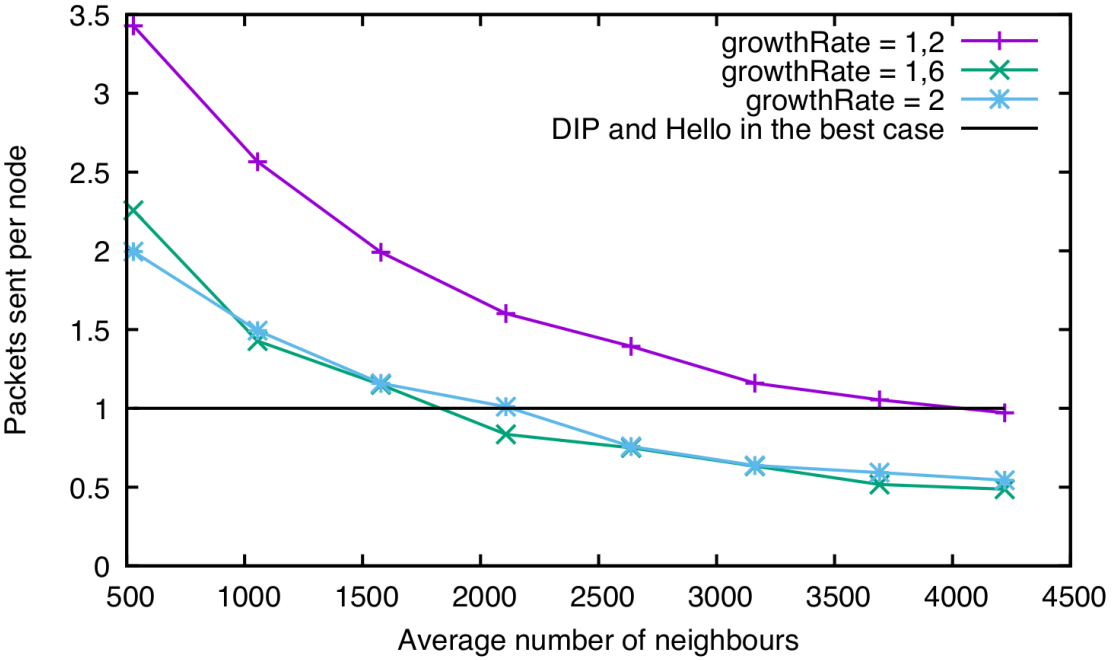
- Plusieurs rounds Ri de durées égales
- Dans chaque round, chaque nœud, après un backoff, émet un paquet avec une probabilité pi
- p0 proche de 0
- pi augmente exp. avec le round : pi *= p0*growthRate^i
- L'algorithme finit quand le nombre de paquets >th, seuil calculé en fonction de la confiance souhaitée (ou quand pi>=1)
- Chaque nœud estime le nombre de voisins en fonction de pr. pi et du nombre de paquets reçus durant le dernier round

Coût de l'algorithme : nombre de paquets échangés

- 2.5mm x 2.5mm
- rayon de comm. 0.5mm
- 100–40000 nœuds placés aléatoirement



emax a une grande influence sur le nb de paquets échangés

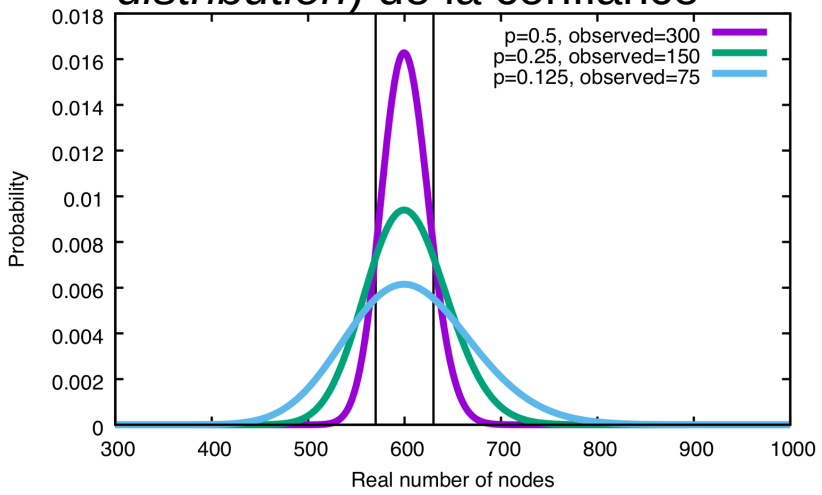


Pour emax = 10%

Qualité de l'algorithme : choix de la confiance et de l'erreur de l'estimation

L'application peut choisir la confiance souhaitée pour une erreur donnée

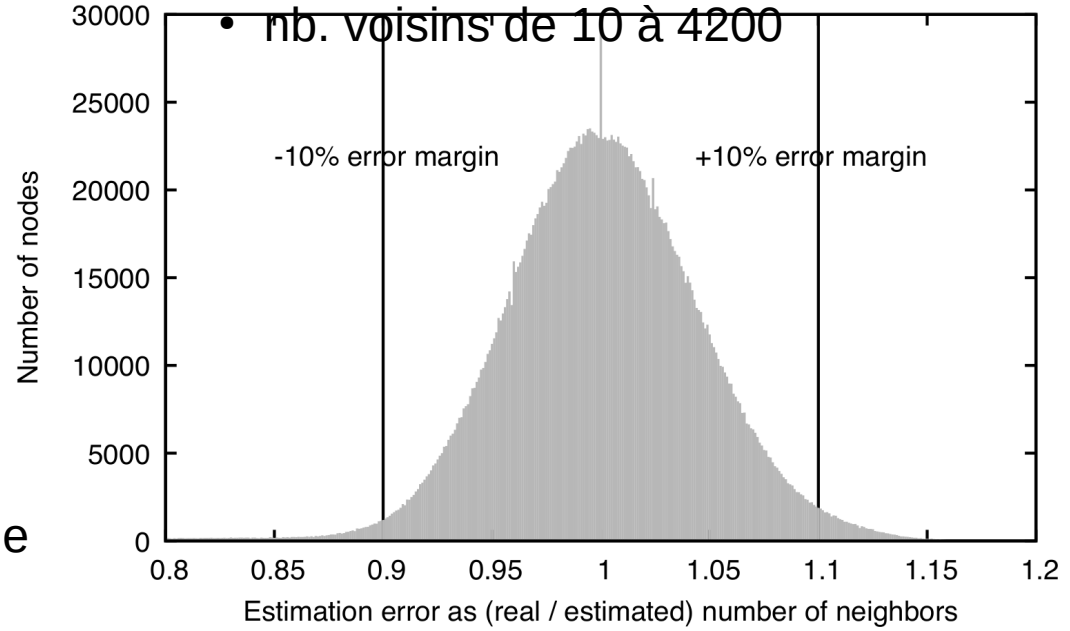
À partir de pi (p) et du nombre de paquets reçus k (observed), il est possible d'afficher la loi de probabilité (*probability distribution*) de la confiance



- 600 voisins comme le plus probable
- traits verticaux : 5% d'erreur

Validation :

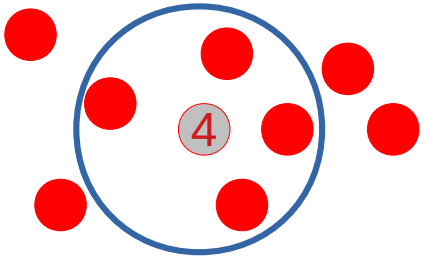
- 2.5mm x 2.5mm
- rayon de comm. 0.5mm
- emax=10%, confiance=95%, k=1.6
- somme de 202 scénarios
- nb. voisins de 10 à 4200

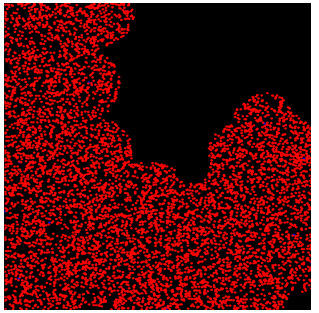


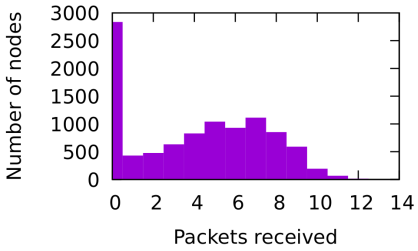
Diffusion : état de l'art

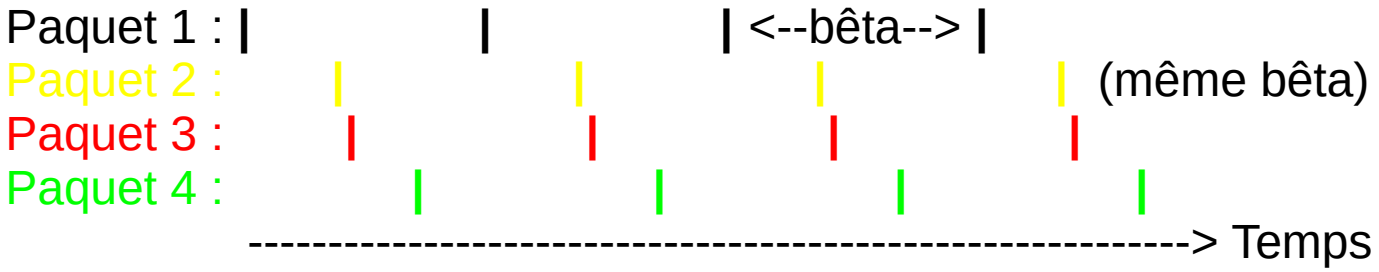
Problèmes des méthodes courantes :

- Flooding pur : énormément de trafic
- Flooding probabiliste adaptatif : probabilité de forward du paquet, la pr. dépend du nombre de voisins : die-out
- Méthodes basées sur compteur : forward si paquet vu maximum *n* fois :
  - à cause du fort multiplexage temporel, de nombreux paquets sont en train d'être reçus avant qu'on puisse décoder *n* paquets









Diffusion : notre méthode

- Solutions aux problèmes :
  - énormément de trafic : estimation la densité
  - die-out : pas de probabilité, mais compteur
  - multiplexage temporel => backoff
- Backoff flooding = méthode hybride incluant :
  - estimateur de densité
  - fenêtre de backoff w :
    w = knt
    avec k constante affectant le nombre de forwarders, n la densité estimée, et t le délai max de forward du paquet (A/R jusqu'au voisin le plus lointain)
  - compteur de paquets – voir figures
- Peu de mémoire et de calcul

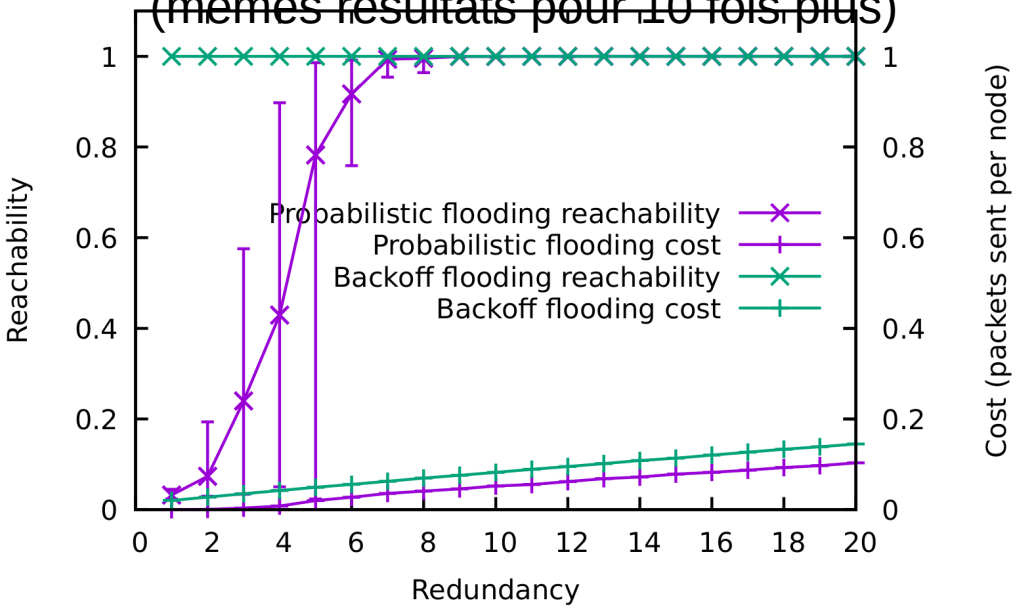
Résultats

- 6mm x 6mm
- le nœud central diffuse un paquet
- rayon de comm. 0.5mm => max 8–9 hops

- ajout de backoff à flooding pur
- atteignabilité = nb nœuds ayant reçu le paquet / nb total nœuds
- coût = nb de paquets / nb de nœuds
- goal : grande atteignabilité avec petit coût
- moyenne de 15 simulations

- Conclusion : notre méthode évite le die-out et est automatique

Atteignabilité et coût en réseaux denses
10000 nœuds (218 voisins)
(mêmes résultats pour 10 fois plus)



Conclusion

- Deux algorithmes efficaces dans les réseaux très denses à multiplexage temporel :
  - algorithme d'estimation de densité (nombre de voisins par nœud)
  - algorithme de diffusion (broadcast)
- Perspectives :
  - les utiliser pour réduire la congestion dans les réseaux saturés
  - optimiser la diffusion en choisissant comme forwarders les nœuds les plus lointains