Vous devez écrire un simulateur implémentant la technique décrite dans ce document ainsi qu'un rapport. Le rapport contiendra une brève description du problème (1/4 du rapport max), une brève description de ce que vous avez implémenté (1/4 du rapport max) ainsi qu'une évaluation des performances du protocole étudié. La troisième séance de TP sera consacrée aux projets. Le travail est à effectuer par groupe de 4 maximum. Le travail est **à rendre pour le 11 mars**.

Consignes :
- Le simulateur est à écrire dans un langage de votre choix parmi C/Java/C++ (si vous préférez le faire dans un autre langage, merci de me le demander).
- Vous devez rendre une archive contenant un **Makefile** ou des **instructions simples** permettant de compiler votre code. Le code doit compiler du premier coup. Un code qui ne compile pas vaut 0.
- Le rapport devra faire **4 pages maximum** en police 10pts (-2pt par page supplémentaire).
- Le rapport est à rédiger dans le format de votre choix (latex, openoffice,...). Ne rendre qu'une version pdf.

Quelques remarques :
- Les différentes questions posées (sur les graphiques ou l'évaluation de performance) ne sont qu'une indication pour vous aider à rédiger votre rapport. Vous êtes encouragés à vous poser d'autres questions et à critiquer vos résultats.
- La limite de pages doit vous pousser à être clair et concis et à faire des choix sur les résultats que vous souhaites présenter.
- Penser à justifier les choix que vous avez faits et à discuter les résultats (est-ce le bon indicateur ? les intervalles de confiance sont-ils bons ? ...)
- Penser à inclure des graphiques clairs et informatifs (voir la *checklist for good graphics*).
- Environ 2/3 des points sera consacré au rapport, 1/3 au code lui-même (fonctions implémentés, qualité et clarté du code,...)

Projet : Réseau ad-hoc

On s'intéresse à un protocole de communication dans un réseau de capteurs. Le protocole qui nous intéresse est un protocole complètement décentralisé. Les réseaux de capteurs sont généralement très variables au court du temps (les capteurs passent leur temps à bouger). On considère donc que les capteurs ne gardent aucune mémoire de la topologie du réseau : quand un capteur i veut envoyer un message à un capteur j, il n'a aucune idée du chemin le plus court, ni même de l'existence d'un tel chemin.

Un capteur qui a un message à envoyer utilise le protocole suivant :

1. Choisir un *time-out* T au delà duquel le message n'est plus valide.
2. "broadcaster" le message à tous ses voisins.

Un capteur i qui reçoit un message utilise le protocole suivant :

1. Si il est le destinataire du message ou qu'il a déjà vu passer le message, il ne fait rien.
2. Sinon :
   - Si $T = 0$, il jette le message.
   - Sinon, avec probabilité q, il "broadcast" le message à tous ses voisins en décrémentant T de 1.

On veut étudier la performance du protocole et en particulier :
- probabilité de réussir à transmettre un message.
- délai de transmission pour émettre un message
- nombre de communications effectuées lors de la transmission d'un message.

Exercice 1. Simulation

Afin de réaliser un simulateur du protocole ci dessus, on fera les hypothèses suivantes :
- La topologie du réseau (*i.e.* qui est connecté à qui) est représentée par un graphe à N sommets.
- Le graphe sera construit aléatoirement en considérant qu'il y a une arrête entre deux noeuds i et j avec probabilité $p > 0$ (indépendamment des autres arrêtes).
- Les paramètres q et T sont communs à tous les noeuds et fixés pour chaque simulation.

- Le temps est discret. Le temps pour envoyer un message depuis un noeud à ses voisins est de 1.
- On s'intéresse à la transmission d'un seul message. Pour cela, on tirera la source i et la destination j aléatoirement puis on simulera le protocole entre i et j.

Écrire un programme qui simule le passage d'un message depuis un noeud à l'autre. Il devra typiquement contenir les données et les fonctions suivantes :

- La représentation du graphe.
- La représentation de l'état des noeuds :
  - Est destinataire
  - A déjà eu le message
  - A en ce moment le message.

Méthodes :

- Tirer un graphe aléatoirement.
- Simuler le protocole entre deux noeuds.

Exercice 2. Quelques courbes

Dans un premier temps, on s'intéresse à la probabilité de réussir à transmettre un message. On note $\mathcal{P}(N, p, q, T)$ la probabilité de réussir à transmettre un message en fonction des paramètres du système.

a. On suppose que $T = \infty$ et $q = 1$: il n'y a pas de time-out et tout le monde transmet le message. Tracer en fonction de p la probabilité qu'un message soit acheminé entre deux noeuds i et j tirés aléatoirement.

On fixe p et N.

b. Tracer $\mathcal{P}(N, p, q, T)$ en fonction de T et en fonction de q.

On s'intéresse maintenant aux nombres de transmissions nécessaires pour transférer un message et au temps de transmission (=le nombre de liens traversés).

c. Tracer le nombre de transmission en fonction de T et q.

d. Tracer le temps de transmission en fonction de q.

Exercice 3. Évaluation des performances

Pour le rapport, on commentera l'efficacité générale du protocole en se basant sur les courbes tracées. En particulier, on pourra se poser les questions suivantes :

- Y a-t-il des comportement radicalement différents en fonction de paramètres ?
- Si on sait que $N \approx 1000$ et $p \approx .1$, et que l'on s'autorise 30% de perte, quels paramètres (q et T) choisir ?
- Est-ce qu'envoyer le même message plusieurs fois améliore grandement ses chances d'être reçu si T est faible ? Si q est faible ?