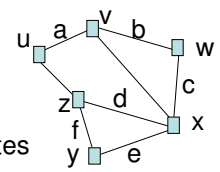
Modèle de l'algorithmique distribuée

Quelques rappels sur les graphes

- Les graphes modélisent une grande variété de problèmes: réseaux de communication, réseaux routiers, structures de molécules, etc
- On se ramène à l'étude de sommets et d'arcs.
- Un graphe simple G est formé de deux ensembles : un ensemble V de sommets et un ensemble E d'arêtes. Chaque arête lie deux sommets (qui sont dits adjacents ou voisins)
- On note G=(V,E)

Exemple

- V = {u,v,w,x,y,z}
- E={a,b,c,d,e,f}



- Le degré d'un graphe:

Nombre d'arêtes adjacentes

deg(x) = 3

Un graphe est régulier de degré r si tous ses sommets son de degré r.

- Lemme de poignées de main:

Soit G=(V,E), la somme des degrés des sommets de V est le double du nombre d'arêtes.

Corollaire : un graphe simple a un nombre pair de sommets de degré impair.

Application: est-il possible de relier 27 ordinateurs de sorte que chaque appareil soit relié avec exactement trois autres ?

Terminologie : graphes

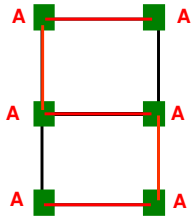
- Sous-graphe: G=(V,E) est sous graphe de G'=(V',E') si V est sous ensemble de V' et E est sous ensemble de E'.
- Chaîne: suite de sommets reliés entre eux par une arête.
- Cycle : chaîne qui revient à son point de départ.
- Graphe connexe: pour toute paire de sommets, il existe une chaîne qui les relie
- Arbre: graphe connexe sans cycle
- Anneau: est un graphe formé d'un cycle
- Longueur d'un chaîne: nombre des arêtes qui composent la chaînes
- Distance entre deux sommets: longueur de la plus courte chaîne joignant les deux sommets
- Diamètre d'un graphe: maximum des distances entre les sommets d'un graphe
- Médian: un sommet qui minimise la somme des distances aux autres sommets

Les arbres

- Soit G un graphe. Nous avons les équivalences suivantes:
  - G est un arbre
  - Entre toute paire de nœuds, il y a un seul chemin simple
  - G est connexe, mais devient non connexe si une arête est supprimée
  - G est connexe et |E|=N-1
  - G est acyclique et |E|=N-1
  - G est acyclique, mais devient cyclique si on rajoute une arête

Arbre recouvrant

- Soit G un graphe. Un arbre recouvrant de G est un sous graphe de G qui est un arbre contenant tous les sommets de G
- Exemple:



Les graphes complets

- Un graphe complet, ou une clique, est un graphe ou chaque paire de sommets est reliée directement par une arête
- Les caractérisations suivantes sont équivalentes:
  - G est un clique
  - |E|=1/2 N (N-1)
  - Chaque sommet est de degré N-1

Rappels sur quelques algos

- Parcours d'un arbre : Visiter tous les sommets
- Parcours en profondeur (DFS): parcourir les branches en profondeur
- Principe de l'algorithme:
  - Visiter un sommet
    - Explorer le premier fils non encore exploré
    - Refaire récursivement

I/ Notion d'algorithme distribué

- Un algorithme distribué sur un système distribué S est une collection de transitions locales à réaliser séquentiellement par chaque process de S
- Ces transitions peuvent s'effectuer suite à des événements (internes ou échange de message)

- Processus:
  - Les instructions d'un processus sont considérées comme atomiques.
  - Il possède une mémoire locale
  - Il possède un état local (ensemble de ses données et des valeurs de ses variables locales)
  - Il peu (ou non) avoir un identifiant
  - Pas ou peu de connaissance des autres processus et de leur état (il peut connaitre l'état des voisins)
  - Il s'exécute en parallèle que les autres processus

- Un algorithme désigne celui d'un seul process et protocole désigne l'ensemble des algos (mais on ne fait cette distinction)
- Événement: interne (changement d'état), envoi de message, réception de message
- Process p: ($etat_i$, $evenmt_j$) ($etat_{i+1}$,$evenmt_{j+1}$)
- Simulation d'un mémoire partagée: 1 process à part qui et chargé uniquement de la communication
- Terminaison: explicite vs implicite

II/ Modèle du réseau

- Process = sommet ; Canal = arête

1. Types de réseaux
   A. Réseaux centralisés: Il existe un seul process dans l'état centraliseur (ou élu) et tous les autres dans l'état battu
   B. Réseaux avec identités: (named network) Chaque process de S est distingué par l'attribution d'une identité unique, distincte de tous les autres et connue par lui seul
   C. Réseaux anonymes: les process sont symétriques du point de vue de l'exécution de l'algorithme et ne possèdent pas d'identités (connues d'eux). Ils sont indiscernables.

2. Propriétés du réseau
   - Le graphe sous-jacent est connexe
   - Les messages sont acheminés le long des canaux. Les délais sont finis mais non bornés (peuvent être arbitrairement longs)
   - Les messages reçus par un process sont traités dans leur ordre d'arrivée (si en //, ordre arbitraire)
   - Pour tout couple (pi,pj), l'ordre de réception des messages est le même que celui de transmission (FIFO)

- Modes de transmission
  - simplex : unidirectionnel
  - half duplex : pas de croisement
  - full duplex : croisement possible (mode général, graphe non orienté)
- Les états peuvent être codés par des étiquettes
- Topologie du réseau (arbre, anneau, complet,…)

III/ Modèle de communication (synchronisme)

1. Synchrone
   - horloge globale
   - L'exécution est partitionnée en rounds

   (un round correspond à un envoi et à une réception de tous les messages. Envoi instantané ou borne fixe)
   - Les process ont la même vitesse

2. Asynchrone
   - Pas d'horloge globale
   - Pas de borne sur le délai de transmissions de messages
   - Les process sont asynchrones (les horloges sont locales)

Modèles de pannes

- Modèle de fautes : 3 grands types de fautes ou pannes
  - Franches : le processus ne fait plus rien
  - Par omission : des messages sont perdus ou non délivrés
  - Arbitraires ou byzantines : le processus renvoie des valeurs fausses et/ou fait « n'importe quoi »
    - Cas de fautes les plus complexes à gérer
    - Les autres fautes peuvent être considérées comme des cas particuliers des fautes byzantines
- Processus correct
  - Processus non planté, qui ne fait pas de fautes
  - Dans le cas où on considère les reprises après erreurs et la relance de processus : processus qui pouvait être incorrect précédemment mais qui est correct maintenant

- ***Fautes par omission***
- Processus ou canal ne réalise pas une action
- Classification des fautes
  - Fail-stop : un processus s'arrête et reste hors-service. Les autres processus peuvent détecter que ce processus est planté.
  - Crash : un processus s'arrête et reste hors-service mais les autres processus ne peuvent pas détecter que le processus est planté.
  - Omission du canal : un message positionné dans un tampon de sortie d'un processus n'arrive jamais dans le tampon d'entrée du destinataire
  - Omission en émission : un processus envoie un message mais il n'est pas placé dans le tampon d'émission
  - Omission en réception : un message est placé dans le tampon d'entrée d'un processus mais le processus ne le reçoit pas

- ***Fautes arbitraires / byzantines***
- Fautes arbitraires
  - Appelées aussi fautes byzantines
  - Fautes quelconques et souvent difficilement détectables
  - Processus
    - Calcul erroné ou non fait
    - État interne faux
    - Valeur fausse renvoyée pour une requête
  - Canal
    - Message modifié, dupliqué voire supprimé
    - Message « créé »
    - En pratique peu de fautes arbitraires sur les canaux car les couches réseaux assurent la fiabilité de la communication
- Fautes arbitraires : classement en 2 catégories
  - Malicieuses : erreurs volontaires (virus, attaques ...)
  - Naturelles : problème non voulu, non déclenché

IV/ Mesures de complexité

Modèle synchrone :

- Nombre de messages : depuis le début jusqu'à la fin de l'algo.
- Taille d'un message : complexité en nombre de bits
- Temps : nombre de rounds

- Modèle asynchrone :
  - Nombre de messages
  - Taille d'un message
  - Temps (moins évident): généralement, on suppose que le temps de transmission sur un canal de communication mesure une unité

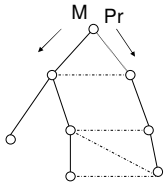
  (ou bornée par une constante)
  - Complexité en espace : espace pour un process
  - Calcul de complexité au pire : max sur toutes les exécutions possibles (généralement le pire des exécutions synchrones)
  - Complexité moyenne : valeur moyenne de la complexité (selon une distribution de probabilités)

V/ Quelques algorithmes de base

1. Un algorithme de diffusion (Broadcast, PI)

- Hypo: un arbre recouvrant donné, un message M
- Un sommet Pr distingué (racine)
- Pr envoie M à tous ses fils
- Si un sommet reçoit le message M de son père, il l'envoie à ses fils

```
Algo pour Pi    i ≠ r
   receive (pere(Pi),M)
      pour tout fils Pj de Pi
           send (Pj,M)
    terminer
```



- Exercice: Calculer la complexité en nombre de messages ? et la complexité en temps ?

2. Inondation et arbre recouvrant :

un graphe G, un message M à envoyer à tous les sommets

2.1/ Inondation, Diffusion, PI (Propagation d'Information)

- Un sommet distingué (racine) P
- P envoie M à tous ses voisins
- Un process Pi qui reçoit M de Pj pour la première fois, l'envoie à tous ses voisins sauf à Pj

Exercice:

Montrer que la complexité en messages est de 2m-n+1

2.2/ Application à la construction d'un arbre recouvrant

Quand Pi reçoit M pour la première fois de Pj, il envoie à Pj un message <parent> et <refus> à tous les autres.

Si Pj reçoit <parent> de Pi alors fils(Pj) = Pi