Université de Paris

UFR Informatique M2 IMPAIR

Algorithmique Répartie

Devoir 3: examen final

Ce devoir est un devoir de confinement il faut le rendre avant le 1-er mai, mais il est conseillé de le rendre avant. Comme pour les précédents vous pouvez le rendre par groupe d'au plus 3 étudiants à la condition que ce soit le résultat d'un travail de vraie collaboration entre tous les auteurs ce qui peut être difficile dans la situation actuelle. Aussi on vous conseille plutôt de le faire seul mais d'échanger avec les autres (et avec moi) pour le réaliser.

Ce devoir est un peu particulier, il s'agit d'une étude sur un problème d'algorithmique distribué classique: le problème de l'attaque coordonnée. On peut voir ce devoir comme un cours avec des questions. Le but est de vous permettre de réfléchir et de comprendre ce problème. La plupart des questions ont une réponse très simple (des fois tellement simple qu'on doute de la réponse). La principale difficulté est le formalisme.

Préliminaires

Le problème de l'attaque coordonnée est le suivant:

Nous sommes en présence de deux armées: l'armée A et l'armée B (le problème peut se généraliser à n armées) qui sont séparées par un territoire occupé par des ennemis. Le seul moyen de communication entre A et B est par l'envoi de messagers qui doivent traverser les lignes ennemies. En traversant les lignes ennemies, ces messagers peuvent être capturés. Si les deux armées attaquent ensemble, elles triomphent et sinon elles perdent. Les deux généraux sont prudents et ne lanceront l'attaque que s'ils sont sûrs de gagner et donc s'ils sont arrivés à se mettre d'accord sur l'attaque. On suppose de plus chaque général a une opinion sur l'attaque et peut être favorable ou défavorable à l'attaque.

Le problème est donc de trouver un moyen d'arriver à un tel accord en échangeant des messagers. Bien sûr le problème est que ces messagers peuvent être capturés et donc que l'envoi d'un messager ne permet pas de savoir si le messager est arrivé. Pour arriver à un accord il ne suffit pas que les deux généraux sachent que l'un et l'autre veulent attaquer.

Par exemple, supposons que A veut l'attaque et qu'il a reçu un message de B lui disant qu'il veut aussi attaquer, ce n'est pas suffisant: en effet B ne sait pas que son messager est arrivé (peut-être a-t-il été capturé?) et donc ne sait pas si A sait qu'il (B) veut l'attaque. La prudence de B l'empêche donc de décider de l'attaque, mais aussi A qui sait raisonner en déduit que B ne va pas attaquer et donc A qui est prudent ne peut pas décider l'attaque.

Mais même si B a reçu un messager lui disant que A a bien reçu le messager indiquant que B voulait l'attaque, cela ne suffira pas: A ne sait pas que ce messager est arrivé jusqu'à B et donc pour lui il n'y a pas de différence avec la situation précédente... et donc il ne peut pas décider de l'attaque. Comme on le voit, on peut prolonger ce raisonnement indéfiniment avec un nombre quelconque de messagers échangés entre A et B. Intuitivement on en déduit qu'il ne sera pas possible que le général A et le général B se mettent d'accord sur l'attaque.

Formalisation

Revenons au problème de l'attaque coordonnée que nous allons un peu formaliser.

Chaque général est un processus $p \in \Pi = \{p_1, \cdots, p_n\}$. Les processus peuvent communiquer par envoi réception de messages les (messagers) qui peuvent être perdus sans restriction. Chaque message a une source (un processus) et une destination (destination). Le graphe de communication est un graphe dont les sommets sont des processus, (p, q) est un arc du graphe si p peut envoyer un message à q. En général, sauf mention explicite du contraire on supposera que ce graphe est un graphe complet: n'importe quel processus peut envoyer un message à n'importe quel autre processus. On supposera toujours que ce graphe est connexe: de façon directe ou indirecte tout processus peut communiquer avec n'importe quel autre. On dira que la communication est *asynchrone* si un message peut être reçu arbitrairement longtemps (possiblement jamais) après avoir été émis, et *synchrone* si un message reçu est (s'il n'est pas perdu) au bout d'une unité de temps. On supposera en général que les processus sont synchrones dans le sens où il ont accès à un horloge parfaite et que chaque instruction prend un temps déterminé et connu (notons que la réception d'un message n'est pas une instruction d'un processus).

Nous allons préciser ce que permet le cas synchrone pour deux processus A et B (en fait on peut généraliser cette définition à un ensemble quelconque de processus): on va considérer des algorithmes en *rondes synchronisées*. Chaque processus exécute des rondes indicées successivement par des entiers de $0, \ldots n, \ldots$ A la ronde i:

- suivant son état au début de la ronde i, A envoie un messager[1]
- A reçoit le message envoyé par B dans la ronde $i-1$ (pour $i > 0$) et B reçoit le message envoyé par A dans la ronde $i-1$
- A change son état en fonction de son état de la ronde $i-1$ (pour $i > 0$ et son état initial sinon) et du message reçu (symétriquement pour B qui change d'état en fonction de son état dans la ronde précédente et du messager reçu)
- suivant le nouvel état obtenu le processus A envoie un messager au processus B (symétriquement pour B qui envoie un messager fonction de son état).

On peut se convaincre (essayez!) que le cas synchrone permet de réaliser des rondes synchronisées telles qu'indiquées ci-dessus. On en déduit que: Pour tout problème P il existe un algorithme permettant de résoudre P dans le cas synchrone si et seulement si il existe un algorithme en rondes synchronisées.

Question 1.— Soit un problème $\mathcal{P}$, montrer (ou plutôt donner des arguments convaincants) que:

1. Le cas synchrone permet de réaliser des rondes synchronisées telles qu'indiquées ci-dessus. En déduire que pour tout problème $\mathcal{P}$ il existe un algorithme permettant de résoudre P dans le cas synchrone si et seulement si il existe un algorithme en rondes synchronisées. (On pourra se restreindre au cas de deux processus)
2. Montrer que s'il n'y a pas d'algorithme pour résoudre $\mathcal{P}$ dans le cas synchrone il n'y en a pas non plus dans le cas asynchrone (on pourra utiliser le fait que l'existence d'un algorithme en asynchrone entraîne l'existence d'un algorithm en synchrone).

Attaque coordonnée

Précisons maintenant la spécification du problème de l'attaque coordonnée. Chaque processus (général) p a:

- une valeur initiale $v_p \in \{0, 1\}$ indiquant l'opinion du processus sur l'attaque (0 le général n'est pas favorable à l'attaque, 1 le général est favorable).

[1] Par envoyer un messager on entend envoyer un messager portant un certain message.

- une variable d_p indiquant la valeur de décision. Cette variable ne peut être écrite qu'une seule fois (la décision est irrévocable). Initialement, elle est indéfinie ($\perp$) et pourra prendre la valeur 1 (attaque) ou 0 (non-attaque). Quand p écrit cette variable on dira que p décide: p décide l'attaque signifie ainsi que p écrit la valeur 1 dans cette valeur.

La spécification de l'attaque coordonnée est donnée par les trois propriétés suivantes:

- *accord:* deux processus ne peuvent décider des valeurs différentes: si $d_p \neq \perp$ et $d_q \neq \perp$ alors $d_p = d_q$
- *validité:*
  - Si tous les processus ont 0 comme valeur initiale alors la seule décision possible est 0.
    $(\forall p \in \Pi : v_p = 0) \Rightarrow \forall p \in \Pi : (d_p \neq \perp \Rightarrow d_p = 0)$
  - Si tous les processus ont 1 comme valeur initiale et aucun message n'est perdu alors la seule décision possible est 1:
    $(\forall p \in \Pi : v_p = 1 \wedge$ aucun message n'est perdu$) \Rightarrow \forall p \in \Pi : (d_p \neq \perp \Rightarrow d_p = 1)$
- *Terminaison:* Tous les processus décident.

On notera que la condition de validité est très faible: elle permet de décider 0 dès que des messages sont perdus ou qu'un des processus a 0 comme valeur initiale, elle oblige à décider quand tout va bien (aucun message perdu et tous les processus étaient favorable à l'attaque) et et à décider 1 si aucun processus n'était favorable à l'attaque.

Nous conseillons à tous de bien vérifier que la définition plus formelle ci-dessus correspond bien à la description informelle de l'attaque coordonnée donnée plus haut.

Le but de ce devoir est de montrer qu'il n'existe pas d'algorithme garantissant les propriétés de l'attaque coordonnée.

Question 2.—

1. Montrer que s'il n'y a pas d'algorithme garantissant l'attaque coordonée pour deux processus A et B parmi n processus, il n'y en a pas non plus dans le cas d'un graphe de communication quelconque (mais connexe). En déduire que pour montrer l'impossibilité de l'attaque coordonnée (c'est-à-dire qu'il n'existe pas d'algorithme) dans n'importe quel graphe de communication il suffit de montrer qu'il n'existe pas d'algorithmes en rondes synchronisées pour deux processus A et B.

Preuve de l'impossibilité

Le principe de la preuve est le suivant: on suppose l'existence d'un algorithme pour l'attaque coordonnée en rondes synchronisées entre A et B et on en déduit une contradiction. Pour cette contradiction, on va considérer différentes exécutions de cet algorithme dans différentes conditions de communication (pertes des messagers) et une relation dite d'*indistingabilité* entre ces exécutions (si deux exécutions sont indistingables elles doivent décider la même chose). Ensuite on construira une séquence d'exécutions chacune étant indistingable de la précédente, telle que le première amène nécessairement à une décision 1 et la dernière nécessairement à une décision 0. De la sorte on obtient une contradiction (les exécutions étant indistingables deux à deux, elles doivent décider la même chose.

Schéma de communication et indistingabilité

Rappelons que l'on suppose qu'il existe un algorithme pour l'attaque coordonnée. Les algorithmes considérés ici sont déterministes: dans nos rondes synchronisées ce que fait un processus dans la ronde i dépend entièrement de son état au début de la ronde et de ce qu'il a reçu (ou n'a pas reçu) comme message. Pour la ronde 0 l'état initial est le même pour toutes les exécutions et ne dépend que la valeur initiale du processus.

- Ce qui se passe à la ronde 0 pour A (respectivement pour B) dépend entièrement de la valeur initiale v_A (respectivement v_B).
- Pour la ronde 1, considérons A par exemple, soit il reçoit le messager de B (qui, ne dépendant que de la valeur initiale de B v_B, aura la même valeur dans toutes les exécutions où B aura la valeur initiale v_B) et donc le comportement de A ne dépendra que la valeur initiale v_A de la valeur initiale v_B et du fait que le messager de la ronde 0 de B vers A arrive ou non à destination.
- Ainsi de suite (en fait par induction) on vérifie que le comportement de A (ou de B) à la ronde i dans une exécution de dépend que de (1) les valeurs initiales v_A et v_B et (2) du fait que dans les rondes précédentes des messagers ont été ou non capturés.

Un *schéma de communication* décrit cette histoire des messagers capturés. Il décrit quels messagers sont capturés. Plus formellement, schéma de communication S est un ensemble de couples $(\rightarrow, i)$ ou $(\leftarrow, i)$ dont a signification est la suivante:

- $(\rightarrow, i) \in S$ si et seulement si le messager de la ronde i de A vers B n'a pas été capturé
- $(\leftarrow, i) \in S$ si et seulement si le messager de la ronde i de B vers A n'a pas été capturé

Par exemple:

- $S = \{(\rightarrow, i)$ pour tout $i \geq 0$ $\}$ correspond au schéma de communication pour lequel tous les messagers issus de A arrivent dans toutes les rondes mais aucun messager issu de B n'arrive en A.
- $S = \{(\rightarrow, i)(\leftarrow, i)$ pour tout $0 \leq i \leq m$ $\}$ correspond au schéma de communication pour lequel aucun messager jusqu'à la ronde m n'est cpaturé, mais tous les messagers après la ronde m sont capturés. Dans la suite on appellera ce schéma $SP(m)$ (sans perte jusqu'à m).

Question 3.—

1. Décrire le schéma de communication de pour lequel aucun messager n'est jamais capturé (on appellera dans la suite ce schéma de communication le schéma SP (sans perte))
2. Décrire le schéma de communication pour lequel tous les messagers sont capturés (on appelera ce schéma TP (tout capturé))
3. Quelles peuvent être les décisions pour le schéma SP si les valeurs initiales sont $(1,1)$[2] ?pour les valeurs initiales $(0,0)$? pour les valeurs initiales $(0,1)$?
4. Quelles peuvent être les décisions pour le schéma TP si les valeurs initiales sont $(0,0)$?

On va écrire plus formellement ce qu'est une exécution. Il est conseillé de vérifier que les définitions formelles qui suivent correspondent à l'intuition. Une exécution α décrit ce qui se passe à chaque ronde: l'état de A et l'état de B, le messager envoyé par A s'il est reçu, le messager de B s'il est reçu.

Une exécution α est déterminée par:

- Processus A:
  - la séquence $(S^A_\alpha(i))$ pour $i \geq 0$ où $(S^A_\alpha(i)$ est l'état de A au début de la i-ème ronde, (en particulier, $S^A_\alpha(0)$ est l'état initial de A c'est-à-dire la valeur initiale v_A de A)
  - $M^A_\alpha(i)$ pour $i \geq 0$ où $M^A_\alpha(i)$ est le messager emis par A à la i-éme ronde *si ce messager est reçu par* B et $\emptyset$ si ce message n'est pas reçu.
- Idem pour le processus B:

[2]quand on dit que les valeurs initiales sont (x, y) cela signifie que la valeur initiale de A, v_A, est égale à x et la valeur initiale de B, v_B est égale à y

– la séquence $(S^B_\alpha(i))$ pour $i \geq 0$ où $(S^B_\alpha(i)$ est l'état de B au début de la i-ème ronde, (en particulier, $S^B_\alpha(0)$ est l'état initial de B c'est-à-dire la valeur initiale v_B de B)

– $M^B_\alpha(i)$ pour $i \geq 0$ où $M^B_\alpha(i)$ est le messager emis par B à la i-éme ronde *si ce messager est reçu par* A et $\emptyset$ si ce message n'est pas reçu.

On notera $\alpha[k]$ l'exécution α jusqu'à la ronde k. Les remarques du début du paragraphe (que l'on pourrait prouver formellement) indiquent qu'étant donné un algorithme une exécution est entièrement déterminée par l'état initial de A et B et un schéma de communication. Dans la suite on identifiera une exécution avec (v_A, v_B) et un schéma de communication.

On va maintenant définir quand deux exécutions sont indistingables pour un processus. Intuitivement deux exécutions sont indistingables pour un processus si pour ce processus il se passe exactement la même chose (il reçoit les mêmes messages et avait le même état initial) dans ces deux exécutions. Pour cela, on dira que α et α' sont indistingables jusqu'à la ronde k (noté $\alpha[k] \equiv_i \alpha'[k]$) pour le processus i (i étant soit A soit B) si et seulement le processus i a reçu les mêmes messages jusqu'à la ronde k dans α et α' et l'état initial de i est le même dans α et α'.

On dira que $\alpha[k] \equiv \alpha[k]$ si $\alpha[k] \equiv_A \alpha[k]$ ou $\alpha[k] \equiv_B \alpha[k]$.

Enfin, on notera $\alpha \equiv \alpha'$ si et seulement si pour tout k tel que $\alpha[k] \equiv \alpha'[k]$.

Question 4.—

1. Montrer que pour tout execution α il existe une valeur de décision unique $d(\alpha)$ (on pourra déduire cette propriété de *terminaison* et de l'*accord*), montrer que si $\alpha \equiv \alpha'$ alors $d(\alpha) = d(\alpha')$ (deux exécutions indistingables pour un processus décident la même chose)

2. Pourquoi dans un algorithme pour l'attaque coordonnée, l'exécution TP (tous les messagers sont capturés) avec la valeur initiale $(0, 0)$ décide forcément 0?

   Pourquoi dans un algorithme pour l'attaque coordonnée, l'exécution SP (aucun messagers n'est capturé) avec la valeur initiale $(1, 1)$ décide forcément 1?

3. Commençons par définir C: le schéma de communication SP (sans aucune perte de messagers) et l'état initial $(1, 1)$. Montrer qu'il existe k tel que A a décidé à la ronde k montrer aussi que cette valeur de décision est 1.

   Nous allons dans ce qui suit construire des séquences d'exécutions (C_i) et (C'_i) pour $(0 \leq i \leq k)$ en enlevant à chaque fois le dernier messager emis par A ou par B:

   (a) Soit C_k: le schéma de communication est tel que tous les messagers à partir la ronde k sont capturés (le schéma de communication est $\{(\rightarrow, i)(\leftarrow, i) : 0 \leq i < k\}$) et l'état initial est $(1, 1)$. Montrer que $d(C_k) = 1$.

   (b) Soit C_{k-1}: le schéma de communication identique à celui de C_k sauf que le dernier messager de A vers B est capturé (schéma de communication de C_k moins $(\rightarrow, k-1)$) et l'état initial est $(1, 1)$. Montrer que $C_{k-1} \equiv_A C_k$, en déduire que $d(C_{k-1}) = d(C_k) = 1$

   (c) Soit C'_{k-1}: le schéma de communication identique à C_{k-1} sauf que le dernier messager de B vers A est capturé (schéma de communication de C_{k-1} moins $(\leftarrow, k-1)$)) et l'état initial est $(1, 1)$. Montrer que $C'_{k-1} \equiv_B C_{k-1}$, en déduire que $d(C'_{k-1}) = 1$

   (d) En continuant de la sorte pour $0 < \ell \leq k-1$ on définit de façon récursive:

   - $C_{\ell-1}$ est identique à C'_ℓ sauf que le dernier messager de A vers B est capturé (schéma de communication de C'_ℓ moins $(\rightarrow, \ell-1)$) et l'état initial est $(1, 1)$).
   - $C'_{\ell-1}$ est identique à $C_{\ell-1}$ sauf que le dernier messager de B vers A est capturé (schéma de communication de C_ℓ moins $(\leftarrow, \ell-1)$) et l'état initial est $(1, 1)$.

   Quel est le schéma de communication pour C'_0? (pour ceux d'entre vous qui sont capturés c'est le schéma TP -tous les messages sont capturés-), montrer que $d(C'_0) = 1$.

(e) Soit D l'exécution avec le schéma de communication TP et l'état initial $(1, 0)$, montrer que $D \equiv_A C'_0$ en déduire que $d(D) = 1$

Soit E l'exécution avec le schéma de communication TP et l'état initial $(0, 0)$, montrer que $E \equiv_B D$ en déduire que $d(D) = 1$

En quoi a-t-on obtenu une contradiction?

(f) Quelle(s) conclusion(s) tirez-vous ce qui précède? (réponse libre)

Que peut-on déduire de ce qui précède en ce qui concerne les systèmes distribués avec des pertes de messages?

Comment fait TCP pour garantir la fiabilité des transmissions de message? Quelle hypothèse est faite sue les pertes de messages pour assurer cette fiabilité des messages?

En guise de conclusion...

Notons que ce qu'il faut pour décider de l'attaque est une propriété très forte. Il s'agit de la propriété de "notoriété" (common knowledge en anglais). Par exemple ce qui vous permet de ne pas vous arrêter à un feu vert c'est que le code de la route est "notoire": vous savez que les autres savent qu'ils doivent s'arrêter au feu rouge et qu'ils savent que vous le savez etc...

La notoriété a quelque chose à voir avec une histoire qu'on raconte sur le ville de Bagdad. Nous allons raconter cette histoire sous sa forme originale malgré les sous-entendus douteux qu'elle contient. Si certains trouvent que cette histoire est au moins de mauvais goût avec des relents sexistes ou racistes, elle permet de comprendre plus facilement les choses

La ville de Bagdad est dirigée par un calife qui sait tout. Les habitants sont très obéissants et sont de parfaits logiciens. Un jour le calife dit: il y a des épouses infidèles à Bagdad et si un mari sait qu'il est trompé, il doit exécuter son épouse à 5h le lendemain matin.

Le principe est qu'un mari trompé ne voit pas qu'il est trompé, par contre il voit les autres maris trompés. Alors que se passe-t-il? Supposons qu'il y a k épouses infidèles à Bagdad. Chaque mari voit $k-1$ épouses infidèles si son épouse est infidèle et k si sa femme n'est pas infidèle. Au matin du premier jour, si $k = 1$ un époux ne voyant aucune femme infidèle en déduit que comme le calife dit vrai son épouse est infidèle et la tue le lendemain matin. Mais si $k > 1$ aucune épouse ne sera assassinée le premier jour. Aussi 2-ème jour si persone n'a été assassiné tout le monde déduit que $k > 1$ si $k = 2$ les deux maris trompés ne voyant qu'une seule femme infidèle en déduisent que c'est la leur - et les assassine le lendemain matin. Si $k > 2$ aucune femme ne sera assassinée le deuxième jour. Si maintenant au m-ème jour aucune femme n'a été assassinée tout le monde en déduit qu'il y a plus que m épouses infidèles. Si $k = m$ alors m maris voient $m-1$ épouses infidèles et les autres voient m épouses infidèles: le lendemain les m maris tuent leur épouses.

On peut se demander quel est l'effet de la phrase initiale du calife. D'abord s'il s'est trompé, et qu'en fait il n'y a pas d'épouses infidèles à Bagdad, que se passe-t-il?

Mais s'il y a au moins deux femmes infidèles alors ce que dit le calife n'apprend rien à personne: tout le monde voit sans ce que dit le calife qu'il y a des épouses infidèles. En fait ce que dit le calife permet faire passer l'assertion suivant laquelle il y a une femme infidèle au statut de la "notoriété": avant si $k > 2$ tout le monde le savait, mais grâce à l'assertion du calife tout le monde sait que tout le monde sait que... il y a une épouse infidèle à Bagdad.

Le devoir montre dans une certaine mesure que dans un système distribué, en présence de défaillances des communications une assertion ne peut jamais atteindre la niveau de la "notoriété".