Master 1 (Maîtrise) d'Informatique

Algorithmes distribués. Synchronisation d'horloges

A. Slissenko

Algorithme de synchronisation d'horloges de Mahany-Schneider.

On considère un modèle *réseau en impulsion avec délais bornés*. Pendant chaque impulsion chaque processus peut envoyer et recevoir un nombre fini de messages. Il envoie d'abord et reçoit aprés l'envoi. Le temps d'arrivée de messages peut varier pour des messages différents, néanmoins les délais se trouvent entre certaines bornes.

Puisque chaque processus peut détecter d'où arrive un message, il peut facilement détecter les processus morts. Pour cette raison on considère comme processus incorrects les processus qui ne suivent pas l'algorithme local donné. On les appelle *byzantins*.

Le problème est de développer un algorithme de synchronisation distribué pour N processus avec délais internes négligeables et avec délais de communications externes avec des bornes données. Les délais externes concernent l'interaction entre les processus, à savoir le temps de communication.

Il y a $B < \frac{N}{3}$ processus parmi N qui peuvent être byzantins. On peut dire que des byzantins veulent empêcher aux autres $\geq (N - B)$ qui sont *corrects* d'arriver à une amélioration de synchronisation. Les corrects exécutent le protocole (algorithme) spécifié, à savoir *ClockSync*, voir Fig. 1. Les processus byzantins ne suivent aucun protocole, on peut penser qu'ils connaissent tout sur les corrects et peuvent faire des coalitions entre eux pour agir contre les corrects.

Par défaut, dans la description de l'algorithme de synchronisation et de ses propriétés on ne parle qu'aux processus corrects. Chaque processus (correct) p a une horloge interne C_p. Au temps réel (temps 'physique') t l'horloge C_p montre le temps $C_p(t)$. La fonction C_p est strictement monotone.

Nous supposons que les horloges sont *parfaites* pendant le fonctionnement de l'algorithme de synchronisation, i. e. $C(t+\tau) = C(t)+\tau$. Les horloges parfaites, une fois synchronisées, ne demandent plus de réglage.

Les horloges d'un système distribué ne montrent pas le temps réel, i. e. $C_p(t) = t$ n'a pas lieu forcement. Les horloges sont *δ-synchronisées* au temps réel t si $|C_p(t) - C_q(t)| \leq \delta$. Le paramètre δ est la *précision* de synchronisation. Le but de l'algorithme de synchronisation est d'améliorer la précision de synchronisation.

Le délai de communication entre les processus est borné du dessous par δ_{min} et du dessus par δ_{max}, où $0 \leq \delta_{min} \leq \delta_{max}$. Plus formellement, si un message est envoyé au temps réel t et reçu au temps réel t', alors

$$\delta_{min} \leq (t' - t) \leq \delta_{max}. \tag{1}$$

Notations : $\delta^+ =_{df} (\delta_{max} + \delta_{min})$, $\delta^- =_{df} (\delta_{max} - \delta_{min})$.

La valeur δ^- caractérise la *désynchronisation de communication*.

Une amélioration de synchronisation est faisable si $\delta \gg \delta^-$ (i. e. que δ est fortement plus grand que δ^-).

Nous supposons que toutes les horloges sont δ-synchronisées initialement.

L'algorithme *ClockSync* de la Figure 1 utilise des fonctions abstraites et pré-interprétées ; ces dernières concernent le temps. Le temps, représente par une sorte *Temps*, est continu.

Estimator (estimateur) est une fonction qui transforme une suite (ou un tableau) A de réels en un réel qui se trouve dans $[\min\{A\}, \max\{A\}]$. Par exemple, on peut prendre comme *estimator* la moyenne de A ou le maximum ou le minimum etc.

L'algorithme utilise les fonctions internes (tableaux) A_p, D_p ; les deux du type $\mathbb{P} \to Temps \cup \{\infty\}$, où $\mathbb{P}$ est l'ensemble de processus. La valeur initiale est ∞ pour tous : $A_p(q) = D_p(q) := \infty$ (ligne 0).

On suppose qu'un "lanceur" envoie à un moment t_0 à tous les processus un signal de lancement de synchronisation. Ce signal arrive à p à un moment $\tau_p \in (t_0 + \delta_{min}, t_0 + \delta_{max}]$. A ce moment-là chaque p (correct) diffuse $C_p(\tau_p)$ à tous les processus (ligne 2). Il attend $2\delta_{max}$ pour recevoir les valeurs des autres et traite chaque valeur reçue pour former le tableau D_p (ligne 3). Pour les processus d'où p n'a rien reçu, il laisse ∞ dans D_p (ligne 2). Ensuite il filtre les valeurs de D_p et obtient A_p qui a ∞ pour les valeurs non reçues ou non retenues par le filtrage (opérateur 4). Le processus p 'complète' ce tableau en remplaçant ∞ par la valeur d'estimator (opérateurs 5–7). A partir du nouveau tableau il calcule la valeur qui corrige son horloge (opérateurs 8–9). La nouvelle valeur de son horloge est calculée dans la ligne 9.

Remarque 1. Il est plus naturel si on suppose que chaque processus connaît les moments de synchronisation $TS_1, TS_2, \ldots$ qui sont les mêmes pour tous et plus grands que la précision de synchronisation. Par exemple chaque p sait qu'il doit commencer une séance de synchronisation aux instants 12h 13h,14h etc., et la précision de synchronisation est 10sec, et les délais de communications et la précision initiale de synchronisation sont largement plus petits que la distance entre TS_i consécutifs. Nous considéron une séance de synchronisation et dénote le moment de début de synchronisation t_0. À cause de désynchronisation le processus p commence sa séance au moment $C_p(t_0)$ qui est différent de t_0. Dans ce cas le décalage du début de travail entre les processus est δ, et il faut attendre $(\delta + \delta_{max})$ pour être sûr que les signaux des autres arrivent.

Une autre version plus naturelle est la suivante. Chaque processus connaît les moments de lancement (qui sont bien séparés par rapport à δ_{max}). Il gère une fonction $session\mathbb{P} \to Bool$ qui lui dit s'il est en session de synchronisation. Après avoir terminé une session, le processus p attend $(\delta + 2\delta_{max})$ temps et met sa fonction à $false$. Un processus p commence une session de synchronisation si : (a) $\neg\, session_p$ et $C_p = momentDeSynchronisation$ ou (b) $\neg\, session_p$ et p reçoit un signal de lancement d'un autre processus. En tout cas, au début d'une session de synchronisation p met $session_p$ à $true$.

Un processus p au moment de lancement d'une session de synchronisation diffuse un signal de lancement. Chaque processus après avoir reçu tel signal commence sa session de synchronisation. Donc si plusieurs signaux sont envouyés à peu près en même temps, un processus réagit au premier signal reçu qui peut être de lui-même. Cette version correspond à la version avec lanceur.

$\underline{ClockSync_p}$:

Tout processus commence son travail après avoir reçu le message le lancement.

0 : **par forall** q **do** $A_p(q) = D_p(q) := \infty$ **endpar** ;
-- Initialisation des tableaux A_p et D_p ;

1 : $H_p := C_p$;

-- sauvegarde du temps du début de travail

2 : **par forall** q **do Send** $\langle \mathbf{val}, H_p \rangle$ **to** q **endpar** ;

3 : **while** $C_p \leq H_p + 2\delta_{max}$ -- Ici C_p joue le rôle du temps courant.
par forall q **do**
if $\langle \mathbf{val}, H \rangle$ est reçu de q
then $D_p(q) := (H - H_p)$
endpar ;
endwhile ;

4 : **par forall** q **do**
if $\#\{r : |D_p(q) - D_p(r)| \leq \delta + (\delta_{max} - \delta_{min})\} \geq (N - B)$
then $A_p(q) := D_p(q)$
endpar ;

5a : $i, q := 1$;

5 : **while** $q \leq N$ **do**
if $A_p(q) < \infty$ **then** $A'_p(i) := A_p(q)$; $i := i + 1$ **endif** ;
$q := q + 1$;
endwhile ;

6 : $esti_p := estimator(A'_p)$;

7 : **par forall** q **do**
if $A_p(q) = \infty$ **then** $A_p(q) := esti_p$
endpar ;

8 : $\Delta_p := \frac{1}{N} \sum A_p$;

9 : $C_p := C_p + \Delta_p$

FIG. 1: Algorithme de synchronisation d'horloges

Exemple 1. Considérons un exemple de fonctionnement de l'algorithmes pour 7 processeurs dont 2, soit 6 et 7, sont byzantins. Soit le temps initial 0. Prenons comme estimateur la fonction max ; avec cet estimateur ce n'est pas la peine de calculer A'_p.

Délais de communication et synchronisation initiale :
$\delta_{min} = 3$, $\delta_{max} = 5$, $\delta = 32$.
On calcule les valeurs utilisées dans l'algorithme :
$\delta^- = (\delta_{max} - \delta_{min}) = 2$.

Définissons les valeurs initiales $C_p(0)$ des horloges des processus corrects et les délais d_p d'arrivée des signaux de début de travail du lanceur :

p	1	2	3	4	5
$C_p(0)$	100,5	80,5	92,3	112,5	103
d_p	3	3	5	5	4.4

TAB. 1: Valeurs initiales des processus corrects

On peut vérifier que les horloges sont 32-synchronisées initialement.

Les processus byzantins peuvent envoyer des valeurs arbitraires :

Envoi à :	1	2	3	4	5
de 6	70	50	58	140	143
de 7	130	50	60	142	133

TAB. 2: Valeurs envoyées par processus byzantins

On calcule des tableaux ci-dessus qu'au début du travail les valeurs d'horloges sont les suivantes :

p	1	2	3	4	5
$C_p(d_p)$	103.5	83.5	97.3	117.5	107.4

TAB. 3: Valeurs H_p

On peut voir que les valeurs s'est écartées davantage et la différence entre eux dépasse δ :
$C_4(5) - C_2(2.3) = 117.5 - 83.5 = 34$,
mais elle de dépasse pas $(\delta + \delta^-) = 34$.

Remarquons que le moment de correction (ligne 9) est $(t_0 + d_p + 2\delta_{max})$, dans notre cas il est $(d_p + 10)$. À ce moment les horloge montrent avant la correction les valeurs qui se trouvent dans Tab. 4

Chaque processus p exécute l'opérateur 3 jusqu'à l'instant $t_0 + d_p + 2\delta_{max} = t_0 + d_p + 10$. On suppose que la vitesse de travail des processus est très grande par rapport à la vitesse de la communication, donc nous considérons que le calcul de chaque processus est instantané. Alors à la fin de calcul les nouvelles valeurs d'horloges sont calculées pour les moments qu'on voie dans les arguments : $C_1(12)$, $C_2(12)$, $C_3(15)$, $C_4(15)$, $C_5(14.4)$. Et pour estimer la synchronisation on les compare avec les valeurs initiales. Donc il faudra les 'normaliser' et comparer les valeurs :

p	1	2	3	4	5
$C_p(t_0 + 2\delta_{max} + d_p)$	113.5	93.5	107.3	127.5	117.4

TAB. 4: Valeurs des horloges au moment d'arrivée à la correction.

$C_1(12) - 12$, $C_2(12) - 12$, $C_3(15) - 15$, $C_4(15) - 15$, $C_5(14.4) - 14.4$.

Maintenant on fait le calcul pour des processus corrects. Les valeurs D_p sont dans le Tab.5.

Après avoir calculé D_p le processus p filtre les valeurs $D_p(q)$ et les met dans A_p : il laisse seulement les valeurs pour lesquelles il trouve au moins $(N - B)$ témoins. Ceci est indiqué dans l'opérateur 4 (dans notre exemple $(N - B) = 5$ et $\delta + (\delta_{max} - \delta_{min}) = 34$) :

$\#\{r : |D_p(q) - D_p(r)| \leq 34\} \geq 5$

Considérons $p = 1$. Voici le calcul des témoins pour chaque $D_1(q)$.

Pour $D_1(1) = 0$: $\{r : |0 - D_1(r)| \leq 34\} = \{1, 2, 3, 4, 5, 6, 7\}$ on a 7 témoins, donc la valeur est retenue. En fait, on peut voir que toutes les valeurs des processus corrects seront retenues (mais un processus correct ne sait pas qui est correct et qui est byzantin !), voir Proposition 1 ci-dessous.

Pour cette raison on va avoir au moins 5 témoins pour tout $D_1(q)$ pour $1 \leq q \leq 5$. Il nous reste de vérifier les valeur reçues des q byzantins.

Pour $D_1(6) = -33.5$: $\{r : |-33.5 - D_1(r)| \leq 34\} = \{1, 2, 3, 6\}$ on a seulement 4 témoins, donc $A_1(6)$ garde sa valeur initiale ∞.

Pour $D_1(7) = 26.5$: $\{r : |26.5 - D_1(r)| \leq 34\} = \{1, 3, 4, 5, 7\}$ on a 5 témoins, la valeur est retenue. Remarquons que cette valeur reçue d'un byzantin devient la valeur de l'estimateur.

Continuons le calcul pour les autres corrects.

Pour $D_2(6) = -33.5$: $\{r : |-33.5 - D_2(r)| \leq 34\} = \{2, 6, 7\}$ on a 3 témoins, la valeur n'est pas retenue.

Pour $D_2(7) = -33.5$: pareil.

Pour $D_3(6) = -39.3$: $\{r : |-39.3 - D_3(r)| \leq 34\} = \{2, 6, 7\}$ on a 3 témoins, la valeur n'est pas retenue.

Pour $D_3(7) = -37.3$: $\{r : |-37.3 - D_3(r)| \leq 34\} = \{2, 6, 7\}$ on a 3 témoins, la valeur n'est pas retenue.

Pour $D_4(6) = 22.5$: $\{r : |22.5 - D_4(r)| \leq 34\} = \{4, 5, 6, 7\}$ on a 4 témoins, la valeur n'est pas retenue.

Pour $D_4(7) = 24.5$: $\{r : |24.5 - D_4(r)| \leq 34\} = \{4, 6, 7\}$ on a 3 témoins, la valeur n'est pas retenue.

Pour $D_5(6) = 35.6$: $\{r : |35.66 - D_4(r)| \leq 34\} = \{4, 6, 7\}$ on a 3 témoins, la valeur n'est pas retenue.

Pour $D_5(7) = 25.6$: $\{r : |25.66 - D_4(r)| \leq 34\} = \{1, 4, 5, 6, 7\}$ on a 5 témoins, la valeur est retenue.

On voit dans le Tab. 5 que après une application de l'algorithme de synchronisation la désynchronisation est 5.6 à la place de 32.

□

Exemple 2. L'algorithme ne marche pas si $B = \frac{N}{3}$. Considérons la situation suivante : les processus 1 et 2 sont corrects, 3 est byzantin, donc $N = 3$, $B = 1$, $(N - B) = 2$. Prenons 0 comme le moment t_0 du début de travail, $\delta_{min} = \delta_{max} = 100$, $\delta = 10$.

Définissons les conditions initiales. Les délais de début de travail sont évidemment les mêmes $d_p = 100$. Prenons comme les valeurs initiales des horloges pour les corrects : $C_1(0) = 0$, $C_2(0) = 10$.

$T_p =_{df} (t_0 + d_p + 2\delta_{max}) = (d_p + 10)$
$C_p(T)$ est la valeur de l'horloge de p corrigée observée au moment de correction.
$C_p(t_0)$ est la valeur corrigée normalisée au moment t_0.

$D_p,\ A_p$	1	2	3	4	5	6	7	correction Δ_p	corrigés $C_p(T_p)$	$C_p(t_0)$
D_1	0	−20	−6.2	14	3.9	−33.5	26.5			
$A_1(init)$	0	−20	−6.2	14	3.9	∞	26.5			
$A_1(final)$	0	−20	−6.2	14	3.9	26.5	26.5	6.4	119.9	106.9
D_2	20	0	13.8	34	23.9	−33.5	−33.5			
$A_2(init)$	20	0	13.8	34	23.9	∞	∞			
$A_2(final)$	20	0	13.8	34	23.9	34	34	22.8	116.3	103.3
D_3	6.2	−13.8	0	20.2	10.1	−39.3	−37.3			
$A_3(init)$	6.2	−13.8	0	20.2	10.1	∞	∞			
$A_3(final)$	6.2	−13.8	0	20.2	10.1	20.2	20.2	9.0	116.3	101.3
D_4	−14	−34	−20.2	0	−10.1	22.5	24.5			
$A_4(init)$	−14	−34	−20.2	0	−10.1	∞	∞			
$A_4(final)$	−14	−34	−20.2	0	−10.1	0	0	−11.1	116.4	106.4
D_5	−3.9	−23.9	−10.1	10.1	0	35.6	25.6			
$A_5(init)$	−3.9	−23.9	−10.1	10.1	0	∞	25.6			
$A_5(final)$	−3.9	−23.9	−10.1	10.1	0	25.6	25.6	3.3	120.7	106.3

TAB. 5: Tableau de calcul des valeurs d'horloges après synchronisation

Donc au début de travail des corrects ont $C_1(100) = 100$, $C_2(100) = 110$. Supposons que le byzantin envoie à 1 la valeur 90 et à 2 la valeur 120. On a $D_1 = (0, 10, -10)$ et $D_2 = (-10, 0, 10)$. Le processus 1 a 2 témoins pour la valeur -10, à savoir lui-même et le byzantin. Pareil pour 2. Donc A_p final est égal à D_p, $p = 1, 2$. La correction Δ_p est 0 pour les deux corrects. Donc les valeurs d'horloges corrigées : $C_1(100) = 100$ et $C_2(100) = 110$. On voit qu'il n'y a pas d'amélioration de synchronisation. □

Exemple 3. Pour 4 processus dont 2 byzantins on peut avoir une détérioration de la précision de synchronisation. Maintenant $(N - B) = 2$, donc pour retenir une valeur un processus a besoin de 2 témoins. Réprenons les mêmes valeurs $\delta_{min} = \delta_{max} = 100$, $\delta = 10$, $t_0 = 0$. Soit les horloges de processus corrects au moment 100 de lancement de travail : $C_1(100) = 100$, $C_2(100) = 110$. Supposons que les byzantins envoient à 1 la valeur 90 et à 2 la valeur 120. Dans ce cas $D_1 = (0, 10, -10, -10)$ et $D_2 = (-10, 0, 10, 10)$. Toutes les valeurs sont retenues et donc A_p final est égal à D_p, $p = 1, 2$. Les corrections : $\Delta_1 = -2.5$, $\Delta_1 = 2.5$. Les valeurs d'horloges corrigées : $C_1(100) = 97.5$ et $C_2(100) = 102.5$. La précision de synchronisation est devenue 15 à la place de 10 initiale. □

Exemple 4. Pour 4 processus dont un byzantin un trucs pareil (comme dans les Exemples 2, 3) ne passe pas. Maintenant $(N-B) = 3$, donc pour retenir une valeur un processus a besoin de 3 témoins. Réprenons les mêmes valeurs $\delta_{min} = \delta_{max} = 100$, $\delta = 10$, $t_0 = 0$. Soit les horloges de processus corrects au moment 100 de lancement de travail : $C_1(100) = 100$, $C_2(100) = C_3(100) = 110$. Supposons que le byzantin envoie à 1 la valeur 90 et à 2 et 3 la valeur 120. Dans ce cas le processus 1 a pour la valeur $(90 - H_1) = -10$ seulement les témoins 1 et 4 et donc cette valeur n'est pas retenue par 1. Les processus 2 et 3 retiennent $(120 - H_p) = 10$ (témoins : 2, 3 et 4). Les tableaux finaux (on prend comme estimateur max) : $A_1 = (0, 10, 10, 10)$, $A_2 = A_3 = (-10, 0, 0, 10)$.

Nous avons $\Delta_1 = 7.5$, $\Delta_2 = \Delta_3 = 0$, donc les nouvelles valeurs d'horloges : $C_1(100) = 107.5$, $C_2(100) = C_3(100) = 110$. L'amélioration de la synchronisation est de 4 fois. □

Notation : d_{pr} est la valeur laquelle p a insérée dans D_p pour r après l'exécution de l'opérateur 3.

Il est évident que $H_p = C_p(t_0) + (\tau_p - t_0)$ (car les horloges sont parfaites), que
$\tau_p \in (t_0 + \delta_{min}, t_0 + \delta_{max}]$
et que

$$d_{pq} = C_q(\tau_q) - C_p(\tau_p)$$
$$= C_q(t_0) + (\tau_q - t_0) - C_p(t_0) - (\tau_p - t_0)$$
$$= C_q(t_0) - C_p(t_0) + \tau_q - \tau_p. \quad (2)$$

Proposition 1 (sur la distance entre d_{pr}.)
Tout de suite après la construction de D_p on a
$$|d_{pr} - d_{qr}| \leq \delta + (\delta_{max} - \delta_{min}) = \delta + \delta^-$$
pour les corrects p, q et r.

Preuve. On a de (2)

$$|d_{pr} - d_{qr}| = |C_r(t_0) - C_p(t_0) + \tau_r - \tau_p - C_r(t_0) + C_q(t_0) - \tau_r + \tau_q|$$
$$= |C_q(t_0) - C_p(t_0)| + |\tau_q - \tau_p| \leq (\delta + \delta^-).$$

■

Théorème (de synchronisation.)
Après une exécution de ClockSync les horloges des processus corrects sont synchronisées avec la precision

$$\widetilde{\delta} =_{df} (\delta_{max} - \delta_{min}) + \frac{2B}{N} \left(\delta + (\delta_{max} - \delta_{min}) \right), \quad (3)$$

i. e. $|C_p(t) - C_q(t)| \leq \widetilde{\delta}$ pour chaque p, q correct, vers le moment $t_0 + 2\delta_{max}$, où t_0 est le moment du début de travail de l'algorithme.

Preuve. Soit C_p l'horloge non corrigée et C'_p l'horloge corrigée, i. e. $C'_p(t) = C_p(t) + \Delta_p$. Fixons un moment $t > t_0 + 2\delta_{max}$, où tous les processus viennent de terminer leur travail.

Dénotons $a_{pq} =_{df} A_p(q)$. Pour p et q corrects on a $a_{pq} = d_{pq}$ (ligne 4 et Proposition sur la distance entre d_{pr}).

Dénotons $w_{pr}(t) = C_p(t) + a_{pr}$. On a pour p, q et r corrects

$$|w_{pr}(t) - w_{qr}(t)| = |C_p(t) + d_{pr} - C_q(t) - d_{qr}|$$
$$= |C_p(t) + C_r(\tau_r) - C_p(\tau_p) - C_q(t) - C_r(\tau_r) + C_q(\tau_q)|$$
$$= |(t - \tau_p) - (t - \tau_q)| = |\tau_q - \tau_p| \leq \delta^-. \quad (4)$$

Considérons le cas où p et q sont correct, mais r est incorrect. Dans ce cas chacun de a_{pr} et a_{qr} doit passer le filtre ou être égal à la valeur de l'estimateur. Si l'estimateur est impliqué alors ce cas est plus favorable que le pire des cas pour les valeurs sauvegardées directement car l'estimateur

donne une valeur entre les valeurs extrêmes. Donc il suffit de considérer le cas où les valeurs envoyées par r ont été sauvegardées.

Soit H_{rp} et H_{rq} les valeurs reçues par respectivement p et q de r. Alors $D_p(r) = H_{rp} - H_p$ et $D_q(r) = H_{rq} - H_q$. Ces deux valeurs passe le filtre, i. e. utilisant la notation

$$E_{ur} =_{df} \{s : |D_u(r) - D_u(s)| \leq \delta + \delta^-\},$$

on a $\#E_{pr} \geq (N - B) > \frac{2N}{3}$ et $\#E_{qr} > \frac{2N}{3}$. Donc $\#(E_{pr} \cap E_{qr}) > \frac{N}{3}$ et, en conséquence cette intersection contient un processus correct, soit s. Pour ce s on a

$$|D_u(r) - D_u(s)| \leq \delta + \delta^- \text{ pour } u = p, q,$$

ou en termes des a_{ur} :

$$|a_{ur} - a_{us}| \leq \delta + \delta^- \text{ pour } u = p, q.$$

Si on développe la partie gauche de la dernière inégalité on obtient

$$|a_{ur} - a_{us}| = |H_{ru} - C_u(\tau_u) - (H_s - C_u(\tau_u))| = |H_{ru} - H_s| \leq \delta + \delta^-. \quad (5)$$

Pour la différence des $w_{pr}(t)$ on a (pour r incorrect)

$$\begin{aligned}
&|w_{pr}(t) - w_{qr}(t)| = |C_p(t) + a_{pr} - (C_q(t) + a_{qr})| \\
&= |C_p(t) + H_{rp} - C_p(\tau_p) - (C_q(t) + H_{rq} - C_q(\tau_q))| \\
&= |C_p(t) - C_p(\tau_p) - (C_q(t) - C_q(\tau_q)) + (H_{rp} - H_{rq})| \\
&\leq |C_p(t) - C_p(\tau_p) - (C_q(t) - C_q(\tau_q))| + |H_{rp} - H_{rq}|.
\end{aligned}$$

On utilise le fait que les horloges sont parfaites pendant une exécution de *ClockSyn*, cela implique

$$= |t - \tau_p - (t - \tau_q)| + |H_{rp} - H_{rq}| = |\tau_q - \tau_p| + |H_{rp} - H_{rq}|.$$

On utilise (5) et on obtient pour r incorrect

$$|w_{pr}(t) - w_{qr}(t)| \leq |\tau_q - \tau_q| + |H_{rp} - H_s| + |H_s - H_{rq}| \leq \delta^- + 2(\delta + \delta^-) = 2\delta + 3\delta^-. \quad (6)$$

Maintenant, on remarque que

$$C'_p(t) = C_p(t) + \Delta_p = \frac{1}{N} \sum_r w_{pr}.$$

Pour borner

$$|C'_p(t) - C'_q(t)| = \left| \frac{1}{N} \sum_r (w_{pr} - w_{qr}) \right|$$

on découpe la somme en la partie pour les processus corrects et en la partie pour les processus incorrects et on utilise (4) et (6). Cela nous donne

$$\begin{aligned}
&\frac{N - B}{N}(\delta_{max} - \delta_{min}) + \frac{B}{N}(2\delta + 3(\delta_{max} - \delta_{min})) \\
&= (\delta_{max} - \delta_{min}) + \frac{2B}{N}(\delta + (\delta_{max} - \delta_{min})).
\end{aligned}$$

■

Remarquons que compte tenu du fait que $\delta \gg (\delta_{max} - \delta_{min})$, on peut estimer la précision $\widetilde{\delta}$ comme $\approx \frac{2}{3}\delta$. Si on peut itérer l'algorithme *ClockSync* i fois, en gardant les horloges parfaites et $\delta \gg i(\delta_{max} - \delta_{min})$, on améliore la précision jusqu'à $\approx \left(\frac{2}{3}\right)^i \delta$.