OTTO

Introduction

Nous avons réalisé une série d'expériences sur une mise en œuvre en simulation du modèle en étoile sous le nom d'OTTO (*Organizing Teams Through Overhearing*) [Legras, 2002]. Les processus de décision de plusieurs agents artificiels y sont simulés grâce à des machines à états finis qui leur permettent de créer, quitter ou vérifier des groupes. Les possibilités de communication sont elles aussi simulées avec des machines à états finis pour chaque paire d'agents. Ceci nous permet d'observer les performances relatives de l'architecture face aux variations des divers paramètres.

7.2 Protocole expérimental

7.2.1 Simulations stochastiques

Afin de simuler un temps continu au sein d'une simulation à temps discret, nous utilisons un intervalle de temps (ou *tick* d'horloge) court par rapport aux autres grandeurs temporelles des simulations (un *tick* de 0,05s comparé à des grandeurs de plusieurs secondes). Les différentes machines à états finis suivent toutes le même principe : en l'absence d'autre interférence, leur évolution est soumise à des lois de Poisson $P(t/\tau)$: à chaque état est associée une constante de temps τ qui correspond au temps moyen pendant lequel la machine reste dans cet état avant de subir une transition.

Du point de vue de la mise en œuvre, les lois de Poisson présentent une propriété très intéressante : si la valeur δ du *tick* d'horloge est petite par rapport à la constante de temps τ de la loi ($\delta \ll \tau$), on peut considérer qu'un événement a une probabilité δ/τ de se produire entre chaque *tick*. Ainsi à chaque pas de la simulation, une machine change d'état naturellement avec une probabilité constante de δ/τ.

Communications

À tout moment, on peut représenter les possibilités de communication entre agents par un graphe non-orienté dont les nœuds sont les agents et les arêtes la possibilité entre les deux agents concernés de communiquer. On appelle la *densité* d'un graphe le rapport entre son nombre d'arêtes et le nombre d'arêtes du graphe complet correspondant ($N(N-1)/2$, voir firgure 7.2). Nous définissons deux paramètres pour caractériser les conditions de communication entre les agents dans une simulation :

1. la *fiabilité* α qui représente la densité moyenne du graphe de communication au cours du temps, et donc la probabilité moyenne qu'a un message d'atteindre un agent dans le système ;
2. la *stabilité* T_{comm} qui représente le temps moyen d'un cycle « communication possible – impossibilité de communiquer » entre deux agents.

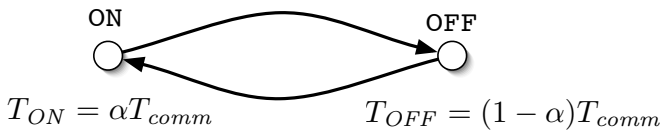

FIG. 7.1 – Machine à états finis associée à chaque paire d'agents représentant leur possibilités de communication.

Nous associons donc une machine cyclique à deux états `(ON,OFF)` à chaque arête du graphe de communication complet (voir figure 7.1). Au début de la simulation, chacune de ces machines a une probabilité α d'être dans l'état `ON` (communication possible entre ces deux agents) et une probabilité $(1-\alpha)$ d'être dans l'état `OFF` (communication impossible entre

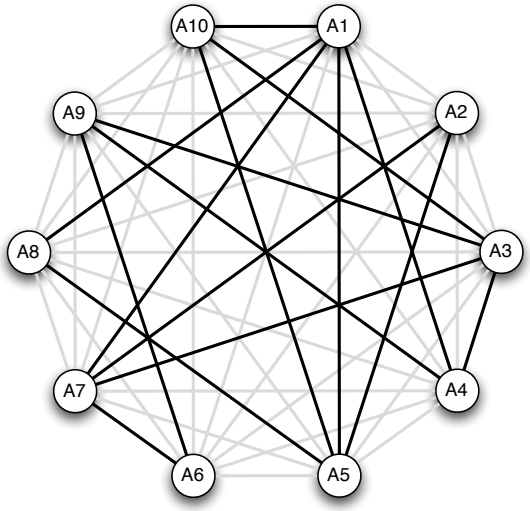

FIG. 7.2 – Exemple de conditions de communication entre dix agents, les arêtes noires correspondent aux possibilités de communication effectives, et les grises aux potentielles (le graphe complet est constitué des arêtes noires et grises), dans cet exemple la densité du graphe est de 16/45.

ces deux agents). Par la suite, ces machines suivent une évolution entièrement déterminée par des lois de Poisson de constantes $T_{ON} = \alpha T_{comm}$ et $T_{OFF} = (1 - \alpha)T_{comm}$. Il vient donc que deux agents ont à un moment donné une probabilité α de pouvoir communiquer et qu'un cycle de ces machines dure en moyenne $T_{comm} = T_{ON} + T_{OFF}$.

Agents

Une simulation concerne un ensemble de N agents homogènes dont l'état et les décisions sont représentés par des machines à états finis. Les figures 7.3, 7.4 et 7.5 représentent ces différentes machines. Leur évolution peut se faire selon trois types de transitions :

1. les flèches noires pleines correspondent aux évolutions suivant une loi de Poisson avec la constante de temps associée au nœud (T_{CHK}, T_{QUIT}, *etc.*) Lorsque ces transitions sont déclenchées, l'agent émet un message du type associé à la transition ;

2. les flèches noires en pointillés correspondent aux évolutions liées à la réception d'un message par l'agent ;
3. les flèches en gris clair correspondent aux transitions déclenchées par l'évolution d'une autre machine à états du même agent.

Trois types de machines sont utilisées :

self : chaque agent est doté d'une telle machine afin de représenter le fait qu'il croit faire partie d'un groupe (`IN`) ou non (`OUT`). S'il est membre d'un groupe, un agent décide de le quitter (ou de le dissoudre s'il en est le leader) au bout d'un temps moyen T_{QUIT}. Si l'agent ne croit pas être membre d'un groupe il crée son propre groupe (AFF) ou tente d'en rejoindre un préexistant (avec un JOIN vers un groupe présent dans sa carte de l'organisation de l'équipe) tous les T_{JOIN} (voir figure 7.3a). Ce temps est généralement assez court, en effet l'idée est d'éviter les agents solitaires. On peut noter au passage que le fait d'émettre un AFF ne fait pas que l'agent se considère membre d'un groupe, ce n'est le cas que lorsqu'il *reçoit* son propre message. D'ailleurs, la seule transition qui permet de passer de `OUT` à `IN` ne peut se faire que par la réception d'un message (figure 7.3b). La figure 7.3c illustre enfin l'évolution liée à la réception d'un message indiquant que l'agent n'appartient plus à son groupe.

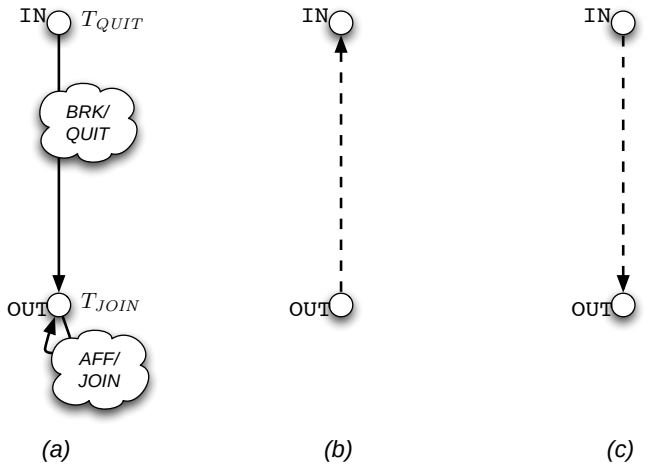

FIG. 7.3 – Trois évolutions possibles de la machine self que possède chaque agent.

doubt : chaque agent est doté d'une telle machine afin de lui donner un comportement de vérification de l'état de son groupe. La machine suit une évolution parallèle à celle de self entre OK et OUT (figure 7.4a), mais dans l'état OK l'agent est susceptible d'émettre un CHK vers son groupe et de passer alors dans l'état DOUBT. Le comportement de la machine dépend alors du statut de leader (ou non) de l'agent. Si l'agent est simple membre (figure 7.4b) il attendra un temps moyen T_{WAIT} avant de QUITter son groupe, à moins qu'il ne reçoive un AFF de son leader l'incluant dans le groupe, le faisant repasser du même coup sur OK (flèche pointillée). Si, au contraire, l'agent est leader de son groupe, il attendra pendant un temps moyen T_{WAIT} que les membres de son groupe se manifestent avant d'exprimer la nouvelle composition du groupe (flèche pleine, figure 7.4c).

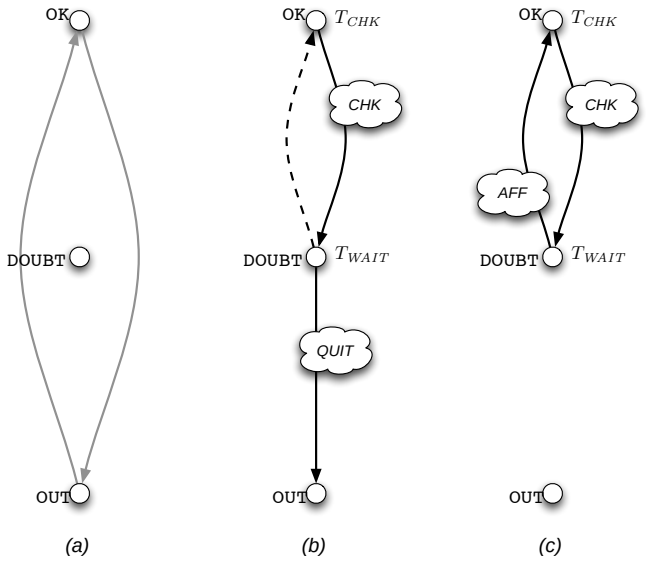

FIG. 7.4 – Évolutions de la machine doubt que possède chaque agent.

member : ces machines sont utilisées par les leaders, ils en utilisent une par agent dans l'équipe (IN pour les membres, OUT pour les autres). Lorsque la machine doubt d'un leader passe sur DOUBT, les machines associées aux membres passent de IN à DOUBT (flèche grise, figure 7.5a). Le fait de recevoir un BEL d'un membre fait repasser la machine correspondante sur IN (voir figure 7.5b). Ainsi, lorsque le leader émet le AFF après avoir patienté pendant T_{WAIT}, il n'inclut dans le groupe que les membres dont la machine

est sur `IN`, ce qui fait passer les autres sur `OUT` (figure 7.5b).

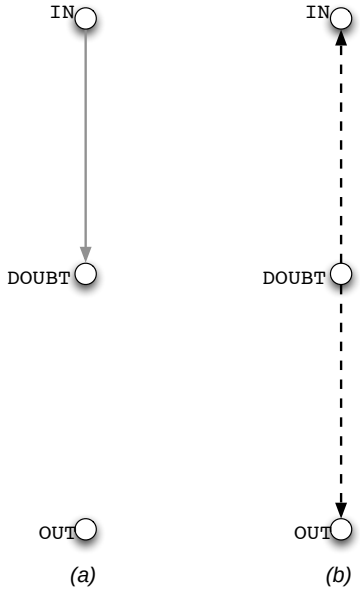

FIG. 7.5 – Évolutions des machines member que les leader associent aux membres de leur groupe.

7.2.2 Paramètres

Nous avons déterminé des valeurs par défaut pour les paramètres de la simulation et les faisons varier un à un. L'effet d'un paramètre est étudié en le faisant varier pendant que les autres gardent leurs valeurs par défaut. Le tableau 7.1 résume ces paramètres avec leurs valeurs par défaut et leurs plages de variation.

7.2.3 Mesure de performance

L'objectif principal du modèle est de maintenir des groupes cohérents au cours des opérations de l'équipe. Nous définissons donc plusieurs notions afin d'évaluer l'état de l'équipe au cours de son évolution :

DÉF. 7.1 – Cohérence faible. *Une équipe est* faiblement cohérente *si et seulement si :*

1. *chaque agent qui croit faire partie d'un groupe en est membre, et*
2. *chaque agent qu'un leader considère comme faisant partie de son groupe en est membre, et*
3. *il existe au moins un groupe qui n'est pas réduit à son leader (il serait inexact de considérer comme cohérente une équipe dans laquelle aucune coopération n'a lieu).*

DÉF. 7.2 – Incohérence. *Une équipe est* incohérente *en l'absence de cohérence faible.*

DÉF. 7.3 – Cohérence forte. *Une équipe est* fortement cohérente *si elle est faiblement cohérente et si chaque agent membre d'un groupe peut communiquer avec son leader.*

DÉF. 7.4 – Cohésion. *La* cohésion *est la proportion d'agents qui sont membres d'un groupe sans en être le leader. Cette notion est définie afin de déterminer si une équipe est fragmentée ou non.*

Pour évaluer une session de simulation, nous comparons le temps durant lequel une équipe est fortement ou faiblement cohérente au temps total de la simulation. Ces ratios sont utilisés pour représenter la cohérence globale de l'équipe au cours d'une simulation. De la même manière, la cohésion a été moyennée sur la durée de chaque simulation, ainsi que le nombre de messages émis par l'équipe.

Paramètre	Nom	Valeur par défaut	Plage
Fiabilité des communications	α	0,5	0,05 – 0,95
Stabilité des communications	T_{comm}	120s	1 – 1000s
Nombre d'agents	N	6	2 – 16
Période entre les tentatives de JOIN ou avant d'émettre un AFF	T_{JOIN}	5s	1 – 1000s
Temps passé dans un groupe avant d'émettre un QUIT / BRK	T_{QUIT}	360s	1 – 1000s
Périodicité des CHKs	T_{CHK}	30s	1 – 1000s
Rectification	*Rectify*	Oui	Oui – Non
Délai d'attente après un CHK	T_{WAIT}	2s	
Durée d'une simulation		7200s	

TAB. 7.1 – Paramètres des simulations dans OTTO.

Avec les paramètres par défaut du tableau 7.1, nous avons obtenu :

- une cohérence faible de 0,8 ;
- une cohérence forte de 0,35 ;
- une cohésion de 0,47 ;
- 0,87 message par seconde.

C'est-à-dire, avec des agents ne pouvant communiquer entre eux que la moitié du temps et ces mêmes conditions de communication changeant toutes les minutes en moyenne, chaque agent émettant un message toutes les 7 secondes (*i.e.* 6 agents à 0,87 messages par seconde), environ la moitié des agents sont membres d'un groupe à tout moment, il n'y a pas d'incohérence 80% du temps et les membres peuvent communiquer avec leur leader 35% du temps.

7.3 Résultats expérimentaux et discussion

Nous présentons ici les résultats les plus significatifs obtenus. Sur les graphiques représentant la cohérence, la cohérence faible est représentée en gris clair et la forte (par définition inférieure) en gris foncé. Toutes les valeurs de temps sont en secondes et le nombre de messages est exprimé en nombre de messages émis par l'équipe par seconde.

7.3.1 Conditions de communication

La figure 7.6 nous montre les effets des conditions de communication sur le comportement de l'architecture. Deux paramètres caractérisent ces conditions : α et T_{comm}. Nous pouvons noter (voir la courbe $coherence - T_{comm}$) qu'alors que la cohérence faible n'est pas affectée par T_{comm} (qui représente la stabilité des conditions de communication), la cohérence forte croît rapidement avec l'augmentation de la stabilité des communications. Cette stabilité est cruciale à la propriété de forte cohérence. En effet la cohérence forte nécessite que les agents soient capables de communiquer, donc si les conditions de communication sont stables, les agents restent capables de communiquer avec leur leader après avoir rejoint un groupe.

Naturellement (voir les courbes $coherence - \alpha$ et $cohesion - \alpha$), la cohésion et la cohérence augmentent avec α, mais alors que l'on obtient une bonne cohérence faible avec une fiabilité médiocre, une très bonne fiabilité est nécessaire pour obtenir de bons scores pour la cohérence forte.

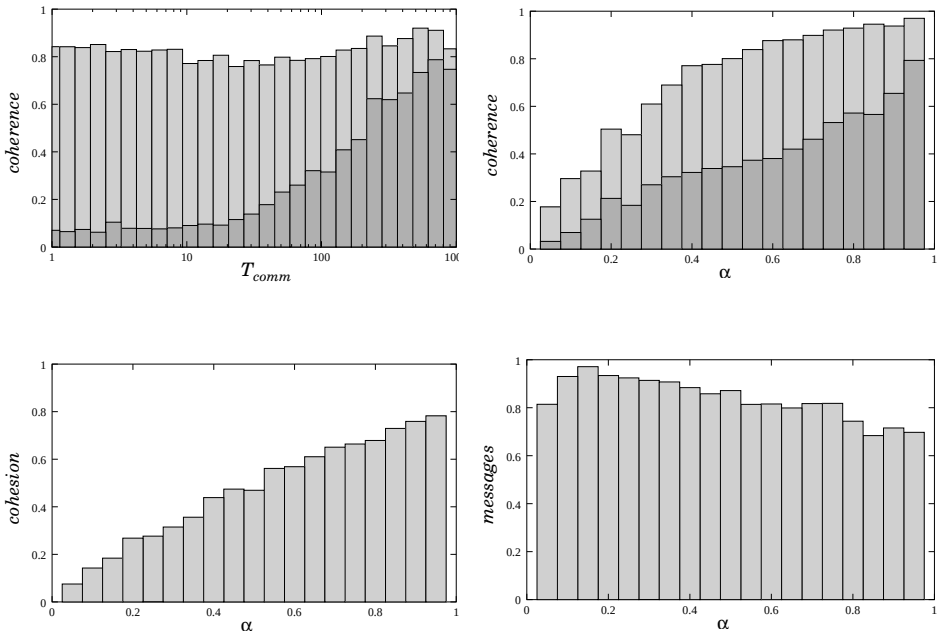

FIG. 7.6 – Effets des conditions de communication.

En ce qui concerne le nombre de messages émis, (voir la courbe $messages - \alpha$), nous pouvons noter un pic pour un α médiocre ($\alpha \simeq 0{,}15$) car les agents ont alors juste assez de possibilités de communication pour tenter de coopérer, mais le font en vain, ce qui provoque un surcroît de messages. Puis, alors que la fiabilité augmente le nombre de messages diminue légèrement.

7.3.2 Nombre d'agents

La figure 7.7, présente l'évolution des performances de l'architecture avec le nombre d'agents N. Tout d'abord (voir la courbe $cohesion - N$) la cohésion atteint rapidement une limite d'environ 0,6. Cela semble indiquer que le nombre total d'agents n'a que peu d'influence sur la taille moyenne des groupes.

En parallèle (voir la courbe $messages - N$) la quantité de messages émis croît linéairement avec l'effectif de l'équipe, ce qui veut dire que le nombre de messages émis par agent et par unité de temps reste constant. Pourtant chaque agent a un nombre croissant d'interlocuteurs potentiels, ce qui aurait pu conduire à une explosion du trafic avec N.

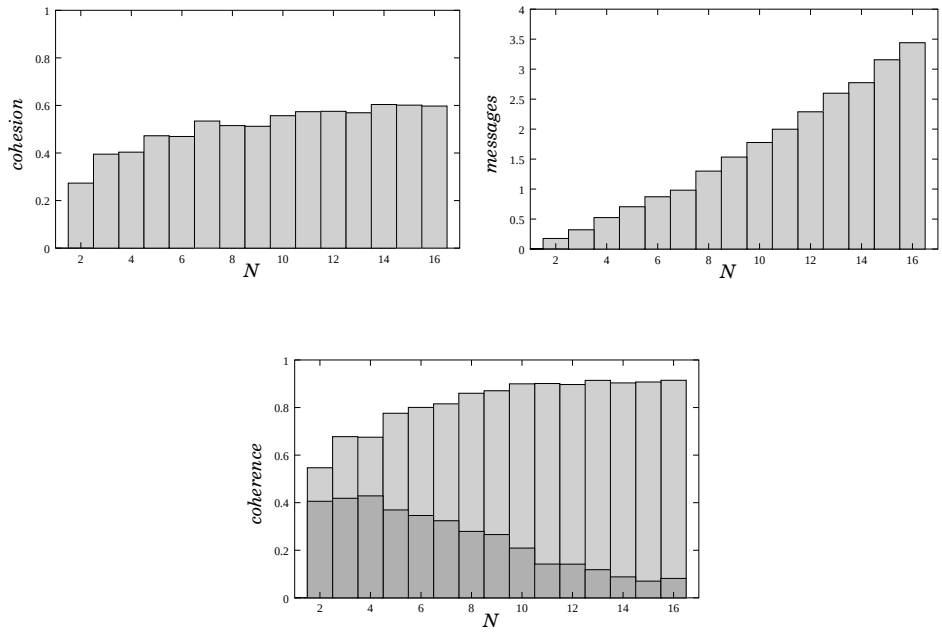

FIG. 7.7 – Effets du nombre d'agents.

En ce qui concerne la cohérence (voir la courbe $coherence - N$), les choses ne sont pas si claires. D'une part, le fait que la cohérence faible augmente avec l'effectif de l'équipe peut être dû à la raréfaction des situations dans lesquelles aucun « membre » n'existe dans l'équipe, et ces situations de groupes réduits à leurs leaders sont incohérente. D'autre part, la décroissance de la cohérence forte quant à elle, semble due à la raréfaction des situations où tous les membres sans exception peuvent communiquer avec leur leader. En effet, il suffit qu'un seul membre d'un seul groupe ne puisse pas communiquer avec son leader pour que nous considérions qu'il n'y a pas cohérence forte (voir définition 7.3). Ce critère ne semble donc pas très adapté à des effectifs importants.

7.3.3 Rectification

Les agents ont la possibilité de se corriger les uns les autres s'ils sont paramétrés pour le faire (*Rectify*). Si c'est le cas, cette rectification est automatique : si, après avoir reçu un message et mis à jour ses croyances, un agent A_i est toujours en désaccord avec ce message à propos d'un agent A_j, A_i émet alors un BEL/AFF décrivant le véritable groupe de A_j.

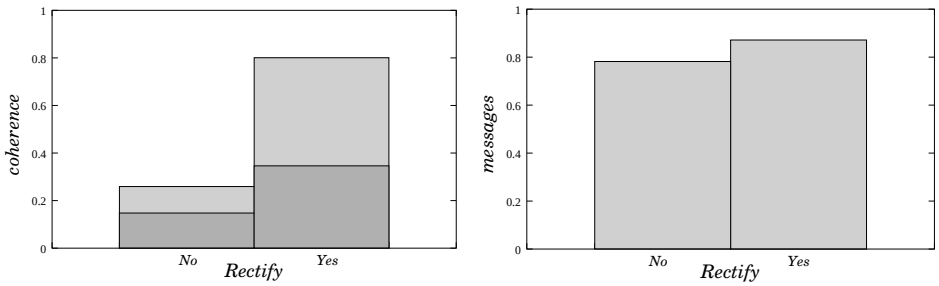

FIG. 7.8 – Effets du paramètre $Rectify$.

La figure 7.8 présente les effets du paramètre $Rectify$ sur la cohérence et le nombre de messages. $Rectify$ a pour effet de tripler la cohérence faible et plus que doubler la cohérence forte, avec une augmentation du nombre de messages qui correspond à seulement un message par agent toutes les 54 secondes. Nous pouvons donc affirmer que la rectification est une part fondamentale de l'architecture qui a un effet très bénéfique sur la cohérence de l'équipe pour très un faible coût du point de vue du trafic.

7.3.4 Fréquence des vérifications

Sur la figure 7.9, nous pouvons observer les effets des variations de T_{CHK} sur la cohérence de l'équipe (voir la courbe $T_{CHK} - coherence$) et sur le trafic (courbe $T_{CHK} - messages$). Tout d'abord, nous pouvons noter que la fréquence avec laquelle les CHK sont émis n'a pas d'influence notable sur la cohérence faible, mais en a une très importante sur la cohérence forte. Cela se comprend bien car pour la cohérence faible, les agents peuvent rejoindre un groupe et ne plus s'en soucier tout en gardant des croyances cohérentes jusqu'à la dissolution du groupe ou leur départ au bout de T_{QUIT}. À l'inverse, le mécanisme de vérification permet aux agents de vérifier leurs possibilités de communication et de quitter un groupe s'ils ne peuvent communiquer avec leur leader. Il permet aussi à un leader d'exclure un membre qui ne répond pas. De faibles valeurs de T_{CHK} (2–3 secondes) donnent de très bons résultats (cohérence forte de 0,8 comparée aux 0,35 de référence), tandis que des valeurs plus importantes donnent de mauvais résultats ($\simeq$ 0,15). Toutefois, ces faibles valeurs de T_{CHK} ont un effet considérable sur le trafic, avec des valeurs six fois supérieures à la valeur de référence.

Il semble donc qu'il faille faire un compromis entre obtenir une bonne cohérence forte et conserver un trafic raisonnable en nombre de messages.

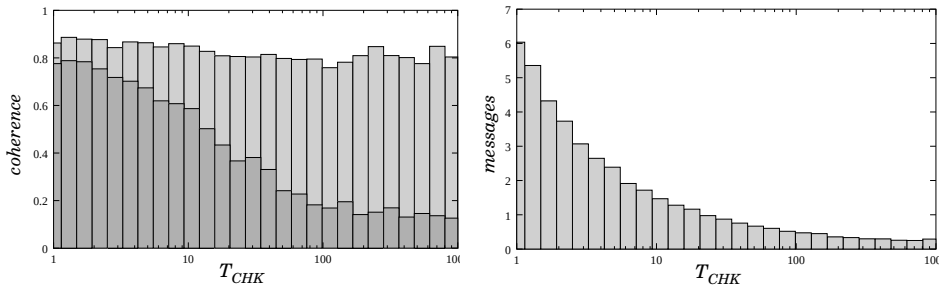

FIG. 7.9 – Effets du paramètre T_{CHK}.

Mais notons que les mécanismes collaboratifs nécessitant la cohérence forte au sein de l'équipe sont tels que les membres ont à communiquer souvent avec leurs leaders. Or, les messages de type BEL/AFF sont de petite taille, et en les couplant aux messages liés à la tâche de l'équipe, on peut obtenir une sorte de *cercle vertueux* : plus l'équipe communique plus sa cohérence forte augmente, alors justement que l'équipe a besoin de cohérence forte pour pouvoir communiquer. Pour profiter des messages échangés entre le leader et ses membres, l'idée est d'utiliser le délai T_{WAIT} afin d'attendre une occasion de communiquer liée à la tâche avant de répondre aux CHK : cela revient à « garder en réserve » sa réponse en attendant de pouvoir la coupler à un autre message lié quant à lui à la tâche de l'équipe et non pas à l'organisation en groupes. Si une telle occasion se présente il suffit alors d'y associer la réponse au CHK en attente.

Mais plus simplement (bien que cela dépende du mode de coopération qu'utilisent les agents), le simple fait de communiquer à propos de la tâche avec son leader (ou inversement avec un membre de son groupe) peut être considéré comme une confirmation implicite de sa volonté de toujours faire partie du groupe, ce qui réduit d'autant le besoin d'émettre des CHK.

7.4 Conclusion

Les simulations OTTO ont mis à l'épreuve les principes du modèle en étoile et le mécanisme de suivi associé. Tout d'abord, ces simulations nous ont permis de confirmer expérimentalement les intuitions qui nous ont conduit à concevoir le modèle en étoile, et notamment que l'utilisation de l'écoute flottante et le suivi des autres membres de l'équipe permettent à l'équipe de se réorganiser dynamiquement en fonction des possibilités

de communication. D'autre part, ces simulations ont permis de dissiper certaines craintes envers le comportement du système, nous aurions pu notamment nous attendre à une explosion du nombre de messages avec l'augmentation de l'effectif (en effet chaque agent reçoit en moyenne les messages de $\alpha(N-1)$ autres agents et est susceptible de réagir à ces messages).

Le résultat le plus intéressant que nous avons tiré d'OTTO est sans doute l'utilisation de la rectification. En effet, nous avons montré que l'utilisation de l'écoute flottante pour détecter des informations incorrectes « circulant » dans l'équipe, combinée à la simple émission à la cantonade de l'information corrigée permet d'accroître considérablement la cohérence du système pour un coût minime en nombre de messages émis.