Un protocole de gestion de groupe et un protocole de diffusion atomique sous contraintes de temps-réel.

David Decotigny

9 & 14 septembre 1998

Résumé

Nous présentons deux protocoles coopérant pour assurer la diffusion atomique de messages pour les systèmes distribués synchrones soumis à des contraintes de temps-réel. Le premier protocole, le protocole de gestion de groupe, assure la mise à jour de la liste des membres perçus corrects du système par tous les nœuds du système. Ce protocole garantit l'accord entre toutes ces *visions du groupe*, ainsi que la détection des défaillances par arrêt et les redémarrages de nœuds en un temps borné connu. Le deuxième protocole, le protocole de diffusion atomique, assure la remise de messages diffusés à tous les nœuds détectés corrects par la gestion de groupe, et destinataires du message. Ce protocole garantit que la remise des messages s'effectue en un temps borné connu. La coopération des deux protocoles permet d'assurer la propriété de *synchronisation virtuelle*, qui signifie que les messages issus de ces deux protocoles sont remis à l'application de façon cohérente.

IRISA IFSIC ENST Bretagne

Cadre du stage

L'Irisa

L'INRIA, le CNRS, l'université de RENNES 1 et l'INSA de RENNES sont partenaires au sein d'une structure de recherche appelée Irisa. Les effectifs de l'Irisa s'élèvent à plus de 340 personnes. L'action de l'Irisa se déroule dans un contexte d'évolution technologique rapide et de très vive compétition scientifique et industrielle. Il est donc essentiel que la recherche conduite soit de la plus haute qualité et que les transferts des résultats soient tout à fait efficaces.

Les activités de l'Irisa vont du développement de composants matériels à la mise en œuvre d'applications avancées. La conception de circuits et d'architectures nouvelles mettant en oeuvre un parallélisme important est maintenant un axe de recherche privilégié. La construction de systèmes et d'applications distribués permettant de rendre transparente l'utilisation des multiples ressources composant l'architecture donne lieu à des travaux tout à fait originaux.

Les domaines applicatifs abordés par l'Irisa sont nombreux. Citons, en particulier les télécommunications et le multimédia, la santé, l'environnement et les transports. L'Irisa entretient des liens contractuels avec de nombreux industriels français ou européens.

L'Irisa est également très présent dans la formation, autant par ses liens très étroits avec l'Ifsic, l'Insa et l'Enssat, que par l'effort important consenti pour l'accueil de doctorants.

Le projet Solidor

Ce projet INRIA de l'Irisa a pour thématique la construction de systèmes et d'applications distribués. De façon générale, un système d'exploitation distribué (plus simplement *système distribué*) peut être défini comme un logiciel complexe permettant de coordonner l'exécution d'applications au-dessus d'une architecture distribuée donnée.

Les évolutions conjointes des architectures alliées à l'apparition de nouvelles applications orientées "grand public" génèrent sans cesse de nouveaux problèmes relatifs au temps de réponse, à la sécurité ou encore à la disponibilité, pour lesquels des solutions doivent être apportées au niveau du système d'exploitation.

Les travaux de recherche du projet Solidor s'articulent autour de la construction de systèmes d'exploitation distribués apportant une réponse à ces problèmes. Le projet Solidor est composé de différent sous-thèmes. Ce stage s'est effectué dans le cadre de la définition de la plate-forme d'exécution HADES.

HADES [1]

Les travaux sur la définition d'une plate-forme d'exécution pour applications à contraintes temps-réel dur visent à fournir une plate-forme exploitable pour une large gamme d'applications distribuées temps-réel à sûreté critique. Cette plate-forme d'exécution, appelée HADES (*Highly Available Distributed Embedded System*), est définie à partir de composants logiciels et matériels standard.

Remerciements

Je tiens à remercier Michel Banâtre pour m'avoir accueilli dans son projet, et pour la confiance qu'il m'a accordée.

Je remercie également très chaleureusement Isabelle Puaut, et Emmanuelle Anceaume, chercheurs dans le thème de recherche HADES, pour m'avoir conseillé tout au long de mon stage, que ce soit par leurs très précieuses suggestions, par leur expérience, ou par leurs critiques. Je les remercie également pour avoir eu la gentillesse de suivre attentivement l'évolution de mes travaux.

Je tiens aussi à remercier André Thépaut, professeur à l'ENST de Bretagne (Brest), pour avoir été mon interlocuteur brestois lors de ce stage, et pour m'avoir secouru dans ma période de grand embarras administratif.

Introduction

La coopération de plusieurs machines pour le traitement de l'information est un enjeu majeur lorsqu'il s'agit de garantir une fiabilité maximale (par réplication), ou d'obtenir une puissance accrue (par répartition des tâches).

Pour permettre l'interaction entre les unités de traitement, de tels systèmes, alors qualifiés de *distribués*, reposent sur un mécanisme de diffusion de l'information. Cette diffusion doit garantir la cohérence de l'information distribuée entre toutes les unités de traitement.

Dans le cas des systèmes dont la fiabilité est la raison d'être de la distribution du traitement entre plusieurs entités, l'ensemble du comportement doit être caractérisé et garanti. Dans certains cas, le comportement temporel fait partie des éléments à caractériser.

Cette catégorie de systèmes distribués, appelés systèmes distribués *temps-réel dur* [1] [2], pour lesquels la correction des tâches effectuées est conditionnée par leur situation temporelle d'exécution, représente l'arrière-plan de notre étude. Par la suite, nous qualifierons ces systèmes soumis à des contraintes de temps-réel dur de systèmes *temps-réel* simplement.

Dans ce document, nous présentons une série de deux protocoles permettant la diffusion de l'information entre tous les nœuds de traitement du système distribué. Ces protocoles garantissent que la diffusion maintient la cohérence du système même en présence de défaillances tant du réseau, que des nœuds, tout en ayant un comportement temporel borné connu.

Pour que cela soit rendu possible, il est nécessaire de se placer dans le cadre des systèmes *synchrones* (le cadre des systèmes asynchrones rendant impossible toute garantie de cohérence, et tout respect de contraintes temporelles [3]). Pour ces systèmes, le comportement temporel est borné et connu en l'absence de fautes, tant au niveau des délais d'acheminement des messages, qu'au niveau des vitesses relatives des processeurs.

Le premier protocole présenté, le protocole de *gestion de groupe* (décrit au paragraphe 5), vise à rendre disponible, à chaque processeur opérationnel du système, la liste des autres processeurs opérationnels du système. Cette liste sera appelée par la suite la *composition du groupe*, ou la *vue du groupe*. Puisque la composition du groupe est une information qui est maintenue localement par tous les processeurs opérationnels, il est nécessaire qu'elle reste cohérente entre tous les processeurs de la vue du groupe, comme toute information distribuée.

Ainsi, le protocole garantit que les décisions concernant la composition du groupe sont prises en *accord*. Plus précisément, les décisions sont identiques, et prises *simultanément* [2] par tous les processeurs de la vue du groupe, malgré les défaillances.

Afin d'intégrer ce protocole dans des systèmes temps-réel, ce protocole de gestion de groupe assure que les défaillances ou les réintroductions de nœuds dans la vue du groupe sont détectées en un temps borné connu.

1. Les systèmes *temps-réel souple* sont ceux pour lesquels un écart du comportement temporel par rapport aux spécifications ne résulte qu'en une diminution de la qualité de service. Les systèmes multimédia en sont des représentants.

2. Moyennant la précision de synchronisation entre les horloges des nœuds du système.

Le deuxième protocole que nous décrivons est un protocole de diffusion atomique de l'information (défini au paragraphe 7). Il repose sur le précédent, puisqu'il lui est nécessaire de disposer de la liste des nœuds vers lesquels assurer la diffusion atomique.

Ce protocole garantit que les messages diffusés sont reçus par tous les nœuds de la vue du groupe au bout d'un délai de diffusion borné connu, et ceci malgré les fautes. Seuls les nœuds faisant partie de la liste des destinataires du message remettent ce message à l'application.

Le protocole garantit que les messages sont tous délivrés dans le même ordre total pour tous leurs destinataires : l'ordre d'émission, sachant qu'à un instant donné, un nœud unique à le droit d'émettre (exclusivité d'émission sur le médium de diffusion). Ceci permet l'*atomicité* de la diffusion.

La coopération de ces deux protocoles possède la propriété de *synchronisation virtuelle* (*Virtual Synchrony* [4]). Cette synchronisation signifie que, entre deux changements de composition du groupe consécutifs, l'ensemble des messages diffusés est identique pour tous les nœuds du groupe qui sont restés opérationnels.

Dans la suite de ce document, nous présentons au paragraphe 1 le contexte lié au paradigme de système distribué, et les propriétés associées. Au paragraphe 2, nous dressons un bref état de l'art en matière de protocoles de gestion de groupe et de diffusion atomique. Au paragraphe 3, nous définissons la terminologie adoptée tout au long du document. Nous établissons précisément, au paragraphe 4, le modèle adopté pour définir le système (en particulier, les modes de défaillance). Le protocole de gestion de groupe est détaillé au paragraphe 5. Le protocole de diffusion atomique l'est au paragraphe 7. Leurs propriétés sont énoncées et prouvées respectivement aux paragraphes 6 et 8.

En annexe B, une variante de ces deux protocoles est présentée, qui est liée à des hypothèses de défaillance plus fortes constituant le cadre de fonctionnement de HADES.

1 Cadre d'étude et définitions

Un environnement distribué de traitement de l'information nécessite au moins une phase de synthèse des résultats du traitement, qui sont en effet *distribués* puisque le traitement associé l'est, voire même une phase de dissémination de l'information. La diffusion de l'information *de* ou *vers* les acteurs du traitement est donc un facteur clef [5] du paradigme de *système distribué*.

Les objectifs d'un système distribué sont de deux ordres :

- Diminuer les temps de traitement de calculs parallélisables, en utilisant une architecture parallèle répartie;
- Accroître la fiabilité d'un système de traitement, grâce à la réplication des unités de traitement.

Dans les deux cas, il est nécessaire de bâtir le système sur un mécanisme de diffusion pouvant masquer les fautes. Ceci garantit, pour la première classe de systèmes, la complétude du résultat (ou de l'information), ou, dans la seconde classe, ceci assure la cohérence de l'information traitée entre les réplicas.

Nous nous plaçons, dans ce document, dans le cadre des systèmes distribués temps-réel. La contrainte de temps réel signifie que la correction du système dépend non seulement de la correction des résultats de l'exécution, mais également du respect de

contraintes temporelles strictes par cette exécution. Ces systèmes se situent le plus souvent dans la classe des systèmes utilisant la réplication, plutôt que dans la classe des systèmes de calculs parallèles répartis, pour lesquels les contraintes de temps réel ne sont pas la priorité fondamentale.

1.1 Consensus et diffusion atomique

Afin d'assurer la cohérence globale du système, tout le problème réside dans la résolution du *consensus* [6, 3].

En considérant un ensemble de processus (déroulement temporel de l'exécution d'une tâche) qui disposent chacun d'une valeur *initiale*, le problème du consensus revient à mettre en place un mécanisme qui permette à tous les processus de s'accorder sur une même valeur résultat. Ainsi, résoudre le problème du consensus, c'est établir les propriétés de :

1. *Terminaison* : Tout processus *correct* (*i.e.* non défaillant) finit par décider;
2. *Accord* : Deux processus ne peuvent décider différemment;
3. *Intégrité* : Un processus décide au plus une fois;
4. *Validité* : La valeur décidée est l'une des valeurs initiales proposées.

Il a été prouvé [7] qu'il y a équivalence entre consensus et diffusion atomique. Le principe de la diffusion atomique est que chaque message finit par être diffusé à tous les processus corrects, en assurant l'*atomicité*. L'atomicité correspond à la *diffusion fiable* (si un message est délivré par un processus, tous les autres processus le délivreront aussi) *ordonnée* (deux processus ne peuvent pas délivrer deux mêmes messages dans des ordres différents). Ainsi, si un processus p délivre deux messages m puis n à l'application, alors nécessairement un autre processus q recevra m et n, et délivrera m puis n à l'application.

Tous les protocoles de diffusion étudiés dans ce document tentent d'atteindre cet objectif en ordonnant globalement tous les messages du système.

Par définition, les propriétés que les protocoles de diffusion atomique vérifient sont :

1. La *terminaison* : Si un message est diffusé par un processus correct, alors celui-ci finit par le délivrer;
2. L'*intégrité* : Un processus remet à l'application au plus une fois un message qui lui a été diffusé;
3. L'*accord* : Si un processus correct délivre un message, alors tous les processus corrects finissent par le délivrer;
4. L'*ordre total* : Si deux processus délivrent les deux mêmes messages, alors ils les délivrent dans le même ordre.

Les deux derniers points définissent ce que nous nommons *cohérence* dans toute la suite.

1.2 Résolution pratique

Résoudre ces deux problèmes (consensus et diffusion atomique) permet de garantir l'accord sur les messages diffusés et reçus entre tous les *nœuds en fonctionnement* du système. Pour que l'accord au sein d'un tel corpus de processeurs (les *nœuds en*

TAB. 1 – *Exemples d'hypothèses de défaillance*

Appellation de l'hypothèse	Défaillances par valeur	Défaillances temporelles		Persistance
	Perception	Type	Perception	
Défaillance par omission	-	retard ∞	cohérente	temporaire
Défaillance par arrêt	-			permanente
Défaillance temporelle	-	quelconque	quelconque	quelconque
Défaillance de réponse	quelconque	-		
Défaillance cohérente	cohérente	quelconque	quelconque	quelconque
Défaillance byzantine	incohérente			

fonctionnement) ait un sens, il est nécessaire que ce corpus soit identifié par tous ses membres. C'est à dire que :

- tous les nœuds du système doivent disposer (grâce à un mécanisme de détection des défaillances [7]) de la liste des *nœuds en fonctionnement* du système;
- il est garanti que cette liste est à chaque instant (du temps réel ou du temps mesuré local à chaque nœud) la même pour tous les nœuds composant cette liste.

Un protocole de *gestion de groupe* doit atteindre ces deux objectifs.

Connaissant le corpus des processeurs opérationnels de système, il incombe ensuite au protocole de *diffusion atomique* de maintenir la cohérence de l'information distribuée entre tous les nœuds opérationnels.

1.3 Modes de défaillance et protocoles concernés

L'enjeu de la résolution du consensus est de passer outre les considérations de défaillances. Il est évidemment impossible de faire abstraction de *toutes* les défaillances, et il est donc nécessaire de définir, pour tous les protocoles, un domaine de validité du protocole, en termes de tolérance aux *fautes* : les hypothèses de défaillance.

1.3.1 Définition et classification des défaillances

Une *faute* [2, 5] est une imperfection ou un phénomène du système qui peut conduire à une *erreur*, c'est à dire qu'une partie du système est dans un état incorrect, invalide ou incohérent. Cette erreur peut elle-même être la source de *défaillance(s)*, *i.e.* fonctionnement non conforme aux spécifications de l'unité de traitement (matériel ou logiciel).

Les défaillances, quand elles ne sont pas d'origine humaine (erreur de conception) peuvent être *permanentes* (un composant qui a trop chauffé, ...) ou *temporaires* (champ magnétique environnant anormalement élevé, ...).

Le problème que posent les défaillances est de les détecter. Ceci peut s'avérer impossible si elles sont *cohérentes*, c'est à dire si, dans le cas de défaillances par valeur, toutes les valeurs sont identiques (accord), mais fausses, ou, dans le cas de défaillances temporelles (le résultat n'est pas fourni pendant l'intervalle de temps spécifié), si les résultats sont fournis dans le même intervalle incorrect (par avance, ou par retard).

Le tableau 1 (issu de [2]) suivant détaille certaines hypothèses de défaillances courantes selon deux axes : les défaillances par valeur, et les défaillances temporelles.

1.3.2 Protocoles associés à la gestion des défaillances

Dans le cas des systèmes distribués, les défaillances permanentes entraînent une modification de la population des nœuds de traitement impliquée dans le protocole de diffusion atomique : le *changement de configuration*. Ce type de défaillances recouvre les défaillances par arrêt des nœuds. Il est essentiel de les détecter et de les signaler au protocole de diffusion. En effet, sans cela, l'état même du groupe est erroné, et le système est donc dans un état incohérent. Ce travail de détection et de signalisation des défaillances permanentes doit être assuré par un protocole de gestion de groupe.

La classe des défaillances temporaires (beaucoup plus fréquentes : d'un facteur 10 à 100 [8]) correspondent à des omissions en émission ou en réception de la part des nœuds du système, ou à des fautes de transmission de la part du réseau. Le travail de masquage des omissions temporaires est en principe assuré par un protocole de diffusion fiable. Tous les protocoles de diffusion présentés au paragraphe 2, sauf RTCAST et Fast CBCast s'attachent à masquer cette classe de défaillances. Pour tous ces protocoles, les omissions sont attribuées à des défaillances du canal de transmission, contrairement au protocole proposé dans ce document qui considère que ce canal est exempt de fautes, et qui ramène la cause de ces omissions à des défaillances des interfaces réseaux des nœuds du système. RTCAST et Fast CBCast considèrent, quant à eux, que ces fautes temporaires équivalent à une défaillance permanente, entraînant le nœud qui en est victime à simuler une défaillance par arrêt.

2 État de l'art

Étant donné que gestion de groupe et diffusion atomique sont très liés, les deux sont souvent traités au sein d'un même projet de recherche. Ces types de protocoles font l'objet de beaucoup de travaux, et nous ne les énumérerons pas tous ici. Ceux auxquels nous faisons référence dans ce bref état de l'art sont : Chang & Maxemchuk [9, 5], PSync [10, 5], Fast Causal Multicast [11, 5], Amœba [12], Transis [13], RTCAST [14], TOTEM [15] et le protocole de Matthew Clegg [16] (fortement inspiré du protocole de Cristian [17, 5]). L'annexe A présente un tableau récapitulatif des différentes caractéristiques des protocoles.

Généralement, gestion de groupe et diffusion atomique sont distincts, et le protocole de gestion de groupe est un prérequis pour la diffusion fiable. Il arrive cependant que les deux soient si enchevêtrés et interdépendants qu'ils ne sont pas distingués. C'est le cas pour RTCAST par exemple. Dans la série de protocoles proposée dans ce document, nous avons choisi de clairement distinguer les deux, et le protocole de gestion de groupe est de niveau inférieur à celui de diffusion fiable : le dernier requiert les informations du premier.

2.1 Gestion de groupe

La gestion de groupe consiste à établir la liste des nœuds opérationnels du système. Tous les protocoles respectent le même schéma suivant. Il s'agit pour tous les nœuds de signaler leur non *défaillance par arrêt*, que ce soit grâce à des messages dédiés à la gestion de groupe, ou grâce à des messages destinés à la diffusion d'informations applicatives (qui intègrent alors le fait que leur émetteur est encore en *vie*).

Les protocoles se distinguent ensuite par leur mode de gestion de la liste des membres opérationnels du groupe (la *vue du groupe*), et par la sémantique associée à la *correction* d'un nœud en présence de fautes. Dans tous les cas, le souci majeur est de garantir la cohérence de cette vue du groupe parmi les nœuds du système malgré les fautes.

À partir des travaux étudiés, on peut distinguer deux catégories de protocoles de gestion de groupe.

Les protocoles qui délèguent la gestion de groupe à un nœud unique. De cette manière, le problème de la cohérence des vues du groupe entre les nœuds du système ne se pose pas. En effet, cette vue du groupe est maintenue par une seule entité qui la diffuse à tous les nœuds de la vue. Deux cas sont possibles :

- L'entité qui maintient la vue du groupe s'occupe de la gestion du groupe tant qu'aucun changement de groupe (*i.e.* départ ou insertion d'un nœud dans la vue du groupe) n'intervient. C'est le principe de fonctionnement qu'a choisi **Amœba** (Vrije Universiteit Amsterdam, 1988-...), pour lequel une entité unique (le *séquenceur*) s'occupe d'établir la liste des membres du groupe, et également d'assurer la diffusion atomique des messages (c'est lui qui estampille les messages, et qui est chargé des retransmissions en cas d'omissions). Dans ce protocole, conçu pour les systèmes asynchrones, les nœuds du système manifestent leur présence par des signes de vie réguliers (intégrés aux messages diffusés s'ils existent) au séquenceur.

  Avec cette approche, le consensus sur la vue du groupe, mais aussi sur les messages diffusés, ne peut pas être garanti quand le séquenceur est victime d'une *défaillance par arrêt*. Si ce dernier est *défaillant par arrêt*, il est alors nécessaire d'en élire un nouveau. La phase de transition n'est pas détaillée, mais ne se plie pas à des contraintes temporelles strictes. Il est en effet coûteux mais néanmoins nécessaire d'établir le consensus sur la composition du nouveau groupe, ainsi que sur les messages reçus de la phase précédente, avant d'entamer la nouvelle phase.
- L'entité qui maintient la vue du groupe s'occupe de la gestion du groupe pendant une fraction du temps. C'est le principe mis en œuvre dans le protocole de **Chang & Maxemchuk**. Dans ce protocole, un jeton qui circule à un rythme fixé entre tous les nœuds, définit le nœud responsable de la cohérence du système. C'est à lui de gérer les retransmissions de messages (le médium de diffusion peut-être la source d'omissions), et c'est à lui de maintenir la vue du groupe.

  Un problème de ce protocole est la forte inefficacité en termes de rentabilité de la bande passante utilisée (chaque nœud doit accuser réception de tous les messages diffusés). L'autre problème est que la gestion de la perte du jeton nécessite un algorithme de négociation du nouveau jeton (par le choix d'un nouveau premier nœud s'occupant de la gestion du groupe). De plus, dans cet état, aucune garantie n'est faite quant à la cohérence du système.

Les protocoles qui distribuent la gestion du groupe entre tous les nœuds. Chaque nœud établit, pour lui seul, la liste des nœuds corrects du groupe telle qu'il peut la voir (*i.e.* conforme à ce que les informations qu'il peut recevoir lui apprennent).

Dans ce cas, la difficulté réside dans l'assurance de la cohérence de l'information (messages diffusés, vue du groupe) entre tous les nœuds. L'approche adoptée est alors de garantir la progression des vues du groupe en phases logiques distinctes

durant lesquelles la vue du groupe ne change pas et est identique pour tous les membres de cette vue : les *configuration*s. Ensuite, au sein de ces configurations, il s'agit de garantir la cohérence sur les messages reçus, grâce à l'établissement d'un ordre entre tous les messages diffusés (pour se rapprocher de la diffusion atomique). C'est la définition même de la "synchronisation virtuelle" [18, 4] (*Virtual Synchrony*).

Au niveau de la gestion du groupe, il s'agit de garantir que les nœuds impliqués voient identiquement les changements de configuration. C'est à dire que tous les nœuds doivent décider de la même nouvelle configuration, avec les mêmes informations (*i.e.* même liste de membres), sachant la provenance de tous les membres (en redémarrage, membre d'une ancienne configuration).

Suivant l'environnement d'utilisation du protocole, la démarche de résolution du problème de la synchronisation des changements de vue est différente.

RTCAST (University of Michigan, 1996), un protocole temps-réel de diffusion atomique, suppose que la seule défaillance possible est la *défaillance par arrêt* des nœuds. Par conséquent, le réseau (supposé à diffusion) est parfait, et les omissions en réception sont assimilées à des *défaillances par arrêt* du nœud qui en est victime.

Avec de telles hypothèses, dès qu'un nœud diffuse un changement de configuration, tous les nœuds opérationnels sont *de facto* en accord (sinon, c'est que le message a été perdu, et la victime de cette perte doit cesser de fonctionner). Mais le prix de cette simplicité de l'assurance de l'accord est l'aspect *exclusif* du protocole (les omissions temporaires en réception d'un nœud p ont pour effet d'exclure p de la vue du groupe).

En pratique, le protocole suppose un réseau synchrone sur lequel un message particulier (le jeton) passe de nœud en nœud. Ce jeton donne le droit, et le devoir (le protocole repose sur la gestion de signes de vie émis par les différents nœuds) d'émettre à son possesseur.

Isis (Cornell University, 1992-1995), et son successeur **Horus** (1993-...), ont donné naissance au concept de "Synchronisation virtuelle" sur réseau diffusant asynchrone.

Ces protocoles reposent sur un jeton-message, et doivent assurer la synchronisation virtuelle des changements de configuration au sein d'un corpus particulier de processeurs : la partition majoritaire (*primary partition*). Puisqu'il ne peut y avoir qu'une seule partition majoritaire, le partitionnement n'est pas autorisé.

TOTEM (Santa Barbara University, 1992-...) repose également sur un réseau asynchrone (local, *i.e.* à diffusion, ou étendu, *i.e.* formé de réseaux locaux interconnectés), mais, à la différence de Isis/Horus, assure la Synchronisation Virtuelle Étendue [4] (*Extended Virtual Synchrony*). Cette propriété assure que des nœuds issus de configurations différentes seront admis dans la même nouvelle configuration, mais en terminant de façon cohérente leur configuration d'origine respective.

Avec cette propriété, ce protocole permet à plusieurs partitions minoritaires de fusionner, tout en garantissant qu'avant la fusion, les partitions minoritaires ont toujours été dans un état cohérent.

La gestion du groupe au sein de chaque nœud est régie par un automate d'états finis, sensibles à des alarmes déclenchées par le temps, à la perte

de messages, et à la perte du jeton. Cet automate a le gros inconvénient de présenter des cycles entre ses états, ce qui interdit tout comportement temporel borné. Ainsi, un changement de configuration ne peut pas être borné dans le temps (mais une loi probabiliste de sa durée peut-être établie), et TOTEM n'est donc pas adapté au fonctionnement en milieu temps-réel.

La thèse de Matthew Clegg (1997) définit un protocole de gestion de groupe sous contraintes de temps-réel, et reposant sur un réseau synchrone. Simple et intuitif, ce protocole présente des propriétés temporelles strictes (à la différence de TOTEM), sans pour autant être trop exclusif (à la différence de RTCAST).

Cependant, les hypothèses sur le système prises par M. Clegg sont incompatibles avec les nôtres. En effet, le protocole de M. Clegg repose sur la réplication des canaux de communication. Or, le projet HADES, qui constitue le cadre de ce document, ne suppose qu'un seul canal de diffusion disponible, et sujet à des défaillances.

2.2 Diffusion atomique

Parmi les protocoles de diffusion atomique étudiés, tous, sauf RTCAST et celui de Clegg, occultent l'aspect temps-réel. Cependant, le temps de latence de la diffusion, même s'il n'est pas strictement borné comme l'impose le temps réel (l'évaluation est en effet *probabiliste* plutôt que stricte), constitue souvent une deuxième métrique de performance (la première métrique étant le plus souvent le débit maximal espéré).

Ce paragraphe est une caractérisation de ces protocoles de diffusion atomique suivant deux axes : leur prise en charge de la tolérance aux fautes, et leur comportement temporel (en particulier la possibilité d'exhiber des bornes temporelles).

2.2.1 Ordres de remise des messages à l'application

Assurer la diffusion atomique, c'est apposer à chaque message destiné à un ensemble de processus une date (logique), et garantir que cette date est unique parmi les messages diffusés.

La sémantique de remise des messages propre à chaque protocole définit ensuite la notion d'unicité de cette estampille (unicité relativement à tous les messages, ou relativement aux messages *causalement liés*, ...). Il s'agit donc d'ordonner les messages. Cet ordre peut être [5, 19, 20, 13, 4, 12] :

Fifo (First In First Out). L'ordre d'émission relativement aux messages émis par un nœud donné est préservé à la réception (par tous les nœuds, considérés indépendamment). Il n'y a donc pas de notion d'ordre global, et plusieurs émissions peuvent se faire simultanément : rien n'indique si tous les nœuds vont recevoir les messages issus de nœuds différents dans le même ordre;

Causal. Une réponse à un message est toujours reçue après le message, qu'elle provienne ou non du même nœud, et un ordre total de précédence des messages au sein d'un même nœud existe. L'ordre est ainsi respecté au sein d'une même session de messages/réponses, mais pas sur tous les messages (en particulier si les flux de messages/réponses sont indépendants);

Ordre Total (ou "**Agreed Order**"). Globalement, sur l'ensemble du système, tous les messages émis par chaque membre du groupe sont soumis à un ordre total. Pour tous les messages différents émis pris deux à deux, il y a forcément une relation de précédence qui caractérise le couple;

Livraison[3] **sûre** (“**Safe delivery**”). Les messages sont totalement ordonnés, et on est assuré de la réception de tous les messages précédents, relativement à l'ordre total implanté, par tous les membres du groupe avant l'émission du suivant.

2.2.2 Principe des protocoles de diffusion

Masquer les défaillances temporaires dues à des omissions en émission, réception, ou transmission, revient à assurer que, si l'information d'un premier message ne parvient pas à destination, elle y parviendra grâce à un deuxième message, et ainsi de suite. Ceci impose la redondance de l'information à hauteur d'un nombre de fautes maximal toléré, tel que défini dans l'énoncé des hypothèses de défaillance (voir le paragraphe 1.3).

Ce principe de répétition de l'information constitue la base de tous les protocoles de communication tolérante aux fautes, que la communication soit de type point à point, ou multipoint. Cette redondance fait appel, en pratique, à une réplication spatiale des canaux (et à l'envoi de l'information sur tous les canaux), ou à une réplication temporelle de l'information (*i.e.* répétition).

Tous les protocoles étudiés, sauf celui de M. Clegg, s'appuient sur la réplication temporelle de l'information, *i.e.* la répétition. Quant à M. Clegg, il a choisi de bâtir son protocole au dessus d'un système dans lequel le canal de diffusion est répliqué.

Le problème que posent la plupart des protocoles de diffusion étudiés et ayant recours à la répétition des messages est que le nombre de répétitions (principe des acquittements positifs ou négatifs) n'y est pas borné. Ceci interdit alors toute prédétermination des délais de diffusion atomique de l'information, et interdit par conséquent l'intégration de ces protocoles au sein de systèmes temps-réel.

Il y a plusieurs façons d'assurer la diffusion de l'information, connaissant la composition du groupe :

Répétition à la demande. C'est la méthode la plus couramment utilisée : elle correspond à utiliser des acquittements (positifs et/ou négatifs sur la réception des messages diffusés). Elle consiste à user de mécanismes permettant à un nœud p de détecter qu'un message ne lui est pas parvenu (utilisation de numéros de séquence sur les messages). Quand p s'aperçoit qu'il lui manque un message, il peut le signaler aux autres nœuds (acquittement négatif), qui vont alors le lui retransmettre.

Les protocoles suivants fonctionnent selon ce principe général.

Chang & Maxemchuk (diffusion atomique). Les nœuds forment un anneau virtuel, et se passent un jeton autour de cet anneau. Le jeton définit le nœud responsable de l'ordre entre tous les messages. C'est lui qui délivre l'estampille ordonnant totalement tous les messages, et permettant aux autres nœuds de détecter les omissions. C'est ce processeur-jeton qui est chargé de la réémission des messages manquants signalés par les processeurs victimes d'omissions.

Chaque message se voit répéter pendant plusieurs tours du jeton consécutifs, donc par plusieurs processeurs-jeton, afin d'assurer le consensus sur la réception du message.

3. Un message *délivré* à l'application signifie que les couches inférieures liées à la gestion de la diffusion transmettent le message aux couches du dessus (au reste de l'application).

Pour partager la responsabilité de ce séquencement (en diminuant le risque que le processeur-jeton soit victime d'une défaillance par arrêt), le jeton circule entre tous les membres du groupe de diffusion. La perte du jeton (détectée suite à une absence d'estampillage d'un message émis, ou après non retransmission de messages comme demandée) est résolue par une procédure de reconfiguration du système.

Amoeba (diffusion atomique). Un nœud particulier est chargé d'attribuer les numéros de séquence, pour garantir l'ordre total : le *séquenceur*. Les messages manquants correspondent à des trous dans la séquence, et le nœud qui en est victime envoie une demande de retransmission au séquenceur. Les messages reçus font automatiquement l'objet d'un acquittement de la part du récepteur. C'est au séquenceur de diffuser l'autorisation de remise des messages à l'application. Cette décision est prise lorsqu'un message a été acquitté par un quorum de récepteurs établi par avance : le taux de résilience du protocole. Ainsi, le séquenceur garantit seulement l'existence d'un nombre minimal de nœuds ayant reçu, de façon sûre, chaque message dans le système.

L'inconvénient de ce protocole est essentiellement qu'il repose sur un nœud particulier. Les changements de configuration sont acceptés, mais, lors de la défaillance par arrêt du séquenceur, l'état du système peut être incohérent.

PSync (diffusion causale). Ce protocole diffuse l'*historique* de chaque message (graphe de précédence entre messages d'une même *conversation*) avec chaque message, afin de garantir l'ordre causal entre tous les message diffusés. Ainsi, à la réception d'un message, il est aisé de détecter les messages manquants, au regard de cet historique. La demande de réémission des messages manquants de cet historique se fait par l'intermédiaire d'un message particulier envoyé à l'émetteur de cet historique (*i.e.* l'émetteur du dernier message reçu, puisque le nœud victime de l'omission reste bloqué jusqu'à réception de ces messages manquants).

L'historique est potentiellement infini, toutefois, en pratique, il est de taille limitée puisque les messages *stables* (*i.e.* les messages qui ont été reçus par tous les participants à la conversation) sont enlevés de cet historique. Mais, en présence d'un mécanisme de gestion de groupe, cet historique peut être borné, puisqu'il est possible de connaître les nœuds, défaillants par arrêt, qui ne vont jamais recevoir les messages de l'historique.

Fast ABCast[4] (diffusion atomique). Isis a introduit le concept de *synchronisation virtuelle* [4] (*virtual synchrony*). Ceci garantit que, entre deux changements de configuration, les membres du groupe qui restent en vie reçoivent et délivrent à l'application les mêmes messages dans le même ordre (ainsi, l'évolution du groupe se trouve découpée en phases logiques distinctes de cohérence de l'information). C'est une propriété clef dans la garantie de la cohérence de l'information répartie.

Le protocole doit reposer sur un protocole de communication point à point fiable (*i.e.* non perte, non duplication, respect de l'ordre des messages – par exemple : TCP), et sur un protocole de diffusion fiable causale (Fast CBCast). Il se déroule en deux phases. L'émetteur d'un message commence par le diffuser en utilisant le protocole de diffusion causale. Ensuite, le processeur-jeton diffuse, par le protocole de diffusion causale, l'estampille

4. Système Isis.

pour le message, autorisant les nœuds à remettre le message en séquence. La tolérance aux omissions est gérée par le protocole de diffusion causale.

Système Transis. La couche protocolaire Trans du système Transis se charge de gérer la diffusion atomique. Elle fonctionne à la manière de PSync, mais en optimisant la bande passante utilisée : les informations d'*historique* (sous forme d'acquittements) sont encapsulées dans les messages ordinaires, permettant de rendre compte de l'historique disponible sur chaque nœud, et les demandes de retransmission sont également intégrées aux messages ordinaires (sous la forme d'acquittements négatifs). Le système Transis peut garantir la synchronisation virtuelle, et même la *synchronisation virtuelle étendue* (voir la définition dans la description de TOTEM suivante).

Système TOTEM. Ce protocole peut gérer des groupes de diffusion interconnectés, ce qui autorise des systèmes distribués de très grands diamètres. Deux protocoles coopèrent pour que ceci soit possible : l'un gère les réseaux locaux (*i.e.* à diffusion), et l'autre gère l'interconnexion de ces réseaux. TOTEM permet ainsi de garantir la synchronisation virtuelle étendue, qui permet de garantir la cohérence de l'information distribuée, même en cas de réunification de plusieurs partitions.

Tous ces protocoles ont l'avantage d'optimiser les ressources réseau, ce qui permet d'atteindre un débit de diffusion des messages utiles optimal en l'absence de défaillances permanentes (autorisant des débits comparables à ceux obtenus en point à point, comme avec TCP). Ils ont cependant le gros inconvénient de n'être pas du tout adaptés aux contraintes du temps réel. En effet, il est impossible de prévoir, avec tous ces protocoles, le nombre maximal de retransmissions[5] qui vont être nécessaires pour la diffusion de chaque message. Or, ces retransmissions ont deux conséquences :

- Elles retardent la remise à l'application du message à retransmettre;
- Elles occupent de la bande passante, et donc retardent également l'émission des autres messages.

Il s'ensuit que cette imprévisibilité des retransmissions engendre l'impossibilité de borner les temps de latence de la diffusion.

Pour s'adapter au temps-réel, il faut donc inclure la connaissance de ces retransmissions dans le protocole. C'est ce à quoi correspondent les deux classes de répétition suivantes.

Répétition par suspection de non réception. C'est la démarche employée par le **protocole de M. Clegg**. Il s'agit d'un protocole à jeton dans lequel le jeton donne le droit de diffusion à son possesseur. Dans ce protocole, les messages sont émis sur plusieurs canaux de diffusion successivement. Quand un nœud n'a pas reçu le message sur le dernier canal d'émission, et qu'il n'a pas eu de nouvelles de son émetteur depuis *trop* longtemps, il suspecte le message de ne pas avoir été reçu par tous les nœuds, Dans ce cas, il le réémet dès que le jeton lui est parvenu. Compte tenu des hypothèses de défaillances, une rediffusion par nœud suspicieux est suffisante avant que le message ait pu être diffusé, de façon certaine, à tous les nœuds du groupe.

Ces hypothèses stipulent en effet que, relativement à tout message émis, en considérant :

f_{in}, le nombre maximal de défaillances en entrée des interfaces réseau des

5. Une évaluation probabiliste est en général donnée, qui peut servir de métrique de comparaison des protocoles, en nombre de messages nécessaires.

nœuds du système (*i.e.* en réception);

f_{out}, le nombre maximal de défaillances en sortie (*i.e.* en émission);

f_{chan}, le nombre maximal de canaux de communication défaillants (*i.e.* muets) simultanément;

c_{max}, le nombre de canaux de communication répliqués (*i.e.* l'indice de réplication);

Le système est alors caractérisé par $f_{in} + f_{out} + f_{chan} < c_{max}$, ce qui veut dire que, dans le pire des cas ($f_{in} + f_{out} + f_{chan} = c_{max} - 1$), un nœud reçoit quand même correctement le message. Lors de la retransmission suivante par ce nœud, ce message ne peut plus être victime de défaillances[6], et donc tous les nœuds corrects le reçoivent correctement.

Avec cette démarche, il est aisé d'évaluer strictement le temps de latence de diffusion atomique d'un message. Ainsi, la remise d'un message à l'application dépend uniquement de la date de son émission.

En réalité, les systèmes-temps réel sont plus exigeants encore; ils nécessitent de connaître les temps de remise pire cas de *tous* les messages, c'est à dire à la fois :

- de ceux qui ont souffert d'une retransmission avant de parvenir à destination;
- de ceux qui n'ont pas eu à être retransmis, mais qui ont du être émis plus tard à cause de la retransmission d'un autre message.

Ainsi, dans le pire des cas, il faut prévoir, dans l'ordonnancement et dans le contrôle de flux, la réémission de tous les messages. Dans ce contexte, l'utilisation du réseau devient équivalente à celle qui serait obtenue en répétant systématiquement *tous* les messages émis.

Répétition systématique. C'est la technique qui est choisie dans le protocole présenté dans ce document. Il est impossible de s'inspirer du principe de suspection du protocole de Clegg précédent, puisque, disposant d'un canal de diffusion unique, nous ne sommes pas en mesure de suspecter, ou de prévoir, la non réception d'un message par un autre nœud.

3 Terminologie

Nous présentons et définissons les termes employés dans toute la suite de ce document. En particulier, nous apportons une grande attention à la définition de la *correction*.

3.1 Le *nœud* et son *processeur*

Nous distinguons les deux termes "*processeur*" et "*nœud*".

Le *processeur* est l'entité de traitement;

Le *nœud* correspond à l'ensemble {processeur + interface réseau d'entrée + interface réseau de sortie}.

6. Il a épuisé son quota de défaillances.

3.2 Le *système*, le *groupe*, les cibles d'un message

Nous distinguons :

- Le *système*, qui définit le contexte d'exécution des protocoles, c'est à dire les paramètres physiques concernant le réseau, le nombre de nœuds, la liste des nœuds, ainsi que les fréquences minimales d'émission/réception des messages de gestion de groupe;
- Le *groupe* est l'ensemble des nœuds du système impliqués dans le protocole de gestion de groupe, c'est à dire l'ensemble des nœuds du système considérés corrects par le protocole de gestion de groupe;
- Les *cibles absolues* d'un message correspondent à la liste des nœuds destinataires du message;
- Les *cibles courantes* d'un message représentent les nœuds de la cible absolue d'un message qui sont considérés opérationnels (*i.e.* dans la vue du groupe). Il s'agit donc de la liste des destinataires effectifs du message.

En résumé, la *cible absolue* est un sous-ensemble du *système* et est liée à chaque message, alors que la *cible courante* est un sous-ensemble de la vue du *groupe* courante (fournie par le protocole de gestion de groupe) et de la *cible absolue* : il s'agit de leur intersection.

Le système est parfaitement déterminé et immuable, c'est à dire que l'introduction de nœuds étrangers au système n'est pas permise.

Le protocole de gestion de groupe s'attache à déterminer l'état des différents nœuds du système (*i.e.* correct ou *défaillant par arrêt*), c'est à dire s'attache à identifier la *composition du groupe*.

Dans ce contexte, le *système* (immuable) est constitué de la population cible du protocole, et la *vue du groupe*, ou *configuration*, ou encore *composition du groupe*, correspond alors à l'état (*i.e. correct* ou *défaillant par arrêt*) associé à chacun des nœuds composant le système. Un nœud est *dans la vue du groupe*, ou *adhérent* au groupe, s'il est *correct*. Il en est exclu si est *défaillant par arrêt*.

Le protocole ne propose pas de mécanisme de *reconfiguration*, au sens de la modification des paramètres du système (en particulier la liste des nœuds le composant). Ainsi, si un nœud est déclaré *défaillant par arrêt*, et exclu de la vue du groupe, ou si un nœud cherche à se réintroduire dans la vue du groupe, le protocole ne cherche pas à définir de nouvelles valeurs pour les paramètres du système, *i.e.* la liste des nœuds du système n'est pas modifiée. En effet, une telle *reconfiguration* (imprévisible de par la nature imprévisible des évènements qui y conduisent), et en particulier l'assurance du consensus sur les nouveaux paramètres du système, engendrerait de façon imprévisible des délais supplémentaires. Ceci serait incompatible avec un test d'ordonnancement.

3.3 Le *réseau* et le *jeton*

Le système est soumis à des contraintes fortes de temps-réel. Afin de disposer d'un temps de diffusion pire cas garanti, le système doit reposer sur un réseau synchrone (le temps de transmission pire cas d'un message est garanti borné). Nous supposons qu'il existe un mécanisme de diffusion non fiable sous-jacent[7].

7. Contrairement à ce qu'utilisent d'autres protocoles dits *multihop*, *i.e.* fondés sur une série de connexions point à point successives.

Afin d'éviter les retransmissions imprévisibles dues à des collisions non prévisibles de messages (une retransmission imprévue risque de violer les contraintes temporelles), nous introduisons un mécanisme d'accès exclusif au médium de diffusion non fiable (ce n'est pas la seule solution possible). Ce mécanisme fait appel au *jeton*, entité abstraite dont la définition est la suivante :

> La possession du *jeton* par un processeur du groupe lui donne le droit d'émettre des messages.

Le *jeton* peut être implanté soit par un message particulier (un *jeton-message*) que les processeurs se transmettent dans un ordre prédéfini (*i.e.* autour d'un *anneau*), soit par un intervalle de temps tel que tous les intervalles d'émission de tous les processeurs ne se recouvrent pas (*i.e.* l'exclusivité de possession du jeton reste respectée).

3.4 Différents types de *correction*s

Les défaillances par omission en émission et en réception sont dites *temporaires* si elles ne durent pas plus de ω (défini au paragraphe 4.1) tours consécutifs du jeton. Dans ce cas, le protocole garantit qu'elles seront masquées, c'est à dire que l'accord sur la composition du groupe sera maintenu. Dans le cas où une défaillance par omission (en émission ou en réception) perdure sur plus de ω tours du jeton, elle devient *permanente* et doit entraîner un arrêt du nœud qui en est victime, c'est à dire simuler une défaillance par arrêt du nœud.

Le terme de *correction* est employé indifféremment dans toute la suite, qu'il s'agisse de correction des nœuds, des processeurs, ou de l'émission de messages. Nous donnons ici la définition de ce qui est associé à ce terme *correction* dans ces trois contextes :

Le processeur d'un nœud est *correct* s'il effectue le traitement qui lui incombe conformément aux instructions du programme qu'il exécute;

Un nœud est *correct* si il n'est pas *défaillant par arrêt*, c'est à dire si le processeur associé est *correct*, et si l'interface réseau (en émission *et* en réception) ne souffre pas de défaillance permanente;

Un nœud *émet correctement un message* s'il arrive à faire parvenir le message jusqu'au médium de communication (*i.e.* ni le processeur, ni l'interface réseau de sortie ne sont défaillants).

Remarquons que, lorsque le processeur est *correct*, il se peut très bien que le nœud correspondant soit *défaillant par arrêt* : c'est le cas si l'interface réseau est défaillante depuis *trop* longtemps (la signification de "*trop*" est présentée au paragraphe 4.1). Inversement, si le nœud est correct alors *a fortiori* le processeur associé l'est aussi. Précisons que, dès qu'un nœud est *défaillant par arrêt*, sa participation dans le décompte des défaillances en émission et réception est annulée (*i.e.* un processeur *défaillant par arrêt* ne souffre pas d'omissions en réception).

3.5 Les *signes de vie*

Un signe de vie correspond à un message qui permet aux nœuds qui vont le recevoir de prendre connaissance que le processeur de p est correct. Ces signes de vie encapsulent les données nécessaires au protocole de gestion de groupe pour fonctionner.

Le protocole de gestion de groupe présenté dans ce document se fonde sur l'échange de tels signes de vie, pour que chaque nœud puisse établir la vue du groupe.

4 Modèle du système

Les paramètres physiques du système correspondent aux données initiales dont disposent les protocoles de gestion de groupe et de diffusion atomique pour fonctionner, et ne peuvent changer au cours du fonctionnement du système. Le tableau 1 présente les paramètres du système, pour lesquels tous les nœuds disposent de la même valeur.

Table 1 Les constantes, paramètres du système

n	Le nombre total de nœuds que compte le système. Cette valeur est utile à l'énoncé des hypothèses de défaillance.
Δ_{trans}	La latence maximale (en temps logique mesurable par tous les nœuds[8]) du réseau en l'absence de défaillances (*i.e.* le temps maximal de transfert d'un message de l'entrée de la carte réseau de l'émetteur, à la sortie de la carte réseau du récepteur).
δ_{send}	L'écart maximal entre deux messages émis par un même nœud.
δ_{recv}	L'écart maximal entre la réception d'un message par la couche réseau, et l'activation de la tâche de réception du protocole de diffusion atomique.
ϵ	La précision de la synchronisation des horloges. C'est à dire que, pour deux nœuds p et q corrects quelconques du système, si $C_p(t)$ et $C_q(t)$ sont les dates indiquées par les horloges locales respectives de p et de q au temps réel t, on est assuré que $\forall t, \lvert C_p(t) - C_q(t) \rvert \leq \epsilon$.

4.1 Hypothèses de défaillance

Les nœuds du groupe sont soumis à des défaillances. Certaines, comme les défaillances par arrêt, n'affectent que les processeurs des nœuds. D'autres, comme les omissions, sont propres aux interfaces réseau. Par contre, le médium de communication est exempt de fautes. Tous ces comportements sont décrits ci-après.

H1. Le processeur d'un nœud ne peut que subir une défaillance par arrêt, c'est à dire qu'une fois le processeur arrêté, le nœud est et reste muet (également dit "*défaillant par arrêt*"). Le nombre maximal de processeurs *défaillants par arrêt* à un instant donné est f_{crash};

H2. Il existe une constante ω telle que :

(a) Sur ω diffusions d'un même message en ω tours de jeton consécutifs par un nœud p dans la vue du groupe courante, p arrive à faire parvenir le message à la majorité des nœuds de la vue du groupe courante, *i.e.* $\lfloor \frac{n}{2} \rfloor + 1$.
Ceci **ne signifie pas** que tous ces autres nœuds perçoivent *la même instance* du message de p pendant ces ω tours du jeton.

(b) En ω tours du jeton, un nœud p dans la vue du groupe courante reçoit des messages émis pendant cette période par au moins une majorité de nœuds dans la vue du groupe, *i.e.* $\lfloor \frac{n}{2} \rfloor + 1$.

Cette hypothèse implique que le système est caractérisé par une majorité de nœuds *corrects* (voir la définition au paragraphe 3.4) : $\lfloor \frac{n}{2} \rfloor > f_{crash}$. Ceci impose le non partitionnement du système, et donc, à tout moment, l'existence

8. Si seule la latence en temps réel est connue, il faut donc établir cette latence en tenant compte de la dérive des horloges.

d'une vue du groupe (*i.e.* d'une partition) unique. Il est alors nécessaire que :

$$f_{crash} < \lfloor \frac{n}{2} \rfloor$$

Cette hypothèse signifie également que, tant qu'un nœud reste correct, il perçoit *régulièrement* des signes de vie d'au moins $\lfloor \frac{n}{2} \rfloor$ autres nœuds corrects, et ses signes de vie sont *régulièrement* perçus par au moins $\lfloor \frac{n}{2} \rfloor$ autres nœuds corrects; Dans toute la suite, si n est le nombre de nœuds que compte le système, alors *une majorité* correspond à au moins $\lfloor \frac{n}{2} \rfloor + 1$ nœuds.

H3. L'interface réseau d'un nœud peut être victime d'omissions en émission, c'est à dire qu'elle n'arrive pas à émettre un message. Le nombre maximal de tours consécutifs du jeton pendant lesquels l'interface réseau d'un nœud peut souffrir de défaillances consécutives en émission est f_{send};

H4. L'interface réseau d'un nœud peut être victime d'omissions en réception. Par message *correctement* émis (*i.e.* parvenu jusqu'au médium de communication), le nombre maximal de nœuds, différents de l'émetteur du message, dont l'interface réseau est ainsi défaillante en réception, est f_{recv}.
Pour assurer qu'un message *correctement* émis est reçu par au moins un processeur correct, il est nécessaire d'avoir :

$$n - 1 - f_{crash} > f_{recv}$$

H5. Le médium de communication (et le commutateur central dans le cas de la topologie en étoile de HADES) est supposé fiable (*i.e.* ni duplication, ni perte, ni partitionnement physique).

Remarque sur l'hypothèse H5 La dernière hypothèse peut sembler forte, mais il s'agit d'un artifice pour démontrer les propriétés du protocole. Elle signifie que l'on déporte les défaillances du réseau aux interfaces réseau de chacun des nœuds.

4.2 Interaction entre les protocoles de gestion de groupe et de diffusion atomique

Nous supposons que le protocole de gestion de groupe s'occupe de rendre disponible la composition du *groupe* impliqué dans la diffusion. Pour cela, le protocole de diffusion doit être informé des défaillances par arrêt, ou des réinsertions de nœuds, mais il ne doit pas s'en soucier. Il doit toujours considérer que la liste des nœuds dont il est informé par le protocole de gestion de groupe contient exactement tous les nœuds *corrects* parmi les *cibles absolues*, c'est à dire correspond à la *cible courante*.

Ainsi, les seules défaillances dont le protocole de diffusion doit se préoccuper sont les défaillances temporaires : ceci concerne les omissions, qui sont propres aux interfaces réseau. Le médium de communication est lui exempt de fautes.

La figure 1 suivante résume les interactions entre les protocoles de gestion de groupe et de diffusion atomique, et entre ces protocoles et le reste du système.

5 Description du protocole de gestion de groupe

5.1 Généralités

Ce document propose un protocole de gestion de groupe adapté aux contraintes du temps réel, et ne nécessitant pas une plate-forme d'utilisation particulière (les canaux

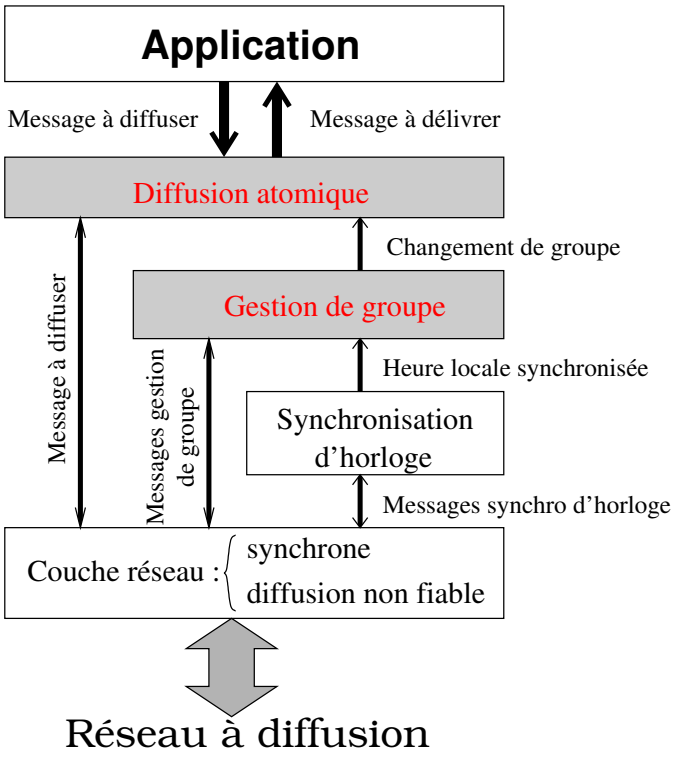


FIG. 1 – *Interaction entre les protocoles, l'application et le système*

n'ont pas besoin d'être répliqués). Il s'appuie sur un réseau à diffusion synchrone, sur lequel l'accès au médium doit être exclusif (en pratique, il est fait recours au TDMA), et l'ensemble de ses propriétés (dont la propriété d'accord est la principale) sont établies relativement à des critères temporels.

Les changements de configuration ne s'effectuent pas à la suite d'une négociation entre les acteurs impliqués, mais suite à une même décision prise à la même heure locale par tous les nœuds impliqués (même ceux qui sont exclus du groupe), de par le fait que tous les nœuds disposent de la même information. Cette propriété essentielle du protocole provient de la réplication temporelle (et non spatiale comme dans le protocole de Matthew Clegg) des informations de gestion de groupe, permettant de garantir une borne supèrieure sur la durée que vont mettre ces informations à se répandre parmi tous les nœuds corrects. Cette propriété tient compte des défaillances possibles. Le protocole masque en effet tant les *défaillances par arrêt* des nœuds, que les omissions en émission/réception, tout en ne nécessitant pas une architecture spécialisée.

5.1.1 Conception générale du protocole

Le principe de la gestion de groupe est de maintenir, sur chaque nœud p :

- l'état que le nœud p perçoit des autres nœuds du groupe afin de détecter les défaillances des autres processeurs et les défaillances en réception des interfaces réseau associées;
- la vision que les autres nœuds ont de p pour détecter les défaillances en émission de p.

Le principe est d'échanger périodiquement les signes de vie qu'un nœud a reçus des autres nœuds du groupe. Ceci permet d'assurer en un temps borné la propagation (la diffusion) des signes de vie les plus récents des différents nœuds du groupe, afin de garantir l'accord des décisions sur la composition du groupe.

5.1.2 Contraintes

La définition du jeton donnée dans le paragraphe 3.3 n'impose pas la nature précise du jeton (jeton message ou fenêtre temporelle).

Ce protocole suppose juste que :

Hypothèse 1 *Le nœud courant doit émettre au moins un message tous les δ_{send} unités de temps.*

Le problème que pose l'utilisation d'un jeton-message, est qu'il faut en gérer la perte et la duplication, ce qui risque de violer les contraintes fortes du temps-réel.

Ainsi, dans toute la suite, nous supposerons que l'accès exclusif au médium de communication est garanti par un partage du temps d'accès à ce médium entre tous les nœuds du système. Le jeton correspond alors à la possession régulière d'un intervalle de temps. En d'autres termes, nous utiliserons une technique d'accès au médium appelée TDMA (Time Division Multiple Access). Cette approche nécessite cependant que les horloges locales des différents nœuds soient synchronisées avec une précision connue.

De plus, conformément à la définition du *groupe* donnée au paragraphe 3.2, les fenêtres temporelles d'accès au médium sont établies une fois pour toute pour l'ensemble du système (c'est un paramètre du système), et ne sont pas négociables, c'est à dire qu'un *changement de composition du groupe* ne les modifie pas.

5.2 Périodicité de l'émission

Dans HADES, et pour des considérations de commodité d'ordonnancement (il est plus aisé d'intégrer une tâche périodique qu'une tâche apériodique dans un test d'ordonnancement), la tâche d'émission sera périodique de période ρ_S [9].

Cas particulier dans lequel on dispose de la précision temps-réel de la synchronisation des horloges.

Soit ϵ_{RT}, une telle précision de la synchronisation des horloges locales aux nœuds corrects du système par rapport au temps réel (*i.e.* si $C_p(t)$ est le temps local à un nœud correct p au temps réel t, on a $\forall t, |C_p(t) - t| \leq \epsilon_{RT}$). Alors, en tenant compte **i)** du temps d'exécution pire-cas de la tâche d'émission r_c, et **ii)** de la synchronisation temps-réel des horloges de précision ϵ_{RT}, le partage exclusif du temps entre tous les processeurs implique :

$$\delta_{send} = \rho_S + r_c + 2\epsilon_{RT}$$

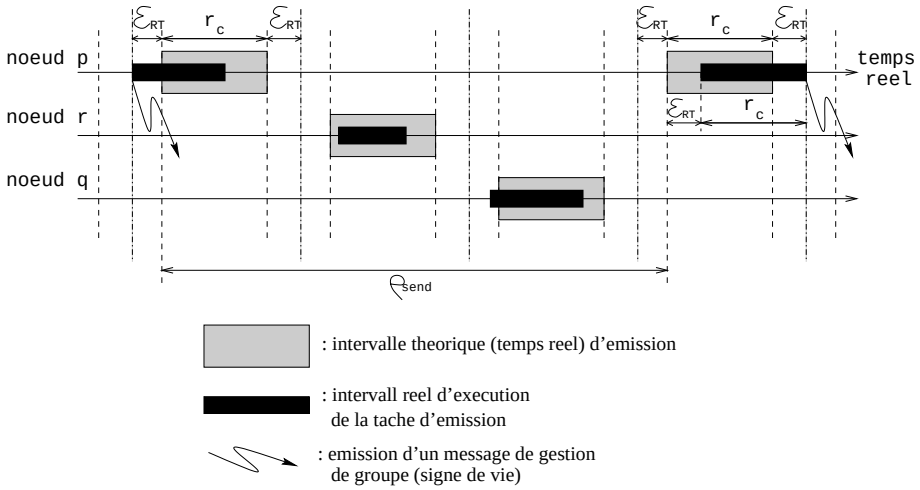


FIG. 2 – *Relation entre δ_{send} et ρ_S, dans le pire des cas*

En effet, comme l'illustre la figure 2, dans le pire des cas, la fenêtre d'émission d'un nœud p est déclenchée ϵ_{RT} unités de temps trop tôt par rapport à sa date de début nominative. À cette date, p émet un message m_1 de gestion de groupe, et il n'en émettra plus dans cette fenêtre d'émission. L'émission du message m_2 de gestion de groupe suivant arrivera au plus tard quand la fenêtre d'émission suivante arrivera en retard de ϵ_{RT}, et quand m_2 sera envoyé à la fin de cette fenêtre. Ainsi, dans ce scénario pire-cas, les deux messages consécutifs m_1 et m_2 sont séparés dans le temps-réel par $\epsilon_{RT} + \rho_S + r_c + \epsilon_{RT}$, d'où la minoration précédente de l'écart pire-cas entre deux messages consécutifs δ_{send}.

9. Il s'agit de la valeur pire-cas de la période d'activation de la tâche d'émission. Lors de l'implémentation, il faut tenir compte de la gigue d'activation. Ainsi, dans le cas d'un ordonnancement du type *Rate Monotonic*, si la périodicité de la tâche d'émission est T_S, alors $\rho_S = 2T_S$.

Remarque : Dans le cas d'horloges synchronisées dans le temps-réel, on ne tient pas compte de la contrainte de Johnson et Sevcik [21] sur les gigues de possession du jeton. Cette contrainte affirme que si le jeton donnant l'exclusivité d'accès au médium de communication a une période de rotation nominative autour de l'anneau $TTRT$, alors, un nœud de l'anneau disposera de ce jeton au pire tous les $2TTRT$ unités de temps. Une telle gigue provient des retards du jeton occasionnés par la diffusion de messages asynchrones (*i.e.* sans contrainte de temps réel). En effet, de tels messages commencent à être diffusés si le jeton est reçu en avance, mais leur émission peut se terminer après l'instant auquel le jeton est attendu (le jeton autorise alors l'envoi des messages synchrones, *i.e.* sous contraintes de temps-réel). Dans HADES, le jeton ne correspond pas à un message particulier, mais plutôt à une fenêtre de temps propre au processeur (donc indépendante des messages et de l'agissement des autres nœuds), et ne peut donc être ni en retard, ni en avance (modulo la précision des horloges). Dans ce contexte, la diffusion de messages asynchrones suivant ce principe est impossible, et la gigue de possession du jeton n'existe pas.

5.3 Aperçu du déroulement du protocole – Principe de transitivité

Pour maintenir la vue du groupe, chaque nœud p doit gérer localement trois informations :

I1. La liste des messages qu'il a reçus des autres nœuds;

I2. La vision qu'ont les autres de son état;

I3. Les dates à partir desquelles les nœuds qui désirent se réintroduire dans le groupe des nœuds corrects, y seront véritablement admis.

Les deux premières informations sont essentielles pour pallier l'indépendance (au sens probabiliste) des défaillances de l'interface réseau en émission et en réception. En effet, dans HADES en particulier, il existe un découplage fort entre les deux canaux d'émission et de réception qui sont, dans le cas d'ATM, deux fibres optiques distinctes.

Il est donc possible que, pour un nœud p considéré :

- soit p ne reçoit aucun message des autres nœuds pendant plus de ω tours du jeton, auquel cas la première information (I1) est importante. Elle signifie que :
  - soit p est victime d'une omission permanente en réception, auquel cas p se voit tout seul dans le groupe. Alors, d'après p, l'hypothèse d'existence de la partition majoritaire de nœuds corrects est contredite, donc p doit simuler une *défaillance par arrêt* (*i.e.* p se suicide);
  - soit tous les autres nœuds sont *défaillants par arrêt*, ce qui est encore contraire à l'hypothèse de l'existence de la partition majoritaire (d'après p, la seule partition dans laquelle il est n'est pas majoritaire, puisqu'elle ne contient que p), et p doit alors se suicider.
- Soit p est victime d'une défaillance permanente en émission : p n'est pas entendu par les autres nœuds. Dans ce cas, p est capable de s'en rendre compte *via* la deuxième information (I2). Le nœud p doit également se suicider puisque son isolement contredit l'hypothèse d'existence de la partition majoritaire (p se voit isolé du reste du groupe).

Le principe du protocole est alors le suivant. Chaque nœud p diffuse (au moins tous les δ_{send} unités de temps) un message à tous les nœuds de la vue du groupe, contenant la vision qu'il a du groupe, c'est à dire la première information (*i.e.* I1, le vecteur contenant la date d'émission du dernier message reçu par p de chacun des

nœuds : c'est $recv_p[]$ défini en 5.4.2), et la troisième information (*i.e.* I3 : le vecteur contenant la date de réintroduction dans le groupe souhaitée par chacun des nœuds, c'est à dire $join_p[]$ défini en 5.4.2). À la réception d'un message, p note comment l'émetteur du message reçu affirme qu'il voit tous les nœuds du groupe. Si cette vision est plus récente que celle de p, p la met à jour. Le nœud p en profite pour noter comment l'émetteur du message le voit (dans la deuxième information : I2, *i.e.* dans le vecteur $last\_received\_from\_us_p[]$ défini en 5.4.2). Dans le cas où la majorité des autres nœuds ne l'a pas entendu depuis trop longtemps (*i.e.* ω tours du jeton), c'est que p est considéré comme défaillant (la majorité s'accorde à le voir comme tel), et alors p se suicide.

De cette manière, comme le montre le sénario de la figure 3 suivante, un signe de vie envoyé par un nœud p peut ne pas arriver directement à un autre nœud correct q, mais être reçu par un autre nœud r, qui, lui, va faire parvenir son signe de vie, ainsi que celui de p qu'il encapsule, jusqu'à q. Cette propagation du signe de vie, par *transitivité*, met en évidence le principe de la réplication temporelle des signes de vie, à la différence du protocole de M. Clegg qui reposait sur une réplication spatiale.

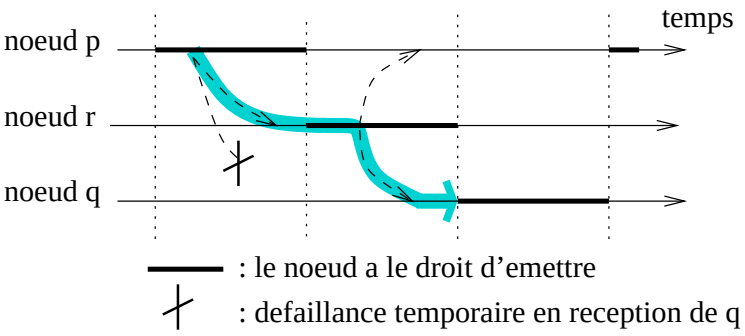


FIG. 3 – *Principe de transitivité*

Pour déterminer si un nœud particulier p est membre du groupe, un nœud q de la vue du groupe doit déterminer à la fois :

- s'il a reçu des nouvelles de p assez récentes;
- et si c'est le cas, le nœud p considéré ne doit pas être en train d'essayer de se réintroduire dans le groupe (car l'accord entre tous les membres de la vue sur la réintroduction de p dans la vue n'est pas garanti).

La troisième information (I3 : la date de réintroduction prévue) permet de garantir l'accord, entre tous les nœuds de la vue du groupe, de la date de changement de vue, marquant l'admission de nœuds qui redémarrent.

Une caractéristique importante de la thèse de M. Clegg a été conservée : les signes de vie sont estampillés par la date locale de leur émission. Ceci facilite l'accord sur les décisions de la composition du groupe (puisque tous les récepteurs disposent de la même donnée sur l'émetteur du signe de vie). Par conséquent, les vecteurs de dates qui sont échangés sont les vecteurs de dates d'émission des différents signes de vie. Afin de permettre l'accord sur les tests de "fraîcheur" des messages reçus (qui déterminent l'adhérence d'un nœud au groupe), il est donc nécessaire de disposer d'un mécanisme de synchronisation des horloges des différents nœuds (précision de synchronisation en temps mesuré).

5.4 Description du protocole

5.4.1 États des nœuds

Dans le protocole présenté, un nœud est potentiellement dans trois états :

défaillant par arrêt (*crashed*). Pendant ce temps, le nœud reste muet, soit à cause d'une défaillance du processeur, soit à cause d'une défaillance prolongée (*permanente*) des interfaces de communication (omissions en émission ou en réception);

redémarrage (*restarting*). C'est un intervalle de temps, succédant la *défaillance par arrêt*, pendant lequel le processeur du nœud correct essaye de se réintroduire dans le groupe des nœuds corrects du système. Pendant cette phase, le nœud se tient au courant de la configuration courante, et signale aux autres son désir d'intégrer le groupe des processeurs corrects. Un nœud dans cet état n'a pas le droit d'appartenir à la *vue* du groupe et donc il n'a pas le droit de participer à la vie du groupe;

opérationnel (*operational*). C'est l'état dans lequel se trouve un nœud considéré comme correct par tous les nœuds corrects. Un nœud dans cet état appartient à la vue du groupe et participe à la vie du groupe.

5.4.2 Paramètres du protocole

Ces paramètres correspondent aux différentes constantes utiles au protocole qui doivent permettre d'assurer l'accord sur les décisions de gestion de groupe (voir tableau 2).

Toutes ces constantes sont utilisées en tant que données d'entrée du protocole. Elles sont déterminées en fonction des paramètres du système et des hypothèses de défaillance (voir le paragraphe 6 énonçant et prouvant les propriétés du protocole).

Parmi elles, la constante Δ_{lat}, à savoir la latence de détection d'une *défaillance par arrêt*, est l'élément central de toutes les propriétés du protocole : c'est grâce à son évaluation correcte que la propriété fondamentale d'accord peut être garantie (propriété 14 au paragraphe 6.13).

Table 2 Les paramètres du protocole

Δ_{lat}	La latence de détection d'une *défaillance par arrêt* d'un nœud, *i.e.* temps maximal entre la *défaillance par arrêt* effective d'un processeur, et sa détection par tous les autres membres du groupe.
Δ_{rlb}	Le temps minimal séparant la *défaillance par arrêt* d'un processeur, de sa réinsertion dans le groupe ("rlb" = "restarting lower bound"). C'est le temps minimal que devra mettre un processeur pour redémarrer, avant de s'insérer et d'être considéré à nouveau correct.
ω	Le nombre maximal de tours de jeton consécutifs pendant lesquels l'interface réseau (entrée et sortie) d'un nœud peut être victime de fautes (omissions en réception ou en émission).

5.4.3 Variables locales à chaque nœud

Le tableau 3 présente les cinq variables locales à chaque nœud p : elles renseignent sur la configuration courante, l'état courant du nœud, la vision que p a du groupe, ou

que le groupe a de p. Les variables $View_p$, $recv_p[]$ et $join_p[]$ sont également appelées à être diffusées au cours du protocole de gestion de groupe.

Table 3 Les variables locales, internes à chacun des processeurs

Time $recv_p[1..n]$; *// Date d'émission du dernier message reçu par p de chacun des nœuds.*

Time $last\_received\_from\_us_p[1..n]$; *// Date d'émission du dernier message que le nœud i a déclaré avoir reçu de p, dans le dernier message que p a reçu de i.*

Time $join_p[1..n]$; *// Date à partir de laquelle un nœud qui a subi une* défaillance par arrêt *pourra être à nouveau considéré comme opérationnel.*

Integer $View_p$; *// Identificateur de la vue courante dans laquelle le nœud p évolue.*

List GMS_list_p; *// La liste des processeurs membres de la vue courante de p.*

$state_p \in \{\text{crashed,restarting,operational}\}$; *// L'état du nœud courant.*

5.4.4 L'ensemble des identificateurs de vue

Nous considérons que les identificateurs de vue sont membres de l'ensemble totalement ordonné $\{0,1,2,3,\cdots\}$.

L'identificateur de vue 0 a un sens particulier. Si cet identificateur vaut 0 pour un nœud p du système, cela signifie que p est en phase de redémarrage, et que la vue courante du groupe est pour lui encore indéterminée.

5.5 Initialisation des variables

Les variables locales à chaque nœud p sont initialisées à chaque redémarrage de p (*i.e.* après une *défaillance par arrêt*). L'algorithme 4 en décrit le déroulement.

Algorithme 4 La fonction d'initialisation (une seule fois après chaque arrêt): $gms\_restart()$

Ensure: Tous les autres processeurs sont vus comme non opérationnels.

Time $local\_time$ = $date\_actuelle()$;

2: $state_p = \text{restarting}$; *// Le nœud p est en phase de redémarrage.*

$View_p = 0$; *// p ne connaît pas encore la vue courante du système.*

4: $GMS\_list_p = \{\emptyset\}$; *// La vue du groupe de p est vide au démarrage.*

for $i = 1$ to n **do**

6: $recv_p[i] = -\infty$; *// p n'a reçu aucun message du nœud i.*

$last\_received\_from\_us_p[i] = -\infty$; *// Les autres nœuds n'ont reçu aucun message de p.*

8: $join_p[i] = \infty$; *// Le nœud i n'est pas en train de se joindre au groupe.*

end for

10: $last\_received\_from\_us_p[p] = recv[p] = local\_time$; *// Le nœud courant p est le plus vivace.*

$join_p[p] = local\_time + \Delta_{rlb}$; *// A partir de cette date, le nœud courant p se verra comme membre du système, et pourra commencer à émettre – ce report d'adhésion au reste du groupe assurera que les autres membres auront le temps de s'accorder pour déclarer le nœud p* crashé *avant de le voir dans le groupe à nouveau, et que p aura le temps de s'accorder avec les autres nœuds corrects avant de faire partie du groupe.*

En pratique, la valeur de ∞ est arbitraire à la condition qu'elle vérifie quand même $\infty > \max(\Delta_{lat}, \Delta_{rlb})$.

5.6 Émission d'un signe de vie

La phase d'émission des signes de vie doit garantir que deux messages consécutifs ne doivent pas être séparés par plus de δ_{send}. Dans le cas de HADES, et pour simplifier le test d'ordonnancement, on supposera que la tâche d'émission des signes de vie (détaillée en figure 5) est déclenchée périodiquement (*i.e.* la possession du jeton correspond à être dans un intervalle de temps donné, répété périodiquement). Ceci suppose néanmoins une synchronisation des horloges avec une précision connue. En effet, si tel n'est pas le cas, l'exclusivité d'accès au médium de diffusion n'est plus garantie, et les collisions de messages imprévisibles risquent de rendre le nombre d'omissions successives supérieur à ω.

Algorithme 5 L'émission d'un message de vie (appelée au moins une fois tous les δ_{send}) par un nœud p : $gms\_send()$

Time $local\_time$ = $date\_actuelle()$;
$last\_received\_from\_us_p[p] = recv_p[p] = local\_time$; *// Le nœud courant p est bien vivace.*
$bcast\_network([View_p, recv_p[1], \cdots, recv_p[n], join_p[1], \cdots, join_p[n]])$; *// p communique sa vision du groupe au reste du groupe.*

5.7 Réception d'un message de vie

Pour un processeur p quelconque, la tâche de réception (appelée au plus tard δ_{recv} après la réception du message par la couche réseau) déroule l'algorithme de la figure 15, qui appelle autant de fois la fonction de réception de la figure 7 qu'il y a de messages dans la queue de réception de la couche réseau.

Une fois tous les messages reçus traités, la composition du groupe est actualisée, c'est à dire que si elle a changé, l'identificateur de vue est incrémenté (par appel à la fonction $gms\_list()$ décrite dans le paragraphe suivant).

5.8 Établir la liste des membres corrects du groupe

Pour déterminer si un nœud i est membre du groupe, l'algorithme de la figure 8 est invoqué sur chaque processeur p.

La liste des membres corrects du groupe, appelée GMS_list_p, telle que chaque nœud p la voit, est actualisée comme le montre la figure 9. Il s'agit de la liste des identificateurs des nœuds corrects du groupe.

Si, depuis le dernier appel à cette fonction $gms\_list()$ d'actualisation, la composition du groupe a changé, l'identificateur $View_p$ de la vue courante est incrémenté : il y a eu changement de configuration, et ce changement est signalé à l'application (par l'invocation de la primitive $signal()$). Ce changement est perçu par tous les membres corrects du groupe à la même heure locale (voir la propriété 14 au paragraphe 6.13).

Note très importante. Afin que les changements de vue soient effectués aux mêmes instants (mêmes heures locales), il est nécessaire que la fonction d'actualisation de

Algorithme 6 La tâche de réception (périodique de période maximale δ_{recv}): $gms\_recv\_task()$

Time $local\_time = date\_actuelle()$;
if $state_p ==$ restarting **and** $local\_time \geq join_p[p]$ **then** *// La phase de redémarrage du nœud p touche à sa fin*
  $state_p =$ operational;
end if
for message [Processor $sender$, Integer $View_{sender}$, Time $t_1, \cdots$, Time t_n, Time $j_1, \cdots, j_n$] in $receive\_queue\_network_p$ **do**
  $gms\_recv(local\_time, state_p, sender, View_{sender}, t_1, \cdots, t_n, j_1, \cdots, j_n)$;
end for
if
$|\{sender \neq p$ such that $local\_time - recv_p[sender] > (f_{send}+1)(\delta_{send} + \Delta_{trans} + 2\epsilon + \delta_{recv})\}| \geq \lfloor \frac{n}{2} \rfloor$
or
$|\{sender \neq p$ such that $local\_time - recv_p[sender] < \Delta_{lat}$ **and** $local\_time - last\_received\_from\_us_p[sender] > \omega(\delta_{send} + \Delta_{trans} + 2\epsilon + \delta_{recv})\}| \geq \lfloor \frac{n}{2} \rfloor$
then *// Les autres nœuds sont perçus défaillants par p (défaillance permanente en réception, ou du commutateur central),* **ou** *les autres nœuds de la vue perçoient p défaillant (défaillance permanente en émission).*
  $state_p =$ crashed;
  signal(GMS_CRASH);
  $gms\_restart()$; *// p informe l'application qu'il a* crashé, *et redémarre.*
end if
if $state_p ==$ operational **then** *// La vue du groupe n'aura de sens pour p qu'une fois la phase de redémarrage terminée (*i.e. *une fois en état* opérationnel*).*
  $gms\_list(local\_time)$; *// On actualise la vue du groupe.*
end if

Algorithme 7 La réception d'un message invoquée par la tâche de réception ($gms\_recv\_task()$) : gms_recv(Time $local\_time$,$state_p \in$ {crashed,restarting,operational}, $Processor\ i$, $Integer\ View_i$, $Time\ t_1, \cdots, Time\ t_n$, $Time\ j_1, \cdots, Time\ j_n$)

if $state_p ==$ restarting **and** $View_i > View_p$ **then** *// Tant que p redémarre, p tente de récupérer le numéro de vue courant, si il est défini pour i*
2: $View_p = View_i$;
end if
4: **if** $join_p[i] < j_i$ **then** *// i s'apprête à redémarrer car on en reçoit un message avant qu'il soit réinséré dans le groupe. On note alors sa date de réinsertion dans le groupe prévue.*
$join_p[i] = j_i$; *// Le nœud i n'est plus considéré dans le groupe au moins jusqu'à cette date.*
6: $last\_received\_from\_us_p[i] = -\infty$; *// Le nœud i qui (re)joint le groupe ne mérite pas la confiance de p, et p doit considérer que i n'a rien reçu de p depuis une éternité.*
end if
8: **if** $recv_p[i] < t_i$ **then** *// Le dernier message que p avait reçu de i avait été envoyé avant celui que p est en train de traiter.*
$recv_p[i] = t_i$; *// Facultatif – voir "***" plus bas*
10: $last\_received\_from\_us_p[i] = t_p$; *// p mémorise comment le nœud i le perçoit.*
end if
12: **if** $local\_time \geq j_i$ **then** *// p ne fait confiance aux informations que i détient sur les autres nœuds du système* **que** *si i est dans l'état* opérationnel
for $j \in \{1, \cdots, n\}$ **do**
14: **if** $recv_p[j] < t_j$ **then**
$recv_p[j] = t_j$; *// *** Si le nœud i a reçu des nouvelles de j depuis plus récemment que p, p le note*
16: **end if**
if $j_j \geq local_time$ **and** $join_p[j] > j_j$ **then**
18: $join_p[j] = j_j$; *// Si le nœud i a déterminé une date de réinsertion de j plus imminente que celle de p, p le note*
end if
20: **end for**
end if

Algorithme 8 Déterminer si un nœud i est membre correct du groupe (par invocation) : $gms\_member(Time\ local\_time, Processor\ i)$

if $join_p[i] \leq local\_time$ and $local\_time \leq recv_p[i] + \Delta_{lat}$ **then** *// i n'est pas déclaré vu* crashé *ou en train de redémarrer, et a donné signe de vie dans les temps, et a reçu de nos nouvelles récemment (*i.e. *ne souffre pas de défaillances permanentes à la réception).*
2: return TRUE; *// le processeur i est membre du groupe.*
else
4: return FALSE; *// le processeur i est soit* crashé*, soit en instance de redémarrage, ou est sujet à des omissions permanentes en émission ou en réception (donc* crashé*).*
end if

Algorithme 9 Détermination de la liste GMS_list_p des membres corrects du groupe par le nœud p : $gms\_list(Time\ local\_time)$

List $GMS\_list_{old} \leftarrow GMS\_list_p$;
$GMS\_list_p \leftarrow \{\emptyset\}$;
for $i = 1$ to n **do**
  if $gms\_member(local\_time,i)$ **then**
    $GMS\_list_p = GMS\_list_p \cup \{i\}$; *// i est rajouté à la liste GMS_list_p des processeurs corrects.*
  end if
end for
if $GMS\_list_p \neq GMS\_list_{old}$ **then** *// La vue a changé.*
  $View_p = View_p + 1$;
  signal(CONFIG_CHANGE); *// Signaler à l'application le changement de vue.*
end if

la composition du groupe soit appelée à la même heure locale pour tous les nœuds. Comme cette fonction est invoquée par la tâche de réception (Algorithme 15), il est donc primordial que la tâche de réception soit appelée à la même heure locale sur tous les nœuds (l'heure d'invocation de cette tâche est un paramètre du système, puisqu'elle est directement liée à la périodicité de cette tâche de réception).

5.9 Synoptique de fonctionnement

Considérons un nœud q, jusque là *défaillant par arrêt*, et qui redémarre en t.

5.9.1 Phase de (re)démarrage

Avant de pouvoir émettre quoi que ce soit, le nœud q désirant redémarrer doit réussir à savoir quand il sera autorisé à émettre. Dans le cas d'un jeton modélisé par une fenêtre temporelle autorisant l'émission d'un message (TDMA), q doit donc attendre d'avoir synchronisé son horloge locale avec celle des autres processeurs, avant de faire quoi que ce soit d'autre.

Une fois les horloges synchronisées, le nœud est en mesure d'émettre lorsqu'il bénéficie de la fenêtre temporelle qui lui est associée statiquement à la constitution du système.

Ensuite, q est autorisé à invoquer le protocole de gestion de groupe, et va donc commencer, à la date t, par initialiser toutes ses variables internes relatives à la gestion de groupe, conformément à l'algorithme 4.

Pendant la période de redémarrage, le nœud q :

- envoie régulièrement des messages de vie (au moins un tous les δ_{send}) conformément à la fonction d'émission correspondant à l'algorithme 5, afin de soumettre aux autres nœuds du groupe sa date de réinsertion dans le groupe;
- se tient au courant de la vue courante du groupe (en particulier, p actualise l'identificateur de la vue courante $View_q$), grâce à la fonction de réception correspondant à l'algorithme 15.

La phase de redémarrage doit durer suffisamment longtemps pour que q :

- puisse prendre connaissance des signes de vie de tous les autres nœuds corrects, afin d'être en accord avec leur vision du groupe avant le passage à l'état *operationnel*;
- puisse diffuser son désir d'adhérer au groupe, et laisser le temps aux autres nœuds corrects de tous se tenir au courant de la date de réinsertion de q dans le groupe.

Finalement, au bout de Δ_{rlb}, à la fois tous les nœuds corrects sont au courant que p fera partie du groupe en $t + \Delta_{rlb}$ (*i.e.* q sera opérationnel), et q disposera de la même vue que tous les autres nœuds corrects.

5.9.2 Fonctionnement en état *opérationnel*

Dans cet état, un nœud q doit réagir à trois événements :

- sa propre défaillance;
- la *défaillance par arrêt* d'un autre nœud;
- la réintroduction d'un autre nœud dans le groupe.

La tâche essentielle est la tâche de réception (algorithme 15) qui a pour rôle d'actualiser les variables locales à q, en fonction des valeurs que les autres nœuds lui communiquent. Le nœud q actualise, par échange avec tous les nœuds du système, pour chacun des autres nœuds i du système, et *via* la fonction de réception (algorithme 7) :

D1. La date à laquelle i a envoyé son dernier message;

D2. La date à partir de laquelle i est (ou a été) appelé à appartenir à la vue du groupe.

Le dépistage de la propre défaillance d'un nœud q (sous forme d'une défaillance permanente en émission) se fait directement à partir de la première donnée (D1) : quand tous les autres nœuds n'ont pas vu q depuis trop longtemps, c'est que q n'arrive pas à se faire entendre, et donc qu'il souffre de défaillances en émission.

De plus, le simple fait que tous les nœuds échangent la première donnée (D1), c'est à dire la date à laquelle tous les nœuds ont été vus pour la dernière fois, permet d'avoir un retour sur la façon dont les autres nœuds perçoivent q. Disposer de cette donnée permet de dépister "la propre défaillance de q", sous la forme d'une défaillance permanente en émission (la ligne 8 de l'algorithme 15).

La défaillance d'un autre nœud provient également de la première information (D1). En effet, un nœud est déclaré défaillant par q (et en même temps par tous les autres nœuds corrects, comme nous le montrerons dans le chapitre suivant) lorsque q n'en a pas reçu de nouvelles depuis trop longtemps (ligne 1 de l'algorithme 8), c'est à dire quand la date d'émission du dernier message que q en a reçu est trop vieux.

Finalement, la deuxième information (D2) permet à q de prévoir quand un autre nœud, qui redémarre, pourra être réintroduit dans le groupe (ligne 4 de l'algorithme 7).

Chacun de ces trois évènements évoqués ci-dessus mène donc respectivement :

- à la *défaillance par arrêt* de q (ligne 9 de l'algorithme 15);
- à un changement de vue (*via* la mise à jour de la vue du groupe, grâce à l'algorithme 9), avec au moins élimination du processeur *défaillant par arrêt*;
- à un changement de vue (*via* la mise à jour de la vue du groupe, grâce à l'algorithme 9), avec au moins intégration du processeur qui le désire à la date qu'il propose.

Indépendamment de la réception, la tâche d'émission diffuse la vision que q a de tout le groupe (c'est à dire les deux informations précédemment mentionnées : les dates d'émission des derniers messages reçus, et les dates de réinsertion dans le groupe), y compris de lui-même, ce qui fait donc office de signe de vie de q.

6 Propriétés du protocole de gestion de groupe

6.1 Problématique

Le protocole conduit tous les nœuds à diffuser l'ensemble des données dont ils disposent, afin que de proche en proche, les signes de vie (vecteur $recv_p[]$), et les dates de réinsertion (vecteur $join_p[]$) se répandent sur l'ensemble du système.

La propriété fondamentale à prouver est la propriété d'accord (correspondant à la propriété 14 de ce document), c'est à dire que :

> "Tous les nœuds corrects voient les mêmes changements dans la composition du groupe, à la même heure locale".

L'accord rigoureux voudrait que ces changements soient perçus à la même heure temps-réel. Nous nous approchons de cet idéal grâce à la synchronisation des horloges.

6.2 Message de vie

Dans les propriétés qui suivent, nous abandonnerons le terme de "signe de vie" au profit de "message de vie", plus riche en information.

Un "signe de vie" correspond à la simple communication d'une donnée, dont le contenu n'a pas d'importance, d'un nœud à un autre, à une date donnée. C'est en quelque sorte un *stimulus*.

Le "message de vie" est lui aussi lié à une communication d'un nœud à un autre à une date donnée, mais c'est son contenu qui est intéressant. C'est en quelque sorte un *signifiant*. Le *message de vie* qu'un nœud q reçoit d'un nœud p correspond alors au triplet :

1. date de l'émission par p du dernier message que q a reçu de p. C'est ce qui correspond à l'élément $recv_q[p]$ du nœud q;
2. date à partir de laquelle le nœud q considère p comme opérationnel, c'est à dire comme membre correct du groupe. C'est ce qui correspond à l'élément $join_q[p]$ du nœud q;
3. identificateur de la vue courante $View_p$ perçue par le nœud p.

Dans le protocole présenté, tous les signes de vie sont en fait des messages de vie.

La *réception* d'un tel *message de vie* de p correspond à recevoir cette information *directement* de p, ou *indirectement*, c'est à dire relayée par différents nœuds du système.

6.3 Suicide par manquement aux hypothèses sur la réception

Avant de prouver la propriété d'accord, nous introduisons deux propriétés qui régissent le comportement d'un nœud lorsque les hypothèses de défaillance sont violées.

Propriété 1 *Un nœud p qui ne reçoit pas de messages de la majorité des nœuds (*i.e. $\lfloor \frac{n}{2} \rfloor$*) de la vue du groupe courante, pendant plus de $f_{send}+1$ tours du jeton consécutifs, se suicide.*

Preuve : En au plus $f_{send} + 1$ tours du jeton, tous les nœuds corrects (en nombre $\geq \lfloor \frac{n}{2} \rfloor + 1$) ont émis correctement un message. Si le nœud p considéré n'a pas reçu de messages d'une majorité de nœuds (*i.e.* $\lfloor \frac{n}{2} \rfloor$) de la vue du groupe courante, il y a partitionnement (car p est isolé), et donc contradiction de l'hypothèse de défaillance H2. Dans ce cas, p doit s'arrêter. ♢

Le protocole présente exactement cette propriété quand $\omega \geq f_{send} + 1$.

6.4 Suicide par manquement aux hypothèses sur l'émission

Propriété 2 *Si pendant ω tours du jeton, un nœud p n'arrive pas à se faire entendre par une majorité des nœuds (*i.e. $\lfloor \frac{n}{2} \rfloor$*) de la vue du groupe courante, alors p se suicide.*

Preuve : Provient directement de l'hypothèse de défaillance H2. ♢

Propriété 3 *Un nœud p correct arrive à transmettre au moins un message de vie à un nœud r correct en au plus $f_{send} + 1$ tours du jeton, soit $(f_{send} + 1)\big(\delta_{send} + \Delta_{trans} + 2\epsilon + \delta_{recv}\big)$ unités de temps.*

Preuve : Dans le pire des cas, p souffre d'omissions en émission pendant f_{send} tours consécutifs du jeton. Avant que p puisse émettre son message de vie, il peut donc s'écouler jusqu'à $f_{send} + 1$ tours du jeton.

Puisque $n - f_{crash} > f_{recv}$, alors, parmi les processeurs de la vue du groupe, au moins un nœud r a reçu le message de vie de p, puisqu'il n'y a pas de défaillances du médium de communication (elles sont reportées aux interfaces réseau des nœuds – voir 4.1). Donc, dans le pire des cas, p arrive à transmettre un message de vie à un nœud r, en au plus $(f_{send} + 1)\big(\delta_{send} + \Delta_{trans} + 2\epsilon + \delta_{recv}\big)$ unités de temps. ♢

Le protocole présente cette propriété quand l'hypothèse 2 suivante est vérifiée :

Hypothèse 2 $\omega \geq f_{send} + 1$.

6.5 Non trivialité

D'après l'hypothèse de défaillance H2 (page 16) sur l'existence de la partition majoritaire de nœuds corrects, la propriété suivante est garantie :

Propriété 4 *Le protocole n'éclate pas le groupe en plusieurs amas distincts.*

Preuve : Si c'était le cas, les nœuds ne recevraient pas de nouvelles d'une majorité de nœuds, ce qui entraînerait, d'après les propriétés 1 et 2, la *défaillance par arrêt* de tels nœuds. De tels amas minoritaires seraient ainsi détruits.

♢

6.6 Propriété fondamentale de la partition majoritaire

Grâce aux propriétés 1 et 2, nous avons la propriété suivante :

Propriété 5 *En ω tours de jeton, un nœud p arrive à faire parvenir un de ses messages de vie à au moins $\lfloor \frac{n}{2} \rfloor$ autres nœuds (*i.e. *la majorité) de la vue du groupe, et à recevoir des messages de vie émis pendant cette période par au moins $\lfloor \frac{n}{2} \rfloor$ autres nœuds de la vue du groupe.*

Preuve : Provient directement des deux propriétés précédentes. ♢

6.7 Diffusion assurée des messages de vie

Disposant de la propriété 5 précédente, la propriété suivante est montrée :

Propriété 6 *Si en t, le processeur d'un nœud p tente d'émettre un message, alors :*

- *soit en $t + 2\omega$ tours du jeton tous les nœuds de la vue ont reçu chacun au moins un message de vie de p émis pendant $[t,t + 2\omega$ tours du jeton$]$;*
- *soit en $t + 2\omega$ tours du jeton aucun des nœuds de la vue n'a reçu de message de vie de p durant $[t,t + 2\omega$ tours du jeton$]$, et de plus p a été victime d'une* défaillance par arrêt *pendant $[t,t + \big((f_{send} + 1)$ tours du jeton$\big)]$.*

Preuve : Considérons un nœud $q \neq p$ appartenant à la vue du groupe au moins pendant $[t,t + 2\omega$ tours du jeton$]$. Pendant les ω premiers tours du jeton, deux cas peuvent se produire :

- soit une majorité M_1 (*i.e.* $\lfloor \frac{n}{2} \rfloor$) des nœuds de la vue du groupe a reçu au moins un message de vie de p émis pendant $[t,t + \omega$ tours$]$ (d'après la propriété 5);
- soit aucun nœud de la vue n'en a reçu. C'est le cas si p est victime d'une *défaillance par arrêt* avant de pouvoir émettre *correctement* un message, c'est à dire, s'il est victime d'une *défaillance par arrêt* avant $t+(f_{send}+1)$ tours du jeton.

Le deuxième cas correspond à la deuxième possibilité énoncée par la propriété.

Sinon (*i.e.* premier cas) :

- soit q fait partie de la majorité M_1, auquel cas q reçoit effectivement un message de vie de p pendant $[t,t + \omega$ tours$] \subset [t,t + 2\omega$ tours$]$;
- soit q ne fait pas partie de cette majorité.

Dans ce dernier cas, q peut recevoir, au cours des ω tours du jeton suivants, directement un message de vie de p émis pendant $[t,t + 2\omega$ tours$]$, ou sinon, d'après la propriété 5, il reçoit des messages de vie d'au moins une majorité M_2 (*i.e.* $\lfloor \frac{n}{2} \rfloor$) des autres nœuds corrects. Or ces deux majorités M_1 et M_2 ont forcément un élément commun, donc un message de vie de p émis pendant $[t,t + \omega$ tours$]$ parviendra à q pendant $[t,t + 2\omega$ tours$]$.

Ainsi, de toutes façons, soit q ne reçoit pas de message de vie de p pendant $[t,t + 2\omega$ tours$]$, et alors aucun autre nœud correct pendant cet intervalle n'en perçoit et p a été victime d'une *défaillance par arrêt* durant $[t,t + (f_{send} + 1$ tours$)]$. Soit q a reçu au moins un message de vie émis par p durant $[t,t + 2\omega$ tours$]$, et tous les autres nœuds corrects durant cet intervalle aussi. Ceci termine la preuve de la propriété. $\Diamond$

Note importante. Cette propriété n'est pas suffisante pour assurer l'accord des décisions de changement de groupe, puisqu'elle n'assure pas que tous les nœuds corrects auront reçu les mêmes messages de vie au bout d'un temps donné.

6.8 Accord sur la réception des messages de vie

6.8.1 Énoncés

Soit la valeur suivante de $\Delta_{diffuse}$, la latence de diffusion d'un message de vie parmi tous les nœuds correct :

Hypothèse 3 *$\Delta_{diffuse}$ correspond à au moins $(n + 1)\omega$ tours du jeton, c'est à dire à $\Delta_{diffuse} \geq (n + 1)\big(\omega\delta_{send} + \Delta_{trans} + 2\epsilon + \delta_{recv}\big)$.*

Sous l'hypothèse 3, la propriété suivante est vérifiée :

Propriété 7 *Tout message de vie correctement émis par un nœud p en t est :*

- *soit reçu, au plus tard en $t + \Delta_{diffuse}$, par tous les nœuds corrects pendant $[t,t + \Delta_{diffuse}]$;*
- *soit reçu par aucun des nœuds corrects pendant $[t,t + \Delta_{diffuse}]$.*

La démonstration de cette propriété essentielle est rédigée dans le prochain paragraphe.

On pourra en déduire immédiatement le corollaire 8 :

Corollaire 8 *Soit r un nœud de la vue du groupe ayant reçu un message de vie d'un nœud p de la vue émis en t. Si r est correct en $t+\Delta_{diffuse}$ alors tous les autres processeurs de la vue pendant $[t,t + \Delta_{diffuse}]$ ont reçu* **ce** *même message de vie (directement ou indirectement* via *un autre nœud, et éventuellement caché*[10] *par un autre message de vie postérieur) avant $t + \Delta_{diffuse}$.*

6.8.2 Preuve de la propriété 7

La démonstration repose sur la diffusion du message de vie de p passant progressivement de nœud en nœud.

Avant de démontrer la propriété 7, nous introduisons la terminologie et la notation suivantes.

Terminologie et notation préliminaires. La difficulté réside dans le fait qu'un message de vie contient uniquement l'information la plus récente concernant les nœuds de la vue, et non l'historique de tous les messages de vie diffusés depuis le démarrage du système. Ainsi, lorsque deux messages de vie $m_{p,t}$ et $m_{p,t'}$ diffusés par un nœud p aux instant t et t' respectivement, avec t et t' appartenant à la même fenêtre temporelle d'émission de p, sont reçus par un nœud r, seul $m_{p,t'}$ est diffusé par r vers le groupe. Ainsi, lorsqu'un nœud q reçoit le message $m_{p,t'}$, nous disons que l'on a une *suite de processeurs* (ici : $p \rightarrow r \rightarrow q$) *qui transmet la connaissance que p a émis un message de vie en t*, et nous noterons : $p \rightsquigarrow_{p,t} q$.

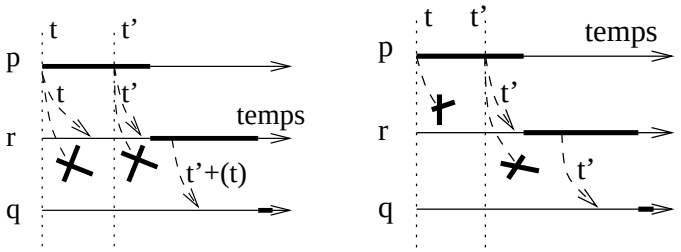
Remarquons que, par transitivité, si $p \rightsquigarrow_{p,t} r \rightsquigarrow_{p,t} q$, alors $p \rightsquigarrow_{p,t} q$.

Cette notation peut être étendue à la notation suivante : $p \rightsquigarrow_{p,[t_1,t_2]} q$, qui signifie qu'il existe une chaîne de processeurs qui transmet à q la connaissance que p a émis au moins un message de vie pendant $[t_1,t_2]$.

Le sénario précédent ne s'applique pas dans le cas suivant. Supposons comme précédemment que p diffuse les messages $m_{p,t}$ et $m_{p,t'}$ dans une même fenêtre temporelle d'émission. Cependant, suite à une omission en émission par p, ou en réception par r, seul le message $m_{p,t'}$ parvienne à r. Comme précédemment, r ne diffuse que $m_{p,t'}$ vers le groupe. Sur réception de ce message, q ne peut pas faire la différence avec le message $m_{p,t'}$ reçu dans le sénario précédent. Cependant, nous n'avons pas la relation $p \rightsquigarrow_{t,p} q$ (mais uniquement la relation $p \rightsquigarrow_{p,t'} q$). Nous noterons cette absence de relation par $p \not\rightsquigarrow_{p,t} q$.

La figure suivante illustre ceci :

10. Voir la définition de "avoir la connaissance qu'un message de vie a été émis en t" dans la démonstration suivante.



$p \rightsquigarrow_{p,t} r$ et $r \rightsquigarrow_{p,t} q$ donc $p \rightsquigarrow_{p,t} q$ $\quad$ $p \not\rightsquigarrow_{p,t} r$, donc $p \not\rightsquigarrow_{p,t} q$

Il est important de noter que cette formulation et la notation associée n'ont de signification que d'un point de vue théorique, pour établir et valider les propriétés du protocole. En effet, en pratique, un nœud q qui ne reçoit que $m_{p,t'}$ d'un nœud r n'a aucun moyen de savoir si ce message apporte également la connaissance que p a émis un message en t.

La notation précédente nous permet de reformuler la propriété 7 à démontrer :

> Si p tente d'émettre un message de vie en t, alors si $\exists q \neq p$ dans la vue du groupe tel que $p \rightsquigarrow_{p,t} q$, alors $\forall r \neq p$ dans la vue du groupe, au plus tard en $t + \Delta_{diffuse}$, on a $p \rightsquigarrow_{p,t} r$.

Remarque : en utilisant cette notation, la propriété 6 précédemment montrée se réécrit comme suit :

> Si p tente d'émettre un message de vie en t, alors si $\exists q \neq p$ correct tel que $p \rightsquigarrow_{p,[t,t+2\omega \text{ tours du jeton}]} q$, alors $\forall r \neq p$ correct, au plus tard en $t + 2\omega$ tours du jeton, on a $p \rightsquigarrow_{p,[t,t+2\omega \text{ tours du jeton}]} r$.

Preuve de la propriété 7. En t, le processeur correct p tente d'émettre un message de vie. Considérons un nœud $q \neq p$ de la vue du groupe (*i.e.* correct), et déterminons le chemin que peut suivre la *connaissance que p a émis un message de vie en t* jusqu'à q.

Amorçage de la démonstration. En supposant l'hypothèse 2, après t, cinq cas peuvent se produire :

1. Le message de vie n'a pas été correctement émis. Dans ce cas, aucun autre nœud ne recevra la *connaissance que p a émis un message de vie en t*.
2. Il existe un nœud correct $r_1 \neq p$ tel que r_1 reçoit le message de vie de p émis en t, c'est à dire $p \rightsquigarrow_{p,t} r_1$. Alors trois possibilités se présentent :
   (a) $r_1 = q$;
   (b) Il existe un nœud $s \neq r_1$ auquel r_1 arrive à faire parvenir la connaissance que p a émis un message de vie en t, en au plus ω tours du jeton, c'est à dire $r_1 \rightsquigarrow_{p,t} s$;
   (c) Tous les nœuds qui ont reçu le message de vie de p (dont r_1 fait partie) sont victime d'une *défaillance par arrêt* avant de retransmettre le message de vie de p à quiconque. Dans ce cas, aucun autre nœud ne recevra la *connaissance que p a émis un message de vie en t*.

Ainsi, après les 2ω premiers tours du jeton (au plus tard), on se retrouve dans l'un des quatre états suivants :

- q a reçu un message de vie de p émis pendant les ω premiers tours du jeton (directement : cas 2.a; ou indirectement *via* r_1 : cas 2.b avec $s = q$). Alors on a

$p \rightsquigarrow_{p,t} q$ (ou $p \rightsquigarrow_{p,t} r_1 \rightsquigarrow_{p,t} q$, ce qui revient au même);

- Un nœud $r_2 \notin \{p,r_1,q\}$, a reçu la *connaissance que p a émis un message de vie en t*, pendant la deuxième série de ω tours (cas 2.b avec $s = r_2$), c'est à dire : $p \rightsquigarrow_{p,t} r_1 \rightsquigarrow_{p,t} r_2$;
- Seuls des nœuds r_2 avec $r_2 \notin \{q,r_1\}$ mais $r_2 \in \{p,r_1\}$, soit $r_2 = p$, ont reçu la *connaissance que p a émis un message de vie en t* (cas 2.b avec $s = r_2 \in \{p,r_1\}$). C'est à dire $p \rightsquigarrow_{p,t} r_1 \rightsquigarrow_{p,t} p$
- le message de vie de p n'a pas été correctement émis en t, et aucun nœud correct n'a reçu de message de vie de p datant de t (cas 1) ou tous les nœud r_1 ont été victimes d'une *défaillance par arrêt* avant de transmettre la *connaissance que p a émis un message de vie en t* (cas 2.c).

Les pires schémas, au sens où ils mènent aux temps pire cas de propagation du message de vie de p, émis en t, à q, ou à la non propagation, apparaissent lorsque :

- on est dans le troisième cas ci-dessus : il y a un cycle entre p et r_1 dans la propagation du message de vie;
- il existe un unique r_2 qui a reçu la *connaissance que p a émis un message de vie en t*, et p et r_1 ont été victime d'une *défaillance par arrêt*. Il y a dans ce cas une forme de chaîne à brin unique qui transporte la *connaissance que p a émis un message de vie en t*, en suivant tous les nœuds corrects, qui sont victimes d'une *défaillance par arrêt* juste après la retransmission du message de vie.

Dans tous les autres cas, soit on a $p \rightsquigarrow_{p,t} q$, soit ni q, ni aucun autre nœud n'a reçu ni ne recevra la *connaissance que p a émis un message de vie en t*.

Finalement, quand la *connaissance que p a émis un message de vie en t* peut se propager au sein du groupe (*i.e.* tant que au moins un nœud non *défaillant par arrêt* en dispose et l'émet *correctement*), deux cas de figure peuvent se présenter :

- Une chaîne à brin unique;
- Une série de cycles.

Cœur de la démonstration. Lors des ω tours suivants, en suivant ce principe de démonstration, on sait que :

1. Si on avait un cycle dans la transmission de la *connaissance que p a émis un message de vie en t*, dans le sens où cette connaissance n'a pu être retransmise, lors des derniers ω tours, qu'à un nœud qui faisait déjà partie de la chaîne de transmission, alors :
   - Le cycle ne recouvre pas une majorité de nœuds corrects, alors *nécessairement*, pendant les ω tours du jeton suivants, $\exists r_k$, **nœud correct qui ne fait pas partie de ce cycle** et qui se rajoute à la chaîne, brisant le cycle.
     En effet, si un tel nœud n'existait pas, aucun des nœuds du cycle n'arriverait à émettre un message de vie en dehors du cycle. Donc, en l'espace de ω tours du jeton, aucun de ces nœuds n'arriverait à transmettre un message de vie à la majorité (*i.e.* à plus de $\lfloor \frac{n}{2} \rfloor$) des nœuds corrects, ce qui contredirait la propriété 5.

     – Si $r_k = q$, alors q a reçu la *connaissance que p a émis un message de vie en t*, et on arrête la démonstration ici;
     – Sinon, on reprend cette partie de la démonstration.

- Le cycle contient une majorité de nœuds (*i.e.* est de taille supérieure ou égale à $\lfloor \frac{n}{2} \rfloor + 1$), et alors, on est sûr que q fera partie de la chaîne au cours des ω tours du jeton suivants.
  En effet, q, d'après la propriété 5, recevra, au cours des ω tours du jeton suivants, les messages de message de vie de la majorité (*i.e.* au moins $\lfloor \frac{n}{2} \rfloor$) des nœuds. Forcément, entre la majorité des nœuds qui fait partie de la chaîne de transmission de la *connaissance que p a émis un message de vie en t*, et la majorité des nœuds dont q recevra un message de vie, on est assuré qu'il y a un élément commun r_k. On a alors $p \rightsquigarrow_{p,t} r_k$ et $r_k \rightsquigarrow_{p,t} q$, d'où $p \rightsquigarrow_{p,t} q$.

2. Si au cours des ω tours du jeton précédents, il y a une chaîne de transmission de la *connaissance que p a émis un message de vie en t*, alors :
   - au cours des ω tours suivants, seuls des nœuds de la chaîne de transmission de la *connaissance que p a émis un message de vie en t* reçoivent cette connaissance, formant alors un cycle. On se retrouve alors dans le cas 1.
   - au cours des ω tours suivants, $\exists r_k$ qui ne faisait pas partie de cette chaîne de transmission, qui reçoit la *connaissance que p a émis un message de vie en t*. Il n'y dans ce cas pas de cycle, et la chaîne de transmission s'est allongée, augmentée de un nœud.
     - Si $r_k = q$, alors q a reçu la *connaissance que p a émis un message de vie en t*, et on arrête la démonstration ici;
     - Sinon, on reprend cette partie de la démonstration.
3. Tous les nœuds qui disposaient de la *connaissance que p a émis un message de vie en t* (dont p) ont été victimes d'une *défaillance par arrêt*, et donc, ni q, ni aucun autre nœud resté correct ne recevra cette connaissance.

Donc, dans le pire des cas de la transmission de la *connaissance que p a émis un message de vie en t*, il est nécessaire d'arriver à former une chaîne (pouvant avoir contenu des cycles) contenant la majorité (*i.e.* de au moins $\lfloor \frac{n}{2} \rfloor + 1$ nœuds) des nœuds (p compris). En effet, une fois ce quorum atteint, on est assuré que, soit que q recevra cette connaissance, soit que, ni q, ni aucun autre processeur correct ne la recevra.

Synthèse. Au total, donc, il faut allonger la chaîne jusqu'à $\lfloor \frac{n}{2} \rfloor + 1$ nœuds, p compris, soit $\lfloor \frac{n}{2} \rfloor$ "étapes de démonstration" comme celle qui précède, et qui peuvent aller jusqu'à prendre 2ω tours de jeton chacune (en cas de formation de cycle). Ensuite, encore ω tours du jeton sont nécessaires pour que ce quorum transmette cette connaissance à q.

Donc, au total, la chaîne de transmission de la *connaissance que p a émis un message de vie en t* de p à q, si elle existe, prendra $\big((2\lfloor \frac{n}{2} \rfloor) + 1\big)\omega \leq (n+1)\omega$ tours du jeton à se créer. Si une telle chaîne n'existe pas, elle n'existe pas non plus pour aucun autre nœud.

Donc, en $\Delta_{diffuse} \geq (n+1)\big(\omega\delta_{send} + \Delta_{trans} + 2\epsilon + \delta_{recv}\big)$, soit tous, soit aucun des nœuds q corrects pendant $[t, t + \Delta_{diffuse}]$ sont tels que $p \rightsquigarrow_{p,t} q$.

$\diamondsuit$

6.9 Vivacité

Des hypothèses sur Δ_{lat}, et des propriétés précédentes, on démontre la propriété de vivacité 9 suivante :

Propriété 9 *Si un nœud q suspecte qu'un autre nœud p est* défaillant par arrêt, *alors :*

- *soit p se suicide ou est défaillant par arrêt;*
- *soit q se suicide.*

Preuve : Si q suspecte p de défaillance, c'est qu'il n'en a pas reçu de message de vie depuis plus de Δ_{lat}.

- Si q a reçu des messages de vie d'une majorité de nœuds dans la vue depuis Δ_{lat}, alors aucun des processeurs de cette majorité n'a reçu de message de vie de p depuis Δ_{lat}. Sinon, il y a contradiction du corollaire 8. Dans ce cas :
  - Soit le processeur du nœud p est défaillant par arrêt;
  - Soit le processeur du nœud p est correct, mais :
    - Soit p a perçu, durant les derniers Δ_{lat}, des messages de vie d'une majorité de processeurs dans la vue, et il se rend compte que moins d'une majorité de processeurs le voient correct, et donc p se suicide à cause d'un manquement aux hypothèses en émission (propriété 2);
    - Soit il n'a pas perçu de message de vie d'une telle majorité de processeurs, et il se suicide également à cause d'un manquement aux hypothèses en réception (propriété 1).

  Donc, dans tous les cas, p se suicide.
- Sinon, q n'a pas reçu de messages de vie d'une majorité de processeurs dans la vue, ce qui correspond à un manquement aux hypothèses en réception, et donc q se suicide (propriété 1).

♢

6.10 Accord sur la date de *défaillance par arrêt* d'un nœud

Sous l'hypothèse :

Hypothèse 4 $\Delta_{lat} \geq \Delta_{diffuse} + \omega$ tours du jeton, *soit* $\Delta_{lat} \geq (n+2)\big(\omega\delta_{send} + \Delta_{trans} + 2\epsilon + \delta_{recv}\big)$.

On a la propriété suivante :

Propriété 10 *Si, au temps t_q local à un nœud q, le nœud q dans la vue du groupe et correct depuis plus de Δ_{lat} n'a pas reçu d'un nœud p de message de vie depuis Δ_{lat}, alors p est* défaillant par arrêt, *et tous les nœuds r dans la vue du groupe depuis au moins Δ_{lat} excluent p de la vue du groupe au temps local $t_r = t_q$.*

Preuve : Notons l_q la date d'émission du dernier message de vie que q a reçu de p. Nous nommerons abusivement cette date "le dernier message de vie reçu de p par q". Soit $t_q = l_q + \Delta_{lat}$.

Première partie : détection de la défaillance de p. À t_q, q n'a reçu aucun message de vie de p depuis Δ_{lat}. Or, puisque $\Delta_{lat} \geq 2\omega$ tours du jeton [11], la propriété 6 assure qu'aucun autre nœud correct depuis au moins Δ_{lat} n'aura également reçu de message de vie de p depuis 2ω tours du jeton, **et que p a été victime d'une *défaillance par arrêt*** pendant $[l_q, l_q + \omega \text{ tours}]$ (puisque $\omega \geq f_{send} + 1$).

11. L'hypothèse d'existence d'une partition majoritaire assure que $0 \leq f_{crash} < \lfloor \frac{n}{2} \rfloor$, ce qui entraîne que nécessairement $n \geq 2$, et donc que 2ω tours du jeton $\leq (n+1)\omega$ tours du jeton (*i.e.* $\Delta_{diffuse}$).

Deuxième partie : le message de vie l_q est le dernier message de vie de p reçu par tous les nœuds corrects. Supposons qu'il existe un nœud $r \notin \{q,p\}$ tel que le dernier message de vie que r a reçu de p est l_r avec $l_r > l_q$. Nécessairement, q aurait reçu l_r au plus tard en $l_r + \Delta_{diffuse} \geq l_q + \omega\ tours + \Delta_{diffuse} \geq l_q + \Delta_{lat}$, soit en t_q (voir la propriété 7 précédente). Donc, en t_q, q aurait reçu le message de vie l_r de p, avec $l_r > l_q$. Ceci contredit donc qu'en t_q, le dernier message de vie que q a reçu de p est le message diffusé en l_q.

D'autre part, en $t_q = l_q + \Delta_{diffuse} \leq l_q + \Delta_{lat}$ tous les autres nœuds corrects auront reçu l_q (propriété 8, sachant qu'au moins q, encore correct, a reçu ce message de vie).

Ce message l_q est donc le dernier message de vie de p reçu par tout nœud correct r (soit $l_r = l_q$), qui décidera que p est *défaillant par arrêt* en $t_r = l_r + \Delta_{lat} = l_q + \Delta_{lat} = t_q$.

En t_q, tous les nœuds r de la vue $View_r$ du groupe excluent p de cette vue. Une nouvelle vue $View_r + 1$ est formée.

♢

6.11 Accord sur l'insertion d'un nœud – propriété de terminaison

La propriété suivante garantit que tous les nœuds appartenant à la vue du groupe courante introduisent dans cette vue du groupe, tous au même instant, un nœud qui (re)démarre.

Propriété 11 *Supposons qu'au temps t, un nœud p passe de l'état* défaillant par arrêt *à l'état* redémarrage*, et tente d'émettre*[12] *son premier message de vie. Alors, pour tous les nœuds q appartenant à la vue du groupe courante pendant au moins $[t,t + 2\omega$ tours du jeton$]$, q voit le nœud p comme correct à partir de Δ_{rlb}, et donc l'intégre au groupe des nœuds corrects à l'heure locale $t + \Delta_{rlb}$.*

Ainsi, sous l'hypothèse 5 suivante, on en déduit immédiatement la propriété 12 suivante.

Hypothèse 5 $\Delta_{rlb} \geq 2\omega$ tours du jeton.

Corollaire 12 *Pour tous les processeurs q corrects pendant $[t,t + 2\omega$ tours du jeton$]$, la décision d'intégration du nœud p dans le groupe est prise au même instant $t + \Delta_{rlb}$.*

Preuve : Tous les nœuds q corrects pendant 2ω tours du jeton ont reçu un message de vie de p (propriété 6), donnant entre autres la date de réinsertion de p dans le groupe (élément j_p d'un tel message), telle que prévue par p, *i.e.* $t + \Delta_{rlb}$. ♢

6.12 Redémarrage d'un nœud

Le problème est qu'un nœud qui redémarre ne fait pas partie de la catégorie des nœuds qui n'ont pas été *défaillants par arrêt* depuis au moins Δ_{lat}, et donc aucune des propriétés précédente n'est respectée. Il faut donc accorder au nœud qui redémarre un intervalle de temps de "mise à niveau", nécessaire pour qu'il puisse connaître la composition du groupe, et pour que tout le monde le voie correct, et qu'il y ait accord sur toutes ces visions.

12. Ce qui ne veut pas dire que l'émission va être correcte.

Il s'agit de montrer sous quelles hypothèses les nœuds qui redémarrent auront connaissance de la vue du groupe. L'objectif est donc de déterminer combien de temps au minimum doit durer la phase de redémarrage d'un nœud, c'est à dire de donner une minoration de Δ_{rlb}.

Propriété 13 *Un nœud qui redémarre est dans l'état* opérationnel *dès qu'il est en accord avec tous les autres processeurs corrects, c'est à dire au bout de Δ_{lat} après l'instant de son redémarrage.*

Ceci implique $\Delta_{rlb} > \Delta_{lat}$.

Preuve : Toutes les propriétés précédentes sur l'accord s'appliquant aux nœuds corrects, s'appliquent pour des nœuds corrects depuis plus de Δ_{lat}. Donc, nécessairement, avant de passer membre du groupe, c'est à dire avant de faire partie de la liste des nœuds parmi lesquels l'accord sur les changements de configuration doit être garanti (propriétés 10 et 11), un nœud qui redémarre en t ne doit pas passer dans l'état *opérationnel* avant $t + \Delta_{lat}$.

Donc nécessairement, $\Delta_{rlb} > \Delta_{lat}$, et après Δ_{lat} après l'instant de son redémarrage, l'accord sur les changements de configuration est garanti. $\Diamond$

6.13 Accord sur la composition du groupe

Grâce à toutes les hypothèses précédentes, on a la propriété suivante :

Propriété 14 *Tous les nœuds du groupe dans l'état* opérationnel *voient les mêmes changements de configuration aux mêmes instants (mêmes heures locales).*

On en déduit immédiatement les corollaires suivants, qui correspondent à la propriété de sûreté du protocole :

Corollaire 15 *Si l'établissement de la composition du groupe (*i.e. *l'appel de la fonction $gms\_list()$) est faite à la même heure locale pour tous les nœuds en état* opérationnel*, à tout instant, la vue sera la même (*i.e. *même identificateur, et même vision du groupe) pour tous ces nœuds.*

Corollaire 16 *Si il n'y a ni* défaillance par arrêt*, ni redémarrage de nœud(s), la vue du groupe n'est pas modifiée.*

Preuve : Un nœud qui est dans l'état opérationnel est un nœud qui est correct pendant plus de $\Delta_{rlb} > \Delta_{lat}$. Dans ce cas, toutes les propriétés sur l'accord s'appliquent, et donc ces propriétés proviennent directement des propriétés 10 et 11. $\Diamond$

7 Description du protocole de diffusion atomique

7.1 Principe général du protocole

L'objectif est, pour chaque nœud p qui reçoit un message, de le rediffuser jusqu'à émission correcte (*i.e.* au moins pendant ω tours du jeton successifs) quoi qu'il arrive, que p fasse partie de la cible du message ou non. Ainsi, par retransmissions successives de proche en proche (suivant le même principe que le protocole de gestion de groupe précédemment décrit), le message pourra parvenir à l'ensemble des nœuds corrects du système, donc à l'ensemble des cibles courantes du message.

7.2 Contraintes

7.2.1 Contrainte sur le protocole de gestion de groupe sous-jacent

Le protocole de diffusion atomique repose sur un protocole de gestion de groupe sous-jacent. Ce protocole peut être quelconque, du moment qu'il est capable de détecter les défaillances par arrêt du groupe, et les réintroductions de nœuds en un temps Δ_{lat} borné connu.

7.2.2 Contrainte sur la nature du jeton

Comme au paragraphe 5, dans toute la suite, nous supposerons que l'accès exclusif au médium de communication est garanti par un partage du temps d'accès à ce médium entre tous les nœuds du système (TDMA).

Ce choix impose par conséquent que la tâche d'émission des messages applicatifs soit périodique[13], de telle sorte qu'elle soit activée pendant la période de possession de la fenêtre temporelle. Il sera donc choisi de donner à la tâche d'émission la période δ_{send}.

Concernant la tâche de réception du protocole de diffusion atomique, aucun choix n'est préconisé. La seule contrainte que doit respecter la tâche de réception est qu'il faut en connaître le délai maximal de son activation (δ_{recv}) après réception d'un message par l'interface réseau.

S'il est décidé que cette tâche doit être activée par interruption, alors δ_{recv} correspond au délai de levée de l'interruption, de sauvegarde du contexte, et doit tenir compte de la levée d'autres interruptions plus prioritaires.

S'il est décidé que la tâche de réception est périodique (pour des raisons de commodité d'ordonnancement la plupart du temps), alors δ_{recv} correspond à la période pire-cas d'activation de la tâche.

7.2.3 Contrainte sur le domaine de validité du protocole

Puisque le protocole de diffusion atomique exige que la liste des *cibles courantes* soit établie, il est nécessaire que le protocole de gestion de groupe sous-jacent soit dans la phase *opérationnelle* de son fonctionnement. C'est à dire que le protocole de diffusion atomique ne devra remettre et diffuser les messages d'application qu'à partir du moment où le protocole de gestion de groupe aura fini son initialisation, *i.e.* à partir de Δ_{rlb} après chaque redémarrage .

7.3 Description du protocole

7.3.1 Paramètres du protocole

Les paramètres du protocole (voir tableau 10) correspondent aux constantes utiles au protocole de diffusion atomique temps-réel pour garantir la diffusion en ordre total assuré (*safe ordered*) des messages. Cet ordre signifie que :

- Tous les nœuds de la même cible de diffusion courante remettent les messages dans le même ordre à leur couche applicative. L'ordre est alors total;

13. Pour le protocole de gestion de groupe, ce choix n'est pas imposé.

- Un nœud p qui remet un message diffusé à sa couche applicative à la date t est assuré que tous les autres nœuds dans la cible de diffusion courante du message en ont fait autant (et en t également).

Toutes ces constantes (immuables) sont utilisées en tant que données d'entrée du protocole. Elles sont déterminées en fonction des paramètres du système et des hypothèses de défaillance (voir le paragraphe 8 énonçant et prouvant les propriétés du protocole).

Table 10 Les paramètres du protocole

Δ_{lat}	La latence de détection d'une *défaillance par arrêt* d'un processeur par le protocole de gestion de groupe, *i.e.* temps maximal entre la *défaillance par arrêt* effective d'un processeur, et sa détection par tous les autres membres du groupe. Dans le cas du protocole de diffusion atomique temps-réel, Δ_{lat} doit également correspondre à la latence de diffusion en ordre total assuré de tous les messages. Si la valeur de Δ_{lat} définie par le protocole de gestion de groupe est inférieure à $(n+2)\omega$ tours du jeton[14], alors il sera nécessaire de redéfinir Δ_{lat} à la fois pour le protocole de gestion de groupe, et pour le protocole de diffusion pour qu'ils équivalent tous deux $(n+2)\omega$ tours du jeton (voir 8). Si le protocole de gestion de groupe est celui précédemment décrit au paragraphe 5, alors nécessairement par construction $\Delta_{lat} = (n+2)\omega$ tours du jeton.

7.3.2 Structure des messages échangés

Les messages manipulés (de type `struct bcast_message_t`) par le protocole de diffusion sont constitués de cinq champs :

```
struct bcast_message_t {
  Time    sent_time;            // Date of the token round when the
                                //    message was _originally_ sent.
  Integer stamp;                // Sequence number for the message
                                //    within the round. Use-
full when
                                //    multiple messages are sent within
                                //    the same token round.
  Integer sender;               // The original sender of the message.
  List of Integer destination;  // The list of nodes concer-
ned with this
                                //    message.
  String  pdu;                  // The proper message.
};
```

7.3.3 Variables locales à chaque nœud

Le tableau 11 présente les variables locales à chaque nœud p : elles renseignent sur l'état du nœud vis à vis des messages diffusés.

14. Avec $(n+2)\omega$ tours du jeton $= (n+2)\Big(\omega\delta_{send} + \Delta_{trans} + 2\epsilon + \delta_{recv}\Big)$.

Le protocole conserve un historique des messages reçus et susceptibles d'être retransmis. Cet historique est une liste dont chaque élément est de type `bcast_retrans_t`:

```
struct diff_retrans_t {
  struct diff_message_t message;  // The message that could be
                                  //    retransmitted
  Integer        nb_retrans;      // The number of times the message
                                  //    has already been retransmitted.
};
```

Table 11 Les variables locales, internes à chacun des processeurs

List of struct `bcast_retrans_t` $retrans\_list_p$; *// Liste des messages reçus et en cours de retransmission.*

$state_p \in \{\text{crashed,restarting,operational}\}$; *// L'état du nœud courant. Il s'agit de la même variable que celle utilisée au paragraphe 5. Cette variable doit être mise à jour par le protocole de gestion de groupe.*

List of Integer GMS_list_p; *// La liste des nœuds corrects du système (et pas seulement de la cible absolue). Cette variable doit être mise à jour par le protocole de gestion de groupe.*

7.3.4 Initialisation des variables

La fonction d'initialisation du protocole de diffusion se limite à signifier qu'aucun message n'a été reçu, et donc qu'aucun message n'est à retransmettre.

Algorithme 12 La fonction d'initialisation (une seule fois après chaque arrêt): $bcast\_restart()$

$retrans\_list_p = \emptyset$; *// Aucun message n'a été reçu ni envoyé, et aucun message ne peut être retransmis.*

Il est laissé à la charge de la fonction d'initialisation du protocole de gestion de groupe d'effectuer les opérations suivantes :

$$\begin{aligned} state_p &= \text{restarting}; \\ GMS\_list_p &= \{\emptyset\}; \end{aligned}$$

7.3.5 Réception d'un message

La tâche de réception (périodique), dont l'algorithme est détaillé en figure 13, se déroule en deux phases.

Lors d'une première phase (lignes 2 à 11), l'historique des messages reçus susceptibles d'être retransmis est mis à jour en fonction des message reçus (figurant dans $receive\_queue\_network_p$). Les messages reçus par la couche réseau d'un nœud p sont insérés dans cette liste s'ils n'avaient pas été reçus auparavant, que p fasse ou non partie des destinataires des messages (ceci provient des hypothèses de défaillances qui sont appliquées à l'ensemble du système, et non aux cibles absolues des messages).

Lors d'une deuxième phase (ligne 12 et suivantes), les messages figurant dans l'historique des messages reçus du nœud courant p sont parcourus. Ceux susceptibles d'être remis à l'application (c'est à dire qui ont été émis au moins Δ_{lat} auparavant) sont enlevés de l'historique, ce qui assure que la taille de l'historique est bornée (si l'on connaît

le nombre maximum de messages qui peuvent être émis par tous les nœuds en un tour du jeton). Ceci permet de garantir que la durée de parcours de cet historique, et donc la durée d'exécution de la tâche de réception est bornée. Parmi ces messages susceptibles d'être remis à l'application, seuls ceux pour lesquels p fait partie de la destination sont remis localement à l'application.

Algorithme 13 La tâche de réception des messages (périodique de période maximale δ_{recv}) : $bcast\_recv\_task()$

```
    Time local_time = date_actuelle();
    // Mise à jour de l'historique des retransmissions.
 2: for struct bcast_message_t m in receive_queue_network_p do
      if local_time − m.stamp ≤ Δ_lat then // Le message n'est pas trop vieux.
 4:     if ∄u ∈ retrans_list_p such that u.message.sender == m.sender and
        u.message.sent_time == m.sent_time and u.message.stamp ==
        m.stamp then // Le message n'a pas déjà été reçu
          struct bcast_retrans_t retrans_tmp;
 6:       retrans_tmp.message = m;
          retrans_tmp.nb_retrans = 0; // Encore aucune retransmission effectuée.
 8:       retrans_list_p = retrans_list_p ∪ {retrans_tmp};
        end if
10:   end if
    end for
    // Remise de messages à l'application si possible.
12: sort_list(retrans_list_p); // Les messages sont classés suivant 1) leur date d'émission, et 2) leur numéro
    de séquence au sein de la vague d'émissions (champ stamp du message), afin que la boucle de parcours suivante
    permette de les délivrer dans le même ordre sur tous les nœuds, indépendamment de leur ordre de réception.
    for struct bcast_retrans_t r in retrans_list_p do
14:   if local_time − r.sent_time > Δ_lat then // Le message peut être remis à l'application
        if State_p == operational and p ∈ r.destination and
        r.message.sender ∈ GMS_list_p then // Le nœud p doit être opérationnel et doit
        faire partie de la liste des destinataires du message. L'émetteur doit également avoir été considéré comme
        correct lors de l'émission.
16:       deliver_app(r.message); // Le message est remis à l'application.
        end if
18:     retrans_list = retrans_list − {r}; // Le message est supprimé de l'historique.
      end if
20: end for
```

7.3.6 Tâche d'émission

La tâche d'émission périodique se charge de diffuser la série de messages qui se trouve dans le tampon d'entrée $send\_queue\_APPLI_p$ (ligne 9 et suivantes). Quand une application désire émettre un message, elle l'insère dans ce tampon, et la tâche d'émission périodique donnée en figure 14 se chargera de le diffuser.

Cette tâche s'occupe également de réémettre, à hauteur de ω tours successifs, les messages qui nécessitent une retransmission (lignes 3 à 8). Les réémissions ne tiennent pas compte des groupes de destinataires (des cibles du message à réémettre) : elles ont lieu quoi qu'il arrive, que le nœud qui réémet fasse ou non partie du groupe de destinataires du message (ceci provient des hypothèses de défaillances qui sont appliquées à l'ensemble du système, et non aux cibles absolues des messages).

Algorithme 14 La tâche d'émission des messages par un nœud p (correspond à la possession du jeton, donc périodique de période δ_{send}) : $bcast\_send()$

```
   Time local_time = date_actuelle();
2: Integer stamp = 0; // Utilisé pour différencier l'émission de plusieurs messages au sein de la même tâche.
   // Réémission des nœuds qui le nécessitent
   for struct bcast_retrans_t r in retrans_list_p do
4:   if r.nb_retrans < ω then // r doit être retransmis.
       bcast_network(r.message); // Diffusion à tous les membres du système.
6:     r.nb_retrans++;
     end if
8: end for
   // Émission des messages de la couche applicative
   if State_p == operational then // Le nœud doit être opérationnel.
10:  for [String pdu, List of Integer destination] in send_queue_APPLI_p do
       struct bcast_message_t bcast_msg;
12:    bcast_msg.sent_time = local_time;
       bcast_msg.stamp = stamp;
14:    bcast_msg.sender = p;
       bcast_msg.destination = destination;
16:    bcast_msg.pdu = pdu;
       bcast_network(bcast_msg); // Diffusion à tous les membres du système.
18:    stamp++;
     end for
20: end if
```

7.4 Remarque sur les domaines de validité du protocole

Dans le paragraphe 7.2.3, il est imposé que les émissions de messages et les remises de messages au sein d'un nœud p ne soient possibles que si p est dans l'état *operationnel*.

Le protocole présenté dans ce document respecte cette propriété (ligne 9 à l'émission, et 14 à la réception), alors qu'il est nécessaire que ce protocole soit lancé dès le démarrage du nœud.

Ainsi, pendant la phase de redémarrage du nœud p pendant laquelle celui-ci n'est pas membre du groupe, mais pendant laquelle p s'informe de la composition du groupe, p va également s'informer des messages diffusés, et participer à leur retransmission, sans pour autant que l'application intervienne, ou soit sollicitée.

8 Propriétés du protocole de diffusion atomique

Avec le recours aux hypothèses de défaillances énoncées au paragraphe 4.1 (en particulier la définition de ω), nous allons montrer dans ce paragraphe les propriétés essentielles suivantes :

- Le protocole de diffusion garantit la **diffusion fiable** des messages (propriété 19). C'est à dire qu'un message émis est reçu par tous les nœuds de la cible courante de la diffusion, tant que l'émetteur est resté correct. Si l'émetteur est détecté incorrect, aucun nœud ne délivrera le message à l'application (même s'il l'a reçu, car il n'est pas certain alors que tous les autres membres de la cible l'aient également reçu).
- Il s'agit d'un protocole de **diffusion en temps borné** (propriété 19). Ainsi, le protocole se caractérise par un délai maximal séparant la demande d'émission d'un message par un nœud, de la réception de ce message par tous les nœuds de la cible courante.
- L'ordre entre tous les messages remis à l'application est respecté (propriété 20), et identique sur tous les nœuds : c'est l'**ordre total**. En effet, d'une part, tous les messages sont ordonnés suivant leur date d'émission (ce qui est rendu possible par la synchronisation des horloges, et par l'exclusion mutuelle en émission sur le canal de diffusion). D'autre part, la remise des messages à l'application conserve cet ordre puisqu'elle a lieu un temps constant après la date d'émission (il s'agit d'une translation dans le temps).
- La **synchronisation virtuelle** (*Virtual Synchrony* - propriété 20) est garantie, puisque la latence de remise des messages à l'application est égale à la latence de détection des défaillances par arrêt. Ceci permet effectivement de délivrer les messages de changement de vue du groupe en cohérence avec les messages diffusés.

8.1 Problématique

Le protocole de diffusion atomique présenté dans ce document repose sur le principe suivant. Il s'agit, pour chaque nœud p qui reçoit un message, de le rediffuser jusqu'à émission correcte (*i.e.* au moins pendant ω tours du jeton successifs) quoi qu'il arrive, que p fasse partie de la cible du message ou non. Ainsi, par retransmissions successives de proche en proche (suivant le même principe qu'au paragraphe 5), le message pourra parvenir à l'ensemble des nœuds corrects du système, donc à l'ensemble des cibles courantes du message.

Pour garantir la cohérence de l'information répliquée du système, la propriété à prouver est la propriété de remise des messages en ordre total (*i.e.* identique sur tous les nœuds) et en temps borné à l'application.

Dans le cas du présent protocole, cette propriété est vérifiée grâce à la remise *synchronisée* des messages (*i.e.* dont la date est uniquement liée à la date d'émission des messages). Toute la difficulté est alors de déterminer la valeur du délai séparant la première émission, de la remise des messages à l'application (Δ_{lat}), afin que ces propriétés soient valides.

8.2 Émission correcte des messages

Compte tenu de l'hypothèse H2a, la propriété 17 suivante est vérifiée.

Propriété 17 *Pour tout message m, un nœud p dans la vue du groupe courante pendant au moins ω tours successifs du jeton arrive à faire parvenir m à une majorité de nœuds de la vue du groupe courante.*

Preuve : Provient du fait que m est répété au cours de ω tours du jeton successifs. Ainsi, d'après l'hypothèse H2a, il est garanti que m sera reçu, au cours de ces ω tours du jeton, par une majorité de nœuds dans la vue du groupe. $\Diamond$

8.3 Diffusion fiable sous contrainte de correction de l'émetteur

La propriété 17 précédente, ainsi que l'hypothèse H2b permettent de prouver la propriété centrale 18 suivante.

Propriété 18 *Pour tout message m émis en t, un nœud p, dans la vue du groupe courante pendant au moins ω tours du jeton successifs, arrive à faire parvenir m à tous les nœuds de la vue du groupe courante pendant $[t,t + 2\omega \text{ tours du jeton}]$ en au plus 2ω tours successifs du jeton.*

Preuve : Considérons un nœud $q \neq p$ appartenant à la vue du groupe pendant au moins $[t,t + 2\omega \text{ tours du jeton}]$.

D'après 17, pendant $[t,t + \omega \text{ tours du jeton}]$, p arrive à faire parvenir m à une majorité M_1 de nœuds dans le vue du groupe. Deux cas peuvent se produire :

- Soit q appartient à la vue du groupe pendant $[t,t + \omega \text{ tours du jeton}]$ et $q \in M_1$. Dans ce cas, la propriété est directement démontrée (q a reçu le message lors des ω premiers tours du jeton);
- Soit
  - $\exists r \neq q \in M_1$ tel que r reçoit m en $t' \in [t,t + \omega \text{ tours du jeton}]$, et tel que r arrive à retransmettre m à q pendant $[t',t' + \omega \text{ tours du jeton}] \subset [t,t + 2\omega \text{ tours du jeton}]$, alors la propriété est à nouveau démontrée.
  - $\nexists r \in M_1$ tel que r réussit à faire parvenir un message à q pendant ω tours successifs du jeton après réception de m (au plus tard en $t+\omega$ tours du jeton). Alors q ne reçoit pas les messages d'une majorité de processeurs (au moins M_1) durant au moins $[t + \omega \text{ tours du jeton},t + 2\omega \text{ tours du jeton}]$. Dans ce cas, q ne fait plus partie de la vue courante du groupe (q est incorrect), ce qui contredit l'hypothèse ci-dessus selon laquelle q est correct pendant cet intervalle. Ce cas est donc impossible.

Par conséquent, à la date $t+2\omega$ tours du jeton, tous les nœuds de la vue du groupe courante ont reçu m. $\Diamond$

8.4 Diffusion en temps réel et remise en temps borné connu des messages à l'application

La propriété 18 précédente suppose deux contraintes de correction (une sur l'émetteur du message, et l'autre sur les récepteurs). La propriété 19 suivante permet de garantir cette correction, et donc de garantir la diffusion fiable assurée de tous les messages.

Propriété 19 *Si en t, un nœud p tente d'émettre un message m, alors, en $t + \Delta_{lat}$, soit tous les nœuds dans la vue du groupe et concernés par m (*i.e. *les cibles courantes) remettent m à l'application, soit aucun d'entre eux ne le remet.*

Preuve : Δ_{lat} est la latence de détection des défaillances par arrêt des nœuds. Pour un nœud q dans la vue du groupe en $t + \Delta_{lat}$, trois cas sont possibles :

- Soit q ne voit pas p dans la vue du groupe en $t + \Delta_{lat}$. Dans ce cas, il n'est pas garanti que p ait été correct durant $[t,t+\omega \text{ tours du jeton}]$, et donc il peut exister un nœud r qui n'ait pas reçu m. Dans ce cas, le message ne doit pas être remis à l'application. Alors, puisqu'aucun autre nœud s de la vue du groupe ne voit p dans la vue du groupe (propriété d'accord du protocole de gestion de groupe), aucun de ces nœuds s et q ne remettra le message m à l'application.
- Soit q voit p dans la vue du groupe en $t+\Delta_{lat}$, et il en est de même pour tous les autres nœuds dans la vue du groupe, et donc par conséquent pour tous les nœuds s formant les cibles courantes de m.
  Le problème est qu'on ne dispose pas de l'information selon laquelle p sera correct jusqu'à $t + \omega$ tours du jeton.
  - Si p est dans la vue du groupe pendant $[t,t + \omega \text{ tours du jeton}]$, alors il est établi (propriété 18) que tous les nœuds formant la cible courante de m reçoivent m avant $t + 2\omega$ tours du jeton, et donc peuvent le remettre en $t + \Delta_{lat}$ à l'application (car $\Delta_{lat} \geq 2\omega$ tours du jeton).
  - Sinon, on procède comme pour le protocole de gestion de groupe décrit en 5 :
    - soit il existe un chemin (comprenant éventuellement des cycles) permettant de faire transiter m, par l'intermédiaire de nœuds qui le retransmettent, de p à une majorité M de nœuds dans la vue du groupe. Alors q recevra m au plus tard en $t + (n + 2)\omega$ tours du jeton $= t + \Delta_{lat}$, $\forall q$ de la cible courante en $t + \Delta_{lat}$. Dans ce cas, tous les nœuds de la cible courante peuvent délivrer le message m à l'application à partir de $t + \Delta_{lat}$;
    - soit un tel chemin n'existe pas, et alors le message m ne parvient à aucun des nœuds de la cible courante de m, et donc aucun de ces nœuds ne pourra le remettre à l'application.

Ainsi, en $t + \Delta_{lat}$, soit tous les nœuds de la cible courante de m disposent de m et peuvent le remettre à l'application à la condition que p soit dans la vue du groupe à cette date, soit aucun d'eux ne le remet à l'application. $\diamondsuit$

Le protocole décrit dans ce document respecte cette propriété puisque, au moment de remttre un message à l'application, un nœud q est nécessairement dans la vue du groupe (ligne 9 à l'émission, et 14 à la réception).

8.5 Ordre total assuré

À partir de la propriété 19, la propriété 20 suivante est immédiate.

Propriété 20 *La remise d'un message m au sein de ses cibles courantes respecte l'***ordre total** *(lié à la chronologie d'émission des messages destinés à ces cibles). De plus, cette remise de m à l'application est* **assurée***, c'est à dire qu'elle a lieu sur tous les nœuds de la cible courante de m.*

De plus, cette remise synchronisée avec :

1. *les messages de changement de configuration;*

2. les messages à l'application
permet de garantir la synchronisation virtuelle *(*Virtual Synchrony*).*

Preuve : Une fois la diffusion fiable *assurée* établie (propriété 19 précédente), cette propriété 20 provient directement du fait que les fenêtres d'émission des nœuds sont disjointes. En effet, les horloges étant synchronisées, le protocole étant tel que la date de remise des messages à l'application est une translation (constante) de leur date d'émission il est garanti que, pour chaque message émis, l'*ordre total* parmi tous les messages de tous les nœuds est respecté (il s'agit de l'ordre chronologique) lors de la remise des messages à l'application.

Quant à la propriété de synchronisation virtuelle, elle provient du fait que les latences de détection des défaillances permanentes, et de diffusion fiable assurée, sont égales (à Δ_{lat}). $\diamondsuit$

9 Conclusion

Ce document présente un protocole de gestion de groupe en milieu synchrone. Ce protocole permet de dresser la liste des nœuds non défaillants par arrêt du système : le *groupe*. Il prend place dans chacun des nœuds du système, et garantit que les décisions de *changement de groupe* sont prises :

en un temps borné connu après les changements d'état de nœuds du système (*défaillance par arrêt* ou réintroduction d'un nœud);

en accord : les décisions prises reflètent la même composition du groupe, et sont prises à la même date locale.

Ce document décrit également un protocole de diffusion atomique. Ce protocole présente un comportement temporel strict permettant de l'intégrer dans des systèmes temps-réel.

Associé à un protocole de gestion de groupe tel que celui décrit dans le présent document, et à un système de synchronisation des horloges, il assure que les messages seront délivrés, sur les nœuds destinataires, à l'application suivant l'ordre total de leur émission et en cohérence avec les messages de changement de vue (ce qui correspond à la synchronisation virtuelle, ou *Virtual Synchrony*). De plus, un message n'est délivré à l'application que si il est certain que tous les nœuds corrects concernés par ce message (*i.e. la cible courante*) l'ont également reçu. Ceci permet de qualifier cet ordre d'ordre total assuré (*safe total order*).

A Tableau récapitulatif des caractéristiques d'autres protocoles

Le tableau 2 récapitule les différentes caractéristiques des protocoles étudiés.

B Variante des protocoles sous contrainte forte de non partitionnement

Dans le cas de HADES, les hypothèses de défaillances sont plus fortes : il est supposé que le système ne peut pas être partitionné. Ceci rend les preuves des mêmes

TAB. 2 – *Caractéristiques des différents protocoles étudiés*

Caractéristique	Chang & Maxemchuk	PSync	Amoeba	Fast ABCast (Isis)	Transis	Totem	RTCAST	Clegg	Ce document
Type de système	Asynchrone	synchrone	asynchrone	synchrone	asynchrone		synchrone		
Topologie	anneau	quelconque		anneau				anneau (réplication des canaux)	anneau
Rôle du jeton	définir le séquenceur	-	-	définir l'émetteur					
Ordre assuré	total	causal	total						
Gestion de l'ordre	centralisée / circulante	à l'émetteur	centralisée	centralisée / circulante	distribuée : reléguée à l'émetteur				
service de gestion de groupe	non	non	oui						
défaillances tolérées	arrêt/omission			arrêt / omissions (Fast CBCast)	arrêt/omission		arrêt	arrêt/omission	
Cohérence conservée aux transitions (Virtual Synchrony)	non	oui	incertain	oui			oui (par construction)	oui	
Tolérance au partitionnement	non	non	non	non	Extended Virtual Synchrony		non		

TAB. 2 – *Caractéristiques des différents protocoles étudiés (suite)*

Caractéristique	Chang & Maxemchuk	PSync	Amoeba	Fast ABCast (Isis)	Transis	Totem	RTCAST	Clegg	Ce document
Contrôle de flux	non	non	non	non	non	oui	oui (par construction)	non	non
Interconnexion de domaines de diffusion	non	-	-	non	(oui)	oui	non		
Empilement de protocoles pour maintenir l'ordre total	non	non	non	oui	oui	non			
Utilisation du réseau espérée	faible	faible	forte (le séquenceur a un effet limitant)	correcte	correcte	élevée (comparable à TCP)	dépend de l'ordonnancement	correcte	dépend de l'ordonnancement
Type d'implémentation	monolithique			empilement de microprotocoles		monolithique		2 protocoles	

propriétés que celles du corps du document moins délicates, et moins intéressantes.

L'ensemble du travail précédent reste identique dans l'ensemble. Nous ne relevons ici que les différences qui apparaissent sous ces nouvelles hypothèses.

B.1 Présentation

HADES suppose que le système ne peut pas se partitionner, c'est à dire qu'il est garanti qu'en un nombre connu de tours de jetons, un message réémis à chaque tour est reçu par **tous** les nœuds corrects [15].

Il découle que la latence pire-cas de diffusion *de proche en proche* (voir le paragraphe 6.8.2) des messages afin d'assurer la cohérence de cette diffusion n'est plus aussi grande. Ainsi, la détection des *défaillances par arrêt* ou des redémarrages de nœuds est plus rapide. Il s'ensuit que la valeur de Δ_{lat} est plus faible.

Pour la même raison, l'établissement de la cohérence au sein des messages diffusés est plus rapide.

B.2 Hypothèses de défaillance

Par rapport à 4.1, seule l'hypothèse H2 est modifiée, et devient :

$\mathcal{H}2$. Il existe une constante ω telle que :

(a) Sur ω diffusions d'un même message en ω tours de jeton consécutifs par un nœud p dans la vue du groupe courante, p arrive à faire parvenir le message à la tous les nœuds de la vue du groupe courante.
Ceci **ne signifie pas** que tous ces autres nœuds perçoivent *la même instance* du message de p pendant ces ω tours du jeton.

(b) En ω tours du jeton, un nœud p dans la vue du groupe courante reçoit des messages émis pendant cette période par tous les nœuds de la vue du groupe.

Ceci impose le non partitionnement du système, et donc, à tout moment, l'existence d'une vue du groupe (*i.e.* d'une partition) unique.

Cette hypothèse signifie également que, tant qu'un nœud reste correct, il perçoit *régulièrement* des signes de vie de tous les autres nœuds corrects, et ses signes de vie sont *régulièrement* perçus par tous les autres nœuds corrects;

B.3 Protocole de gestion de groupe

B.3.1 Description

Le protocole de gestion de groupe ne diffère de celui présenté au paragraphe 5.4 que suivant un point : la tâche de réception.

En effet, les critères pour considérer un nœud *défaillant par arrêt*, ou pour détecter sa propre défaillance par arrêt aux yeux des autres ne sont pas les mêmes. Il ne s'agit plus de raisonner sur une majorité de nœuds, mais sur l'ensemble des nœuds du système.

15. Et pas seulement une majorité.

Algorithme 15 La tâche de réception sous l'hypothèse forte de non partitionnement (périodique de période maximale δ_{recv}) : $gms\_recv\_task()$

Time $local\_time = date\_actuelle()$;
if $state_p ==$ restarting **and** $local\_time \geq join_p[p]$ **then** *// La phase de redémarrage du nœud p touche à sa fin*
  $state_p =$ operational;
end if
for message [Processor $sender$, Integer $View_{sender}$, Time t_1, $\cdots$, Time t_n, Time j_1, $\cdots$, j_n] in $receive\_queue\_network_p$ **do**
  $gms\_recv(local\_time, state_p, sender, View_{sender}, t_1, \cdots, t_n, j_1, \cdots, j_n)$;
end for
if
$\wedge_{sender \neq p} local\_time - recv_p[sender] \geq \Delta_{lat}$
or
$\wedge_{sender \neq p} local\_time - last\_received\_from\_us_p[sender] > \Delta_{lat}$
then *// Les autres nœuds sont perçus défaillants par p (défaillance permanente en réception, ou du commutateur central),* **ou** *les autres nœuds de la vue perçoient p défaillant (défaillance permanente en émission).*
  $state_p =$ crashed;
  signal(GMS_CRASH);
  call $gms\_restart()$; *// p informe l'application qu'il a* crashé, *et redémarre.*
end if
if $state_p ==$ operational **then** *// La vue du groupe n'aura de sens pour p qu'une fois la phase de redémarrage terminée (*i.e. *une fois en état* opérationnel*).*
  $gms\_list(local\_time)$; *// On actualise la vue du groupe.*
end if

B.3.2 Propriétés

Suicides par manquement aux hypothèses.

Les deux propriétés suivantes sont directement issues de l'hypothèse de défaillance $\mathcal{H}2$.

Propriété 21 *Un nœud q qui ne reçoit pas de messages de tous les processeurs de la vue pendant plus de $f_{send} + 1$ tours du jeton consécutifs, se suicide.*

Propriété 22 *Si pendant ω tours du jeton, un nœud n'arrive pas à se faire entendre par tous les nœuds dans la vue du groupe, alors il contredit l'hypothèse de non partitionnement, et donc se suicide.*

Diffusion d'un signe de vie.

Les propriétés 21 et 22 conduisent à la valeur minimale pour ω :

Hypothèse 6 $\omega \geq f_{send} + 1$.

Sous cette hypothèse, et grâce aux propriétés 21 et 22, nous avons aussi la propriété suivante :

Propriété 23 *En ω tours de jeton, un nœud p arrive à faire parvenir un de ses messages de vie à tous les nœuds de la vue du groupe, et à recevoir des messages de vie émis pendant cette période par tous les autres nœuds de la vue du groupe.*

Disposant de la propriété 23 fondamentale précédente, la propriété suivante est montrée :

Propriété 24 *Si en t, le processeur d'un nœud p tente d'émettre un message, alors, en ω tours du jeton :*

- *soit, en $t + \omega$ tours du jeton, tous les nœuds de la vue ont reçu chacun au moins un message de vie de p émis pendant $[t,t + \omega$ tours du jeton$]$;*
- *soit, en $t + \omega$ tours du jeton, aucun des nœuds de la vue n'a reçu de message de vie de p durant $[t,t + \omega$ tours du jeton$]$, et de plus p a été victime d'une* défaillance par arrêt *pendant cet intervalle.*

Accord sur la réception des messages de vie

Cette propriété représente la différence centrale entre les hypothèses prises pour le corps du document.

Définition de la latence de diffusion des signes de vie. Soit la valeur suivante de $\Delta_{diffuse}$, la latence de diffusion d'un message de vie parmi tous les nœuds correct :

Hypothèse 7 $\Delta_{diffuse}$ *correspond à au moins 2ω tours du jeton, c'est à dire à* $\Delta_{diffuse} \geq 2(\omega\delta_{send} + \Delta_{trans} + 2\epsilon + \delta_{recv})$.

Sous l'hypothèse 7, les propriétés 3, 7, ainsi que le corollaire 8 présentés dans le corps du document sont vérifiés.

Preuve de la propriété 7 sous la nouvelle hypothèse $\mathcal{H}2$. La démonstration repose sur la diffusion du message de vie de p passant progressivement de nœud en nœud (en fait ici, au plus un intermédiaire sera nécessaire). Nous utilisons la notation "$\rightsquigarrow_{p,t}$" introduite en 6.8.2.

En t, le processeur correct p tente d'émettre un message de vie. Considérons un nœud $q \neq p$ de la vue du groupe (*i.e.* correct), et déterminons le chemin que peut suivre la *connaissance que p a émis un message de vie en t* jusqu'à q.

Après la tentative d'émission du message, trois cas sont possibles :

- Le message de vie n'a pas été correctement émis, auquel cas ni q, ni aucun nœud correct du groupe ne recevra *l'information que p a émis un message de vie en t*;
- Le message de vie a été correctement émis. Puisque $f_{recv} < n - f_{crash}$, $\exists r$ correct, $r \neq p$, qui a reçu le message de vie de p. Dans ce cas :
  - Si $r = q$, q a directement reçu le message de vie de p émis en t, et donc $p \rightsquigarrow_t q$;
  - Sinon :
    - Si r arrive à transmettre correctement un message de vie avant de *crash*er, alors q le recevra en ω tours du jeton (propriété 23), et donc on a $p \rightsquigarrow_t r \rightsquigarrow_t q$, d'où $p \rightsquigarrow_t q$;
    - Si ni r, ni aucun autre nœud correct différent de p et qui avait reçu l'*information que p avait émis un message de vie en t* n'arrive à transmettre cette information avant de *crash*er, alors, ni q, ni aucun nœud correct du groupe ne recevra *l'information que p a émis un message de vie en t*.

Ainsi, en un maximum de 2ω tours du jeton, on a :

- Soit $p \rightsquigarrow_t q$;
- Soit ni q, ni aucun nœud correct du groupe ne recevra *l'information que p a émis un message de vie en t*.

Accord sur la défaillance ou le redémarrage d'un nœud.

Défaillance d'un nœud. Sous l'hypothèse :

Hypothèse 8 $\Delta_{lat} \geq \Delta_{diffuse} + \omega$ tours du jeton, *soit* $\Delta_{lat} \geq 3(\omega\delta_{send} + \Delta_{trans} + 2\epsilon + \delta_{recv})$.

On a la propriété suivante :

Propriété 25 *Si, au temps t_q local à un nœud q, le nœud q dans la vue du groupe et correct depuis plus de Δ_{lat} n'a pas reçu d'un nœud p de message de vie depuis Δ_{lat}, alors p est* défaillant par arrêt, *et tous les nœuds r dans la vue du groupe depuis au moins Δ_{lat} excluent p de la vue du groupe au temps local $t_r = t_q$.*

Preuve : Notons l_q la date d'émission du dernier message de vie que q a reçu de p. Nous nommerons abusivement cette date "le dernier message de vie reçu de p par q". Soit $t_q = l_q + \Delta_{lat}$.

Première partie : détection de la défaillance de p. À t_q, q n'a reçu aucun message de vie de p depuis Δ_{lat}. Or, puisque $\Delta_{lat} \geq 2\omega$ tours du jeton, la propriété 24 assure qu'aucun autre nœud correct depuis au moins Δ_{lat} n'aura également reçu de message de vie de p depuis 2ω tours du jeton, **et que p a été victime d'une *défaillance par arrêt*** pendant $[l_q, l_q + \omega \text{ tours}]$ (puisque $\omega \geq f_{send} + 1$).

Deuxième partie : le message de vie l_q est le dernier message de vie de p reçu par tous les nœuds corrects. Supposons qu'il existe un nœud $r \notin \{q,p\}$ tel que le dernier message de vie que r a reçu de p est l_r avec $l_r > l_q$. Nécessairement, q aurait reçu l_r au plus tard en $l_r + \Delta_{diffuse} \geq l_q + \omega\ tours + \Delta_{diffuse} \geq l_q + \Delta_{lat}$, soit en t_q (voir la propriété 7 précédente). Donc, en t_q, q aurait reçu le message de vie l_r de p, avec $l_r > l_q$. Ceci contredit donc qu'en t_q, le dernier message de vie que q a reçu de p est le message diffusé en l_q.

D'autre part, en $t_q = l_q + \Delta_{diffuse} \leq l_q + \Delta_{lat}$ tous les autres nœuds corrects auront reçu l_q (propriété 8, sachant qu'au moins q, encore correct, a reçu ce message de vie).

Ce message l_q est donc le dernier message de vie de p reçu par tout nœud correct r (soit $l_r = l_q$), qui décidera que p est *défaillant par arrêt* en $t_r = l_r + \Delta_{lat} = l_q + \Delta_{lat} = t_q$.

En t_q, tous les nœuds r de la vue $View_r$ du groupe excluent p de cette vue. Une nouvelle vue $View_r + 1$ est formée.

$\Diamond$

Redémarrage d'un nœud.

Propriété 26 *Supposons qu'au temps t, un nœud p passe de l'état* défaillant par arrêt *à l'état* redémarrage, *et tente d'émettre*[16] *son premier message de vie. Alors, pour tous les nœuds q appartenant à la vue du groupe courante pendant au moins $[t,t + 2\omega$ tours du jeton$]$, q voit le nœud p comme correct à partir de Δ_{rlb}, et donc l'intégre au groupe des nœuds corrects à l'heure locale $t + \Delta_{rlb}$.*

Ainsi, sous l'hypothèse 9 suivante, on en déduit immédiatement la propriété 27 suivante.

Hypothèse 9 $\Delta_{rlb} \geq 2\omega$ tours du jeton.

Corollaire 27 *Pour tous les processeurs q corrects pendant $[t,t + 2\omega$ tours du jeton$]$, la décision d'intégration du nœud p dans le groupe est prise au même instant $t + \Delta_{rlb}$.*

Preuve : Tous les nœuds q corrects pendant 2ω tours du jeton ont reçu un message de vie de p (propriété 24), donnant entre autres la date de réinsertion de p dans le groupe (élément j_p d'un tel message), telle que prévue par p, *i.e.* $t + \Delta_{rlb}$. $\Diamond$

Le problème est qu'un nœud qui redémarre ne fait pas partie de la catégorie des nœuds qui n'ont pas été *défaillants par arrêt* depuis au moins Δ_{lat}, et donc aucune des propriétés précédente n'est respectée. Il faut donc accorder au nœud qui redémarre un intervalle de temps de "mise à niveau", nécessaire pour qu'il puisse connaître la composition du groupe, et pour que tout le monde le voie correct, et qu'il y ait accord sur toutes ces visions.

Il s'agit de montrer sous quelles hypothèses les nœuds qui redémarrent auront connaissance de la vue du groupe. L'objectif est donc de déterminer combien de temps au minimum doit durer la phase de redémarrage d'un nœud, c'est à dire de donner une minoration de Δ_{rlb}.

Propriété 28 *Un nœud qui redémarre est dans l'état* opérationnel *dès qu'il est en accord avec tous les autres processeurs corrects, c'est à dire au bout de Δ_{lat} après l'instant de son redémarrage.*

Ceci implique $\Delta_{rlb} > \Delta_{lat}$.

16. Ce qui ne veut pas dire que l'émission va être correcte.

Preuve : Toutes les propriétés précédentes sur l'accord s'appliquant aux nœuds corrects, s'appliquent pour des nœuds corrects depuis plus de Δ_{lat}. Donc, nécessairement, avant de passer membre du groupe, c'est à dire avant de faire partie de la liste des nœuds parmi lesquels l'accord sur les changements de configuration doit être garanti (propriétés 25 et 26), un nœud qui redémarre en t ne doit pas passer dans l'état *opérationnel* avant $t + \Delta_{lat}$.

Donc nécessairement, $\Delta_{rlb} > \Delta_{lat}$, et après Δ_{lat} après l'instant de son redémarrage, l'accord sur les changements de configuration est garanti. $\Diamond$

Accord sur la composition du groupe.

De la même manière que dans le corps du document, à partir des propriétés précédentes, la propriété d'accord 14 est démontrée.

B.4 Protocole de diffusion atomique

Le protocole n'est pas modifié lorsque l'hypothèse de défaillance H2 est remplacée par $\mathcal{H}2$.

Avec $\Delta_{lat} \geq 3(\omega\delta_{send} + \Delta_{trans} + 2\epsilon + \delta_{recv})$, seules les propriétés sont modifiées.

Diffusion fiable des messages sous contrainte de correction de l'émetteur.

Compte tenu de l'hypothèse $\mathcal{H}2$, la propriété 29 suivante est vérifiée.

Propriété 29 *Pour tout message m, un nœud p dans la vue du groupe courante pendant au moins ω tours successifs du jeton arrive à faire parvenir m à tous les nœuds de la vue du groupe courante.*

Preuve : Provient du fait que m est répété au cours de ω tours du jeton successifs. Ainsi, d'après l'hypothèse $\mathcal{H}2$, il est garanti que m sera reçu, au cours de ces ω tours du jeton, par tous les nœuds dans la vue du groupe. $\Diamond$

Diffusion en temps réel et remise en temps borné connu des messages à l'application.

La propriété 29 précédente suppose deux contraintes de correction (une sur l'émetteur du message, et l'autre sur les récepteurs). La propriété 30 suivante permet de garantir cette correction, et donc de garantir la diffusion fiable assurée de tous les messages.

Propriété 30 *Si en t, un nœud p tente d'émettre un message m, alors, en $t + \Delta_{lat}$, soit tous les nœuds dans la vue du groupe et concernés par m (*i.e. *les cibles courantes) remettent m à l'application, soit aucun d'entre eux ne le remet.*

Preuve : Δ_{lat} est la latence de détection des défaillances par arrêt des nœuds. Pour un nœud q dans la vue du groupe en $t + \Delta_{lat}$, trois cas sont possibles :

- Soit q ne voit pas p dans la vue du groupe en $t + \Delta_{lat}$. Dans ce cas, il n'est pas garanti que p ait été correct durant $[t, t+\omega \text{ tours du jeton}]$, et donc il peut exister un nœud r qui n'ait pas reçu m. Dans ce cas, le message ne doit pas être remis à l'application. Alors, puisqu'aucun autre nœud s de la vue du groupe ne voit p dans la vue du groupe (propriété d'accord du protocole de gestion de groupe), aucun de ces nœuds s et q ne remettra le message m à l'application.

TAB. 3 – *Application numérique*

ω	2					
Δ_{trans}	5ms					
δ_{send}	10ms					
δ_{recv}	10ms					
ϵ	100ms					
Nombre de nœuds	Hypothèse de majorité correcte			Hypothèse forte de non partitionnement		
	4	6	8	4	6	8
Δ_{lat}	1.410s	1.880s	2.350s	0.235s		

- Soit q voit p dans la vue du groupe en $t + \Delta_{lat}$, et il en est de même pour tous les autres nœuds dans la vue du groupe, et donc par conséquent pour tous les nœuds s formant les cibles courantes de m.
  Le problème est qu'on ne dispose pas de l'information selon laquelle p sera correct jusqu'à $t + \omega$ tours du jeton.

  – Si p est dans la vue du groupe pendant $[t,t + \omega$ tours du jeton$]$, alors il est établi (propriété 29) que tous les nœuds formant la cible courante de m reçoivent m avant $t + 2\omega$ tours du jeton, et donc peuvent le remettre en $t + \Delta_{lat}$ à l'application (car $\Delta_{lat} \geq 2\omega$ tours du jeton).
  – Sinon, si, durant ω tours du jeton, p n'arrive à transmettre m qu'à un nœud r dans la vue du groupe, deux cas sont possibles :
    – Soit r arrive à émettre correctement m, auquel cas q, comme tous les autres nœuds corrects du système recevront m au cours des ω tours du jeton suivants;
    – Soit r est défaillant par arrêt avant d'arriver à émettre m correctement, auquel cas ni q, ni aucun autre nœud de la vue du groupe ne recevra m.

Ainsi, en $t + 2\omega$ tours du jeton $= t + \Delta_{lat}$, soit tous les nœuds de la cible courante de m disposent de m et peuvent le remettre à l'application à la condition que p soit dans la vue du groupe à cette date, soit aucun d'eux ne le remet à l'application. $\Diamond$

Le protocole décrit dans ce document respecte cette propriété puisque, au moment de remttre un message à l'application, un nœud q est nécessairement dans la vue du groupe (ligne 9 à l'émission, et 14 à la réception).

C Application numérique

Nous donnons dans le tableau 3 un échantillon de valeurs numériques concernant les temps de latence de détection des *défaillances par arrêt* des nœuds, et de diffusion atomique assurée des messages.

Table des matières

Références

[1] E. Anceaume, G. Cabillic, P. Chevochot, and I. Puaut. Modélisation des applications et positionnement de l'étude par rapport au projet trdf. Technical report, Projet Hades, équipe Solidor, IRISA, November 1997.

[2] P. Chevochot and I. Puaut. Tolérance aux fautes dans les systèmes d'exploitation temps-réel à sûreté critique. Technical Report 1142, IRISA, November 97.

[3] M.J. Fischer, N.A. Lynch, and M. Patterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM, 32(2)*, pages 374–382, April 1985.

[4] L.E. Moser, Y. Amir, P.M. Melliar-Smith, and D.A. Agarwal. Extended virtual synchrony. In *Proceedings of the 14th IEEE International Conference on Distributed Computing Systems*, pages 56–65, Poznan, Poland, June 1994.

[5] Emmanuelle Anceaume. *Algorithmique de fiabilisation de systèmes répartis*. PhD thesis, Université Paris XI Orsay, January 1993.

[6] J. Turek and D. Shasha. The many faces of consensus in distributed systems. *Computer*, 25(6):8–17, June 1992.

[7] T.D. Chandra, V. Hadzilacos, and S. Toueg. The weakest failure detector for solving consensus. In *Journal of the ACM*, 1996. Version préliminaire dans 11th ACM Symposium on Principles of Distributed Compulting, Août 1992.

[8] H. Kopetz and G. Grünsteidl. Ttp – a protocol for fault-tolerant real-time systems. In *IEEE COMPUTER*, University of Vienna, January 1994.

[9] J. Chang and N. Maxemchuk. Reliable broadcast protocols. In *ACM Transactions on Computer Systems*, volume 2 (3), August 1984.

[10] L. Peterson, N. Buchholz, and R. Schlichting. Preserving and using context information in interprocess communication. In *ACM Transaction on Computer Systems*, volume 7 (3), August 1988.

[11] K. Birman, A. Schiper, and P. Stephenson. Fast causal multicast. Technical Report TR-1105, Cornell csd, April 1990.

[12] M.F. Kaashoek and A.S. Tanenbaum. Efficient reliable group communication for distributed systems. Technical report, Vrije Universiteit Amsterdam, ftp://ftp.cs.vu.nl/pub/papers/amoeba/group94.ps.Z, 1994.

[13] D. Dolev and D. Malki. The transis approach to high availability cluster communication. In *Communications of ACM (ftp://ftp.cs.huji.ac.il/users/transis/cacm-96-4.ps)*, volume 39 (4), The Hebrew University of Jerusalem, April 1996.

[14] T. Abdelzahner, A. Shaikh, F. Jahanian, and K.G. Shin. Rtcast: A lightweight multicast for real-time process groups. In *IEEE Real-Time Technology and Applications Symposium*, Boston, MA, June 1996.

[15] L.E. Moser, P.M. Melliar-Smith, D.A. Agarwal, R.K. Budhia, C.A. Lingley-Papadopoulos, and T.P. Archambault. The totem system. In *Proceedings of the 25th International Symposium on Fault Tolerant Computing*, pages 61–66, Pasedena, CA, 95.

[16] Matthew Clegg. *Kernel Services for Supporting Hard Real-Time Active Replication*. PhD thesis, San Diego University (CA), 1997.

[17] F. Cristian. Synchronous atomic broadcast for redundant channels. Technical Report RJ 7203, IBM Research, December 1989.

[18] R. Friedman and R. van Renesse. Strong and weak virtual synchrony in horus, August 1995.

[19] D.A. Agarwal. *Totem: A Reliable Ordered Delivery Protocol for Interconnected Local-Area Networks*. PhD thesis, Department of Electrical and Computer Engineering, 94.

[20] Y. Amir, L.E. Moser, P.M. Melliar-Smith, D.A. Agarwal, and P. Ciarfella. The totem single-ring ordering and membership protocol. In *ACM Transactions on Computer Systems 13, 4*, pages 311–342, November 1995.

[21] Jean Walrand. *Note on FDDI*. http://www.path.berkeley.edu/˜wlr/228/FDDI.html.