N° d'ordre : 412

Année 2005

THÈSE

présentée en vue de
l'obtention du titre de

DOCTEUR
de
L'ÉCOLE NATIONALE SUPÉRIEURE
DE L'AÉRONAUTIQUE ET DE L'ESPACE

ÉCOLE DOCTORALE : Informatique et télécommunications
SPÉCIALITÉ : Informatique
par

Benoît BRÉHOLÉE

Interconnexion de simulations distribuées HLA

Soutenue le 8 mars 2005 devant le jury :

MM.	B.	LÉCUSSAN	Président
	Ch.	CHASSOT	
	N.	GIAMBIASI	Rapporteur
Mme	Ch.	MORIN	Rapporteur
M.	P.	SIRON	Directeur de thèse

Thèse préparée au sein du Département Traitement de l'Information et Modélisation de l'ONERA-Centre de Toulouse.

Remerciements

Cette thèse intitulée *Interconnexion de simulations distribuées HLA* a été préparée à l'Office National d'Études et de Recherches Aérospatiales, centre de Toulouse, au Département Traitement de l'Information et Modélisation.

Je tiens à remercier tout particulièrement mon directeur de thèse, Pierre SIRON, pour ses conseils et son soutien tout au long de ces années.

Je remercie également les autres membres du jury pour leurs remarques et leurs encouragements : Bernard LECUSSAN qui a présidé cette commission d'examen, Norbert GIAMBIASI et Christine MORIN pour l'intérêt qu'ils ont porté à ce travail en acceptant d'être rapporteurs de cette thèse, et enfin Christophe CHASSOT qui a également examiné ce mémoire.

Enfin, je souhaite aussi saluer les membres du DTIM pour leur accueil chaleureux et remercier les chercheurs, ingénieurs et étudiants avec qui j'ai travaillé sur le thème de HLA et du projet CERTI.

Table des matières

Table des figures

Introduction

La simulation consiste à étudier le comportement d'un système à partir de celui d'un modèle. L'utilisation de la simulation est nécessaire ou préférable dans de nombreuses situations : elle présente un intérêt économique en permettant de travailler sur un modèle plus accessible que le système, de multiplier plus aisément les expériences, ou de manipuler des prototypes entièrement virtuels lors d'un travail de conception.

Lorsque le modèle est particulièrement complexe ou détaillé, et qu'il nécessite le traitement d'une très grande quantité de données, l'approche séquentielle de la simulation peut se révéler insuffisante : il est alors indispensable de distribuer l'exécution du modèle sur un grand nombre de machines ou de processeurs. La simulation distribuée, en répondant à ce besoin, se trouve confrontée aux nécessités de modèles de taille importante : la modélisation monolithique devient inefficace, la réutilisation de composants doit être facilitée en raison de leur grand nombre et de leur coût de développement, et enfin l'interopérabilité entre composants provenant de disciplines différentes doit être privilégiée.

Des architectures de simulation distribuée à événements discrets ont été proposées dans ce but, en particulier HLA[1], initialement développée par le DoD[2] étasunien et devenue une norme OMG et IEEE. Cette architecture décompose la simulation en *fédérés*, des simulateurs individuels, interconnectés par l'intermédiaire d'un noyau, le RTI (Run Time Infrastructure). La simulation globale produite est alors appelée *fédération*. Cette structure incite bien à la création de simulations modulaires ; et par son caractère de norme ne ciblant pas un domaine d'application particulier, HLA facilite l'interopérabilité et la réutilisation de ces fédérés. Cependant ces avantages peuvent montrer des limites, en particulier dans le cas de simulations à grande échelle. L'aspect modulaire se réduit à une décomposition sans proposer de véritable hiérarchisation : ceci n'est pas toujours satisfaisant du point de vue de la

[1]High-Level Architecture
[2]Department of Defense

conception, de la sécurité, mais aussi des performances. De même, si HLA apporte une interopérabilité, celle-ci apparaît parfois trop fastidieuse.

Si ces différents points constituent des avancées significatives, leur mise en œuvre dans HLA présente des limites, et laisse en perspective des travaux visant à les améliorer. La notion classique de « domaine », absente des simulations monolithiques, joue un rôle essentiel dans chacun de ces avantages. L'objectif de cette thèse est donc de reprendre cette notion, et de la renforcer afin d'envisager l'amélioration de ces objectifs initiaux de HLA, en particulier dans le cadre des simulations développées avec le prototype de RTI de l'ONERA, CERTI.

Plan du mémoire

Ce mémoire est composé de sept chapitres. Les trois premiers proposent un état de l'art tandis que les suivants sont consacrés à notre contribution.

Le premier chapitre présente la simulation distribuée à événements discrets, et en particulier les aspects intervenant directement dans l'utilisation de HLA comme les différentes approches de la gestion du temps.

Le deuxième chapitre est principalement consacré à HLA : son architecture, sa définition, ses caractéristiques et son utilisation pour le concepteur de simulateur. Quelques architectures ayant précédé HLA sont également présentées.

Le troisième chapitre concerne la notion de domaine, les objectifs et intérêts qui y sont associés, et différents travaux la concernant à des niveaux très différents : d'une abstraction allant au-delà ce que prévoit la norme (multirésolution, interfédérations) aux questions de plus bas niveau concernant les protocoles et l'architecture, en passant par les services HLA qui lui sont dédiés.

La seconde partie commence par présenter, au chapitre 4, le prototype de RTI réalisé par l'ONERA : son architecture et l'histoire de son développement, ainsi que notre contribution au cours de sa diffusion en tant que logiciel libre.

Les chapitres suivants reviennent sur la notion de domaine, dans le cas particulier des interfédérations : des interconnexions de fédérations. Nous les appréhendons tout d'abord au sens le plus général au chapitre 5 et présentons différentes manières de les réaliser ainsi que leurs topologies possibles.

Le sixième chapitre est consacré au cas le plus fréquent d'interfédération : les ponts de fédérations. Nous indiquons dans le détail le comportement qu'ils doivent suivre en tant que participants à des fédérations HLA, afin de réaliser des interfédérations cohérentes. La manière de réaliser ces comportements est aussi discutée, car elle dépend fortement des objectifs et des RTI disponibles.

Puis, au dernier chapitre, nous terminons par présenter nos choix lors de la conception d'un pont de fédérations, et les résultats obtenus avec le prototype réalisé.

Première partie

État de l'art

Chapitre 1

Simulation distribuée

1.1 Généralités, objectifs

La simulation consiste à représenter, imiter aussi fidèlement que possible un système ou phénomène réel, à l'aide d'un modèle. Le modèle apporte de nombreux avantages sur le système réel, en particulier une plus grande simplicité, que cela concerne sa définition, avec un niveau d'abstraction adapté à l'étude, ou sa manipulation expérimentale. Certains systèmes peuvent en effet être difficilement manipulables, ou même ne pas l'être du tout : prototypes à réaliser, phénomènes que l'observation perturberait, ou dans le cadre de conditions d'observation difficilement reproductibles. Après avoir évoqué plus en détail les intérêts et les objectifs de la simulation, nous reviendrons sur la démarche de conception du modèle et d'exécution de la simulation.

Il est possible de distinguer deux grandes catégories d'objectifs dans la simulation. Il peut s'agir de concevoir un modèle d'un système physique, avec comme principal objectif la réalisation de ce modèle : la définition d'un modèle conforme au système et à l'abstraction recherchée. Ceci peut aussi bien concerner des simulations dans lesquelles le système à représenter existe dans la réalité (par exemple à des fins d'étude, d'anticipation) que des simulations dans lesquelles ce système n'existe pas encore (par exemple dans le cas du prototypage).

L'objet de la simulation peut également être l'étude du système réel, l'étude, l'observation ou l'utilisation de celui-ci au travers d'un modèle éprouvé. Bien sûr ce type de simulation peut être réalisé à l'aide de modèles conçus précédemment. Les deux catégories d'objectifs sont complémentaires, les deux pouvant être nécessaires au cours d'une étude donnée. Par exemple une première étape peut consister à concevoir un modèle, validé à l'aide d'un ensemble de résultats connus et observés dans la réalité. Une deuxième étape

consiste alors à utiliser le modèle réalisé dans le cadre de nouveaux scénarios, ou pour la réalisation d'autres modèles.

Les domaines d'application de la simulation sont nombreux. Comme nous l'avons vu, la pertinence de l'emploi de la simulation peut provenir de la difficulté, voire l'impossibilité, de manipuler le système étudié. On trouve des exemples dans le domaine de la météorologie, dans l'anticipation de trafic routier ou de télécommunications.

Un autre exemple est celui de la conception de prototypes pour architectures parallèles. Un prototype simulé, sur une architecture simulée, sera plus simple et surtout moins coûteux à réaliser et à tester que son équivalent réel.

De manière plus générale, l'emploi de la simulation peut être motivé par de nombreux objectifs et contraintes :

- le prototypage, l'évaluation de performances ;
- la création de modèles ;
- l'étude ou la validation d'un système dans des conditions particulières (par exemple difficiles ou coûteuses à mettre en place, conditions aux limites, sans perturbation par l'observateur) ;
- l'anticipation, l'aide à la prise de décision ;
- l'entraînement.

Bien sûr un projet peut faire appel à plusieurs de ces éléments, souvent complémentaires.

Lorsqu'il s'agit d'étudier un système complexe, lorsque le modèle choisi pour représenter le système est lui-même complexe, ou lorsque la quantité de données impliquées devient très importante, une simulation séquentielle (c'est-à-dire fondée sur un seul processeur, une seule machine) devient inadaptée : l'exécution d'une telle simulation est lente et inefficace en raison du manque de mémoire ou de puissance de calcul. Dans ce cas, la simulation parallèle, s'exécutant sur un plus grand nombre de processeurs et ayant accès à une mémoire totale plus importante, doit être envisagée. Le passage d'une simulation séquentielle à une simulation parallèle ne se fait pas sans introduire des inconvénients, en particulier des temps de traitement supplémentaires dus aux mécanismes du parallélisme, tels que la synchronisation. La parallélisation n'en reste pas moins une solution permettant un accroissement global des performances.

1.2 Notions de base

Zeigler [ZPK00] a proposé la définition de quatre grandes notions de base de la simulation : le système étudié, son modèle, l'exécution de ce modèle, et

le cadre de cette exécution.

Tout d'abord, le *système*[1] est l'objet que l'on cherche à représenter, étudier. Il s'agit souvent d'un système physique, mais celui-ci n'est pas nécessairement réel. Par exemple dans la simulation d'un véhicule à réaliser, celui-ci n'existe pas encore, il s'agit de le concevoir avant de réaliser un prototype. Certains éléments peuvent cependant exister (des constituants, des éléments extérieurs), d'autres non. Si la différence entre objet réel et virtuel a peu d'importance pour le simulateur chargé de les représenter, le processus de conception du modèle utilisé par celui-ci va dépendre de la nature de cet objet.

La deuxième notion essentielle est le *modèle*, c'est-à-dire l'objet avec lequel il s'agit de travailler. On peut le définir comme l'abstraction qui sera manipulée, animée lors de la simulation. Le modèle est une représentation du système, plus simple mais suffisamment pertinente pour convenir à l'étude à mener. Le travail de modélisation n'est pas spécifique à la simulation, il s'agit de la même démarche qui dans d'autres disciplines propose de concevoir et valider le modèle d'un système ou d'un phénomène, puis de travailler avec. Pour la simulation comme pour d'autres domaines, l'intérêt de travailler avec un modèle peut provenir de la difficulté à travailler avec le système réel, que ce soit pour des problèmes pratiques ou de coûts.

Le modèle peut être défini de différentes manières. Il peut être un ensemble de règles, d'équations, de contraintes permettant de définir un comportement possible. Le comportement observable du modèle doit être similaire à celui du système, ou l'imiter avec une précision suffisante.

Un *simulateur* est défini comme un agent capable d'obéir aux instructions d'un modèle, et donc d'engendrer réellement le comportement. En effet, à moins que le modèle soit relativement simple ou s'y prête tout particulièrement, il n'est pas possible de déterminer aisément tous les comportements d'un modèle à partir de sa définition. Il s'agit d'exécuter le modèle, de suivre des scénarios possibles de son évolution et non d'envisager toutes les possibilités. Un seul scénario est suivi au cours d'une exécution.

Enfin, le *cadre expérimental* précise les conditions d'observation, spécifie un contexte particulier correspondant à la situation étudiée. On fait ainsi la distinction entre le système étudié à l'aide de son modèle et l'environnement dans lequel il se trouve. Nous indiquons cette dernière entité proposée par Zeigler cependant nous n'insisterons pas sur son rôle : en effet, nous nous

[1]Les autres termes souvent rencontrés pour désigner le système sont : « système source », « système représenté », « monde représenté », « système physique », etc.

intéressons à la simulation proprement dite, à son exécution, c'est-à-dire à une phase postérieure aux choix du système et du modèle et même souvent du simulateur. Suivant cette approche, le cadre expérimental est une donnée du problème qui peut être vue comme appartenant au système, et incluse dans le modèle de système global. D'une manière plus générale cependant, la mise en évidence du cadre expérimental est essentielle lors de l'étude d'objets réels, la conception de prototypes, etc.

Formalismes

Comme nous l'avons vu, le modèle est une abstraction du système. Une abstraction parmi d'autres, mais choisie parce qu'elle convient à l'étude à réaliser. Les modèles possibles d'un système sont a priori nombreux, et le sont d'autant plus que le système est complexe. Le choix s'effectue en fonction de nombreux critères que l'on retrouve principalement dans les objectifs de la simulation et les capacités des machines à les supporter, en terme de ressources (mémoire et processeurs). Un choix similaire est effectué lorsqu'il s'agit de mettre en œuvre le simulateur proprement dit. Si le modèle est défini, son évolution, en particulier vis-à-vis du temps, peut être exécutée suivant différentes approches. Ces approches sont appelées *formalismes*.

Par exemple les modèles à équations différentielles proposent des fonctions continues permettant de déterminer l'évolution de l'état du modèle en fonction du temps. Ces modèles sont adaptés à la description de grandeurs continues. Le temps y est une variable similaire aux variables d'état, qui définissent entièrement l'état du modèle. Mais le même modèle peut être appréhendé avec un formalisme différent, par exemple un formalisme à temps discret. Un modèle à temps discret passe par des états ponctuels successifs, chacun associé à un instant en temps simulé. Ainsi un même modèle peut être exécuté par un simulateur soit à l'aide d'un formalisme à équations différentielles, soit avec un formalisme à temps discret. Ces formalismes correspondent aux deux grandes catégories de simulation : simulations à événements discrets et simulations continues [LK91].

En pratique, les contraintes du modèle ou les capacités du simulateur peuvent imposer un formalisme plutôt qu'un autre. Un modèle à temps discret s'impose lorsque la complexité du modèle ne permet pas d'employer un formalisme à équations différentielles. Des valeurs discrètes peuvent constituer une approximation suffisamment précise d'une grandeur continue d'un système. Dans la suite de ce mémoire, nous n'envisagerons que les simulations à base de modèles à temps discret.

1.3 Simulation à événements discrets

Nous nous intéressons donc aux simulations impliquant des modèles à temps discret. Dans ces simulations, les variables du modèle évoluent de manière discrète : une exécution est une suite d'états successifs du modèle (l'état étant l'ensemble des variables). Les changements d'état sont discrets, ils correspondent à un avancement dans le temps, et obéissent aux instructions du modèle. La valeur d'un nouvel état dépend en général des états passés.

1.3.1 Temps simulé

On représente cette évolution des états sur un axe appelé *temps simulé* : si le temps n'est pas une variable, il faut cependant que chaque événement de la simulation, chaque changement d'état, soit associé à un instant afin de représenter le temps dans la simulation. Le temps simulé est bien distinct du temps réel, ainsi que du temps du simulateur (le temps de son processus) : il avance plus ou moins rapidement suivant la vitesse d'exécution de la simulation, et dépend donc aussi bien du simulateur que du modèle.

Une simulation très complexe risque d'avoir pour conséquence un temps simulé avançant plus lentement que le temps réel. Dans ce cas, le calcul de quelques secondes de simulation nécessitera des minutes ou des heures. Inversement, si l'exécution a lieu rapidement, l'avancement du temps simulé est plus rapide que celui du temps réel. Alors on peut éventuellement synchroniser ces deux temps, comme dans le cas de l'entraînement.

Il faut cependant remarquer que ces comparaisons entre vitesses d'avancement dans le temps n'ont aucune raison d'être vérifiées tout au long d'une simulation. Par exemple une phase d'une simulation peut représenter le déroulement d'un phénomène complexe, qui provoquera un avancement très lent du temps simulé, tandis que la phase suivante, plus simple, avancera plus rapidement. Le temps simulé n'est pas uniforme et donc dans le cas de simulation d'entraînement évoqué précédemment, le temps simulé doit être plus rapide que le temps réel à tout instant, et non en moyenne.

1.3.2 Événements

Nous revenons sur la définition de quelques notions évoquées précédemment.

Événement Un événement désigne une action ponctuelle ayant lieu dans la simulation. Il peut s'agir d'un changement d'état ou la communication de toute information ponctuelle.

Estampille L'estampille[2] d'un événement désigne l'instant de son occurrence en temps simulé : l'événement doit se produire, c'est-à-dire être exécuté par le simulateur, à cet instant du temps simulé.

1.3.3 Le temps discret

Le temps réel est une grandeur qui évolue de manière continue. Les modèles continus sont donc adaptés à sa représentation, mais ce formalisme ne convient pas aux modèles trop complexes. On peut alors envisager de représenter l'évolution du modèle de manière discrète en assimilant un changement continu à un ensemble fini de changements ponctuels. Alors que le temps est continu, les événements représentés doivent être ponctuels et en particulier les phénomènes continus doivent être discrétisés.

Cette discrétisation doit être effectuée en tenant compte de la simulation : la représentation sera d'autant plus proche de la réalité que le nombre d'instants considérés sera élévé. Avec un échantillonnage trop faible, des phénomènes observables en temps continu peuvent ne plus apparaître. À l'inverse, une trop grande résolution peut n'apporter que peu de précision supplémentaire, mais significativement augmenter les ressources nécessaires à l'exécution de la simulation. La précision sera excessive par exemple dans le cas de grandeurs continues qu'il serait aisé d'interpoler à partir des dernières valeurs connues.

Les modèles à temps discrets se fondent donc sur la mise en œuvre de *pas de temps*, au cours desquels ont lieu les seules évolutions possibles des variables du modèle. Il s'agit de représenter successivement des états du modèle, à différents instants. La figure 1.1 montre un exemple de modèle dont l'état varie lors du passage par des instants successifs. En dehors de l'aspect discret, rien n'est imposé à l'évolution du temps.

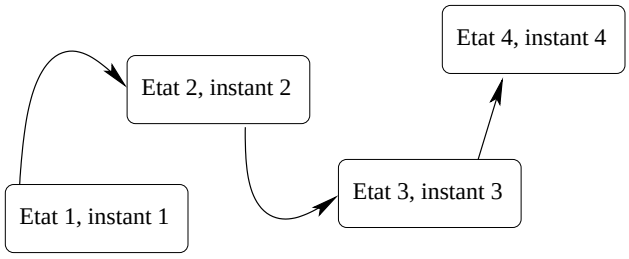

FIG. 1.1 – Temps discret

On distingue deux approches de cette gestion discrète du temps : syn-

[2]on emploie aussi : « instant », « date », ou « estampille temporelle »

chrone et asynchrone.

Temps discret synchrone

Dans le cas synchrone[3], les instants successifs du temps simulé sont séparés par un intervalle de taille fixe : le pas de temps. Pratiquement, on ne considère pas que les événements ont lieu en ces instants, mais que les événements ayant lieu dans un intervalle apparaissent à la fin de cet intervalle. La seule information est qu'ils ont eu lieu au cours du dernier intervalle de temps. Sur la figure 1.2, la simulation avance de l'instant 0 à l'instant T, puis $2T$, $3T$, etc. Les événements e_2, e_3 et e_4 ont lieu au cours du deuxième intervalle et seront perçus à la fin de celui-ci, au temps $2T$.

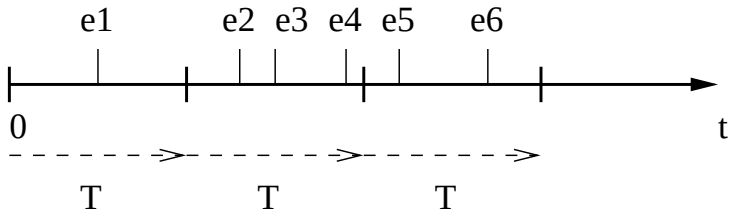

FIG. 1.2 – Événements discrets : approche synchrone

Dans ce type de modèle, il est important de choisir correctement la valeur du pas de temps. Si cette valeur est trop grande, alors la représentation des événements risque d'être incorrecte, en particulier dans le cas de phénomènes continus périodiques. Les relations de cause à effet ne peuvent être exprimées qu'entre un instant (événement « apparition de la cause ») et l'instant suivant (événement « manifestation de l'effet »). Tout enchaînement de relations de cause à effet ne peut donc avoir lieu qu'au rythme d'une action par pas de temps.

Un pas de temps trop court apporte également des inconvénients. S'il existe un grand nombre de pas de temps séparant deux événements successifs, le simulateur traite un ensemble d'avancées dans le temps qui auraient pu se résumer à une seule. Dans le cas de pas de temps petits par rapport aux intervalles entre occurrences d'événements, le temps de traitement de ces avancées non pertinentes n'est pas négligeable et détériore les performances.

[3]ou « dirigé par le temps », ou « à pas de temps fixe »

Temps discret asynchrone

L'approche asynchrone[4] par contre n'impose aucun intervalle minimal. La simulation avance de manière variable, en suivant les dates des événements successifs. Sur la figure 1.3, chaque avancée dans le temps correspond au déplacement d'une occurrence d'événement à une autre.

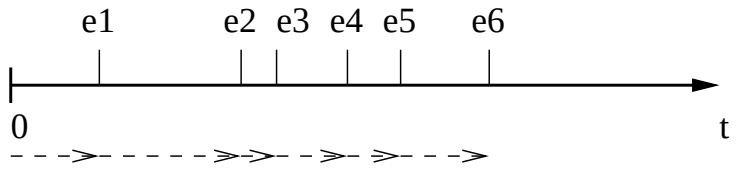

FIG. 1.3 – Événements discrets : approche asynchrone

On trouve bien sûr comme avantages de cette approche ce qui faisait défaut à l'approche synchrone : seuls les instants significatifs, c'est-à-dire associés à des événements, sont pris en compte. Si une longue période de temps simulé ne connaît aucun événement, le simulateur avancera dans le temps en couvrant toute cette période en une seule étape. De plus il ne se pose aucun problème de précision de la représentation puisque chaque événement est bien vu à l'instant auquel il est censé avoir lieu selon le modèle.

Cependant la gestion d'un instant par événement peut se faire au détriment des performances quand les événements sont nombreux. L'approche synchrone permet de rassembler les événements proches en les associant à un unique instant. Si ceci ne constitue pas une erreur d'approximation dissuasive, l'approche synchrone peut alors donner de meilleures performances.

1.3.4 L'exécution des événements

Dans un simulateur séquentiel, lorsque le modèle a évolué jusqu'à une date donnée, le simulateur a accès à l'ensemble des données courantes du modèle, et à la définition de ce modèle. Il peut donc calculer un ensemble d'événements à venir, et ainsi déterminer l'état de l'instant suivant. Mais tous les événements déterminés ne concernent pas uniquement l'instant suivant. L'ordre dans lequel ils ont été déterminés ne correspond pas a priori à leur ordre chronologique en temps simulé.

Le simulateur doit exécuter les instructions du modèle afin de faire apparaître un scénario d'une évolution possible de ce modèle. Le temps simulé n'est pas directement lié au temps réel, mais il doit en garder certaines propriétés. En particulier, les événements doivent avoir lieu (c'est-à-dire être

[4]ou « dirigée par les événements », « orientée événement »

exécutés) dans l'ordre croissant de leurs estampilles. Si deux événements E et E' ont lieu respectivement aux instants t et t' et que $t < t'$, alors E doit être exécuté avant E'. Ne pas respecter cette règle constitue une *faute temporelle*.

Comme le simulateur séquentiel a accès à l'ensemble des événements à venir déjà déterminés, il est en mesure de les ordonner afin de les exécuter correctement, sans faute temporelle. Le tri des événements est facilité par l'aspect séquentiel du simulateur : toutes les informations relatives à ces événements sont accessibles. Par contre le cas de la simulation parallèle introduit une répartition du simulateur en plusieurs processus pour lesquels un tel tri n'est pas immédiat (leur connaissance des événements à venir étant partielle).

1.4 Simulation parallèle à événements discrets

1.4.1 L'approche distribuée

Les éléments que nous avons présentés jusqu'à présent constituent les bases de la simulation à événements discrets séquentielle. Ces simulations, historiquement les premières mises en œuvre, sont toutefois fortement limitées par la puissance de calcul disponible sur la machine supportant l'exécution. Afin d'appréhender des modèles et des exécutions plus complexes, l'approche parallèle s'est imposée. Elle reprend les notions essentielles de la simulation à événements discrets vues précédemment.

Souvent, les termes *parallèle* et *distribué* sont employés de manière interchangeable. Cependant, le terme *parallèle* est préféré lorsque la parallélisation provient d'un besoin en puissance de calcul, tandis que le terme *distribué* est plus adapté lorsqu'une répartition géographique de la simulation est nécessaire. Bien sûr, certaines situations font apparaître ces deux besoins.

La simulation parallèle ou distribuée consiste à répartir l'exécution d'une simulation sur un ensemble de processeurs. L'ensemble des processus simulateurs concurrents peut être vu comme un unique simulateur : son rôle est toujours d'animer un modèle donné.

On retrouve bien sûr la démarche évoquée précédemment, avec le choix d'un modèle adapté et l'exécution de celui-ci par un simulateur. Mais cette fois, modèle et exécution sont répartis. Concernant l'exécution, le simulateur global est constitué d'un ensemble de *processus logiques* concurrents, chacun étant chargé d'exécuter une partie du modèle. Cette parallélisation nécessite une synchronisation entre processus logiques afin que l'exécution reste cohérente. Le modèle doit quant à lui être fragmenté afin de permettre à chaque processus logique d'en gérer une partie. L'ensemble du modèle est

ainsi pris en compte par l'ensemble des processus logiques. Cette démarche peut être réalisée en déterminant tout d'abord quels sont les processus physiques qui composent le système simulé, chaque processus logique étant alors défini comme simulateur de l'un d'entre eux.

Un processus logique est donc un simulateur se consacrant à un ensemble de données modélisées. Il a la responsabilité d'engendrer les événements liés à ces entités du modèle, mais aussi d'appliquer les conséquences des événements qu'il observe, c'est-à-dire qui ont eu lieu dans l'ensemble de la simulation. Un processus logique a une vue limitée à ses données et aux événements reçus. Il n'existe aucune mémoire partagée entre les processus logiques. Les échanges de données sont réalisés à l'aide de messages estampillés : tout événement donne lieu à l'envoi d'un message contenant sa description et sa date d'occurrence. On peut ainsi définir :

Message Information transmise d'un processus logique à un autre.

Estampille Valeur représentant une date d'occurrence associée à un message. L'estampille d'un message transportant la description d'un événement correspond à l'estampille de cet événement.

Enfin, il nous faut revenir sur la notion de temps simulé. Comme nous l'avons vu en §1.3.1, il s'agit du temps dans la simulation, et il s'écoule indépendamment du temps réel. En pratique il correspond à l'estampille de l'événement que le simulateur est en train d'exécuter. À un instant donné du temps réel, il n'y a donc qu'une seule valeur du temps simulé dans le cas de la simulation séquentielle. Ce n'est plus le cas lorsque la simulation est distribuée : chaque processus logique avance dans le temps simulé. Il est nécessaire de maintenir une synchronisation entre processus logiques pour la cohérence de la simulation, mais à un instant donné du temps réel, tous n'en sont pas au même point du temps simulé. On désigne alors par *temps logique* d'un processus logique son avancement dans le temps simulé.

1.4.2 La causalité

Le problème de causalité représente l'incohérence qui consiste, pour un processus logique, à tenter d'influer sur le passé d'un autre processus logique. Cette situation se produit dès qu'un message possède une estampille de valeur inférieure au temps logique du processus logique destinataire. Un exemple proposé par Fujimoto [Fuj98] est le suivant : trois processus concurrents sont exécutés, chacun suivant son propre temps simulé. Sur la figure 1.4, le premier processus simule un événement T (le tir d'un projectile). Cet

événement est envoyé aux autres processus logiques à l'aide d'un message, et est reçu plus tard. Dans l'exemple, cette réception est marquée (T) et a lieu pour le processus 3 puis plus tard pour le processus 2. L'événement D (la destruction d'une cible gérée par le processus 3), conséquence de l'événement T est aussitôt exécuté dans le processus 3, et transmis aux autres processus (de même, la réception est notée (D) sur la figure). Les délais des échanges de messages de cet exemple font que l'événement D est vu dans le processus 2 avant que l'événement T n'ait été reçu par ce processus. Dans ce processus, la conséquence apparaît avant la cause, la destruction a lieu avant le tir.

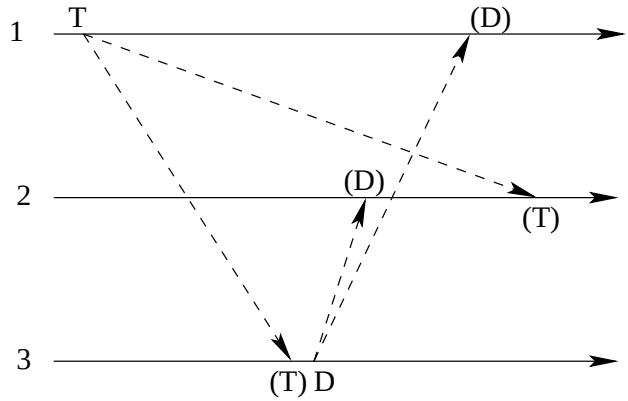

FIG. 1.4 – Problème de causalité

Il s'agit donc en simulation distribuée de maintenir une cohérence entre les événements en respectant la causalité.

1.5 La gestion du temps

En simulation distribuée, les processus logiques exécutent chacun une partie de la simulation globale, et communiquent par échanges de messages afin de tenir compte de l'évolution de leurs modèles respectifs. La gestion du temps simulé, et en particulier le respect des contraintes de causalité, nécessite des algorithmes assurant la cohérence de ces échanges.

Ainsi, pour respecter la contrainte de causalité, ce qui revient à s'assurer que deux événements seront exécutés dans l'ordre imposé par leurs estampilles, il suffit que dans un processus logique les événements soient exécutés dans l'ordre croissant de leurs estampilles [Fuj90]. Pour cela, il existe principalement deux approches, dites optimiste et conservative.

L'approche optimiste consiste à ne tenir aucun compte des contraintes de causalité, mais à détecter toute faute temporelle et à agir en conséquence (en particulier par un « retour en arrière »).

L'approche conservative a pour objectif d'éviter toute erreur de causalité, c'est-à-dire qu'un événement ne sera exécuté qu'avec l'assurance que tous les événements antérieurs ont effectivement été traités. À aucun moment il ne peut être demandé à un processus logique de revenir dans son passé.

Ces deux approches ont pour inconvénient la mise en œuvre de mécanismes de synchronisation, coûteux en temps de traitement. Ces délais diffèrent d'une approche à l'autre puisque dans le cas de l'approche conservative, on les retrouve pour chaque événement, tandis que dans le cas optimiste, ils ne sont présents qu'en cas d'incohérence.

1.5.1 L'approche conservative

Afin de conserver la cohérence temporelle, l'approche conservative (ou pessimiste) propose de maintenir en permanence un ordre correct des exécutions d'événements, en s'assurant qu'aucun message futur n'aura d'estampille inférieure à son temps logique. Aucun avancement dans le temps laissant le risque d'une faute temporelle n'est toléré, quitte à ce que cela entraîne un blocage.

Le *lookahead*

Avant de présenter les approches conservatives, nous avons besoin d'introduire la notion de *lookahead*, définie comme une capacité de prévision d'un simulateur. Si un simulateur à un instant t est en mesure de dire qu'il ne produira pas d'événement avant l'instant $t + L$, alors L est appelé le *lookahead*.

Comme nous le verrons plus loin, une valeur élevée des *lookaheads* des processus logiques améliore les performances de la simulation. Certaines situations assurent l'existence d'une valeur strictement positive du *lookahead* bien qu'il n'y en ait aucune garantie dans le cas général :

- la latence du modèle vis-à-vis des événements susceptibles de l'influencer ;
- le formalisme employé : dans le cas de modèles à pas de temps fixe, le *lookahead* vaut au moins la valeur du pas de temps, pour chaque processus logique.

Chandy, Misra et Bryant

Chandy et Misra [CM79], ainsi que Bryant [Bry77] ont proposé un algorithme conservatif de gestion du temps.

Dans cette proposition, il n'y a pas de synchronisation régulière des processus logiques : la synchronisation de fait est réalisée à l'aide des messages

estampillés servant à décrire les événements. Il s'agit alors, pour chaque processus logique, d'envoyer les messages par ordre croissant d'estampille, et d'autre part d'exécuter les événements également produits et reçus dans cet ordre. Pour ce deuxième point, un tri des événements à exécuter est nécessaire.

Chaque processus logique communique avec un autre processus logique par un canal. Dans chaque canal, les messages arrivent par ordre croissant d'estampille : c'est ainsi qu'ils sont envoyés, et on considère que le transport du message maintient cet ordre. À tout instant, un processus logique a donc accès à un ensemble de messages qui déterminent l'événement futur d'estampille minimale : chaque canal fournissant la valeur minimale des estampilles à venir, l'estampille minimale des canaux est aussi l'estampille minimale de tous les messages à venir. Le processus logique peut alors exécuter l'événement associé en ayant l'assurance qu'aucun message d'estampille inférieure n'arrivera par la suite.

Cependant, lorsque des canaux sont vides, la dernière information est l'estampille du dernier message arrivé. Il n'existe aucune information sur le message suivant, et il est alors impossible de déterminer avec certitude la prochaine estampille minimale tant qu'un nouveau message n'est pas arrivé. Cette situation conduit à un blocage : le processus logique attend jusqu'à l'arrivée d'un message dans le canal vide.

Ceci a donné lieu à d'autres propositions, basées sur cet algorithme de Chandy et Misra, et fournissant une solution au problème de blocage.

Principe du message NULL

Le principe du message NULL proposé dans [CM79] consiste à envoyer des messages ne correspondant pas à des événements. Ces messages possèdent une estampille, mais pas de contenu décrivant un événement d'où leur nom de message NULL. L'intérêt de cette estampille est qu'elle peut être utilisée comme précédemment pour déterminer l'estampille minimale du prochain événement : le processus logique émetteur assure qu'aucun message d'estampille inférieure ne sera envoyé ensuite.

Il existe plusieurs variantes : une solution consiste à envoyer un message NULL à tous les non-destinataires d'un message, afin que la totalité des processus logiques reçoivent l'information contenue dans l'estampille. Une autre solution consiste à envoyer un message NULL sur requête d'un processus logique qui cherche à faire cesser sa situation de blocage. L'approche systématique est plus consommatrice de messages, entraînant un ralentissement global. À l'inverse, l'approche par requête donne lieu à des messages supplémentaires, et donc des coûts, plus ponctuels, mais plus importants.

Détection d'interblocage

Enfin une autre approche fondée sur la méthode conservative de Chandy et Misra consiste à détecter l'interblocage une fois qu'il a eu lieu plutôt qu'à le prévenir. Cette solution proposée par Chandy et Misra [CM81] met en œuvre un processus ayant pour but de détecter tout interblocage en observant les processus à court de message (canal vide) puis en déterminant une suite acceptable de la simulation en désignant un processus capable de « reprendre ».

Algorithmes à fenêtre temporelle

Une approche différente de celle suivie par Chandy et Misra consiste à introduire l'idée de fenêtre temporelle : en ajoutant comme hypothèse qu'un processus logique ajoute un délai minimum à une estampille produite (un délai minimum de propagation), alors des processus logiques suffisamment « éloignés » ne pourront émettre d'événements concernant un intervalle de temps à venir suffisamment petit. Le principe des algorithmes à fenêtre temporelle est de réduire les événements à prendre en compte à ceux qui arrivent dans cet intervalle, ce qui en pratique réduit le nombre de processus logiques à prendre en compte (les plus éloignés peuvent être ignorés).

L'algorithme *Conservative Time Window* d'Ayani et Rajaei [AR92] en est un exemple.

L'algorithme du *Bounded Lag* de Lubachevsky [Lub88] utilise également une fenêtre temporelle. Dans cet algorithme, on ne considère qu'une seule file d'événements par processus logique. Les processus logiques susceptibles de fournir un événement à traiter dans l'immédiat se trouvent dans une zone de recherche (arbitrairement limitée) du fait de l'existence d'un délai de propagation. Lubachevsky propose alors une estimation de la date du prochain message, en fonction des messages en attente et des délais de propagation entre processus logiques.

Le principe de causalité est respecté, et il n'existe pas de situation de blocage. Cependant, des phases de resynchronisation sont nécessaires : il faut diffuser à tous les processus une valeur d'horloge globale de la simulation (qui est la borne inférieure globale de la fenêtre).

1.5.2 L'approche spéculative

L'approche spéculative ou optimiste prend comme point de départ l'hypothèse inverse de l'approche conservative : aucune prévention des fautes temporelles n'est effectuée, par contre de telles fautes doivent être corrigées

lorsqu'elles ont lieu.

Le principal mécanisme optimiste, le *Time Warp*, a été proposé par Jefferson [Jef85]. Dans cette approche, un processus logique exécute les événements au fur et à mesure de leur réception, son temps logique étant alors la date d'occurrence du dernier message traité. Un mécanisme de retour en arrière ou *rollback* est exécuté lorsqu'une faute temporelle apparaît. Il est nécessaire pour cela que le processus logique sauvegarde régulièrement son état afin d'y revenir. Il faut aussi annuler tout message incorrect envoyé suite à cette exécution incorrecte : ceci est réalisé à l'aide d'anti-messages correspondants aux messages envoyés par erreur.

La solution la plus simple, dite *aggressive cancellation* consiste à annuler tous les messages postérieurs à l'instant de la faute temporelle. Jefferson a défini le *Global Virtual Time* qui représente la date minimale d'un retour en arrière. Il est calculé à partir des temps logiques des processus et des estampilles des messages en transit. Ainsi, un processus logique qui doit sauvegarder les états successifs de son passé, peut limiter ces sauvegardes grâce au GVT. Cette indication peut être utilisée pour tous les éléments sauvegardés par le processus logique, en particulier les messages et anti-messages traités ou émis.

Cette approche spéculative est efficace en l'absence de retour en arrière, ceux-ci en constituant le principal inconvénient, avec le besoin en sauvegardes, et le calcul et la diffusion du GVT qu'ils impliquent.

On trouve donc de nombreuses propositions d'amélioration. En particulier la *lazy cancellation* [Gaf88] dont le but est d'annuler le minimum de messages : certains ne sont pas annulés immédiatement s'ils sont engendrés même après correction. Cette approche peut néanmoins être aussi coûteuse, en particulier elle peut avoir pour effet de laisser des fautes temporelles qui seront à traiter, et que l'approche d'*aggressive cancellation* aurait supprimées.

1.5.3 Les approches mixtes et adaptatives

Il existe également des approches mixtes, visant à conjuguer les deux approches précédentes. Par exemple, une approche peut être avant tout spéculative avec l'éventualité de retours en arrière, tout en cherchant à favoriser un ordre correct des événements, comme l'imposent les approches conservatives. Inversement, l'éventualité de retours en arrière peut être envisagée pour faire face à une situation particulière d'une approche qui reste globalement conservative.

Ces approches correspondent à des algorithmes suivis tout au long de l'exécution de la simulation. Les approches adaptatives proposent par contre de varier entre les approches conservative et spéculative en fonction de l'avancement de la simulation.

Pour une présentation plus complète des différentes approches, et en particulier de ces méthodes mixtes et adaptatives introduites plus récemment, le lecteur pourra se référer à [NF94] qui en fait une synthèse exhaustive.

Chapitre 2

Architectures de simulation distribuée

Les principes de la simulation distribuée peuvent être mis en pratique dans la réalisation de simulateurs à l'aide de langages, d'outils, d'architectures facilitant plus ou moins leur mise en œuvre. Les langages généralistes de programmation peuvent être utilisés, le travail comprend alors d'une part la modélisation du système dans le programme, et d'autre part la gestion de son exécution.

Des langages et des logiciels de simulation ont été conçus afin de simplifier la tâche du développeur, et de lui permettre de se consacrer autant que possible à la partie métier de son simulateur.

Nous nous intéressons aux architectures qui ont pour objectif de fournir un tel support dans le domaine de la simulation distribuée à événements discrets.

Comme nous l'avons évoqué au chapitre précédent, la simulation parallèle ou distribuée peut être envisagée lorsque les ressources d'un simulateur installé sur une seule machine sont insuffisantes, en terme de puissance de calcul par exemple. Mais le besoin de répartir une simulation en processus logiques vient aussi d'un problème de conception : les simulateurs complexes et monolithiques présentent de nombreux inconvénients concernant leur maintenance et leur pérennité, et n'offrent pas de solution d'interopérabilité et de réutilisation. Tout en gardant un simulateur complexe, les architectures de simulation distribuée, et en particulier les architectures telles que HLA que nous aborderons plus loin, répondent à ces problèmes :

- la maintenance et la pérennité sont facilitées par une architecture modulaire ;
- la répartition en éléments plus simples aide à la réutilisation de cer-

taines parties d'une simulation dans une autre ;
- partageant des protocoles de communications communs, des simulations doivent pouvoir interopérer avec un travail d'adaptation minimal.

Pour être plus précis, l'objectif de *réutilisabilité* exprimé ci-dessus peut-être résumé par cette définition proposée par le DMSO[1] :

> Utilisation de ressources de modélisation et simulation (par exemple, modèles, simulations, bases de données, algorithmes, outils) pour des objectifs allant au-delà de ce pourquoi elles ont été développées. Ces réutilisations peuvent avoir lieu au sein d'une organisation ou de plusieurs, et peuvent concerner des domaines d'application différents.

Ceci constitue, avec l'interopérabilité, l'un des deux principaux objectifs de l'architecture HLA. Nous allons présenter, dans l'ordre chronologique, les architectures qui l'ont précédée et qui ont influencé sa définition.

De 1983 à 1990, le projet SIMNET [Pop89] de l'ARPA a mené à l'élaboration d'un protocole de communication utilisé pour l'échange d'information entre simulateurs de véhicules militaires, placés dans un environnement virtuel commun. Ces travaux se sont poursuivis par la création du protocole DIS.

2.1 DIS

DIS[2] [Com94] a été réalisé sur la base des principes concluants de SIMNET, avec l'objectif de proposer une norme plus complète favorisant l'interopérabilité entre simulateurs.

DIS propose des échanges de données asynchrones standardisés, avec la définition des PDU[3] qui permettent de transmettre des événements. Ces PDU suivent un format très précis : il n'en existe que 27 sortes, correspondant à des types d'événements déterminés et relevant dans l'ensemble du domaine militaire (collision, tir, détonation, ravitaillement, réparation, etc.). Chaque simulateur crée des événements, signale la création et les mises à jour d'entités.

Cette approche engendre un trafic réseau très élevé : non seulement la taille de chaque PDU est importante (souvent pour quelques informations

[1] Defense Modeling and Simulation Office
[2] Distributed Interactive Simulation
[3] Protocol Data Unit

pertinentes seulement) mais la cohérence de la simulation est assurée en envoyant chaque événement à *tous* les destinataires.

Il faut noter également que l'origine militaire de DIS a aussi orienté cette architecture vers les simulations d'entraînement : on s'attend à ce que l'homme prenne part à ces simulations. DIS ne propose pas de simulation en temps coordonné, les événements sont émis et doivent arriver au plus tôt aux destinataires afin que l'illusion donnée à l'observateur humain soit correcte.

DIS a donc apporté une norme permettant l'échange de données entre simulateurs, en définissant des événements plus ou moins génériques, et une manière de représenter l'environnement. Ces éléments sont essentiels à l'interopérabilité entre simulateurs, cependant DIS reste une architecture dédiée à un domaine d'application spécifique. HLA a constitué l'étape suivante en évitant d'imposer la sémantique des messages échangés.

2.2 ALSP

Dans le même temps, l'architecture ALSP[4] [WW94] a été développée. Celle-ci proposait une partie logicielle, une interface pour l'échange de messages et un ensemble de simulations existantes, utilisant ces composants. Contrairement à DIS, orientée entraînement, où l'homme agit directement dans la simulation, ALSP permet l'étude de scénarios exécutés en temps coordonné. Comme DIS, les travaux sur ALSP ont contribué au développement de HLA, qui autorise les deux approches.

La figure 2.1 présente l'architecture d'une simulation globale ALSP (ou *confédération*), regroupant deux simulations (ou *acteurs*). Cette architecture s'organise autour d'un composant central (bien qu'il puisse être présent sous forme de plusieurs instances), l'ABE[5], chargé des communications entre les différents composants de la simulation, les ACM[6]. C'est dans ces ACM que sont gérés le temps et les objets, tandis que les ABE ont un rôle de routage des données, avec comme objectif de minimiser le trafic réseau (il est possible d'avoir plusieurs ABE dans ce but).

Les ABE et les ACM sont reliés à des moniteurs :

- les ACT[7] sont attachés à un seul ACM ou ABE, et permettent notamment de visualiser ou modifier le statut de celui-ci ;

[4] Aggregated Level Simulation Protocol
[5] A Broadcast Emulator
[6] A Common Module
[7] A Control Terminal

– un CMT[8] ayant un rôle similaire, mais relatif à plusieurs ACM ou ABE, voire à tous ceux de la confédération.

Enfin, nous avons dit que les ACM correspondaient aux composants de la simulation. En effet, chaque simulateur individuel doit posséder un traducteur ALSP qui convertit les données de la simulation au format proposé par ALSP et les transmet via un ACM.

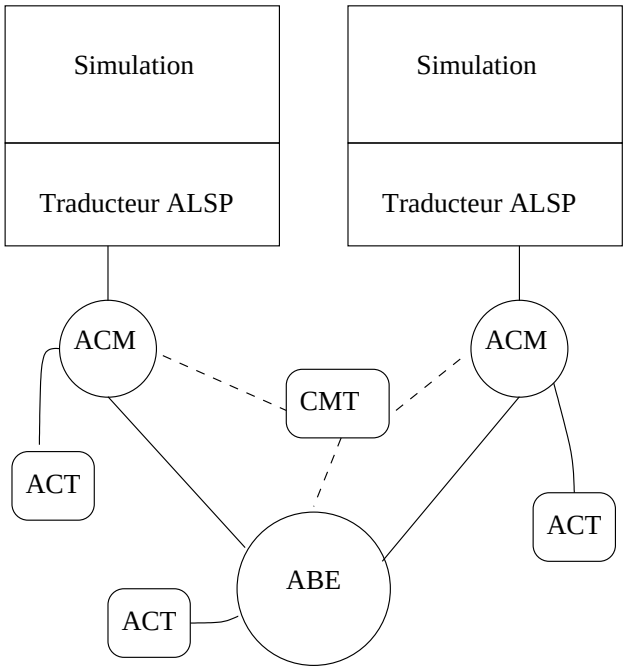

FIG. 2.1 – Architecture ALSP

Cette architecture est donc bien différente de celle de DIS, et annonce déjà celle de HLA.

2.3 HLA

HLA[9] est une architecture de simulation distribuée développée par le DMSO du DoD[10] étasunien, et dont la première version est parue en 1996. Cette spécification a fait l'objet d'une normalisation de l'OMG [OMG02] et de l'IEEE [IEE00b, IEE00a, IEE00c]. HLA est une spécification, et non une implémentation particulière ; le DMSO a néanmoins fourni une telle implémentation gratuitement de 1997 à 2002.

[8] Confederation Management Tool
[9] High-Level Architecture
[10] Department of Defense

HLA fait partie des solutions proposées par le DMSO pour répondre à un ensemble d'objectifs relatifs à sa stratégie de développement de nouvelles simulations [oDfAT95]. En effet, le premier de ces objectifs consiste à développer une infrastructure technique commune pour la modélisation et la simulation, favorisant l'interopérabilité et la réutilisabilité.

De 1996 à 1998, la spécification HLA a évolué, en particulier pour corriger certaines incohérences présentes dans la version 1.0. Par exemple, des services de distribution des données ayant pour but d'optimiser les échanges présentaient des incompatibilités avec leurs équivalents non optimisés[11]. La version finale de HLA proposée par le DMSO est donc le résultat de ce processus de correction et d'amélioration, qui n'a pas permis de conserver la compatibilité ascendante entre versions.

Cette version finale, HLA 1.3, a ensuite servi de base à la proposition de standardisation par l'IEEE, qui a donné la version « IEEE 1516 » de HLA. À nouveau, ces deux spécifications sont incompatibles, en raison des importantes modifications apportées. Les principes initiaux de HLA sont conservés, mais les formats de données dont désormais exprimés en XML, certains services sont ajoutés et d'autres simplifiés. Une synthèse des différences entre HLA 1.3 et HLA IEEE 1516 est présentée dans [LLL00].

Dans la suite de ce mémoire, nous nous intéresserons principalement à HLA 1.3.

2.3.1 Architecture générale

De nouveau, il s'agit de réunir plusieurs simulateurs individuels au sein d'une même exécution de simulation. HLA a eu comme objectif d'améliorer fortement l'interopérabilité et la réutilisation de ces simulateurs, par rapport à ses prédécesseurs. De plus, cette architecture n'est pas spécifique au domaine militaire, et propose au contraire de réunir au sein d'une même simulation des simulateurs provenant de métiers différents. Dans cette approche, il s'agit d'imposer le minimum de contraintes aux fédérés, afin que l'architecture soit suffisamment généraliste.

Une simulation globale HLA est appelée *fédération*, et un simulateur y participant est appelé *fédéré*. Il est toutefois possible que des fédérés ne soient pas des simulateurs au sens strict, par exemple s'ils sont simplement observateurs de l'exécution. Un fédéré peut aussi recevoir des données extérieures

[11] Il s'agit des groupes de service *Declaration Management* et *Data Distribution Management* que nous présentons plus loin.

correspondant à un objet réel ou un humain et ainsi le représenter dans la simulation.

À ces deux notions s'ajoute un autre composant essentiel d'une simulation HLA : le RTI[12], un moteur d'exécution chargé de fournir un ensemble de services aux fédérés, afin qu'ils puissent communiquer. En effet, chaque fédéré ne communique qu'avec ce RTI, et n'a aucune connaissance directe des autres fédérés. Tout échange de données d'un fédéré à un autre passe par le RTI, comme indiqué sur la figure 2.2 qui donne un exemple de simulation HLA.

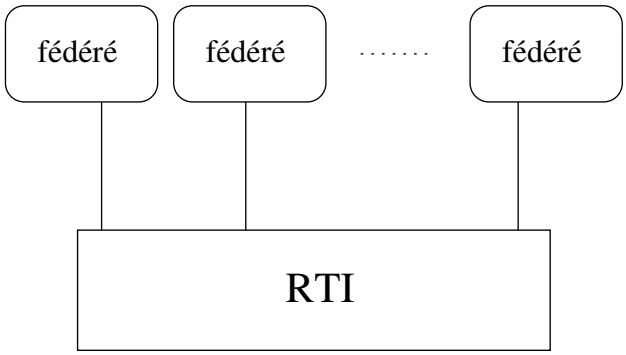

FIG. 2.2 – RTI et fédérés d'une simulation HLA

2.3.2 Définitions

Quelques définitions sont nécessaires à la suite de cette présentation de HLA. Nous commençons par les composants évoqués ci-dessus :

Fédération C'est une simulation HLA, prenant pour base un ensemble de simulateurs s'échangeant des informations ;

Fédéré Un simulateur individuel participant à une fédération — en général il simule un ensemble d'entités du monde représenté ;

RTI Le moteur d'exécution, noyau de la simulation — tout échange de données entre fédérés passe par lui ;

Objet Une entité du monde représenté, ayant une durée de vie, et des caractéristiques pouvant évoluer au cours de la simulation (par exemple, un véhicule et sa position) — ces caractéristiques sont appelées *attributs* ;

Interaction Une information ponctuelle émise par un fédéré, il s'agit d'événement ne correspondant pas a priori à un changement d'état d'un objet — comme les objets, leur valeur correspond à un ensemble de caractéristiques, appelées cette fois *paramètres* ;

[12]RunTime Infrastructure

Classe Comme en conception ou programmation objet, les objets semblables par leurs caractéristiques correspondent à des instances d'une même classe ; HLA utilise également cette dénomination, les types des attributs d'un objet sont définis dans sa classe d'objet, et il en va de même pour les classes d'interaction ;

Événement Un événement correspond à la définition générale vue au chapitre 1 : il s'agit ici d'une modification d'un objet ou de l'envoi d'une interaction ;

OMT ou *Object Model Template* : il s'agit d'un modèle objet employé pour décrire des entités, des caractéristiques d'une simulation ou d'un fédéré ;

FOM ou *Federation Object Model* : le modèle objet de la fédération décrit les données du monde représenté, c'est-à-dire les données vues et manipulées par les fédérés ;

SOM ou *Simulation Object Model* : le modèle objet des données propres à un fédéré.

2.3.3 Les documents de définition

La spécification HLA est constituée de trois documents de définition : les règles, le modèle objet, et l'interface.

Les règles

La spécification HLA 1.3 définit un ensemble de règles [DoD98c]. Leur respect est indispensable au bon déroulement d'une simulation. Les cinq premières règles concernent le fonctionnement global des fédérations.

Règle 1 Une fédération doit avoir un modèle objet de fédération conforme à l'OMT de HLA. Le FOM doit décrire toutes les données échangées au cours de l'exécution d'une fédération, et doit indiquer les conditions de ces échanges.

Règle 2 Toutes les représentations d'instances d'objet associées à la simulation doivent être dans les fédérés et non dans le RTI.

Règle 3 Au cours de l'exécution de la fédération, les données décrites dans le FOM ne peuvent faire l'objet d'échanges entre fédérés que par l'intermédiaire du RTI.

Règle 4 Au cours de l'exécution d'une fédération, les fédérés doivent interagir avec le RTI conformément à la spécification de l'interface.

Règle 5 Au cours de l'exécution d'une fédération, une instance d'attribut ne peut être la propriété que d'un seul fédéré à un instant donné.

Les cinq règles suivantes s'intéressent aux fédérés.

Règle 6 Un fédéré doit avoir un modèle objet de simulation conforme à l'OMT de HLA. Le SOM correspond aux informations qui pourront être rendues publiques au cours de l'exécution de la fédération.

Règle 7 Les fédérés doivent mettre à jour ou prendre en compte les modifications d'attributs, ainsi qu'envoyer ou recevoir des interactions conformément à la spécification indiquée dans leur SOM.

Règle 8 Les fédérés doivent pouvoir transférér ou accepter dynamiquement la propriété d'attributs au cours d'une exécution de fédération, conformément aux spécifications indiquées dans leur SOM.

Règle 9 Les fédérés doivent pouvoir faire varier les conditions sous lesquelles ils fournissent des mises à jour d'attributs.

Règle 10 Les fédérés doivent être capables de gérer un temps local d'une manière qui permet les échanges de données avec les autres membres de la fédération.

Ces différentes règles indiquent donc notamment :
- les exigences en termes de documents de description à produire (FOM pour la fédération, SOM pour les fédérés, les deux suivant le format indiqué dans l'OMT) ;
- le rôle central du RTI, passage obligatoire des échanges d'information entre fédérés ; malgré ce rôle central, le RTI ne conserve pas les données du monde représenté : celles-ci ne sont présentes que dans les fédérés ;
- la base de la gestion de la propriété : appartenance d'un attribut par au plus un seul fédéré, et possibilité de transfert d'un fédéré à un autre ;
- la possibilité pour les fédérés d'échanger des données (modification de leurs objets, envoi d'interactions) en respectant l'interface HLA, en associant un temps à ces échanges, en cohérence avec les autres fédérés.

Le modèle objet

L'OMT [DoD98b] décrit le formalisme qui doit être employé pour définir le FOM d'une fédération, ou le SOM d'un fédéré. En particulier dans le cas du FOM, ce modèle donne une manière de décrire les éléments du monde simulé, identique dans toutes les simulations HLA. La réutilisation de portions de ces descriptions facilite la création de nouvelles simulations.

Les éléments décrits sont principalement les objets et les interactions, qui sont au cœur des échanges entre fédérés.

Les objets sont des entités simulées par les fédérés. Comme en conception ou programmation orientée objet, ils sont vus comme des instances d'une

classe qui correspond à leur type. L'OMT permet de définir des attributs pour chaque classe d'objet, ainsi que des relations de spécialisation entre classes et sous-classes. Lors de l'exécution d'une fédération, les objets sont instanciés par des fédérés, qui peuvent ensuite les modifier, les détruire, etc.

Une interaction correspond à un événement, une information transmise entre fédérés. En général, il s'agit d'événements ne changeant pas l'état d'un objet. Comme pour les objets, l'OMT propose un moyen de définir des classes d'interactions, toujours avec d'éventuelles relations de spécialisation. Les différents éléments qui composent une interaction sont appelés paramètres. Contrairement aux objets, les interactions n'existent que ponctuellement : elles sont créées par des fédérés, et transmises à leurs destinataires.

Enfin l'OMT permet de décrire d'autres éléments. Par exemple, nous verrons dans la partie consacrée à la gestion de la distribution des données les notions de dimension et d'espace de routage. Une fédération les employant doit disposer d'un FOM les décrivant, suivant le modèle OMT.

L'interface

L'interface HLA [DoD98a] définit un ensemble de services permettant l'échange d'informations entre fédérés. Ces informations ne transitent pas directement d'un fédéré à un autre, mais via le RTI conformément à la règle 3. Les services de l'interface HLA appartiennent donc à deux grandes catégories :

- Les services à l'initiative des fédérés (ou *federate initiated*), c'est-à-dire les services fournis par le RTI, auxquels peuvent accéder les fédérés ;
- Les services à l'initiative du RTI (ou *RTI initiated*), c'est-à-dire les services fournis par les fédérés, auquel peut accéder le RTI.

Il est d'usage de distinguer ces deux catégories en ajoutant le symbole « † » à la suite du nom d'un service à l'initiative du RTI. Par exemple, une mise à jour d'attribut est effectuée par un fédéré au moyen du service *Update Attribute Values*, qui a pour conséquence l'appel du service *Reflect Attribute Values* † des fédérés concernés par le RTI.

Il faut donc bien noter que la réalisation d'un fédéré HLA implique la mise en œuvre d'un ensemble de services que ce fédéré devra fournir au RTI.

Indépendamment de ce sens d'utilisation, les services HLA sont réunis en plusieurs groupes. Il y a principalement six groupes de services concernant chacun un ensemble cohérent de mécanismes :

Federation Management Ce groupe concerne la gestion de la fédération, c'est-à-dire d'une part tout ce qui concerne l'accès d'un fédéré à une fédération (créer, rejoindre, quitter ou détruire une fédération), et d'autre part les mécanismes qui ne se conçoivent qu'à l'échelle de la fédération (sauvegarde, restauration, synchronisation).

Declaration Management Ce groupe concerne la gestion des déclarations, c'est-à-dire la déclaration par les fédérés des messages qu'ils sont susceptibles d'envoyer, et l'information par le RTI des messages que les fédérés sont susceptibles de recevoir. Ces déclarations se font au niveau des classes d'objet et d'interaction.

Object Management Ce groupe concerne la gestion des objets et des interactions. Le monde simulé évolue par la création d'objets, leurs éventuelles mises à jour et destruction, et l'envoi d'interactions. Ces événements sont créés et observés par les fédérés à l'aide de ce groupe de services, dans le cadre des déclarations effectuées à l'aide du groupe précédent.

Ownership Management Ce groupe concerne la gestion de la propriété des attributs d'objet. Conformément à la règle n° 8, des transferts de propriété d'attributs d'objets sont possibles, et les services de ce groupe définissent plusieurs mécanismes permettant de réaliser ces transferts.

Time Management Ce groupe concerne la gestion du temps, et permet en particulier aux fédérés d'avancer dans leur temps logique, de déclarer un *lookahead*, etc. Les services correspondant aux envois de messages estampillés sont dans *Object Management*, l'estampille y étant optionnelle.

Data Distribution Management Ce groupe concerne la gestion de la distribution des données, et en pratique consiste à affiner les déclarations du groupe de services *Declaration Management*. Il s'agit ici d'exprimer les intérêts non plus au niveau classe, mais au niveau instance et valeur d'instance.

Il existe deux autres groupes de services, considérés comme annexes à ces groupes principaux :

Support Services Il s'agit de services annexes de support, comme par exemple l'accès aux identifiants des éléments de la fédération à partir de leur nom (et inversement).

Management Object Model Il s'agit d'un ensemble de services indiquant l'utilisation des services du RTI par les différents fédérés. L'utilisation de ces services peut entraîner une violation de la règle n° 2, et ne va pas dans le sens de l'interopérabilité. Le MOM est donc à réserver aux

fédérés gestionnaires de fédération ou collecteurs d'information, et non aux fédérés simulateurs.

2.3.4 Caractéristiques

Le résumé de ces trois documents donne un premier aperçu de HLA, cependant il nous faut insister sur certains aspects essentiels. Dans [KWD99], Kuhl *et al.* soulignent que HLA suit trois des styles d'architectures logicielles recensées dans [SG96] :

- HLA possède des aspects d'*architecture organisée en couches hiérarchiques* : dans une telle architecture, chaque couche propose des services à la couche supérieure, et inversement utilise les services de la couche inférieure. L'avantage provient de l'interface, qui permet de changer l'implémentation d'une couche sans conséquences sur les autres. Ceci se retrouve donc dans HLA puisque le RTI a un comportement défini par une spécification, et qu'un fédéré peut être conçu et réalisé indépendamment du RTI utilisé. De plus, la couche RTI permet de proposer au concepteur de fédéré des mécanismes de simulation de haut niveau, et de cacher dans des couches inférieures des aspects de bas niveau (protocoles de communication, spécificités d'une plate-forme, etc.)
- HLA propose une *architecture d'abstraction des données*, au sens usuel des types abstraits ou de l'orientation objet. La spécification impose une interface pour le RTI et une interface pour les fédérés. Derrière cette interface, le travail du fédéré est caché au RTI, et inversement le fonctionnement du RTI est caché au fédéré. Ceci assure une indépendance entre RTI et fédéré : une fédération peut être exécutée sur toute implémentation de RTI conforme à la norme, et la seule contrainte d'un fédéré est de respecter son interface, ce qui autorise de nombreuses possibilités : simulateurs logiciels, moniteurs, simulateurs complexes avec leur propre RTI, logiciels rapportant des résultats provenant de capteurs et de systèmes physiques en général.
- HLA est une *architecture orientée événement* : le RTI diffuse les données aux fédérés qui se sont enregistrés en conséquence. Ainsi un fédéré, après avoir indiqué les types d'événements qu'il souhaite recevoir, n'a pas à rechercher activement l'apparition de tels événements : ses propres services seront directement appelés par le RTI. Ce style d'architecture a pour avantage de faciliter l'évolution et la réutilisation.

2.3.5 Données objet

L'OMT permet de décrire des modèles objet pour les fédérés et les fédérations, en proposant un formalisme souvent qualifié d'orienté objet. On retrouve en effet la notion de classes définies par leurs attributs, et de relations de spécialisation entre classes ; les objets sont bien des instances de classe, avec leur propre identité. Cependant l'OMT ne permet pas de définir [KWD99] [Lut] :

- des comportements ou méthodes : ceux-ci sont à la charge des fédérés qui gèrent des objets ;
- des attributs de classes, c'est-à-dire des attributs n'appartenant pas à une instance mais à une classe (et partagés par toutes les instances de cette classe) : là encore leur équivalent est généralement mis en œuvre par un fédéré dédié à la classe concernée ;
- d'autres types d'associations entre classes que les associations d'héritage (par exemple la composition, l'agrégation) ;
- une véritable encapsulation : tous les attributs déclarés sont publics, l'équivalent d'attributs privés doit être géré à l'intérieur du fédéré, sans apparaître dans le FOM.

L'absence de méthodes associées aux objets, et donc de ce qui permet les communications d'objet à objet, est en partie compensée par la présence des interactions [Lut].

Comme il s'agit essentiellement de restrictions, l'utilisation de formalismes objet tels que les diagrammes de classe UML reste possible, pourvu que l'on se limite aux notions présentes dans HLA.

2.3.6 Services

Gestion de fédération

Dans ce groupe, *Federation Management*, quatre services concernent la création et la destruction des fédérations, et la participation des fédérés. Les autres services correspondent à des mécanismes à l'échelle de la fédération (sauvegarde, restauration, synchronisation).

Concernant la première catégorie de services, un fédéré peut créer une fédération à l'aide du service *Create Federation Execution*. Ce fédéré pourra alors ultérieurement détruire la fédération créée avec le service *Destroy Federation Execution*. Lorsqu'une fédération existe, les fédérés (et en particulier le créateur de la fédération) participent à la fédération à l'aide du service *Join Federation Execution* et peuvent la quitter avec le service *Resign Federation Execution*.

Les opérations à l'échelle de la fédération (sauvegarde, restauration et synchronisation) impliquent plusieurs services, employés dans des mécanismes assez similaires. Dans le cas particulier de la sauvegarde de fédération, il s'agit de demander aux fédérés de sauvegarder leur état afin de pouvoir procéder ultérieurement à une restauration. Une sauvegarde s'effectue à la requête d'un fédéré qui appelle le service *Request Federation Save*. Le RTI demande alors à tous les fédérés de démarrer une sauvegarde en utilisant leur service *Initiate Federation Save* †. Ceux-ci indiquent l'avancement de leur sauvegarde en appelant *Federate Save Begun* lorsque celle-ci a commencé, puis *Federate Save Complete* lorsqu'elle s'est terminée (éventuellement par un échec). Lorsque le RTI a reçu toutes les réponses des fédérés, il termine par un appel à *Federation Saved* † qui indique à tous les fédérés le résultat de la sauvegarde.

La restauration d'une sauvegarde, et l'établissement d'un point de synchronisation suivent des mécanismes similaires. On trouve des différences dans le détail, mais les quatre étapes principales restent :

1. requête initiale de la part d'un fédéré ;
2. transmission de l'ordre à tous les fédérés concernés, par le RTI ;
3. opération et envoi du résultat au RTI, par chaque fédéré ;
4. envoi du résultat global à tous les fédérés.

La gestion des déclarations

HLA étant orienté événement, les fédérés reçoivent les informations relatives aux objets et aux interactions par des appels de service effectués par le RTI. Afin d'éviter des appels inutiles, le fédéré déclare son intention d'utiliser ou de produire certains types de données, à l'aide des services de ce groupe, *Declaration Management*. Le RTI peut alors router les données plus efficacement en déterminant la pertinence des messages pour chaque fédéré destinataire.

Par exemple, un fédéré utilisera le service *Publish Object Class* pour indiquer qu'il a l'intention de créer (et probablement faire évoluer) des objets d'une certaine classe. Seuls les fédérés ayant manifesté un intérêt pour les objets de cette classe, à l'aide du service *Subscribe Object Class Attributes*, recevront des informations relatives à ce type d'objets (créations, mises à jour, suppression). Pour annuler ces souscriptions et intentions de publication les fédérés disposent des services *Unsubscribe Object Class* et *Unpublish Object Class*. Il existe quatre services similaires pour les publications et souscriptions de classes d'interactions.

Enfin, à partir de ces indications, le RTI peut déterminer si un fédéré a effectivement besoin d'informer les autres fédérés de ses mises à jour d'objet ou

d'interactions. Si aucun autre fédéré n'a exprimé d'intérêt, alors il est inutile d'envoyer ces informations. Quatre services servent à donner ces indications.

Gestion des objets

Ce groupe, *Object Management*, contient les services permettant d'envoyer des *messages* aux autres fédérés, par l'intermédiaire du RTI. Les messages sont les événements de la simulation : envois d'interactions d'une part, créations et suppressions d'objets, mises à jour de leurs attributs d'autre part. Chaque transfert d'information correspond à l'utilisation de deux services :

1. envoi de l'information au RTI par un fédéré ;
2. envoi de cette information aux fédérés concernés par le RTI.

Ainsi, la création d'un objet par le service *Register Object Instance* sera perçue par les fédérés abonnés, par l'appel de leur service *Discover Object Instance* †.

Les deux couples de services les plus courants dans une simulation HLA sont la mise à jour d'attributs : *Update Attribute Values* et *Reflect Attribute Values* †, et l'envoi d'interaction : *Send Interaction* et *Receive Interaction* †. La figure 2.3 présente quelques échanges de messages à l'aide de ces mécanismes. Si plusieurs fédérés sont abonnés et donc concernés par l'échange de messages, alors un *Update Attribute Values* donne lieu à autant de *Reflect Attribute Values* † qu'il y a de destinataires.

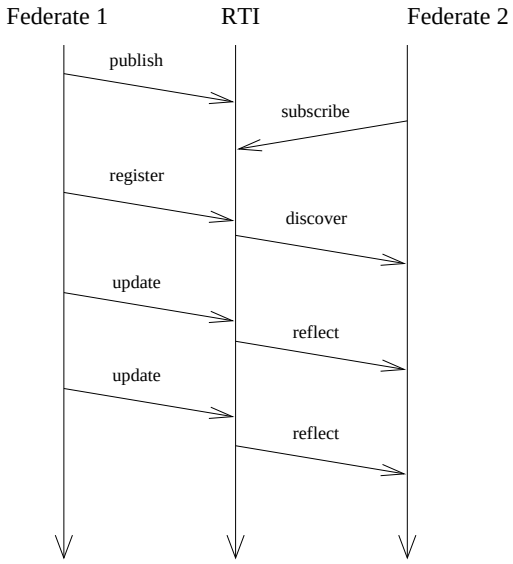

FIG. 2.3 – Échange de messages entre fédérés

Enfin, ce groupe propose des services permettant de définir le type de *transport* à utiliser : fiable ou non ("*reliable*" ou "*best-effort*"). Par exemple, des informations ponctuelles doivent en général être fiables, contrairement aux données représentant des phénomènes continus, qui peuvent être extrapolées, ou tout simplement dont l'absence d'une mise à jour est sans incidence. En fonction de cette caractéristique, le RTI choisit parmi les types de transmission dont il dispose le plus adapté à l'envoi d'un message. Généralement les données *reliable* sont transmises à l'aide du protocole TCP tandis que les messages *best-effort* sont envoyés en UDP.

La gestion de la propriété

Ce groupe *Ownership Management* et les suivants, contrairement aux trois premiers, ne sont pas indispensables à toute fédération.

Comme l'indiquent les règles de HLA, les attributs d'instances appartiennent à des fédérés, et à un instant donné, un attribut ne peut pas appartenir à plus d'un fédéré. Par contre il est possible que tous les attributs d'un objet n'appartiennent pas au même fédéré.

Initialement, les attributs d'un objet appartiennent au fédéré qui a créé cet objet. Ensuite, des transferts de propriété entre fédérés peuvent être effectués. Il y a pour cela deux types de mécanismes :

- un fédéré commence par abandonner la possession d'un attribut : dans ce cas le RTI en informe les autres fédérés, et accepte une éventuelle demande d'acquisition ;
- un fédéré souhaite acquérir un attribut qui est déjà possédé par un autre fédéré : il en fait la requête, et le RTI demande à ce fédéré d'abandonner la possession de cet attribut. Si la demande est acceptée, le transfert a lieu.

Il apparaît donc qu'un attribut peut n'appartenir à aucun fédéré. Dans ce cas il n'est plus mis à jour, mais pourra l'être de nouveau dès qu'un fédéré en aura fait l'acquisition.

La gestion du temps

HLA propose un ensemble de services de gestion du temps. Ceux-ci ne se restreignent pas à une seule approche, comme dans le cas de DIS qui imposait un fonctionnement en temps réel.[13] Ce fonctionnement similaire à DIS est néanmoins possible, en n'utilisant pas les services de gestion du

[13]Le terme de temps réel est fréquemment utilisé pour désigner une simulation dans laquelle le temps avance à la même vitesse que dans la réalité, sans tenir compte des temps de transmission – il ne s'agit donc pas de temps réel dur avec des contraintes d'échéances.

temps. Mais une simulation peut aussi s'exécuter en temps coordonné, en suivant une approche conservative ou spéculative.

Nous avons vu au premier chapitre plusieurs approches de la gestion du temps coordonné, en particulier les approches optimistes et conservatives. Cette dernière est imposée dans l'architecture ALSP, qui utilise l'algorithme de Chandy, Misra et Bryant. L'architecture DIS n'a pas ces exigences et vise à exécuter les événements dès que possible, sans vérification de causalité.

Avec son impératif d'interopérabilité, HLA ne peut pas imposer une de ces approches, c'est pourquoi les services de *Time Management* sont définis de manière à permettre la coexistence de fédérés qui évoluent dans le temps suivant des approches différentes.

Tout d'abord, comme pour le groupe de services précédent, les services de gestion du temps ne sont pas indispensables dans une simulation, et donc ne le sont pas pour un fédéré donné. Un fédéré ne les utilisant pas évoluera suivant une approche « temps réel » similaire à celle de DIS.

Pour les approches mettant effectivement le temps en œuvre, un *temps logique* est défini : il s'agit de la notion vue précédemment à propos des processus logiques. Mais ces approches sont nombreuses et HLA a non seulement pour but de permettre l'exécution de fédérations suivant l'une ou l'autre de ces approches, mais aussi de permettre l'interopérabilité entre fédérés qui suivent des approches différentes. Un fédéré, utilisant le RTI afin de suivre le mécanisme de gestion du temps pour lequel il a été conçu, doit interagir en toute transparence, quels que soient ces mécanismes. C'est pourquoi ceux-ci doivent rester cachés des autres fédérés.

Les événements estampillés doivent alors être exécutés lorsque le temps logique du fédéré atteint l'instant indiqué par l'estampille : les événements de la simulation sont situés sur un axe des temps commun à toute la fédération : si deux instants T_1 et T_2 y sont placés, T_1 précédent T_2, alors T_1 correspond à un instant ayant lieu avant l'instant T_2, et tout événement ayant lieu à l'instant T_1 doit être exécuté avant un événement de date T_2. Cette ordre d'exécution des événements doit être vérifié dans chaque fédéré exécutant ou observant ces événements, même si deux fédérés n'exécuteront pas ces événements au même moment.

Il s'agit donc avant tout pour le RTI de pouvoir s'assurer qu'un message n'est pas envoyé dans le « passé » d'un fédéré, afin que celui-ci puisse l'exécuter sans faute temporelle, dans le cas d'une approche conservative. Le RTI a également besoin de pouvoir signaler l'annulation de messages précédents, dans le cas d'une aproche optimiste.

Pour cela HLA propose :

- des services d'avancement dans le temps ;
- des catégories de messages indiquant comment ceux-ci doivent être traités ;
- des services pour l'annulation de messages.

Nous verrons plus en détail certains services d'avancement dans le temps : il faut avant tout retenir qu'un fédéré ne peut avancer dans le temps qu'avec l'autorisation du RTI. Un fédéré émet une requête d'avancement dans le temps, et en fonction des messages à venir, peut être autorisé par le RTI à faire cet avancement (ou un avancement d'une moindre importance).

Afin de simplifier le travail du RTI et de permettre plusieurs approches, les messages sont caractérisés par un type d'ordre, qui indique la manière de les transmettre. Il existe cinq catégories [DoD96] :

- L'ordre de réception (*Receive Order* ou RO) indique que les messages doivent être transmis au fédéré dans l'ordre où ils ont été reçus : c'est le cas le plus simple à traiter, il correspond à une file FIFO, et ne fait pas intervenir les estampilles de messages ;
- L'ordre par priorité (*Priority Order*) place les messages dans une file à priorité (l'estampille du message étant utilisée pour déterminer la priorité), ce qui n'est toutefois pas suffisant pour empêcher des messages d'arriver dans le « passé » de leur fédéré destinataire ;
- L'ordre causal (*Causal Order*) permet de maintenir l'ordre entre deux événements reliés par une relation de cause à effet : leur instant d'occurrence dans un même fédéré détermine cette relation, tandis que l'ordre de deux événements provenant de deux fédérés différents ne pourra pas toujours être assuré ;
- L'ordre causal et totalement ordonné (*Causal and totally ordered*) reprend les caractéristiques de l'ordre causal ci-dessus et impose, de plus, que deux événements quelconques soient reçus dans leur ordre d'occurrence dans les fédérés les observant tous les deux ;
- enfin l'ordre des estampilles (*Time stamp order* ou TSO) assure que les messages sont fournis conformément à l'ordre indiqué par leur estampille, et n'arrivent jamais dans le « passé » du fédéré destinataire.

En pratique seuls RO et TSO ont été maintenus. L'utilisateur intéressé par un ordre causal doit l'émuler à l'aide du type TSO.

Ces caractéristiques des messages interviennent sur l'avancement des fédérés, puisque si les messages RO peuvent être transmis à tout moment et sans contrainte, les messages TSO doivent arriver suivant l'ordre des estampilles et uniquement lorsque le temps logique du fédéré coïncide avec ces dernières.

Il faut également noter qu'un fédéré peut employer différents types d'ordre pour les différents messages qu'il envoie. Comme dans le cas de la fiabilité

vue précédemment, des données continues pourront être mises à jour avec des messages RO tandis que des événements ponctuels devront être transmis par des messages TSO.

Ces envois n'ont pas lieu lors d'appels de services de *Time Management* mais lors des envois d'événements de *Object Management* : création d'objet, mise à jour, etc. Le groupe *Time Management* propose par contre des services liés à l'avancement du temps logique des fédérés.

Tout d'abord un fédéré peut indiquer s'il est régulateur (c'est-à-dire s'il envoie des messages TSO) et s'il est contraint (c'est-à-dire s'il reçoit des messages TSO). Cette information est utile au RTI et a pour effet de transformer tout message TSO non attendu : un message envoyé par un fédéré non régulateur ne sera pas considéré comme TSO même si son contenu l'indique, et inversement un message TSO perd son statut s'il est à destination d'un fédéré non contraint. Bien sûr un même message, TSO à son émission, peut donner lieu à différents messages qui seront TSO ou non suivant leurs destinataires respectifs.

Ensuite, HLA demande aux fédérés régulateurs de fournir un *lookahead* : il s'agit de la notion vue au premier chapitre, et qui doit permettre au RTI d'estimer la date minimale d'un événement futur produit par un fédéré. En effet un fédéré, de temps logique t et de *lookahead* L assure de ne pas envoyer de message ayant une estampille de valeur inférieure à $t + L$. Un fédéré peut modifier son *lookahead* cependant, en cas de réduction, la valeur effective de ce *lookahead* ne peut varier que dans le respect des valeurs précédentes annoncées.

Enfin les services d'avancement dans le temps permettent au fédéré de demander un avancement au RTI, il en existe deux :

- *Time Advance Request* permet de demander l'avancement du temps logique jusqu'à une valeur spécifiée ;
- *Next Event Request* permet de demander l'avancement jusqu'au prochain événément.

L'avancement ne sera effectif qu'après l'appel du service *Time Advance Grant* † par le RTI. Cet appel précise le nouveau temps logique, qui n'est pas nécessairement celui demandé dans le cas de *Time Advance Request*.

Lorsqu'un fédéré est en attente d'avancement, il reçoit les messages dont l'arrivée doit précéder cet avancement : en particulier, les messages TSO dont l'estampille est supérieure au précédent temps logique du fédéré, et inférieure au prochain.

Ces services, avec l'emploi de messages TSO, permettent la mise en œuvre de fédérés suivant des mécanismes conservatifs de gestion du temps. Il faut toutefois ajouter un service dont le but est de fournir à chaque fédéré une

valeur appelée LBTS[14] et définie ainsi : $LBTS = min(T_i + L_i)$, minimum calculé sur l'ensemble des fédérés susceptibles d'envoyer des messages TSO au fédéré concerné, T_i et L_i représentant respectivement le temps logique et le *lookahead* de tels fédérés.

Il faut enfin noter l'existence du service *Retract* dont le but est de supprimer un événement précédemment envoyé, afin d'autoriser une approche optimiste de la gestion du temps.

L'emploi de ces services permet donc aux fédérés d'évoluer suivant différents mécanismes : avancement par pas de temps, ou orienté événement, dans le cadre d'une approche conservative du temps. Une approche optimiste, ainsi qu'une approche sans prise en compte du temps sont possibles, et les services ont été conçus de manière à permettre, dans une même fédération, l'interopérabilité entre fédérés n'obéissant pas aux mêmes mécanismes.

La gestion de la distribution des données

Le groupe de services *Data Distribution Management* a pour objectif l'optimisation des échanges de données, en fournissant aux fédérés un moyen de préciser les informations qui leur sont pertinentes. Il s'agit ici de reprendre le principe des publications et des souscriptions, non plus au niveau des classes mais au niveau des instances.

Ainsi HLA définit la notion d'*espaces de routages* et de *régions* appartenant à ces espaces. Les espaces de routage sont définis comme la donnée d'un ensemble de dimensions nommées. Chaque fédération précise quels sont les espaces qu'elle possède, et quelles sont leurs dimensions. Il est alors possible pour les fédérés de créer des régions situées dans ces espaces. Par exemple la figure 2.4 présente un espace à deux dimensions avec trois régions.

Il s'agit ensuite d'exprimer les actions et les intérêts des fédérés à l'aide de telles régions, de manière similaire aux services de *Declaration Management*. Ainsi certaines régions sont utilisées pour indiquer quelles sont les actions d'un fédéré : celles-ci sont localisées en une région d'un espace de routage, et correspondent à des événements. Il faut bien remarquer qu'il n'y a pas de relation directe entre les valeurs caractéristiques d'un événement (par exemple, les nouvelles valeurs lors d'une mise à jour d'attributs) et les bornes d'une région à laquelle cet événement est associé. Une telle relation existe, dans la conception du fédéré, mais n'est pas directement traduite par HLA, c'est pourquoi des services spécifiques doivent être utilisés.

[14]Lower Bound on the Time Stamps

On peut prendre comme exemple le déplacement d'un véhicule sur un terrain représenté en deux dimensions. On y associe un espace de routage à deux dimensions, et on fait correspondre les coordonnées sur le terrain (donc les coordonnées des véhicules s'y déplaçant) aux coordonnées de l'espace de routage. Alors si un véhicule ne se déplace que dans une région donnée, on exprime cette information par la création d'une région de routage associée à cet objet : les coordonnées de cette région correspondent à la zone où le déplacement est possible. S'il y a coïncidence entre les deux coordonnées, c'est parce qu'elles ont été définies ainsi, mais toute autre représentation est possible (changement d'échelle, ou toute autre relation). Tout comme le fédéré pouvait utiliser les services de *Declaration Management* pour indiquer ses actions (publication d'un type de classes), il peut utiliser une région pour indiquer quelles seront ses actions, au niveau instance (la région est associée à un objet) et valeur (les bornes de la région définissent la région d'action, d'après la relation choisie).

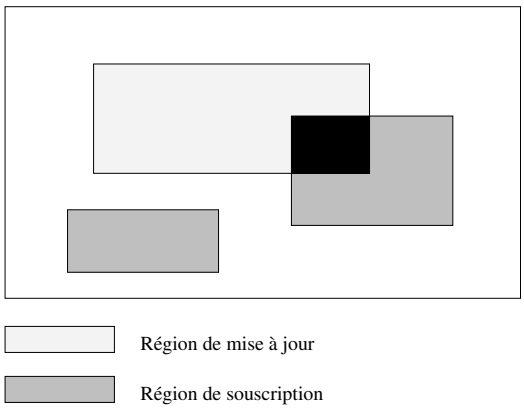

FIG. 2.4 – Régions de routage

De même, les fédérés peuvent définir des régions utilisées cette fois pour exprimer leur intérêt : elles sont créées avec les mêmes services, et les fédérés ont la possibilité de souscrire à des classes, en précisant une région. Ils expriment ainsi leur intérêt pour tout événement qui sera associé à cette région. Dans l'exemple précédent du véhicule se déplaçant sur un terrain, une région entourant le véhicule pourraît être définie, afin de représenter la région physique observable depuis ce véhicule.

La pertinence de l'information est déterminée par le recouvrement éventuel de régions de souscription et de régions de mise à jour : la figure 2.4 présente un cas de recouvrement partiel entre deux de ces régions. Ceci se produit lorsqu'un événement a été produit avec une certaine région, et qu'un fédéré est intéressé par tout événement se produisant dans une partie de cette

région (la partie commune). À moins que les deux régions ne se recouvrent exactement, il est tout de même possible que l'information ne soit pas pertinente, par exemple si celle-ci est située dans une partie non commune des régions. Pour reprendre notre exemple de véhicule, ce recouvrement permet de définir qu'un fédéré est intéressé par les événements se situant à proximité du véhicule qu'il simule, et non par les événements hors de portée.

La tâche du RTI est de conserver toutes les informations relatives aux régions, et de déterminer si elles sont utilisées et s'il y a recouvrement lors de l'occurrence d'un événement. Seules les informations pertinentes selon ces critères seront transmises. Dans le cas du véhicule, si le terrain est grand et que des centaines de véhicules s'y déplacent, le service de souscription de *Declaration Management* obligera à recevoir les mises à jour relatives à tous les véhicules. Avec les régions de routage, seules les mises à jour des véhicules proches seront transmises.

Cet emploi des régions concerne tous les événements : création, suppression, modification d'objet, et envoi d'interaction.

Enfin il faut préciser que cette présentation concerne les services *Data Distribution Management* de HLA 1.3 : la version IEEE a été simplifiée, le principe reste le même mais dispose simplement de dimensions dans un unique espace de routage.

La création, la modification et la suppression de région représentent toutefois des appels supplémentaires et donc un coût. Pour présenter les régions de souscription, nous avons considéré un véhicule entouré par une région. Ceci suppose que la région est déplacée au fur et à mesure que le véhicule se déplace. Cette utilisation est souvent peu performante, et un trop grand nombre de régions rend la détermination des recouvrements trop longue à calculer. Nous verrons différentes approches tentant d'appréhender ce problème dans le chapitre 3, consacré aux domaines.

2.3.7 Logiciels HLA

HLA est une spécification et non une implémentation. Il existe donc plusieurs logiciels relatifs à HLA et en particulier des RTI :

- des RTI commerciaux tels que pRTI de Pitch, MäK RTI de MäK Technologies, et le RTI-NG (initialement gratuit et développé par le DMSO), eRTI ;
- des logiciels libres, en particulier CERTI [BS02] que nous présentons plus loin, yaRTI [Can99], XRTI [Kap03]. Ces RTI ne proposent pas tous les services de HLA mais fournissent les plus fréquemment employés.

On trouve également des outils pour la conception de fédération, des modèles objet, etc. mais aussi pour la transition de simulations ALSP ou DIS vers HLA.

2.3.8 Conclusion

HLA résulte des travaux réalisés à partir des architectures qui l'ont précédé, principalement DIS et ALSP. HLA a eu pour objectifs principaux l'interopérabilité et la réutilisation, tout en se tournant vers une indépendance vis-à-vis du domaine d'application. Réutilisation et interopérabilité sont particulièrement favorisées dans le cas de la gestion du temps qui autorise la participation de fédérés suivant des mécanismes différents dans une même fédération : c'est un point sur lequel HLA se distingue particulièrement de ses prédécesseurs [IS01].

Chapitre 3

Domaines

3.1 Généralités

Nous nous intéressons à l'introduction de la notion générale de domaine dans la simulation distribuée et en particulier dans les simulations HLA.

Un *domaine* est une région logique d'une simulation, regroupant des éléments de la simulation, qu'ils correspondent à des simulateurs ou à des éléments du monde représenté. L'introduction des domaines consiste donc à regrouper des fédérés ou des objets de simulation en groupes logiques ou physiques.

Réaliser ces regroupements, mettre en œuvre des domaines, peut être motivé par plusieurs objectifs.

L'interopérabilité et la réutilisation sont deux des principaux objectifs des architectures de simulation distribuée telles que HLA. Ces deux éléments sont indispensables à la réduction des coûts de réalisation de simulations à grande échelle. Outre le besoin de modularité conduisant à l'abandon du simulateur monolithique, la réutilisation consiste à récupérer des modules d'anciennes simulations lorsque les objets à représenter se retrouvent dans de nouveaux modèles. L'objectif est de rendre les coûts de réutilisation minimaux en ayant très peu d'adaptations à réaliser si un groupe de composants de simulation requis a déjà été développé dans une précédente simulation.

Un autre objectif consiste à améliorer l'extensibilité d'une simulation : il s'agit de la capacité d'une simulation à supporter des changements d'échelle, par exemple une forte augmentation du nombre de fédérés participants. Idéalement les ressources nécessaires devraient être proportionnelles à la complexité de la simulation, mais en pratique les performances se dégradent bien plus rapidement. On peut alors envisager l'utilisation de domaines afin de regrouper, de factoriser des échanges de données. La plupart des implémen-

tations actuelles de HLA conviennent à des simulations de taille moyenne sur réseau local. Sur des simulations plus grandes, ou au travers d'un WAN, les performances décroissent rapidement. Parmi les optimisations possibles de ces implémentations, la notion de domaine apparaît à plusieurs niveaux : au plus bas niveau en agissant sur les protocoles de communication employés, ou au plus haut niveau en intervenant sur la conception du contenu de la fédération.

Bien qu'il ne s'agisse pas ici d'un des premiers objectifs de ce travail sur l'introduction de domaines, le besoin de sécurité trouve des solutions à l'aide des domaines. Par exemple les données échangées entre deux fédérés sont visibles dans toute la fédération : il suffit généralement de s'abonner aux classes correspondantes. Certaines structures hiérarchiques permettent de définir des domaines dans lesquels ces informations seront visibles ou non aux différents fédérés.

Il s'agit là d'objectifs très généraux. Il ne s'agit pas, pour chaque approche envisagée, de satisfaire à tous ces objectifs, mais de faire face à certains de ces problèmes, même si d'autres ne sont pas solutionnés.

Nous allons présenter différents travaux qui introduisent la notion de domaine plus fortement que dans le cadre d'une fédération classique :

- RTI à noyau distribué
- Multirésolution
- Les régions de routage
- Le multicast
- Les ponts de fédérations

3.2 RTI à noyau distribué

Une fédération HLA est constituée d'un ensemble de fédérés réunis autour d'un unique composant, le RTI. Ce RTI est généralement implémenté par un ensemble de processus accompagnant les fédérés, tous connectés à un unique processus central (éventuellement, il peut exister un tel processus pour chaque fédération en cours d'exécution).

La norme HLA n'impose toutefois pas de contrainte quant à cette implémentation, et un noyau lui-même distribué peut être envisagé. On retrouve alors l'architecture proposée par ALSP avec plusieurs ABE (§ 2.2, figure 2.1).

Ce type d'architecture a été proposé dans [Sir98] dans le cadre du développement de CERTI. Ce RTI a dès le départ été envisagé comme un ensemble de processus communicants afin d'effectuer des traitements locaux sur les machines des fédérés, et de ne passer par le processus central qu'en cas de

nécessité. Le même principe est envisagé dans le cas de fédérations réparties sur plusieurs sites (figure 3.1) : afin d'éviter que le composant central (appelé RTIG) ne soit présent que sur un seul site, une version distribuée, avec donc un processus par site, est envisagée dans le but de factoriser certaines communications entre les sites. Les processus accompagnants les fédérés (les RTIA) sont bien sûr toujours présents.

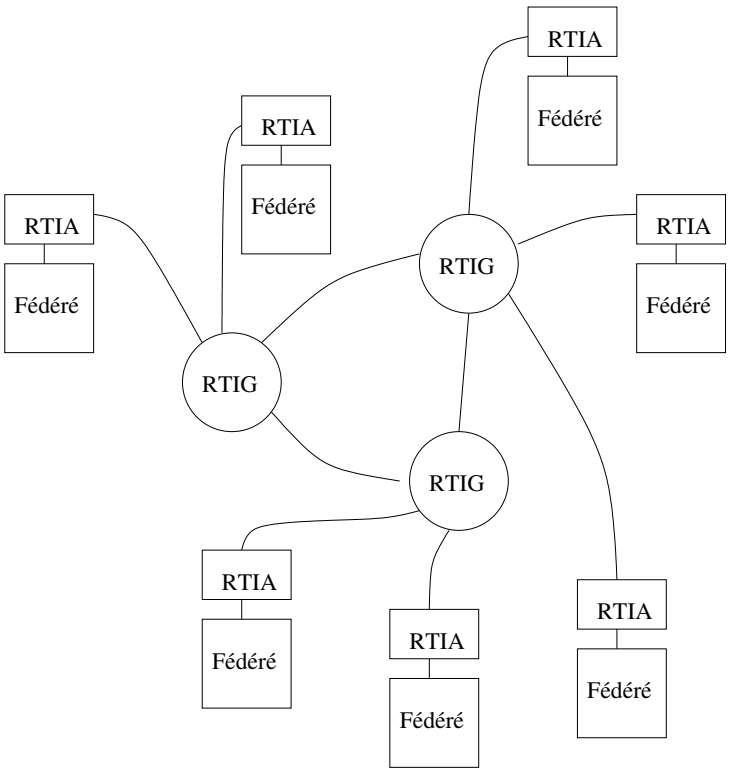

FIG. 3.1 – Noyau RTI distribué (cas de CERTI)

Il s'agit ici avant tout d'un objectif d'optimisation. Cependant cette architecture faciliterait l'ajout de fonctionnalités permettant la confidentialité de certaines données, chaque site contrôlant un RTIG et donc un éventuel filtrage.

3.3 Multirésolution

La multirésolution consiste a décrire différemment certaines entités de la simulation, en fonction du niveau de détail de la description requise. Ces entités sont parfois vues comme un seul objet, parfois comme un ensemble d'objets. Il en va de même pour leur comportement, qui est soit celui d'un

seul objet, soit un ensemble de comportements différents. On a donc affaire à des objets, occasionnellement groupés afin d'adapter le niveau d'abstraction. Ce regroupement peut avoir pour objectif une factorisation des données échangées, une optimisation des communications.

Mais comprendre les mécanismes de la multirésolution présente aussi un intérêt vis-à-vis de la démarche de réutilisation privilégiée par HLA. En effet, si HLA favorise l'utilisation de modèles, de fédérés, déjà utilisés et éprouvés dans le cadre de précédentes fédérations, les fédérés et modèles récupérés pour une nouvelle simulation n'ont pas forcément le degré d'abstraction souhaité. Le modèle d'un objet complexe présentera peu d'intérêt si, malgré sa qualité, sa complexité est trop grande dans la nouvelle simulation, autrement dit si un modèle plus simple convient et est moins coûteux en temps de calcul.

La mise en œuvre de la multirésolution nécessite la présence d'opérations d'agrégation et de désagrégation : au cours de la première, plusieurs objets sont réunis en un seul et des informations sont perdues (l'abstraction est plus élevée), tandis que la seconde augmente de nouveau le niveau de détail et nécessite alors une création ou une récupération de données [SHB00]. La multirésolution est généralement présentée dans le cas d'une entité décrite à deux niveaux d'abstraction différents, mais plusieurs niveaux intermédiaires peuvent exister.

Un exemple concret développé à l'ONERA est fourni dans [AS01] : il s'agit d'une simulation de combat aérien dans laquelle une patrouille est une entité constituée de plusieurs aéronefs. Il s'agit donc de faire apparaître ces aéronefs soit individuellement, soit au sein d'une patrouille, ces deux représentations étant exclusives l'une de l'autre.

Concernant la représentation des entités, deux approches sont identifiées :

- une approche centralisée dans laquelle un fédéré se charge de la simulation des deux niveaux d'abstraction ; l'avantage principal est de centraliser les informations communes aux deux résolutions, et de faciliter le passage de l'une à l'autre ;
- une approche distribuée dans laquelle chaque forme est simulée par un fédéré distinct : dans l'exemple présenté, il s'agit donc d'avoir un fédéré pour la patrouille et un fédéré pour chaque aéronef ; cette approche a pour avantage de permettre la distribution de ces fédérés sur des machines différentes, et présente l'intérêt d'être fondée sur des fédérés plus facilement réutilisables.

Cette simulation utilise des interactions pour signaler aux fédérés concernés le déclenchement d'une agrégation ou d'une désagrégation : c'est en effet la seule manière standard de procéder, la multirésolution n'étant pas gérée nativement par HLA. Des services de gestion de la multirésolution (exten-

sions à HLA) sont également proposés.

Enfin, l'emploi de la multirésolution dans une simulation soulève de nouveaux points à étudier lors de la conception de la fédération :

- les conditions d'agrégation/désagrégation ;
- la cohérence entre formes agrégée et désagrégée, et en particulier la détermination des nouvelles informations lorsque le niveau de détail augmente ;
- les possibilités d'agrégation si un élément désagrégé est détruit ou quitte la fédération.

La multirésolution peut donc être mise en œuvre dans les simulations HLA ; elle propose d'adapter les modèles utilisés par les fédérés en fonction du niveau de détail réellement utile de ces modèles, et permet ainsi de diminuer la quantité globale d'information, en particulier la quantité d'informations échangées. Toutefois, il faut que la fédération cible s'y prête et dispose d'entités à agréger et désagréger. Le cas d'un grand nombre d'objets au comportement fortement indépendant en profitera peu. Il est aussi possible que le niveau de résolution soit imposé quand il concerne les modèles que l'on souhaite observer à l'aide de la simulation.

3.4 Gestion de la distribution des données

Le principe de la gestion de la distribution des données (*Data Distribution Management*) a été été présenté en § 2.3.6. Ce groupe a pour but d'optimiser l'exécution d'une simulation en fournissant des services permettant aux fédérés de mieux exprimer leurs intérêts. On trouvera des présentations détaillées par exemple dans [MS97] et [HRC96].

Dans la plupart des implémentations de HLA, le groupe de gestion de la distribution des données n'a pas été implémenté en priorité, l'optimisation de la gestion des objets étant assurée simplement par le service de gestion des déclarations (*Declaration Management*). Il s'agit d'une optimisation plus grossière, fondée uniquement sur les classes, mais néanmoins satisfaisante pour des simulations à faible nombre d'objets et de fédérés.

Cependant les simulations plus complexes nécessitent le raffinement proposé par *Data Distribution Management* car sans ces services, le nombre d'échanges de données non pertinents augmente fortement et ralentit la simulation :

- en augmentant le trafic sur le réseau ;
- en augmentant le nombre de messages à traiter par les fédérés.

HLA fournit avec *Data Distribution Management* un moyen pour les fédérés d'indiquer leur intérêt avec plus de précision, par l'intermédiaire de régions dans les espaces de routage de la fédération. Ainsi on s'attend à ce que la proportion de messages pertinents arrivant à un fédéré augmente considérablement, avec comme conséquence une réduction du trafic et des temps de traitement.

Mais nous avons évoqué un problème lié aux régions de routage : leur utilisation correspond à des appels de services et à des opérations effectuées dans le RTI. Si des améliorations sont obtenues concernant les données circulant sur le réseau, ces mécanismes ont eux-mêmes un coût qu'il s'agit de maîtriser. Un exemple simple met en évidence cet inconvénient de *Data Distribution Management* : si deux mobiles utilisent des régions de routage afin d'indiquer quel est leur environnement proche, chaque déplacement peut donner lieu :

- à un appel du service *Modify Region* pour indiquer la nouvelle région entourant le mobile ;
- un calcul de recouvrement entre régions, effectué par le RTI.

Alors si les deux mobiles sont proches la plupart du temps (soit en raison du scénario de la simulation, soit à cause de régions mal dimensionnées), le coût de gestion des régions risque de dépasser celui économisé par l'absence d'échanges lorsque les mobiles sont éloignés. De nombreuses causes peuvent mener à ces situations où les coûts de *Data Distribution Management* sont excessifs.

Il s'agit donc de proposer des stratégies d'utilisation des régions ; celles-ci concernent la conception de la fédération, des fédérés et du RTI.

3.4.1 Cas d'utilisation

Très tôt des scénarios généraux pour l'utilisation de *Data Distribution Management* ont été identifiés. [Coh97] propose des cas d'utilisation mettant en évidence l'importance du choix des dimensions des espaces de routage. Un des exemples les plus simples est celui de l'utilisation de fréquences radio pour différentes communications sur un aéroport. Les émetteurs utilisent des fréquences réservées, connues des destinataires : il suffit à ceux-ci de choisir la fréquence qui les intéresse. Un espace de routage ayant une dimension associée à la fréquence permet d'établir de telles communications, sans envoyer de messages aux fédérés pour lesquels ces informations ne sont pas pertinentes.

Un autre exemple consiste à employer un espace de routage à deux ou trois dimensions pour représenter un espace géographique : c'est le cas de l'observation visuelle ou par des radars. Seuls les événements proches sont observés.

Il est alors possible d'effectuer certaines combinaisons : les deux cas ci-dessus peuvent être regroupés pour des communications avec une fréquence et une certaine portée. On voit également apparaître à ce niveau différentes approches concernant la géographie : l'utilisation de régions peut être *statique* si une région est créée et utilisée, puis n'est plus modifiée au cours de l'exécution de la fédération. C'est par exemple le cas d'un radar fixe, qui récupère des informations localisées dans une région centrée sur lui-même et dont la taille reste fixe.

Un avion peut utiliser des informations en provenance des ces radars fixes, mais le pilote obtient aussi des informations par observation visuelle directe : dans ce cas, une région de souscription représentant cette vision est *dynamique* car elle doit être modifiée à chaque déplacement de l'avion.

Enfin, parmi ces cas d'utilisations, on retrouve une préoccupation proche du problème de la multirésolution : par exemple, un observateur global peut être intéressé par l'évolution de la fédération, mais pas à un niveau de détail aussi grand que celui utilisé par les fédérés simulateurs d'objets. Il peut s'agir d'une résolution ou d'une fréquence de mise à jour plus faible. Alors, on peut envisager que des mises à jour de position concernent plusieurs attributs, correspondant à plusieurs résolutions, chaque résolution étant associée à un espace de routage différent. Les attributs de faible résolution n'étant pas mis à jour aussi fréquemment, un fédéré abonné à ce type d'attribut ne reçoit pas les mises à jour de haute résolution.

On voit à travers ces cas d'utilisation que le principe des régions de *Data Distribution Management* se prête particulièrement bien à certains scénarios. Bien qu'il s'agisse d'un groupe de services généralement associé à des objectifs d'optimisation, on constate que les régions peuvent avoir une réelle signification pour le modèle. Une région peut être utilisée pour modéliser la portée d'un objet (en terme d'action ou d'observation), et le modèle ne reçoit ou n'émet d'informations que dans un cadre conforme à la réalité qu'il modélise. L'optimisation finalement obtenue peut être vue comme la résultante de deux filtrages, qu'il s'agit de mettre en œuvre dans la fédération :

- une restriction aux informations effectivement accessibles, d'après le modèle ;
- une restriction aux informations pertinentes, parmi les informations accessible d'après le modèle.

3.4.2 Performances

De nombreux critères entrent en jeu dans les performances d'une fédération utilisant les services de *Data Distribution Management*. Il s'agit de com-

parer une fédération se servant de ces services et une fédération équivalente, ne les utilisant pas. Pour effectuer cette comparaison, [MBD99] présente deux mesures relatives de la rapidité d'exécution et quatre critères représentatifs de l'utilisation qui est faite de *Data Distribution Management*. La rapidité est mesurée à l'aide d'un nombre de cycles CPU et d'une latence définie comme le temps nécessaire à la réception d'une donnée pertinente. Ces deux mesures doivent tenir compte, dans le cas où *Data Distribution Management* n'est pas utilisé, du temps nécessaire au traitement des données non pertinentes. Les critères caractéristiques d'un scénario utilisant *Data Distribution Management* sont alors les suivants :

- le nombre de régions : chacune nécessite de la mémoire et occasionne un temps de traitement ;
- le taux de modifications de régions : ces modifications doivent être prises en compte par le RTI qui peut, par exemple, recalculer tous les recouvrements entre régions dès qu'il y a modification ;
- le nombre de recouvrements entre régions : ceux-ci doivent être revérifiés à chaque modification de région ;
- le taux de changement des recouvrements : s'il est élevé et que les effets des changements de région ne sont pas immédiats, un plus grand nombre de données non pertinentes risque d'être reçu.

Ces critères sont bien différents les uns des autres, bien qu'ils ne soient pas complètement indépendants. Ils peuvent être utilisés pour établir, avec d'autres informations, des algorithmes proposant des estimations des gains représentés par l'utilisation de *Data Distribution Management*. Mais ces critères ont aussi pour intérêt d'indiquer au concepteur de fédération quels éléments peuvent dégrader les performances de la simulation.

Le cas le plus représentatif est celui de régions suivant un objet. Il est donné en exemple dans [HW01] qui présente l'implémentation de *Data Distribution Management* dans le RTI-NG : il est tentant pour le concepteur de fédéré de centrer des régions sur des objets, et de les déplacer en même temps que ceux-ci. Le cas évoqué est celui d'un avion AWACS avec une région de souscription destinée à recevoir toute information relative aux aéronefs suffisamment proches.

C'est une approche très coûteuse dans le cas du RTI-NG et de la plupart des implémentations, car la mise à jour des régions et le calcul des intersections dégradent les performances. Il faut en effet diffuser ces informations dans tout le RTI, et en particulier vers ses composants locaux qui accompagnent les fédérés.

La solution préconisée consiste à utiliser des régions statiques, qui ne sont pas modifiées au cours de l'exécution. Ceci peut se faire en partitionnant un espace de routage avec un ensemble de régions : le fédéré souscrit alors à la

région dans laquelle l'objet qu'il gère se trouve (ou un ensemble de régions proches). Un inconvénient de cette approche apparaît immédiatement concernant la taille des régions. Dans l'exemple classique de l'espace géographique en 2 dimensions, le partitionnement est une grille de régions identiques. La résolution de la grille doit être adaptée à la fédération, car une résolution trop grande multiplie à nouveau les appels aux services DDM, tandis qu'une résolution trop faible fait perdre l'intérêt des régions.

Une autre solution qui peut être acceptable consiste à surdimensionner les régions de manière à ne pas changer la région associée à un objet à chaque déplacement de celui-ci. Il est cependant difficile de donner une solution générale, ou d'affirmer qu'une telle solution peut convenir : en pratique, il faut tenir compte de la fédération effectivement utilisée et de ses caractéristiques.

3.4.3 Régions et multicast

Jusqu'à présent nous avons considéré les régions de routage uniquement sur un aspect logiciel permettant de réduire les échanges de données. Afin d'améliorer les performances, il est intéressant de conjuguer régions de routage et utilisation de protocoles multicast.

Nous reviendrons plus en détail sur les propositions relatives au multicast (indépendamment de l'emploi des régions de *Data Distribution Management*), néanmoins la prise en compte du multicast est aussi pertinente dans le cas des régions : en effet, les régions amènent à définir des destinataires particuliers pour certaines informations, du point de vue logiciel, ce qu'un protocole multicast se propose de réaliser sur un plan plus matériel, en définissant des groupes de destinataires physiques.

Une diffusion générale des informations (en *broadcast*) permet des envois effectués rapidement, mais le travail des fédérés destinataires est plus grand car un plus grand nombre de messages non pertinents sont envoyés. À l'inverse, une diffusion point-à-point assure que seuls des destinataires bien définis recevront les informations, mais c'est cette fois l'envoi qui nécessite un traitement coûteux, avec la détermination des destinataires et l'envoi d'un message pour chacun d'entre eux. Le multicast se place entre ces deux approches, à l'aide de groupes de destinataires : l'envoi est plus simple qu'en *broadcast* et seuls les destinataires souhaités reçoivent l'information.

L'emploi d'un protocole multicast amène alors de nouvelles conditions sur l'utilisation de *Data Distribution Management* : par exemple le nombre de groupes multicast étant limité, il n'est pas possible d'associer une région de routage à un groupe dès qu'un grand nombre de régions est mis en jeu [MBT99]. Une contrainte telle que le coût de modification d'un groupe multicast a conduit à des solutions fondées principalement sur des approches

statiques de la distribution des données.

3.4.4 Implémentations

Nous allons présenter différents types de mise en œuvre des espaces et régions de routage. Les plus simples d'entre elles ne nécessitent pas l'utilisation du multicast, bien que celui-ci améliore généralement très fortement les performances.

Avec le service *Data Distribution Management*, la définition de régions de mise à jour précises et de régions d'abonnement pertinentes doit permettre l'établissement d'un routage des données efficace.

Si ces régions permettent de minimiser les communications, elles nécessitent un travail supplémentaire de gestion de la part du RTI. Notamment parce qu'en pratique, les fédérés indiquent a priori des régions de routage différentes d'un fédéré à l'autre. Plusieurs implémentations permettent d'appréhender ce problème, [BBSd98] en présente quatre, partant de l'approche statique la plus courante à une proposition fortement dynamique.

Grille fixe

Dans cette implémentation, l'espace de routage est partitionné sous forme de grille (si on le considère en 2 dimensions). Lorsqu'une région de souscription S et une région de mise à jour U partagent une même cellule, le fédéré ayant défini U envoie des informations au fédéré ayant défini S. Mais il peut arriver, comme dans le cas de la figure 3.2, que la résolution de la grille soit faible devant la taille de régions, et que S et U partagent la même cellule sans pour autant se recouvrir.

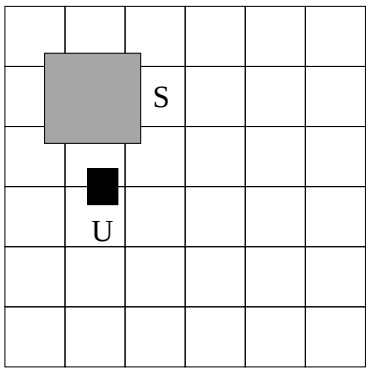

FIG. 3.2 – Espace de routage partitionné par une grille fixe

Cette méthode a pour avantage d'être simple à implémenter, et de ne demander que peu de ressources au cours de l'exécution. Par contre, le cas de la figure 3.2 est un inconvénient considérable. En effet, s'il est possible de choisir la résolution de la grille, il est difficile de trouver une valeur optimale dans le cas de régions de souscription et de mise à jour très nombreuses, en raison de la variété des fédérés. De plus, ces régions sont a priori dynamiques, tandis que la grille est statique.

Grille multi-niveaux

Afin de profiter des avantages de la grille fixe précédente, et d'en réduire les inconvénients, l'approche multi-niveaux se place dans le cadre d'une simulation à grande échelle, et propose des grilles de résolutions différentes sur les réseaux locaux et sur le réseau étendu (cf. fig. 3.3)

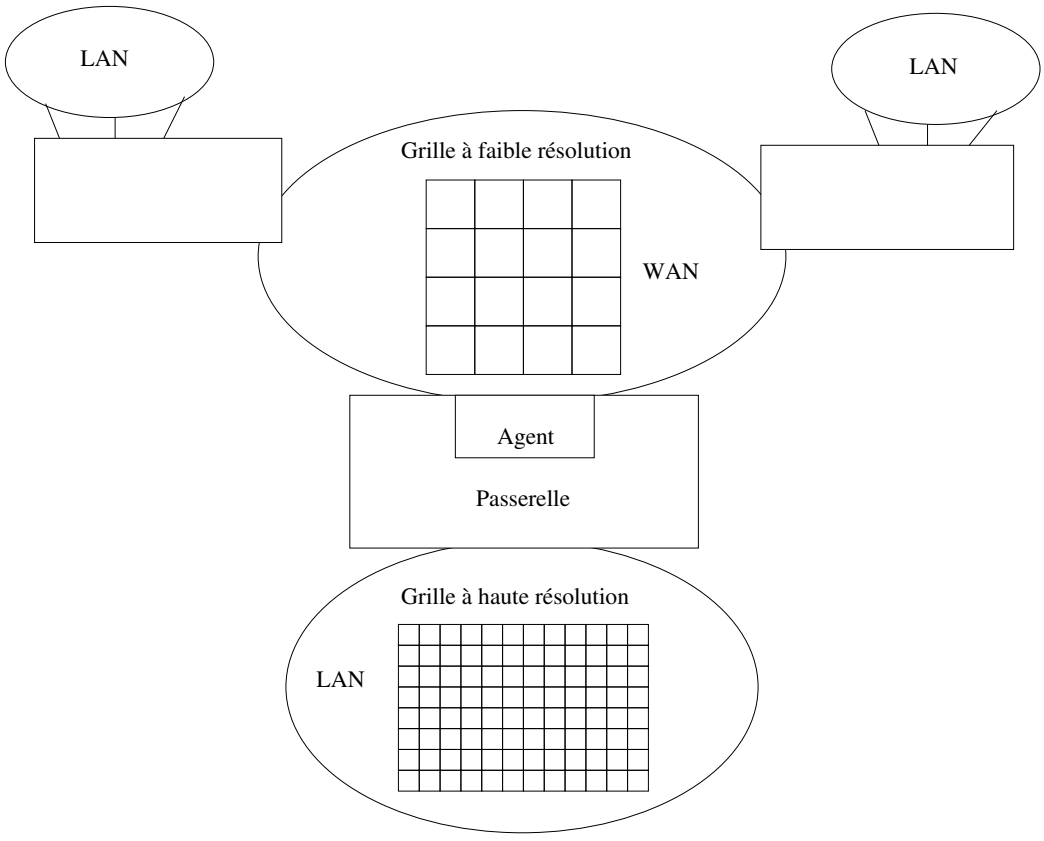

FIG. 3.3 – Grille à 2 niveaux

La résolution de la grille étant faible au niveau WAN, le nombre de cellules

n'est pas dissuasif vis-à-vis des ressources nécessaires. Par contre, le filtrage est loin d'être suffisant, mais il ne s'agit que d'un premier filtrage, le second étant effectué au niveau de la grille du LAN. Cette méthode est bien meilleure que la précédente, notamment parce qu'elle ne nécessite pas d'augmenter le nombre de groupes multicast. Cependant, ces allocations étant toujours statiques, elle reste peu performante.

Regroupements

Il s'agit ici de former des groupes multicast constitués d'objets ou fédérés qui ont des régions de souscription/mise à jour identiques. Cette méthode est efficace pour de petites simulations, mais le nombre de groupes multicast devient vite trop élevé dans le cas de grandes simulations. Il est alors possible de définir les groupes non pas en se basant sur des régions identiques, mais des régions proches : a priori, des régions proches dans l'espace de routage devraient avoir de nombreux abonnés communs. La détermination des groupes ou *clusters* revient à grouper les régions de mise à jour, de telle sorte que celles qui appartiennent à un cluster, soient plus proches les unes des autres que de n'importe quelle autre région de mise à jour (appartenant à un autre cluster.) Il est possible de déterminer de telles configurations, et ce travail représente une faible charge dans le cas statique. Mais pour être efficace, cette méthode doit être employée dynamiquement, et la détermination dynamique des régions peut avoir un coût trop élevé.

Approche hybride grille/clusters

La première phase de cette méthode consiste à créer initialement des régions de l'espace de routage nommées *clusters* et destinées à être associées à des groupes multicast. Les clusters sont déterminés en suivant le principe de la section précédente. Des algorithmes permettent de réaliser cette opération efficacement [BAM82][JD88] ; étant donné que cette répartition est effectuée une seule fois lors de l'initialisation, la complexité n'est pas dissuasive.

La deuxième phase a lieu au cours de l'exécution de la fédération. Il s'agit de maintenir une bonne répartition des clusters, alors que les régions de mise à jour évoluent dans l'espace de routage. Il s'agit donc de déterminer, lorsqu'une région se déplace, si elle doit rester dans le même cluster ou s'il faut l'assigner à un autre.

Une approche directe consiste à calculer dynamiquement toutes les distances entre les centres des clusters et les régions de mise à jour, en déduire les nouvelles associations entre régions et clusters, et recalculer les positions des centres des clusters ; ce calcul est trop lourd pour être effectué dynami-

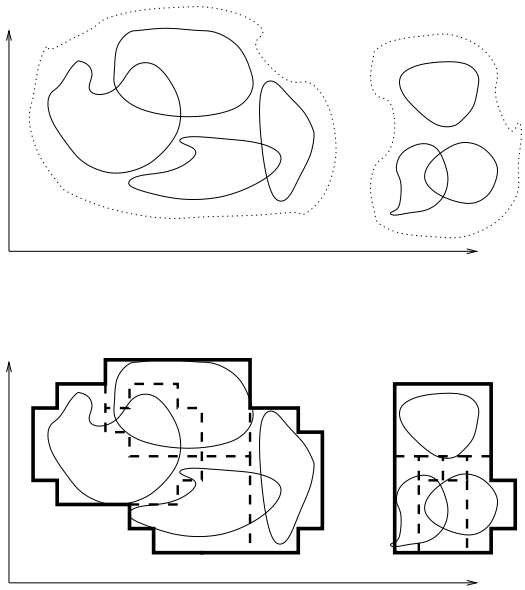

FIG. 3.4 – Approche hybride des groupes multicast

quement, même si l'on ne prend en compte que les clusters susceptibles d'être modifiés, car la première partie (calcul des distances) a tout de même lieu à chaque fois qu'une région de mise à jour est modifiée.

Cette deuxième phase sera donc de préférence appréhendée de la manière suivante (fig. 3.4). Chaque cluster, constitué d'un ensemble de régions de mise à jour, est discrétisé sur une grille de cellules à taille fixe, équivalente en résolution à la grille de bas niveau de la figure 3.3. Les régions de mise à jour sont placées sur cette grille en inscrivant le nom du fédéré qui publie dans chaque cellule recouverte. Ainsi, un cluster est constitué de cellules et chacune de ces cellules indique au moins un fédéré qui y publie. Une cellule vide ne peut pas appartenir à un cluster, et chaque cellule non vide n'appartient qu'à un seul cluster.

Les mises à jour de clusters sont déclenchées uniquement lorsqu'une cellule devient vide, ou lorsqu'une cellule vide devient occupée. On y associe deux opérations, respectivement *leave* et *join*.

- Lors du *leave*, la cellule est retirée du cluster auquel elle était associée, et le centre du cluster est recalculé.
- Lors du *join*, la cellule est ajoutée au cluster le plus proche. La distance d'un cluster sera la distance à son centre, ou la distance à sa cellule la plus proche ; la première distance permet de maintenir des clusters compacts, la deuxième a l'avantage d'être plus simple à calculer.

- Pour éviter qu'un cluster s'étende trop, il faut également prévoir une opération *split* permettant de couper un cluster en deux. Il faut pour cela qu'il y ait des groupes multicast libres.
- Ce sera plus probablement le cas si l'on dispose également d'une opération de fusion *merge* permettant de réunir deux clusters similaires (ie. de centres proches) en un seul.

Avec cette structure en place, le routage de données a lieu lorsque des régions de mise à jour et de souscription couvrent le même cluster. Un groupe multicast est associé à chaque cluster. Chaque fédéré s'abonne aux groupes multicast dont le cluster associé recouvre tout ou partie de la région de souscription. Les mises à jour d'attribut et les interactions ne sont envoyées qu'aux groupes multicast dont le cluster associé couvre la région de mise à jour de l'objet qui publie.

3.4.5 Conclusion

Les services de gestion de la distribution des données permettent aux fédérés de définir des domaines d'intérêt, des domaines d'action, afin de mieux déterminer les transferts d'information pertinents dans l'exécution d'une fédération. L'emploi de ces services peut améliorer la cohérence du modèle en cachant effectivement des informations inaccessibles à l'objet modélisé, ou en localisant la production d'information. Ces précisions fournies au RTI ont un coût, mais permettent d'envisager une optimisation des échanges de données. Les tentatives d'implémentation de ces optimisations ont montré que le gain sur des domaines très dynamiques ne compensait pas la gestion de tels domaines : la plupart des implémentations attendent l'utilisation de régions statiques adaptées à la fédération exécutée. D'une manière plus générale, il n'existe pas d'approche unique de l'implémentation de ces services qui serait adaptée à toute fédération. La solution passe donc par le développement de plusieurs stratégies possibles et fortement paramétrables.

Enfin, l'emploi de multicast peut améliorer les performances en associant des régions ou des groupes de régions à des groupes multicast. Mais l'implémentation de *Data Distribution Management* doit alors aussi tenir compte des limitations imposées par le multicast, comme le nombre de groupes et le coût de reconfiguration des routeurs multicast.

3.5 Multicast

Enfin, la notion de domaine apparaît avec l'emploi de protocoles multicast évoqués précédemment : leur utilisation ne se limite pas au cas particulier de

Data Distribution Management. Il s'agit, d'une manière bien plus générale, de profiter des caractéristiques du multicast dans l'ensemble des échanges de messages d'une simulation. En effet, le transfert d'information d'une source à plusieurs destinataires (et non l'ensemble des destinataires possibles) est un schéma très fréquent dans l'exécution des simulations. Le multicast a été envisagé dès DIS pour réduire des coûts de communication qui sans son utilisation se traduisent soit par un excès de messages inutiles (destinataires non concernés) soit par une plus grande complexité des échanges (réalisation par un ensemble d'envois point à point).

Cependant, il faut insister sur le type de protocole multicast requis par les simulations : au moins pour une partie des données à transmettre, la fiabilité est essentielle. Le problème est donc bien l'emploi d'un protocole multicast *fiable*.

Le protocole multicast le plus répandu est UDP. Il permet de transmettre des informations d'une source vers plusieurs destinataires, réunis en *groupes multicast*. Mais ce protocole ne prévoit ni la conservation de l'ordre des paquets, ni la vérification de leur réception (pas d'accusé de réception, positif ou négatif). UDP peut donc être utilisé sur un réseau fiable, ou lorsque les messages ne nécessitent pas d'être reçus avec une grande fiabilité.

Le problème de la fiabilité est pris en compte dans un protocole (non multicast) tel que TCP. TCP s'occupe des paquets perdus par accusés de réception positifs, et ordonne correctement ces paquets. L'inconvénient est la lourdeur du protocole en raison de la quantité de vérifications à effectuer. C'est de plus un protocole reliant un unique expéditeur à un unique destinataire. La transmission d'un message nécessite l'établissement d'autant de connexions qu'il y a de destinataires.

Un multicast fiable peut être approché à partir de ces deux protocoles, TCP et UDP. Deux approches sont possibles :

- soit le protocole TCP est utilisé, mais il faut alors gérer les aspects multicast qu'il ne possède pas, c'est-à-dire l'envoi à de multiples destinataires en réalisant une connexion pour chacun d'entre eux ;
- soit le protocole UDP est utilisé, et il faut améliorer sa fiabilité par l'ajout de vérifications par exemple à l'aide d'accusés de réception.

Dans chacune de ces approches, l'ajout nous éloigne de ce qui est attendu dans un véritable protocole multicast fiable. Le premier cas simplifie la tâche de l'utilisateur mais techniquement cela reste de l'unicast et les performances ne sont pas améliorées. Le second cas alourdit fortement les échanges en raison des messages de vérification.

Enfin il faut noter les propositions de protocole de communication repo-

sant sur des réseaux adaptatifs ou des routeurs intelligents. Un des principes utilisés est de bâtir un arbre sur les moyens de communication et de s'en servir pour construire d'une manière distribuée les accusés de réception du réseau. Pullen a étudié l'application de ces nouvelles propositions à la simulation distribuée [Pul99].

Il faut alors envisager le contexte de notre utilisation : la simulation. Dans les adaptations de TCP et d'UDP ci-dessus, aucune distinction n'est faite entre les différents échanges de messages ayant lieu au cours d'une exécution. Or il est possible d'exploiter le fait que les messages peuvent être classés en deux grandes catégories :

- ceux nécessitant la fiabilité : en général il s'agit d'informations ponctuelles devant être absolument connues des destinataires ;
- ceux ne nécessitant pas la fiabilité en raison de leur mise à jour périodique (par exemple la représentation de phénomènes continus).

Différentes propositions de protocoles consistent donc à adapter la prise en charge de chaque message en fonction de ces caractéristiques.

3.6 Interfédérations

Enfin, la notion de domaine apparaît lorsque l'on réalise des simulations à l'aide de plusieurs fédérations interconnectées. Ces simulations sont appelées *interfédérations*. Les simulations HLA classiques sont constituées d'une seule fédération : un unique RTI et des fédérés. Il s'agit donc ici d'obtenir une simulation reposant sur plusieurs fédérations, et en particulier plusieurs RTI.

Ces interfédérations sont réalisées en interconnectant plusieurs fédérations à l'aide de ponts. Il en existe plusieurs sortes : on peut par exemple connecter directement deux RTI. HLA ne prévoit pas ce cas, cependant la définition d'un protocole permettant la communication directe d'un RTI à un autre permet de l'envisager. Un autre type de pont, plus fréquemment rencontré, est le *fédéré-pont* : il s'agit d'un fédéré ayant rejoint plusieurs fédérations et transmettant les informations de l'une à l'autre simplement par l'interface de HLA [BL97].

L'emploi d'interfédérations peut être motivé par des objectifs très différents : interopérabilité, sécurité, performances, amélioration de la modélisation. Nous allons rapidement évoquer ici ces intérêts, et nous y reviendrons au cours des chapitres 5 et suivants, consacrés à notre contribution sur ce sujet.

3.6.1 Interopérabilité

Un fédéré-pont relie deux fédérations en étant connecté à chacune d'entre elles et en reproduisant dans l'une des événements observés dans l'autre. Ce principe n'est pas limité à des simulations HLA : nous nous intéresserons principalement aux ponts n'impliquant que HLA, mais les ponts sont également employés afin de réaliser des simulations reposant sur des architectures de simulation différentes. Cet usage a en particulier été considéré aux débuts de HLA, alors qu'il existait de nombreuses simulations DIS. Avant que la transition vers HLA ne soit complète, il était possible de profiter de composants de simulation DIS en les connectant à une fédération HLA avec un tel pont.

Un pont entre fédérations HLA peut être considéré pour les mêmes motivations, car bien qu'il s'agisse de HLA de part et d'autre, tous les RTI ne sont pas au même stade de développement et ne fournissent pas les mêmes API (langage utilisé, services implémentés).

Enfin le pont peut apporter une interopérabilité immédiate en effectuant des conversions sur les valeurs des données échangées. Deux fédérés n'utilisant pas le même type de coordonnées peuvent être utilisés dans une même simulation si un pont convertissant correctement les valeurs relie leurs fédérations respectives [PHS00].

3.6.2 Sécurité

Étant le passage obligé entre deux fédérations, un fédéré-pont se trouve dans une situation privilégiée pour effectuer un filtrage des informations. Avec un tel filtrage, il est possible de ne faire apparaître certains événements que dans une partie de la simulation : une fédération donnée. Le pont peut notamment profiter des services de *Data Distribution Management* pour cela [PHS00]. Si ce filtrage est essentiel dans le cas du besoin de confidentialité, il présente aussi un intérêt pour le point suivant, la modélisation.

3.6.3 Modélisation

HLA propose de séparer les composants du modèle à simuler en fédérés de même niveau. Mais le modèle de la simulation est souvent plus complexe : il serait plus pertinent de faire apparaître des sous-systèmes de manière hiérarchique. Par exemple, une simulation peut naturellement se décomposer en plusieurs simulateurs de complexités diverses. Les plus simples pourront être réalisés avec un seul fédéré, mais les plus complexes nécessiteront plusieurs fédérés correspondant à leurs propres composants. Après une telle décompo-

sition, HLA ne propose que de placer tous les fédérés déterminés au même niveau, il n'y a pas de hiérarchie. On peut alors envisager d'utiliser des interfédérations afin que chaque simulateur complexe soit réalisé par une fédération à part entière, un unique fédéré servant alors de façade à la fédération principale. Ainsi le fonctionnement interne de ce simulateur n'apparaît qu'aux fédérés concernés, les autres n'ayant accès qu'à une vue externe.

On trouve cette approche dans les *fédérations hiérarchiques* proposées par [CTG98]. Mais la répartition de la simulation en plusieurs fédérations peut aussi avoir pour but de mettre en œuvre un ensemble de fédérations homogènes plutôt qu'une fédération hétérogène : dans [Lak98], Lake considère des critères de regroupement tels que les débits d'événements plutôt que la hiérarchie du modèle. Il préconise le développement séparé de ces fédérations homogènes pour ensuite les relier à l'aide de ponts, et les exécuter chacune sur du matériel adapté à leurs besoins.

Chapitre 4

Le projet CERTI

En 1996, HLA est devenu le standard du DoD pour les nouvelles applications de simulation, toutes les anciennes simulations devant s'y conformer également dans les cinq ans.

L'ONERA s'est également tourné vers cette norme pour le développement de simulations distribuées. On trouve parmi les objectifs de cette initiative :

- le souhait de comprendre et maîtriser l'architecture HLA, de déterminer des méthodologies de développement et de programmation adaptées aux utilisateurs de l'ONERA (notamment pour des simulations distribuées impliquant des métiers différents) ;
- envisager des travaux exploitant les compétences acquises dans le domaine de la sécurité et des systèmes distribués ;
- le développement d'une architecture de simulation distribuée sécurisée.

4.1 Historique et évolutions

Cet intérêt pour HLA s'est notamment concrétisé par le développement d'un prototype de RTI dès 1996. En effet certains objectifs relatifs à la sécurité ou l'optimisation exigent d'intervenir directement sur le RTI. HLA était alors une norme très récente ne disposant pas d'implémentations facilement réutilisables (par exemple libres, ou « *Open Source* »).

Afin de mieux cerner HLA et d'obtenir rapidement des simulations fonctionnelles, l'objectif principal a été de développer un prototype minimal proposant un ensemble restreint de services. Ainsi la gestion de la propriété et la gestion de la distribution des données n'ont pas été considérées dans un premier temps. Les autres groupes de services n'ont pas été développés entièrement, certains mécanismes comme la sauvegarde et la restauration de fédérations n'étant pas pertinents pour cette première étape. Ont égale-

ment été privilégiés le fonctionnement en temps coordonné conservatif, et des échanges de données fiables (avec le protocole TCP).

Ce prototype, appelé CERTI[1] a atteint ce stade en 1997, et a ensuite évolué au fur et à mesure de travaux relatifs à HLA : soit tout simplement par le développement de nouveaux services de la norme HLA, soit par l'ajout de fonctionnalités spécifiques ou par des optimisations.

En 1998, une deuxième version a été réalisée, apportant principalement des optimisations (en particulier sur la taille des messages transitant sur le réseau).

La version suivante a apporté des extensions permettant des communications sécurisées entre certains fédérés. L'objectif recherché était d'envisager la réalisation de fédérations dont les fédérés sont fournis par plusieurs entreprises ou organisations : HLA donne une situation privilégiée au RTI et ne permet pas de restreindre les données partagées à des groupes de fédérés.

CERTI a ainsi évolué en suivant les services décrits dans la norme 1.1 de HLA. En 2000, une nouvelle version a réalisé la transition vers HLA 1.3, la dernière version proposée par le DoD (l'évolution de HLA n'est plus envisagée que dans sa version de l'IEEE).

Enfin, les dernières versions ont apporté les services de gestion de la propriété, la conformité des descriptions de fédérations à HLA 1.3, et une partie des services de gestion de la distribution des données. À partir de 2002, nous nous sommes occupés de la distribution de CERTI en tant que logiciel libre sous licence *copyleft*[2] conformément à la politique scientifique de l'ONERA (cette contribution est détaillée dans l'annexe A, p. 125.)

4.2 Mise en œuvre

4.2.1 L'architecture de CERTI

Dès son premier prototype, CERTI a été conçu selon une architecture de processus communicants. Il est ainsi constitué de trois composants :

- Le RTIG ou *RTI Gateway*, composant central de l'architecture ;
- Le RTIA ou *RTI Ambassador*, composant local accompagnant chaque fédéré ;
- La libRTI, bibliothèque HLA fournissant les services requis aux fédérés.

Ces trois composants sont présentés sur la figure 4.1.

[1] Contraction de CERT (Centre d'Études et de Recherches de Toulouse – ONERA Toulouse) et de RTI.

[2] GNU GPL et GNU LGPL

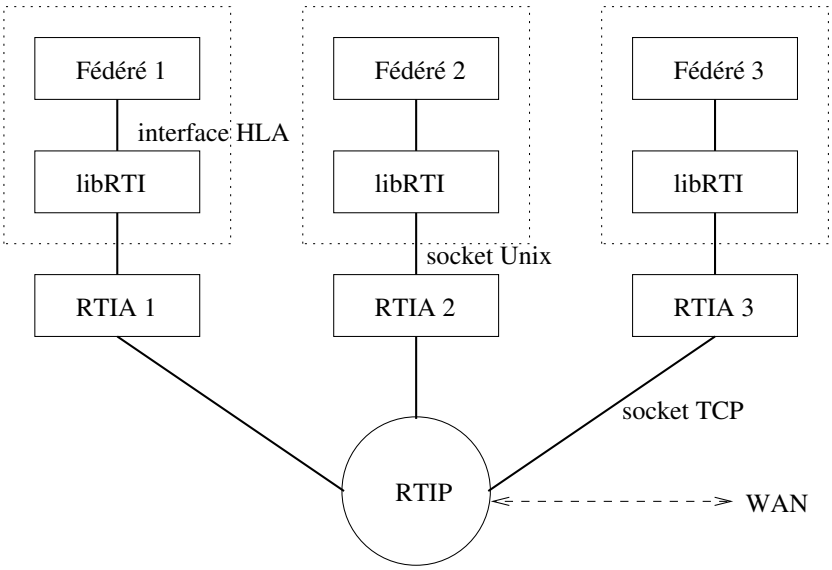

FIG. 4.1 – Architecture de CERTI

Chaque fédéré doit être compilé et lié à la bibliothèque *libRTI* : cette bibliothèque correspond à l'API de HLA, elle fournit des services au fédéré, et utilise ceux du fédéré. Comme le reste de CERTI, cette bibliothèque est écrite en C++, et ne permet donc d'exécuter avec CERTI que des fédérés écrits en C++ et compilés de la même manière.

Le travail de cette libRTI consiste essentiellement à traduire les appels de services HLA en messages et à les envoyer vers un composant RTIA avec des sockets Unix. De même, les réponses du RTIA sont traduites soit en valeur de retour de la fonction appelée, soit en appels de services du fédéré (*callbacks*). Ainsi, conformément à l'API C++ fournie par HLA, un fédéré utilise le RTI uniquement par des appels de fonction.

Les informations relatives aux appels de services effectués par un fédéré parviennent donc au RTIA qui l'accompagne. Si ce composant local est en mesure de réaliser seul l'opération demandée, celle-ci est effectuée et la réponse est retournée au fédéré par l'intermédiaire de la libRTI. C'est par exemple le cas d'une demande d'identifiant (*handle*) à partir d'un nom de classe. Si le service demandé intervient sur l'ensemble du RTI, par exemple s'il s'agit d'une mise-à-jour d'attributs à diffuser vers d'autres fédérés, la requête est transmise au composant central, le RTIG. Pratiquement ceci est effectué par un nouvel échange de messages, cette fois avec une connexion TCP.

Enfin, le RTIG est le point de passage obligatoire de toute information transitant d'un fédéré à un autre. Aucun transfert de données n'a lieu entre

deux RTIA : celles-ci vont de l'un à l'autre via le RTIG. Le RTIG rassemble tout ce qui est de nature globale dans la fédération : les informations sur les fédérés, les publications et souscriptions de classes, l'état concernant les procédures de sauvegarde, de restauration et de synchronisation. Il a aussi pour but de transmettre les informations d'un RTIA à un autre en fonction de la pertinence de ces échanges. Ainsi, lors de la mise à jour d'un attribut, seuls les RTIA des fédérés abonnés recevront un message informant de cette mise à jour. Le RTIG réalise alors une forme de multicast logiciel : cette opération est transparente du point de vue des fédérés. Cependant elle n'est pas réalisée à l'aide d'un protocole multicast mais par un ensemble de connexions TCP. Ceci est illustré par la figure 4.2 : un fédéré informe de la mise à jour d'un attribut avec le service *Update Attribute Values* (noté UAV), et se voit notifier la mise à jour d'un attribut d'un autre fédéré par le service *Reflect Attribute Values* † (noté RAV). Si une telle mise à jour de la part d'un fédéré 1 doit être vue par un fédéré 2, l'information passe par le RTIA du premier fédéré, le RTIG, le RTIA du second fédéré et enfin ce fédéré. La figure ne montre qu'un seul fédéré destinataire : s'il y en a plusieurs, le RTIG envoie autant de messages que nécessaires aux RTIA concernés.

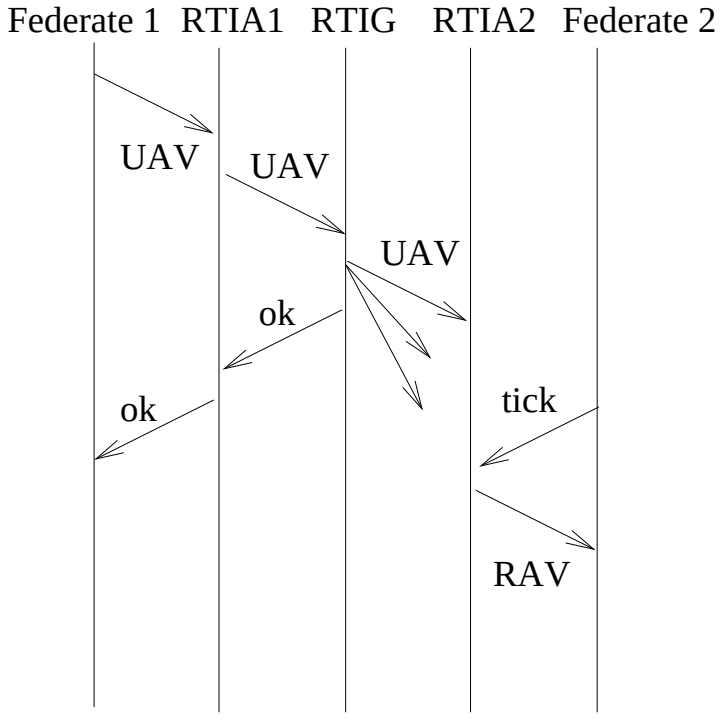

FIG. 4.2 – Transfert de données dans CERTI

4.2.2 L'emploi de standards

CERTI est développé sur la base de standards existants. Ce choix évite d'une part des redéveloppements inutiles d'outils existants et d'autre part facilite la maintenance et la portabilité.

Ainsi CERTI est développé en C++ : bien que la norme ISO de ce langage n'ait été publiée qu'en 1998, CERTI a utilisé dans un premier temps les éléments de ce langage qui étaient alors des standards de fait. Avec la publication de la norme C++ et la prise en compte progressive de celle-ci par les compilateurs courants, c'est maintenant l'ensemble de ce langage et de sa bibliothèque standard qui est considérée et privilégiée pour le développement.

Mais CERTI ne peut se limiter aux possibilités du C++. Là encore des protocoles et des bibliothèques standard sont utilisées : TCP/IP, standard Unix. Le même choix a été effectué pour des extensions de sécurité avec GSSAPI [Lin97, BCSZ98].

Au cours des travaux concernant la gestion de la distribution des données, le format XML de description des fédérations de la version IEEE de HLA a été utilisé. Suite à la distribution en logiciel libre de CERTI, un contributeur a également utilisé XML pour les services de sauvegarde et de restauration de fédération. Pour ces deux utilisations, *Libxml2*[3], une bibliothèque libre implémentant de nombreux standards relatifs à l'accès aux documents XML a été employée.

À ces standards utilisés pour le développement proprement dit de CERTI, on peut bien sûr ajouter HLA : la pérennité de sa définition est tout aussi importante que celle des technologies impliquées.

4.2.3 Les performances

Le test de performances de la figure 4.3 montre le nombre de pas de temps effectués par une simulation en temps coordonné et à pas fixe pendant une durée donnée, en fonction du nombre de fédérés de la simulation (chaque fédéré est exécuté sur une machine différente, le nombre de machines augmente donc avec le nombre de fédérés). Les fédérés font des mises à jour d'attributs et sont tous abonnés aux classes associées.

Il apparaît donc que les performances sont rapidement dégradées lorsque le nombre de fédérés augmente : c'est en partie dû à la complexité du monde simulé, qui augmente avec le nombre de fédérés. Mais le RTIG, par sa position centrale, limite aussi l'exécution de l'ensemble de la simulation.

La figure présente les résultats obtenus avec CERTI et le RTI-NG du DoD. Sur cet essai CERTI semble plus rapide, mais l'ordre de grandeur des résultats

[3]http ://www.xmlsoft.org

reste le même (en particulier la dégradation des performances avec le nombre de fédérés). La différence peut aussi être attribuée à l'absence d'optimisation dans la configuration du RTI-NG lors de ce test.

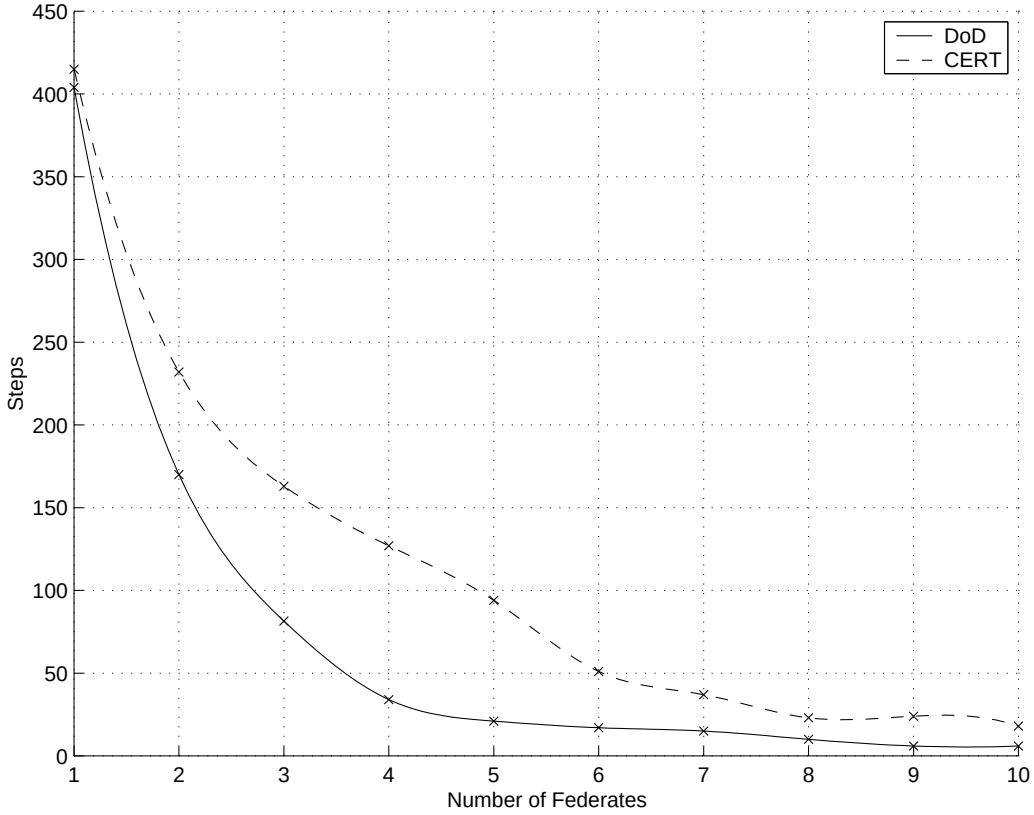

FIG. 4.3 – Performances de CERTI et du RTI-NG

4.3 Travaux associés

Si CERTI représente la part la plus significative de l'initiative HLA de l'ONERA, les objectifs concernent néanmoins le développement de simulations distribuées complètes, et sont donc aussi à placer du côté utilisateur. Nous présentons rapidement quelques travaux concernant HLA et fortement liés à CERTI.

4.3.1 Extensions de sécurité

Une étude a concerné le cas des simulations distribuées constituées de fédérés appartenant à différentes organisations ou compagnies [BCSZ98]. Dans

ce cas, les données partagées par les fédérés sont a priori accessibles de *tous* les fédérés. Or l'une des organisations peut fournir un ensemble de fédérés dont les données partagées seraient à ranger en deux catégories :

- des données privées, vues par ces fédérés et non par les fédérés de fournisseurs différents ;
- des données publiques correspondant à la véritable interface publique de ce système de fédérés.

Le résultat a été la proposition d'une architecture basée sur CERTI avec des extensions de sécurité, avec la définition de niveaux de sécurités associés aux données, et l'utilisation de données sécurisées avec l'interface GSSAPI [Lin97]. La figure 4.4 donne un exemple dans lequel les communications sont illimitées au sein de chacune de deux organisations, tandis que seules des données autorisées peuvent passer de l'une à l'autre. Le RTIG doit de plus être exécuté par un tiers n'ayant pas d'intérêt dans l'obtention des données restreintes.

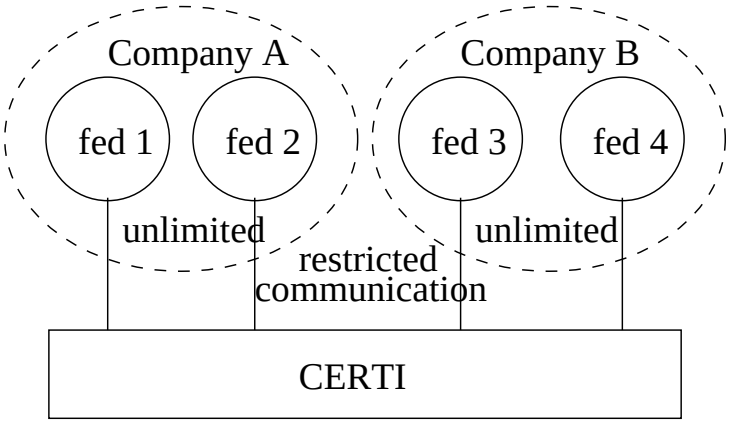

FIG. 4.4 – Simulation avec extensions de sécurité

4.3.2 Applications

Des applications de simulation ont été développées à l'aide de CERTI. En particulier une application de défense aérienne [AS00], à partir de laquelle a été étudié le problème de multirésolution [AS01] évoqué précédemment.

Des applications plus complexes sont actuellement développées :

- un projet d'*Aéroport du futur* [Ade99] initialement développé à l'aide du RTI-NG ;
- une application de détectabilité de cibles par un radar passif impliquant plusieurs métiers (plusieurs départements) et éventuellement plusieurs sites.

4.3.3 Contributions relatives aux domaines

Enfin, nos contributions relatives aux domaines ont également été effectuées à l'aide de CERTI. Il s'agit principalement du travail sur les interfédérations présenté dans la deuxième partie de ce mémoire, mais aussi de l'ajout de services de *Data Distribution Management* (présenté succinctement en annexe, § A.4).

Deuxième partie

Contributions

Chapitre 5

Interfédérations

5.1 Principe

Une *interfédération* (ou interconnexion de fédérations) est un ensemble de fédérations coopérant les unes avec les autres afin de réaliser une unique simulation globale. Contrairement aux simulations HLA classiques où l'on a un seul RTI auquel se connectent des fédérés, il s'agit dans une interfédération d'avoir plusieurs fédérations, c'est-à-dire plusieurs RTI. Chaque fédération reste bien sûr constituée de fédérés.

Bien que ce ne soit pas systématique, une interfédération a souvent pour objectif de proposer une simulation équivalente à celle qui serait constituée d'une seule fédération supportant les mêmes fédérés. Les fédérés, les objets et les événements sont les mêmes, mais différentes raisons peuvent inciter à l'utilisation d'une interfédération plutôt que d'une fédération simple.

Dans un premier temps, nous n'allons considérer que cette définition d'interfédération, sans envisager de mise en œuvre particulière. Cependant, il est préférable d'évoquer dès maintenant un exemple d'interfédération : celui des ponts de fédérations, que nous présenterons en détail au chapitre suivant.

HLA définit très précisément les relations entre les composants d'une fédération : un fédéré n'échange des informations qu'avec son RTI, et le RTI ne communique qu'avec des fédérés. Lorsqu'une information est transmise d'un fédéré à un autre, c'est par le biais du RTI, et aucun mécanisme n'est prévu pour des échanges entre plusieurs RTI. Dans ce cadre, une mise en œuvre d'interfédération possible consiste à établir des liens entre fédérations à l'aide de fédérés, que l'on appelle alors *ponts de fédérations* ou *fédérés-ponts*.

Quand l'interconnexion de deux fédérations est réalisée à l'aide d'un *fédéré-pont*, ce pont est en fait un fédéré appartenant à plusieurs fédérations (comme dans l'exemple de la figure 5.1). Le fédéré-pont ne gère pas

d'objets lui-même, mais doit fournir à chaque fédération des *représentations* des objets de l'autre fédération. L'objectif est d'obtenir un comportement de ces objets représentants identique à celui des objets originaux. C'est toute une fédération qu'un pont doit simuler à une autre fédération.

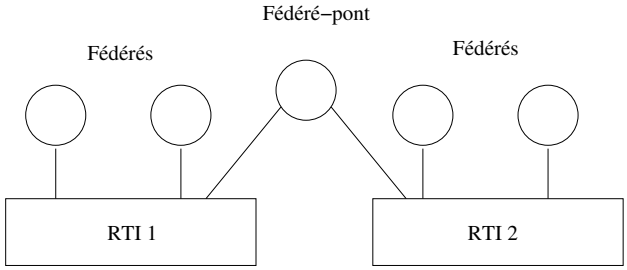

FIG. 5.1 – Interfédération simple

5.2 Intérêts

Plusieurs objectifs peuvent motiver la conception et l'utilisation d'interfédérations : optimisation et extensibilité, sécurité, interopérabilité et réutilisation. Souvent, ces intérêts ne sont pas spécifiques à une mise en œuvre particulière telle que celle des fédérés-ponts ci-dessus : il s'agit avant tout de considérer l'interfédération comme une interconnexion de fédérations, quel que soit le moyen employé pour réaliser cette interconnexion. Nous utiliserons cependant les ponts de fédérations et le cas de CERTI comme exemples.

5.2.1 Extensibilité, optimisation

Dans une fédération classique, le RTI est le point de passage obligatoire d'un grand nombre d'informations.

L'architecture de CERTI est constituée d'un processus central (le RTIG) et de processus locaux (les RTIA) accompagnant les fédérés. Bien que l'utilisation d'un composant local du RTI avec chaque fédéré allège la charge du RTIG, ce dernier reste le goulot d'étranglement limitant la vitesse d'exécution des fédérations à grand nombre de fédérés. Une solution initialement envisagée est de distribuer le RTIG en un ensemble de processus communicants. Une autre approche consiste à utiliser une interfédération : pour obtenir un RTIG distribué, le rôle du RTI est réparti sur plusieurs fédérations, utilisant chacune un RTI. Pour une même simulation (c'est-à-dire une interfédération

dans laquelle les fédérés mis en place sont les mêmes que dans la fédération simple équivalente), le nombre de fédéré connectés à un RTI devient plus faible, et d'autant plus que le nombre de fédérations interconnectées est grand. Mais bien sûr la gestion de l'interfédération devient également plus complexe avec le nombre de fédérations. L'objectif est de renforcer la notion de domaines, afin d'éviter tout transfert de données inutile entre fédérations, en particulier à l'aide des services de gestion de la distribution de données [PHS00].

Ces objectifs d'optimisation peuvent provenir des difficultés rencontrées dans des fédérations hétérogènes [Lak98]. Il s'agit de fédérations dans lesquelles ont peut répartir les fédérés en plusieurs groupes, certains groupes engendrant de très forts débits d'événements. S'il est possible d'utiliser des réseaux physiques adaptés à ces débits, le besoin d'un tel réseau peut n'apparaître que pour une partie des fédérés. Il s'agit donc dans ce cas de réaliser plusieurs fédérations, chacune étant homogène par le débit d'événements de ses fédérés et mise en œuvre sur un réseau physique adapté. La simulation est ensuite réalisée en connectant les différentes fédérations à l'aide de fédérés-ponts.

5.2.2 Sécurité

Les objectifs de sécurité apparaissent lorsque se pose le problème de la visibilité des données échangées. Le modèle proposé par HLA permet de définir une représentation du monde simulé, l'ensemble de ces informations étant public : tout fédéré peut y avoir accès, pourvu qu'il procède à l'abonnement des informations voulues.

L'existence de plusieurs fédérations pour une même simulation permet d'envisager, sans extension de la norme HLA, l'introduction de visibilités différentes pour les éléments partagés de la simulation. Chaque fédération possède une description indiquant quelles sont les informations partagées entre fédérés : définition de classes d'objets et d'interactions, etc. Chaque fédéré s'abonne aux classes qui l'intéressent, et indique s'il publie des objets ou des interactions de ces classes. Ces informations partagées sont globales, dès lors qu'un fédéré s'abonne, il est assuré de recevoir les informations correspondantes.

Avec une interfédération, il est possible de considérer des descriptions de fédérations différentes. Bien sûr, pour que des données soient échangées entre fédérations de la même manière qu'au sein d'une unique fédération, il est nécessaire d'avoir une intersection commune entre les FOM des deux fédérations. Tout type d'information existant dans plusieurs fédérations peut être

considéré comme si l'on avait affaire à une description commune pour chaque fédération. Par contre, toute information dont le type n'existe que dans une fédération ne pourra pas être reproduite dans les autres fédérations, les classes supportant cette information n'existant pas. Et de manière plus générale, on peut choisir un ensemble d'informations à considérer comme privées : alors, même si le même type d'information existe dans deux fédérations différentes, le contrôle de la liaison entre les fédérations permet de filtrer les messages et de ne laisser passer que ce qui est considéré comme public. Ce type d'utilisation rejoint les objectifs des extensions de sécurité réalisées sur la plate-forme CERTI [BCSZ98], et est employé dans les fédérations hiérarchiques [CTG98].

5.2.3 Interopérabilité

Une interfédération étant par définition l'interconnexion de plusieurs fédérations, c'est-à-dire de fédérés regroupés autour d'un RTI, elle implique soit un RTI exécutant les différentes fédérations, soit plusieurs RTI. Ces deux cas sont similaires, les fédérations étant indépendantes les unes des autres, même si elles sont gérées par un seul RTI. Cette différence est importante du point de vue des charges des machines exécutant les RTI, mais pas d'un point de vue logique.

Dans le cas le plus fréquent où les RTI de l'interfédération sont répartis sur plusieurs machines, il est possible d'utiliser des RTI de fournisseurs différents. HLA fournit la spécification de ces RTI mais il existe des différences entre les différents RTI existants, et on peut préférer un RTI plutôt qu'un autre pour différentes raisons : par exemple, certains RTI fournissent des interfaces pour plusieurs langages comme Java ou Ada, tandis que dans le cas de CERTI, seule l'API C++ est implémentée. On peut aussi tout simplement préférer un RTI parce qu'il est plus performant avec des fédérations particulières, ou plus facile à déployer. Ces préférences peuvent ne concerner qu'une partie d'une simulation (un ensemble de fédérés) et plusieurs RTI peuvent apparaître préférables, chacun pour une partie donnée de la simulation. Un exemple dans lequel cet aspect a un caractère obligatoire est celui de l'utilisation principale de CERTI pour une fédération, bien que certains fédérés de celle-ci nécessitent une API Java. Une interfédération permet d'exécuter ces fédérés avec un RTI supportant Java tel que le RTI-NG, tandis que le reste des fédérés utilise CERTI.

Ici une autre solution serait d'écrire une API Java pour CERTI, ou une bibliothèque intermédiaire dans le cas de plusieurs RTI propriétaires. Un autre problème d'interopérabilité vient du fait que HLA définit l'API du RTI, mais aucune norme binaire : le nom des bibliothèques associées à la libRTI et plus généralement tout ce qui concerne l'interface binaire de la

bibliothèque n'est pas imposé. Dans cette situation où il n'existe pas de standard, si un fournisseur de fédéré ne souhaite pas diffuser son code source, le fédéré doit être distribué compilé pour un RTI particulier. Dans le cas de plusieurs fédérés propriétaires, compilés pour des RTI différents, l'utilisation d'une interfédération est nécessaire.

5.3 Types d'interfédérations

Nous avons principalement évoqué les fédérés-ponts pour la mise en œuvre d'interfédérations. Il existe d'autres possibilités : HLA définit les rôles du RTI et des fédérés, et on peut envisager d'établir un lien entre deux éléments de fédération quelconques (RTI ou fédéré). Ainsi la connexion entre fédérations peut être réalisée :

- Par un lien entre deux RTI : ceci se rapproche de l'idée d'un RTI distribué, et pose comme principal problème celui de la communication entre deux RTI. Un tel protocole n'est pas prévu par HLA et n'est donc envisageable que si les fournisseurs de RTI définissent un tel protocole. Certains intérêts comme l'interopérabilité disparaissent alors.
- Par un lien entre deux fédérés : à nouveau, ceci nécessite un protocole à définir pour l'échange de données entre ces fédérés (dans HLA un fédéré ne communique jamais directement avec un autre fédéré).
- Par un lien entre un RTI et un fédéré : c'est le cas du fédéré-pont (en plus d'être connecté normalement à un RTI, le fédéré est lié à un autre RTI). S'il semble moins performant que le lien entre deux RTI, il a l'avantage d'utiliser directement HLA et ne nécessite pas l'ajout d'un protocole d'échange de données. Mais le transfert d'informations d'une fédération à une autre pose des difficultés liées au fait que le pont est seulement un fédéré et n'a pas les connaissances globales que possède un RTI. Le chapitre suivant est notamment consacré à ce problème.

Notre objectif d'interopérabilité étant impératif, nous nous sommes donc principalement intéressés au cas du fédéré-pont, et à sa réalisation exclusivement à l'aide de services standard de HLA.

5.4 Éléments de traduction

Le transfert d'informations d'une fédération à une autre correspond à une traduction : à chaque événement de l'une doit être associé un événement représentant dans l'autre. Le détail de cette traduction dépend du type d'interfédération choisi. Par contre le problème posé est similaire en ce qui

concerne le document FOM qui décrit les fédérations. Si l'on souhaite faire de l'interfédération une simulation se rapprochant le plus possible d'une fédération classique, on utilisera pour chaque fédération le même FOM. Ainsi, un fédéré pourra souscrire à une classe, quelle que soit la fédération dans laquelle celle-ci est définie. Un lien entre fédérations a notamment pour tâche de faire la correspondance entre les deux fédérations, si une même classe ou un même objet n'a pas le même identifiant d'une fédération à l'autre.

Dans le cas d'une interfédération motivée par des questions de sécurité et par le besoin de classes privées, présentes dans une seule fédération, on pourra avoir des FOM différents. L'étape de traduction est alors encore plus essentielle car on a la certitude que les descriptions (et donc les identifiants créés par les RTI) diffèrent.

5.5 Topologies

Un pont a pour objectif de simuler une fédération à au moins une autre fédération, le plus souvent de manière bidirectionnelle. C'est ce qui permet de relier effectivement deux fédérations, c'est-à-dire de partager des événements communs, et plusieurs types de ponts et de structures d'interfédérations sont possibles.

Nous mettons en avant l'utilisation de fédérés-ponts mais des interfédérations utilisant des liens de RTI à RTI peuvent avoir des topologies similaires.

5.5.1 Interfédérations acycliques

Le cas le plus simple d'interfédération est constitué de deux fédérations et d'un pont. Le pont a un rôle double : il observe une fédération et crée des représentants dans l'autre. Et ceci dans deux directions : le pont observe chaque fédération et produit également un comportement dans chacune d'entre elles.

Nous ne considérons dans un premier temps que la simulation complète d'une fédération à une autre. Pour étendre le principe de l'interfédération précédente à un nombre quelconque de fédérations, deux approches sont possibles.

Interfédérations en étoile

Dans l'exemple précédent, un pont observe une fédération, et produit un comportement dans une autre fédération. Le but est en fait d'observer ce qui se passe dans une fédération, et de faire en sorte que tous les événements aient lieu dans le reste de l'interfédération. Autrement dit, si nous avons

une interfédération de n fédérations, il s'agit d'observer l'ensemble des comportements d'une fédération, et de les reproduire dans les $n-1$ fédérations restantes. On peut alors envisager de connecter plus de deux fédérations à un pont : celui-ci aura pour tâche d'observer chaque fédération, et d'y reproduire les comportements observés dans toutes les autres (Fig. 5.2). Chaque fédération verra donc apparaître des objets simulés de différentes fédérations.

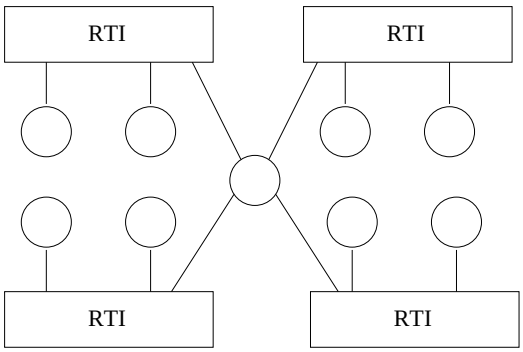

FIG. 5.2 – Interfédération en étoile

L'avantage de cette structure est que l'on aura toujours le même nombre de transports d'information pour les événements reproduits, c'est-à-dire simplement le passage par *un seul* pont. L'inconvénient est qu'avec un grand nombre de fédérations, le pont va être particulièrement chargé.

Interfédérations linéaires

Une autre approche consiste, toujours en partant de l'exemple simple de deux fédérations, à considérer l'interfédération ainsi réalisée comme une fédération classique. En effet, l'objectif des interconnexions de fédérations est de créer des simulations qui soient similaires aux fédérations équivalentes : les fédérés ne « voient » pas les autres fédérés mais seulement leurs objets. Ceci est aussi vrai pour les fédérés appartenant à des fédérations différentes : ils ne seront vus que par leurs objets, leurs actions, quelle que soit leur fédération d'origine. Le RTI également n'a pas connaissance de l'interfédération, puisque le pont apparaît comme un fédéré.

Puisqu'une interfédération peut être considérée comme une fédération par le RTI et par tous les fédérés y participant (en dehors du pont), nous pouvons créer une nouvelle interfédération à l'aide d'un pont simple, reliant deux fédérations, mais en connectant cette fois une fédération classique à une interfédération. Et comme le résultat peut de nouveau être vu comme une fédération, on peut y ajouter un nouveau pont et une nouvelle fédération (ou

interfédération), et ainsi de suite pour un nombre quelconque de fédérations. (Fig. 5.3)

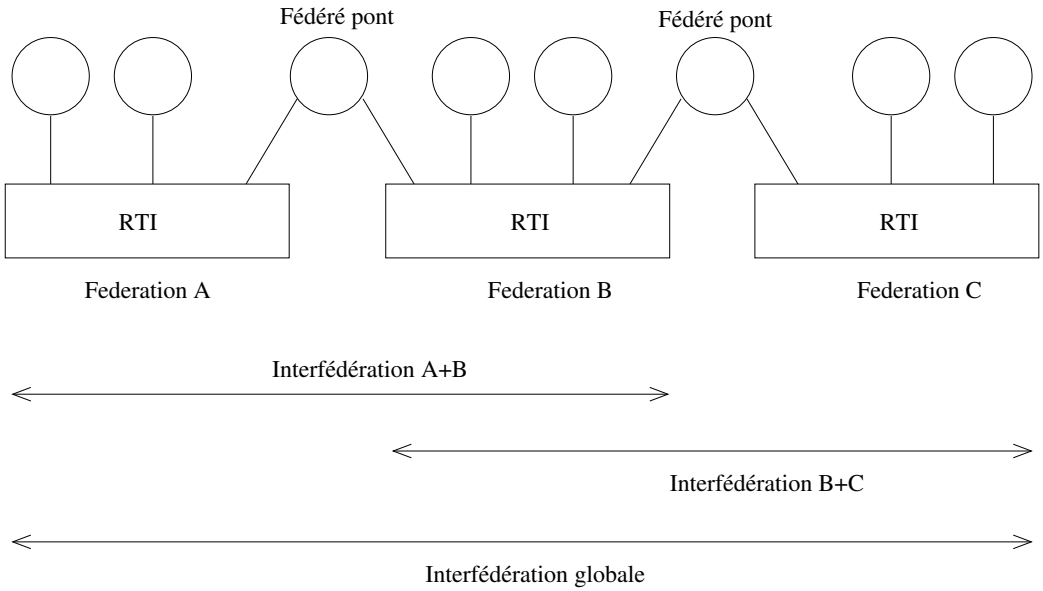

FIG. 5.3 – Interfédération linéaire

L'avantage ici est que le pont n'a pas besoin d'être capable de gérer un nombre quelconque de fédérations : deux suffisent. Un autre avantage est de distribuer ce rôle de représentation en plusieurs ponts, et non en un seul comme dans le cas précédent. Mais l'inconvénient cette fois est que la distance entre événements réels et événements simulés est plus grande : il peut être nécessaire de passer par plusieurs fédérations et ponts. Et bien que ce type d'interfédération ne soit pas forcément formé d'une seule branche, la distance entre deux fédérations peut être grande.

Interfédérations acycliques générales

Comme nous l'avons vu dans le cas précédent, une interfédération peut être vue comme une fédération habituelle, aussi bien d'un point de vue fédéré que d'un point de vue RTI. Interfédérations en étoile et interfédérations linéaires peuvent donc être mélangées sans difficulté, pourvu que les ponts permettant de les implémenter existent. Certaines fédérations d'une interfédération en étoile peuvent être connectées à d'autres fédérations de manière linéaire, et on pourrait aussi remplacer certains ponts simples d'interfédérations linéaires par des ponts capables de relier un nombre quelconque de fédérations. (Fig. 5.4)

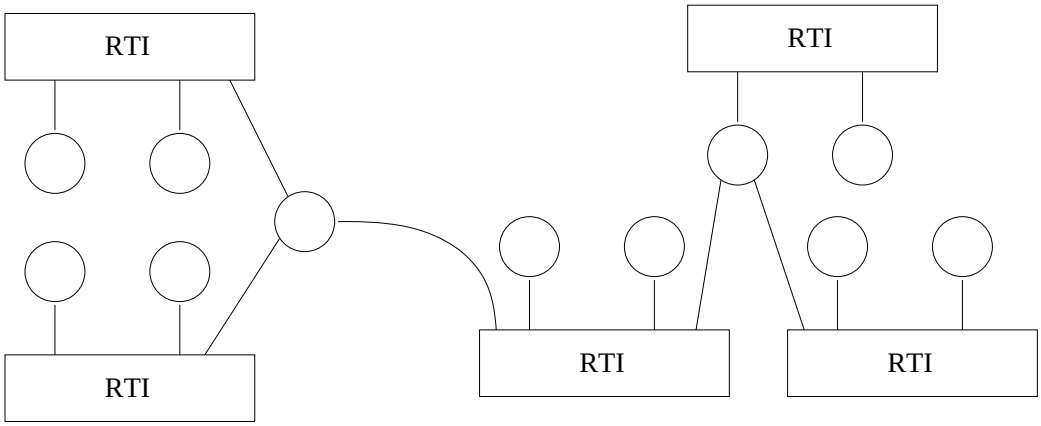

FIG. 5.4 – Interfédération acyclique générale

5.5.2 Interfédérations cycliques

D'autres besoins indiquent une solution intéressante à envisager. Dans une interfédération acyclique, chaque pont voit passer toutes les informations à transmettre d'une partie à l'autre de l'interfédération. Ceci représente une quantité importante d'informations. On peut penser au problème rencontré par CERTI quand le nombre de fédérés augmente : le RTIG, le point central, sous une forte charge, devient le facteur limitant. L'interfédération, nécessitant un travail de traduction et de simulation, se trouve dans la même situation ; si toutes les informations doivent être transmises, le pont risque d'être surchargé. Mais il est possible d'envisager plusieurs chemins pour la transmission d'information, se répartissant les différents événements à reproduire. Chaque chemin, donc chaque pont, voit alors sa charge diminuer. Comme bien sûr il s'agit de ne pas reproduire plusieurs fois un même événement, les informations doivent être correctement réparties entre les chemins. Ceci pourrait apparaître déjà dans le cas simple de deux ponts à sens unique, remplaçant un pont à deux fédérations. Plus généralement, il faut déterminer pour chaque élément à reproduire les ponts à utiliser.

Jusqu'ici nous avons considéré des ponts transmettant tous les événements observés d'une fédération à l'autre. Les interfédérations envisagées étant acycliques (d'une fédération à une autre, il n'existe qu'un seul chemin), ce type de pont pouvait être utilisé à chaque connexion entre fédérations.

Mais dès lors que l'on souhaite réaliser une structure d'interfédération cyclique (avec plusieurs chemins possibles entre deux fédérations, comme dans l'exemple de la figure 5.5), l'utilisation exclusive de tels ponts ne convient pas. En effet, dans le cas des interfédérations acycliques, chaque pont sépare

l'interfédération en un certain nombre de parties, et tout ce qui est présent d'un côté est reproduit dans les autres. Avec le même comportement des ponts, dans une interfédération cyclique, les objets sont reproduits de proche et proche, et se trouvent réintroduits sous forme d'objets représentants dans leur fédération d'origine. Il est donc nécessaire de couper cette boucle, sans quoi plusieurs objets représentants seront créés pour un même objet original (on obtiendrait en fait une infinité de représentants [DMA+98]). Pour cela, on peut tout d'abord envisager de permettre aux ponts de détecter ces effets. Mais ceci nécessite une communication entre les différents ponts, car par la fédération, le pont ne peut obtenir aucune information : tous les événements sont vus de la même manière, qu'ils soient originaux ou reproduits par un pont. Cette approche ne fait que retourner à la situation acyclique, puisqu'avec une telle information, un pont détectant une cyclicité cesserait son activité, et bien qu'il soit physiquement connecté, les échanges d'informations seraient ceux d'une interfédération acyclique. Une telle rupture est bien nécessaire mais n'a pas besoin d'être localisée dans un seul pont pour tous les événements. Par exemple si deux fédérations sont reliées par deux ponts, chaque pont peut prendre en charge la moitié des événements à reproduire.

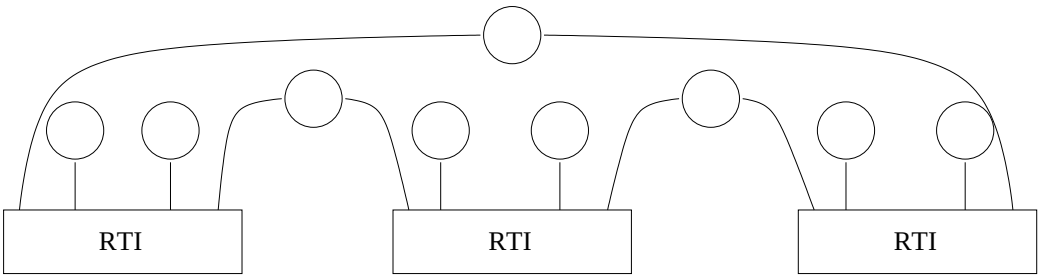

FIG. 5.5 – Interfédération cyclique

Conclusion

On peut remarquer que, pourvu que chaque objet existant soit représenté dans chaque fédération, on peut utiliser toutes ces formes de pont dans une même interfédération. La structure dépend alors directement des besoins qui justifient la mise en œuvre d'une interfédération. La seule contrainte liée à la topologie adoptée est que chaque événement soit représenté de manière cohérente, c'est-à-dire qu'il ne soit ni oublié ni représenté en plusieurs exemplaires dans une même fédération.

Chapitre 6

Ponts de fédérations

6.1 Présentation

Nous nous intéressons à un cas particulier d'interfédération : les ponts de fédérations réalisés à l'aide de fédérés, ou fédérés-ponts. Le choix de ce type de pont vient essentiellement du fait qu'il peut être réalisé à l'aide de HLA et sans introduire de nouveaux protocoles qui feraient perdre l'avantage de l'interopérabilité.

Le pont de fédérations est souvent présenté comme un unique fédéré. Il s'agit en fait d'un logiciel se présentant comme fédéré à plusieurs RTI (Fig. 6.1). Du point de vue de HLA, il faudrait le considérer comme un ensemble de fédérés communicants, chaque fédéré appartenant à l'une des fédérations reliées.

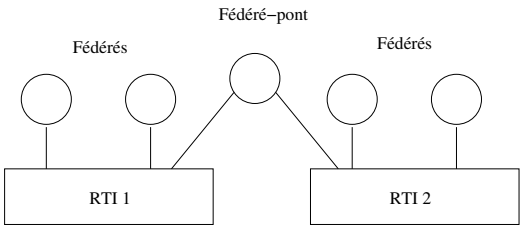

FIG. 6.1 – Interfédération réalisée avec un pont

Dans le cas général, une interfédération ayant pour but de réaliser une simulation équivalente à la fédération simple constituée des mêmes fédérés, tout événement ayant lieu dans une fédération de l'interfédération doit être propagé dans les autres fédérations. C'est par exemple le cas de la création d'objet, de la mise à jour d'attributs, de l'envoi d'interactions, etc. Ces événements, locaux aux fédérations, ne peuvent pas être transmis directement

entre deux fédérations à l'aide de mécanismes HLA, et le pont doit donc les reproduire dans les fédérations cibles. Les événements à transmettre peuvent être ponctuels, comme des envois d'interactions, mais peuvent aussi être associés à des créations d'objets. Alors, chaque objet créé dans une fédération et observé par le pont doit être reproduit, par la création d'un *représentant* dans les autres fédérations. Par la suite, le pont doit donner à ce représentant le même comportement que l'objet réel (mises à jour d'attributs, destruction, etc.)

Accès aux informations

Le principal problème rencontré dans la conception et la réalisation de ponts de fédérations est directement lié aux spécifications HLA : en effet, un pont est avant tout, dans chaque fédération reliée, un fédéré. La simulation dans son ensemble est constituée du RTI, dans lequel toutes les informations relatives à la fédération sont accessibles, et de fédérés, n'ayant connaissance que d'informations locales ou autorisées. En pratique, un fédéré peut observer ce qui se produit dans la fédération, mais il ne peut pas avoir connaissance des origines de ces événements (par exemple, par quel fédéré un événement a été produit.) Cette encapsulation joue un rôle essentiel pour la réutilisabilité des fédérés, qui réagissent en fonction des événements de l'environnement simulé et non de l'implémentation de ceux-ci ou de la structure de la fédération.

Le pont de fédérations n'a donc de la fédération que la vision que peut avoir un fédéré, c'est-à-dire une vue très restreinte par rapport à celle du RTI. Or la tâche que l'on souhaite confier au pont est bien celle d'un composant ayant une vue globale de la simulation. La plupart des difficultés de conception du comportement du pont viennent de cet aspect.

Lorsque cette situation pose une difficulté pour la transposition des événements d'une fédération à l'autre, une solution très générale consiste à employer le MOM, objet défini par HLA, fourni par le RTI, et qui permet aux fédérés d'avoir accès à des informations relatives à l'implémentation de la fédération et aux services appelés dans l'ensemble de la fédération, en particulier par d'autres fédérés. Cependant ce MOM n'est pas implémenté dans CERTI et ne peut donc être utilisé. Si sa réalisation peut être envisagée, cet objet n'a pas été prioritaire dans CERTI et ne l'a pas été dans les RTI non commerciaux qui n'ont pas implémenté toute la norme HLA. C'est pourquoi nous avons souhaité privilégier des approches se passant du MOM :

- soit en transmettant les événements sans celui-ci si cette opération est possible ;

– soit en envisageant une transmission des événements limitée à des cas particuliers mais suffisamment acceptables.

Nous allons donc présenter les principaux mécanismes de HLA et les solutions envisagées pour les transposer d'une fédération à une autre, en privilégiant les approches alternatives à l'utilisation du MOM.

6.2 Mécanismes généraux de HLA

Les mécanismes basés sur les services proposés par HLA sont de complexités diverses. Nous allons tout d'abord présenter deux types de mécanismes utilisés par plusieurs groupes de services. Puis nous présenterons les mécanismes de transfert de propriété et de gestion du temps.

6.2.1 Mécanisme direct, inconditionnel

Certains mécanismes de HLA sont relativement simples : par exemple la mise à jour d'attributs est un mécanisme direct et inconditionnel. Cet exemple utilise deux services : dans un premier temps, un fédéré indique une mise à jour d'un attribut de l'un de ses objets, et utilise le service *Update Attribute Values* du RTI. Ensuite, le RTI informe les fédérés concernés par cette mise à jour, à l'aide du service *Reflect Attribute Values* †. Ce second appel est finalement le seul et le dernier qui soit visible par d'autres fédérés lors de ce mécanisme en deux temps. On a vu qu'un pont de fédération crée des représentants des objets ou des événements qu'il observe dans une fédération. Dans l'exemple précédent, le pont reçoit le *Reflect Attribute Values* † envoyé par le RTI, et utilise cette information pour mettre à jour le représentant de l'attribut concerné. La nouvelle valeur est a priori la même, de même que l'estampille temporelle si le message en possédait une (nous détaillerons le cas de la gestion du temps dans une section suivante).

Le transfert par le pont d'un événement inconditionnel et transmis à tous les fédérés constitue donc un des cas les plus simples :

1. Le fédéré émetteur envoie l'information au RTI ;
2. Le RTI informe éventuellement les fédérés concernés.

Ce type d'échanges est représenté sur la figure 6.2.

Transposée à une interfédération, la situation n'a d'intérêt que si le RTI retransmet effectivement l'information à d'autre fédérés (sinon l'information s'arrête naturellement au premier RTI). S'il est bien abonné aux informations à transmettre, le pont les reçoit et peut alors les transmettre :

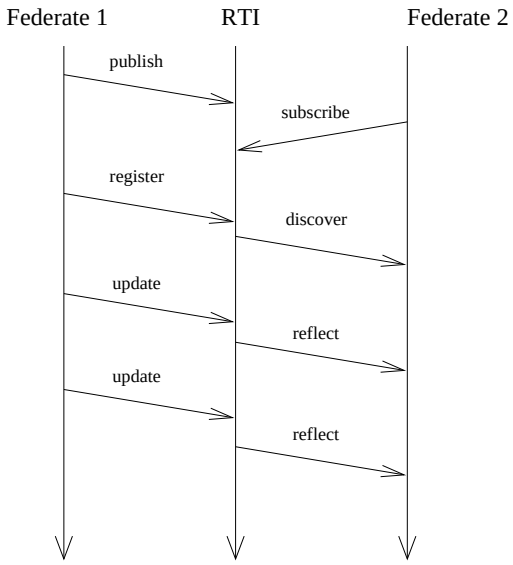

FIG. 6.2 – Échange de messages inconditionnels

1. Le fédéré émetteur envoie l'information au RTI ;
2. Le RTI informe les fédérés concernés, dont le pont ;
3. Le pont transmet l'information à l'autre RTI ;
4. Ce second RTI informe les fédérés de sa fédération.

Ce qui est représenté sur la figure 6.3.

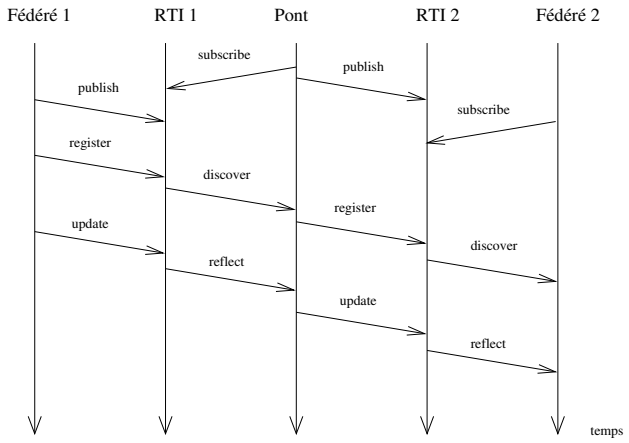

FIG. 6.3 – Messages inconditionnels au travers d'un pont

6.2.2 Mécanisme à consensus

Un mécanisme plus complexe est celui de la sauvegarde de fédération [BL97, DGD02]. C'est un mécanisme global qui fait intervenir plusieurs fédérés lors de son déroulement. Une requête de sauvegarde est envoyée au RTI par un fédéré. Si celle-ci est valide, alors le RTI informe tous les fédérés du début d'une procédure de sauvegarde, leur indiquant qu'ils doivent tous procéder à la sauvegarde de leur état. Chaque fédéré indique, en deux étapes, l'avancement de cette sauvegarde avec un message pour en signaler le début, et un second pour en signaler la fin, et le statut — succès ou échec. Après avoir obtenu toutes les réponses nécessaires, le RTI peut terminer par un message global indiquant le résultat de la sauvegarde. Les services de gestion de la fédération possèdent deux autres mécanismes assez similaires : la restauration d'une sauvegarde, et la synchronisation. Ils diffèrent dans le détail mais le principe est toujours celui du consensus : le dernier service est l'annonce par le RTI d'un résultat qui dépend des réponses d'un ensemble de fédérés. La procédure mise en œuvre peut se résumer ainsi :

1. Le fédéré émetteur envoie la requête au RTI ;
2. Le RTI informe les fédérés concernés du mécanisme en cours ;
3. Les fédérés répondent au RTI, indiquant un résultat local ;
4. Le RTI calcule et envoie le résultat global à l'ensemble des participants.

Avec une interfédération, le pont reçoit la demande de participation comme les autres fédérés. Il doit assurer l'établissement d'un mécanisme identique dans l'autre fédération, en lançant lui-même la requête.

C'est ici que se situe le problème. La notion d'interfédération n'existant pas au niveau du RTI, la même procédure est lancée de part et d'autre, et les RTI y travaillent parallèlement mais sans coopérer. Puis le pont n'a ni accès aux résultats ni la possibilité de donner le résultat final.

Le rôle d'un pont est de créer dans une fédération la représentation d'une autre fédération, ce rôle devant être symétrique. Si une réponse unanime des fédérés est nécessaire à l'achèvement d'une procédure, le pont doit fournir une réponse correspondant à la fédération représentée. Mais il ne peut pas obtenir de réponse de la part d'un RTI si l'autre RTI n'a pas donné sa réponse, et ceci est valable pour les deux fédérations. Ainsi le résultat d'une fédération A dépend de celui d'une fédération B, mais ce dernier dépend aussi de celui de la fédération A.

Bien sûr, le pont peut envoyer une réponse par anticipation pour que la situation ne soit pas bloquée, et obtenir une réponse de part et d'autre. Il obtient alors les réponses des deux fédérations, et en déduit le résultat pour l'interfédération. Mais il n'est pas en position d'annoncer ce résultat : chaque

RTI aura déterminé un résultat, et il est donc possible que ces résultats ne concordent pas. Plusieurs solutions sont envisageables.

Utilisation du MOM

Le MOM possède des attributs reflétant des caractéristiques de la fédération, et peut lancer des interactions associées aux services du RTI. Ainsi, un fédéré s'abonnant à ces attributs ou à ces interactions recevra des messages correpondant aux services appelés (qui peuvent être à l'initiative de fédérés ou du RTI). L'utilisation du MOM place le pont dans une situation plus proche de celle du RTI, en ayant à disposition toutes les informations nécessaires sur la fédération. Le pont peut alors gérer la situation de mécanisme global en observant les messages individuels des fédérés au RTI. Cependant, nous recherchons des solutions permettant d'éviter l'utilisation de ce composant.

Restriction du résultat à l'initiateur

Si l'on n'utilise pas le MOM, on peut souvent, en ajoutant des conditions sur l'interfédération, proposer un comportement du pont permettant d'éviter le blocage. Par exemple, on peut imposer que le résultat d'une sauvegarde de fédération ne soit utilisé que par le fédéré qui a initialement effectué la demande de sauvegarde. Dans ce cas, seul le résultat donné par le RTI de cette fédération est important, et le pont peut se déclarer sauvegardé dans les autres fédérations même si ce n'est pas le cas.

Prenons l'exemple de la sauvegarde de fédération. La demande de sauvegarde émane d'un fédéré qu'on nommera l'*initiateur*. Le RTI transmet la demande à tous les fédérés dont le pont. Le pont doit transmettre la demande à l'autre RTI et pour que ce RTI puisse valider la sauvegarde de sa fédération, le pont se déclare sauvegardé (par anticipation.) Le pont et tous les fédérés de la seconde fédération reçoivent le résultat indiquant la réussite ou non de la sauvegarde de cette fédération. Ce résultat est alors donné comme réponse au premier RTI. Un problème se pose si la seconde fédération a réussi la sauvegarde tandis que la première échoue. Tous les fédérés de la première fédération sont informés correctement par leur RTI de l'échec de la fédération mais tous ceux de la seconde fédération ont reçu un message indiquant la réussite de la sauvegarde, qui est vraie au niveau fédération, mais pas au niveau interfédération.

Le fédéré initiateur est dans la première fédération, il reçoit le bon résultat. Les fédérés qui ne sont pas initiateurs n'ont aucun moyen de déterminer s'ils sont dans la même fédération que le fédéré initiateur. Seul le fédéré initiateur est certain d'avoir reçu le bon résultat concernant la sauvegarde. Une

solution est donc de restreindre l'utilisation de ce résultat à cet unique fédéré.

Si cette restriction peut être acceptée dans le cas de la sauvegarde et de la restauration de fédération, où un unique fédéré peut être responsable de ces opérations, elle n'est pas envisageable dans le cas des points de synchronisation où tous les fédérés attendent une réponse fiable.

Ajout de services

Il faut également noter que les informations manquantes sont souvent essentielles mais peu nombreuses. Une solution consiste donc à intervenir sur la norme HLA en proposant certains services supplémentaires [DGD02, Lak98] destinés aux ponts de fédérations. Cette solution pose un problème d'interopérabilité encore plus grand que le cas du MOM, puisque de telles propositions ne sont pas actuellement dans HLA. Cependant en cas d'ajout dans la norme et d'adoption par les RTI, ces solutions devraient être les plus adaptées à la résolution de ces problèmes.

6.2.3 Transfert de propriété

Le groupe de services *Ownership Management* concerne la gestion de la propriété : il permet à des fédérés de donner la propriété de certains des attributs de leurs objets à d'autres fédérés.

Principe général

Dans une fédération F simple, un cas de transfert de propriété est par exemple le suivant : l'attribut a d'un objet O^A appartenant à un fédéré A devient la propriété d'un autre fédéré, B (tout en restant attribut de l'objet O^A).

Transposée à une interfédération, la situation est par exemple la suivante (Fig. 6.4) : deux fédérés, A_1 et B_2 (l'indice est simplement utilisé pour rappeler le numéro de la fédération) appartiennent respectivement aux fédérations F_1 et F_2. A_1 gère un objet O_1^A, dont il possède l'attribut a_1. Le but est de transférer la propriété de a_1 de A_1 à B_2.

Afin de simuler F_1 à F_2, le pont crée un objet représentant O_1^A dans F_2 : O_2^A, et cet objet possède un attribut représentant a_1 : a_2.

Le fédéré A_1 étant inconnu dans F_2, transférer la propriété de l'attribut consiste en deux transferts : dans F_1, le pont prend la propriété de a_1 ; et dans F_2, le pont donne la propriété de a_2 (l'attribut représentant) à B_2 (Fig. 6.5).

Les rôles d'attribut original et d'attribut représentant sont alors inversés : c'est a_2 qui sera modifié par un fédéré participant à la fédération F_2, et le

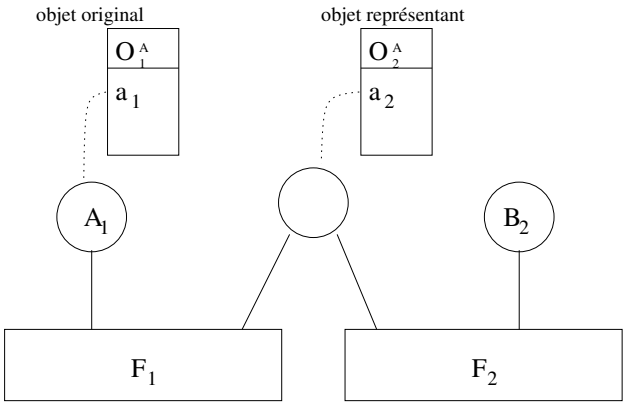

FIG. 6.4 – Propriété dans une interfédération

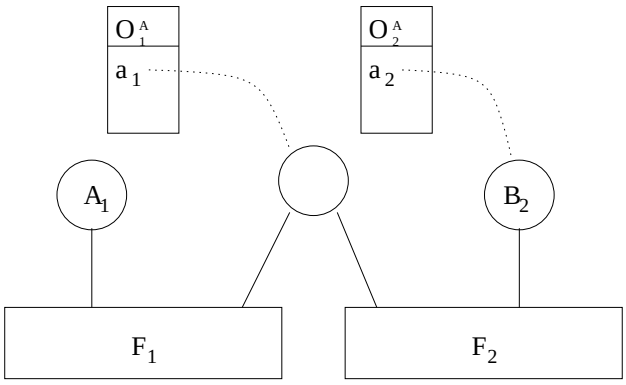

FIG. 6.5 – Transfert de propriété au travers d'un pont

pont modifiera en conséquence a_1.

Cas envisagés

Ownership Management possède de nombreux services. Deux mécanismes généraux (qui se décomposent en plusieurs variantes) sont les suivants :

- un fédéré souhaite obtenir la propriété d'un attribut qu'il ne possède pas : le RTI tente de faire le transfert si le fédéré possédant l'attribut accepte de s'en désister ;
- un fédéré ne souhaite plus posséder un attribut donné : le RTI tente alors de donner l'attribut à un autre fédéré (l'attribut peut temporairement n'appartenir à aucun fédéré).

Le premier cas ne présente pas de difficulté : demander la propriété d'un attribut d'une autre fédération revient à demander la propriété d'un attribut représentant, c'est-à-dire possédé par le pont. Le pont peut alors tenter d'obtenir la propriété de l'attribut représenté, et s'il l'obtient, autorise le transfert.

Le second cas est plus complexe car le RTI effectue une démarche pour trouver un fédéré acceptant de prendre la propriété de l'attribut. Pour reproduire dans l'interfédération ce qui se passerait dans une fédération, il faudrait que le pont propose de donner la propriété de l'attribut représentant. En effet, c'est le seul moyen de proposer le transfert de propriété à tous les fédérés de la simulation, comme c'est le cas dans une fédération simple. Mais ceci n'est pas possible sans remettre en cause l'intégrité de l'interfédération. En effet si le pont propose de donner la propriété d'un attribut représentant, les deux transferts de propriété potentiels (celui de l'attribut original, et celui du représentant) sont gérés par deux RTI différents, et sont donc susceptibles d'avoir lieu en même temps. On se trouverait dans ce cas dans une situation où aucun des deux attributs ne serait contrôlé par le pont ; chaque attribut évoluant dans sa fédération, la simulation dans l'une et dans l'autre ne serait pas la même.

Pour conserver l'intégrité de l'interfédération, le pont doit donc assurer qu'il possède toujours au moins un des deux attributs d'un couple d'attributs (original/représentant). Lorsque le pont reçoit un message du RTI indiquant qu'un fédéré souhaite donner la propriété d'un attribut, deux alternatives sont possibles :

- ignorer le message, c'est-à-dire considérer qu'un candidat sera trouvé dans la même fédération ;
- accepter de prendre cette propriété, et si celle-ci est effectivement obtenue, proposer de donner la propriété du représentant dans l'autre fédération.

Les différents scénarios possibles n'ont pas été envisagés plus précisément. Mais on peut imaginer un comportement du pont visant à réunir ces deux solutions : ignorer la proposition dans un premier temps, puis prendre la propriété si celle-ci est toujours en attente, et ainsi de suite, alterner entre les deux fédérations. Ceci permettrait de proposer la propriété à tous les fédérés, même si cette solution n'approche qu'imparfaitement la situation d'une fédération unique.

6.2.4 Gestion du temps

À la propagation des messages de *Object Management*, il faut ajouter la gestion du temps : si le principe de l'échange d'information via les mécanismes tels que *Update Attribute Values/ Reflect Attribute Values* † se transmet au travers du pont, les messages permettant de réaliser ces mécanismes doivent aussi pouvoir transporter une estampille temporelle indiquant l'instant de l'événement.

Plus précisément, un type d'échange nécessite ces estampilles. Il s'agit du transport de messages TSO d'un fédéré régulateur à un fédéré contraint.

Le fédéré-pont a pour but de représenter au mieux une fédération ou un ensemble de fédérations. On considère une même échelle de temps utilisée pour tous les fédérés des fédérations (tout comme on le fait pour une fédération classique). Il s'agit donc de transmettre les messages estampillés en utilisant cette échelle de temps.

Statut contraint et régulateur

Pour cela, il faut dans un premier temps remarquer que lorsqu'un message TSO est envoyé par un fédéré non régulateur, ou reçu par un fédéré non contraint, le message va perdre son statut TSO : il apparaîtra comme un message RO à la réception. Le premier cas n'a pas de sens, un fédéré non régulateur n'envoie normalement pas de messages TSO, et s'il le fait, ceci revient à considérer ce message comme RO. Un fédéré non contraint peut recevoir des messages étant TSO à l'origine, mais son statut non contraint indique que le message doit être vu comme message RO.

Les messages RO, c'est-à-dire sans estampille significative, peuvent être transmis par un pont en suivant simplement les mécanismes évoqués précédemment. En effet les messages ne sont employés que dans des mécanismes inconditionnels. Dans une simulation de ce type, par exemple sans temps coordonné, la simple transmission de messages RO suffit : le mécanisme est reproduit, et le temps n'est pas pris en compte.

Les messages TSO par contre doivent être transmis au travers du pont. En particulier le pont doit être capable d'une part de recevoir les messages TSO et d'autre part de produire de tels messages. Lorsqu'un message TSO est correctement produit (c'est-à-dire qu'il est déclaré TSO, qu'il est envoyé par un fédéré régulateur, et qu'il possède une estampille temporelle), il sera reçu en tant que TSO par tout fédéré concerné pourvu que ce fédéré soit contraint. Le pont doit donc fonctionner en temps contraint. De la même manière, pour qu'un message TSO arrive bien en tant que TSO, il faut non seulement que le fédéré de destination soit contraint, mais aussi que l'expéditeur soit régulateur : pour les messages correspondant à des représentations d'événements, cela implique que le pont soit régulateur. Le pont étant constitué de plusieurs composants fédérés, ce sont plus précisément les observateurs qui doivent être contraints, et les simulateurs qui doivent être régulateurs. La représentation étant souvent à effectuer dans les deux sens, chaque fédéré-pont sera à la fois contraint et régulateur dans le cas général. Les cas où ceci n'est pas nécessaire sont par exemple des ponts à sens unique, ou des inter-fédérations dans lesquelles une fédération ne possède aucun fédéré contraint ou aucun fédéré régulateur.

Avancement du temps

On note F_R l'ensemble des fédérés régulateurs d'une fédération, F_C l'ensemble de ses fédérés contraints, F l'ensemble de tous les fédérés et f_i un fédéré de F. Alors, les notations suivantes sont utilisées :

- $Time(f_i)$ désigne le temps logique du fédéré f_i ;
- $Lookahead(f_i)$ désigne le *lookahead* du fédéré f_i ;
- $LBTS(f_i)$ désigne le LBTS[1] du fédéré f_i.

On a alors dans une fédération :

$$\forall f_i \in F_C, LBTS(f_i) = \min_{f_j \in F_R, j \neq i} (Time(f_j) + Lookahead(f_j))$$

Enfin, on note RTI^F le RTI supportant les fédérés F. Le processus d'avancement du temps d'un fédéré dans une fédération est le suivant :

1. Le fédéré f_i de temps logique $Time(f_i)$ demande à RTI^F un avancement au temps t ($t > Time(f_i)$). Tout message M engendré dans cette phase doit avoir une estampille $TS(M)$ telle que $TS(M) \geq t + Lookahead(f_i)$;
2. RTI^F envoie à f_i les messages M tels $TS(M) < t$, s'il y en a ;

[1] *Lower Bound on Time Stamps*, cf. § 2.3.6 p. 51

3. RTI^F envoie un *Time Advance Grant* † à f_i, indiquant qu'aucun message d'estampille temporelle inférieure à t ne sera envoyé. Cet avancement dans le temps peut être déclenché dès que $LBTS(f_i) > t$.

Dans le cas des interfédérations, plusieurs approches sont possibles. De manière générale, le LBTS d'un pont dans une fédération représente la plus petite valeur possible d'estampille des futurs messages TSO susceptibles de devoir être reproduits dans une autre fédération. Cette valeur peut être utilisée par le pont pour demander, dans une autre fédération, un avancement dans le temps. Dans le cas d'un pont joignant deux fédérations cette valeur peut être utilisée directement. Dans le cas d'un pont rejoignant plus de trois fédérations, l'avancement dans le temps dans une fédération cible dépend de la plus petite valeur d'estampille possible à transmettre, en provenance d'une fédération source quelconque. Le pont est donc en mesure d'avancer dans une fédération F_i jusqu'à un temps logique t tel que $t = \min_{j \neq i} LBTS(p_j)$, où $LBTS(p_j)$ est le LBTS du pont dans une fédération F_j. Ce temps correspond à une valeur possible du temps logique du pont dans une fédération F_i, c'est-à-dire que le pont peut atteindre cette valeur de temps sans se trouver dans une situation engendrant des incohérences : tous les messages reçus par la suite auront une estampille supérieure à ce temps, et pourront dont être transmis par le pont avec cette même estampille. Cependant cette valeur n'est qu'une valeur de temps idéale pour le pont dans une fédération, ce qui ne constitue pas une méthode d'avancement du pont : il faut d'une part déterminer la valeur à demander effectivement lors d'une demande d'avancement dans le temps, et d'autre part vérifier que cet avancement est possible et permet la réception de messages (le schéma d'avancement ci-dessus rappelle que les messages TSO d'estampille t ne sont envoyés à un fédéré par le RTI que si ce fédéré a demandé un avancement dans le temps au moins jusqu'à t).

Tout d'abord, la manière la plus juste de représenter le temps consiste à utiliser pour chaque message reproduit la même valeur d'estampille que celle du message original. La solution proposée dans [Lak98] consiste à utiliser des ponts de fédérations, qui en tant que fédérés, ont toujours un lookahead nul. Le LBTS d'un pont dans une fédération permet de déterminer qu'aucun événement n'aura lieu dans cette fédération au-delà d'un instant logique, et permet donc de demander l'avancement dans le temps, jusqu'à cette date, dans l'autre fédération. Cependant l'avancement effectif n'est pas assuré si de part et d'autre les fédérés fonctionnent en lookahead nul. La solution proposée consiste donc à déterminer un LBTS global à l'interfédération. Celui-ci est calculé à l'aide de l'algorithme *Bounded Lag* de Lubachevsky [Lub89], mais nécessite des extensions du RTI. Cette contrainte limite l'utilité d'un tel pont

à des objectifs d'optimisation, ces extensions n'étant pas applicables à un RTI aux sources non disponibles (à moins d'être intégrées à la norme HLA dans une future révision).

Une deuxième approche, celle que nous considérons principalement pour le prototype de pont, ajoute des hypothèses à la situation précédente. Nous nous plaçons dans le cas où aucun fédéré (y compris le pont) ne fonctionne en lookahead nul. Ceci correspond aux fédérations mises en œuvre avec CERTI. Le problème posé concerne le temps d'avancement demandé par le pont : ce temps doit être tel qu'en y ajoutant le lookahead minimal imposé par cette situation, le temps obtenu soit la borne inférieure des estampilles des futurs messages. Autrement dit, si une telle borne t_{min} est déterminée, et qu'il existe un lookahead minimal L_{min}, le pont ne peut pas demander un avancement au-delà de $t = t_{min} - L_{min}$ (contrainte de l'étape 1). Or, si le temps logique reste à cette valeur de t, le fédéré ne peut pas recevoir les messages d'estampille t_{min}, car ceux-ci ne peuvent être délivrés que lorsque le pont a demandé un avancement de son temps logique à cette valeur. Pour ne pas bloquer l'avancement global, il est nécessaire de faire avancer le pont dans le temps, ce qui nécessite des hypothèses supplémentaires. Le besoin est d'atteindre cet instant t_{min} tout en gardant un lookahead L_{min}. Si le pont demande un avancement jusqu'à t_{min}, il ne sera pas en mesure de transmettre correctement les messages TSO dont l'estampille a une valeur supérieure à t_{min} et inférieure à $t_{min} + L_{min}$. Dans le cas d'une simulation pas à pas, dans laquelle les fédérés avancent d'un même intervalle de temps, on a l'assurance que de tels messages ne se présenteront pas si L_{min} est inférieur à un pas de temps. Cette approche convient donc à cette catégorie de simulation. On peut en fait généraliser ceci dès qu'il existe un intervalle de temps indivisible : tous les intervalles d'avancement employés par les fédérés sont alors des multiples de celui-ci, et avec un L_{min} de cette valeur, l'avancement est donc possible. Il faut bien rappeler cependant que nous sommes toujours dans le cas où aucun fédéré n'a de lookahead nul.

Ces deux approches permettent de créer des messages représentants transportant une estampille de valeur identique à celle du message d'origine. On peut également envisager de transmettre un événement avec une estampille différente. Cette approximation entraîne une incohérence dans l'interfédération : deux événements ne sont pas observés avec la même date par deux fédérés différents. Ceci ne peut donc pas convenir dans le cas général, mais comme dans le cas précédent, ce comportement pourrait être suffisant pour la bonne exécution de certaines fédérations, à déterminer. Cette piste n'a pas été envisagée, mais on peut faire deux remarques. D'une part, dans le cas d'interfédérations complexes, mettant en œuvre une chaîne de ponts et de fédérations, comme l'estampille d'un événement est retardée à chaque

passage par un pont, l'erreur est d'autant plus grande que l'événement et l'observateur sont éloignés dans la structure d'interfédération. D'autre part, même si l'on se restreint cette fois à un pont unique ou à un faible nombre de ponts séparant deux fédérés, il faut vérifier au cas par cas les conséquences de l'erreur introduite.

6.3 Autres éléments à traduire

6.3.1 Abonnements et publications

Declaration Management est un ensemble de services concernant la gestion des déclarations de publication et d'abonnement.

Dans une fédération, les fédérés indiquent s'ils sont susceptibles de modifier des valeurs d'attributs d'instances de classes données (publication d'attributs). Ils indiquent de même s'ils sont susceptibles d'engendrer des instances de classes d'interactions (publication d'interactions). D'autre part, les fédérés indiquent s'ils sont intéressés par ce type d'informations (changements de valeurs d'attributs, lancement d'interactions), classe par classe. Ces services permettent d'éviter un certain nombre d'envois de messages non pertinents aux fédérés.

Huit services de *Declaration Management* concernent ce type de déclarations : ce sont des services à l'initiative de fédérés, grâce auxquels les fédérés indiquent s'ils souhaitent commencer (ou arrêter) à publier (ou à être abonné à) certains attributs ou certaines classes d'interaction.

Quatre autres services, à l'initiave du RTI, permettent au RTI d'indiquer aux fédérés la pertinence de leurs publications. En pratique ces services permettent au RTI de prévenir un fédéré si les informations qu'il publie n'intéressent aucun autre fédéré : dans ce cas, celui-ci peut éviter d'envoyer les messages correspondants.

Tout d'abord, on constate que le pont peut publier et s'abonner à toutes les classes (ou au sous-ensemble commun si les fédérations diffèrent). Ainsi il recevra tous les messages provenant des fédérés, et pourra les réexpédier. Cette solution n'est pas la plus pertinente mais elle montre que les services *Declaration Management* n'engendrent pas de blocages comme certains des mécanismes précédents.

Une optimisation possible consiste à ne s'abonner, dans une fédération, qu'aux classes que l'on est susceptible de publier dans une autre fédération.

Les services indiquant la pertinence des publications concernent en particulier le pont. Celui-ci s'abonne en début d'exécution à toutes les classes possibles, puis peut tenir compte des indicateurs de pertinence pour cesser

un abonnement et éviter ainsi certains envois de messages inutiles dans une fédération.

6.3.2 Informations à l'initiative de fédérés

Nous avons vu que la principale difficulté de conception d'un pont se passant de MOM venait de l'inaccessibilité de certaines informations. Il s'agit en pratique des informations des appels de services à l'initiative des fédérés.

Certaines informations contenues dans ce type de services sont absolument inaccessibles sans l'utilisation du MOM. Ce sont celles qui apportent une information au RTI sans que cette information ait des conséquences visibles pour les fédérés. C'est par exemple le cas de services isolés, qui ne participent à aucun mécanisme. Ce cas se retrouve même dans tout un groupe de services : *Data Distribution Management*. Nous avons vu qu'il s'agit d'un groupe de services destinés à l'optimisation des échanges de messages. Les fédérés indiquent en détail leurs intérêts mais ces informations ne sont utilisées que par le RTI pour décider de la pertinence des messages à expédier éventuellement.

Pour *Data Distribution Management* et certains autres services, on peut se contenter de ne pas tenir compte de leur existence. Comme pour *Declaration Management* où une solution non optimale consiste à publier et à s'abonner à toutes les classes, ne pas tenir compte de *Data Distribution Management* fait perdre, au travers du pont, tout l'intérêt de ce groupe de services, mais n'entraîne aucun blocage. Transmettre les informations relatives aux régions de routage et à leur dynamique nécessite l'utilisation du MOM.

Toute autre solution impose des restrictions qui cette fois sont très fortes. Le pont n'ayant aucune connaissance des régions, la seule solution est de lui fournir séparément toutes les informations nécessaires. Ceci pourrait être envisagé dans le cas de régions statiques et si des objets ne parcourent que certaines d'entre elles. Cette solution ne convient pas aux cas plus complexes avec par exemple des modifications fréquentes de région ou une incertitude totale quant à l'utilisation faite de ces régions par les fédérés.

Ces difficultés liées à *Data Distribution Management* ne concernent que la tentative de transmettre ces services d'une fédération à une autre. Un pont peut être par ailleurs utilisateur de ces services de manière classique. On peut ainsi décider que la transmission d'informations au travers du pont ne concerne que certaines régions, pour des raisons d'optimisation ou de sécurité. C'est aussi une manière possible de sélectionner les données à transmettre pour éviter les redondances dans le cas des interfédérations cycliques.

6.3.3 Traduction d'identifiants

Les différentes entités de la simulation sont référencées à l'aide d'identifiants, et éventuellement d'un nom. Les identifiants ou *handles* sont créés par le RTI lors de la création des entités. Les noms éventuels sont indiqués dans le FOM (classes d'objets et d'interactions, attributs, etc.) ou fournis par les fédérés dans le cas d'entités dynamiques (par exemple les objets). L'association entre nom et identifiant est réalisée par le RTI, et rien ne permet de déterminer a priori comment un RTI attribue des valeurs d'identifiants.

Dans le cas des entités statiques, lues dans le document de description de la fédération, deux exécutions d'un même RTI peuvent donner systématiquement les mêmes identifiants, comme c'est le cas avec CERTI, mais rien ne permet de s'y fier. Les entités dynamiques auront quant à elles des identifiants différents en raison de l'existence d'une interfédération. C'est par exemple le cas de l'interfédération basée sur l'application de test *Billard* : chaque fédéré crée un objet « Bille », qui est initialement le seul objet de la fédération. Dans le cas de CERTI, chacun de ces objets se voit attribuer l'identifiant 1 par son RTI. Dès que les fédérations sont reliées par un pont, chaque RTI découvre les objets des autres fédérations. Dans le cas du Billard, chaque objet Bille donne lieu à la création d'un objet représentant, qui se voit alors attribuer l'identifiant 2, un objet Bille d'identifiant 1 étant déjà présent. Un même objet représenté n'a donc pas partout le même identifiant. La traduction d'identifiants au travers du pont apparaît alors indispensable.

On choisit donc de traduire l'intégralité des identifiants. Afin d'établir la correspondance entre entités de fédérations différentes, le nom indiqué dans le FOM est utilisé. Par exemple, deux classes déclarées par un même nom, chacune dans son FOM, seront considérées identiques. Toute instance de l'une dans une fédération sera vue comme instance de l'autre dans la seconde fédération : c'est le choix le plus naturel lorsque le FOM est identique de part et d'autre. On peut cependant remarquer qu'un pont permettrait de proposer une association entre des classes de noms différents, pourvu que leur définition soit identique. Cette possibilité permettrait de partager, sans modification des fédérations, des concepts partagés mais nommés différemment. Ce principe peut être étendu aux valeurs (par exemple des changements d'échelle) pourvu que les règles de traduction nécessaires soient fournies au pont.

Un fédéré-pont doit donc conserver l'ensemble des correspondances entre identifiants et effectuer une traduction lors de chaque transmission d'événement.

Chapitre 7

Implémentation et résultats

7.1 Implémentation d'un prototype de pont

Un prototype de pont a été réalisé. Il ne nécessite pas la disponibilité d'un MOM. Ceci est bien sûr motivé par l'absence de MOM dans CERTI, mais aussi par le fait que les services essentiels (mise à jour d'attributs, envoi d'interactions, etc.) peuvent être réalisés sans MOM. Ce pont relie un nombre quelconque de fédérations, et reproduit dans les fédérations cibles les objets et les événements observés dans les fédérations sources.

7.1.1 Conception et interfédérations

Le prototype réalisé permet de mettre en œuvre tous les types d'interfédérations évoqués précédemment. En effet, ce pont est constitué d'un ensemble quelconque de composants fédérés, chaque fédéré étant observateur dans une fédération, et simulateur dans les fédérations des autres fédérés du pont. Ceci correspond au type de pont nécessaire pour réaliser une interfédération en étoile, utilisant un pont central et un nombre quelconque de fédérations (1 pont pour n fédérations). Ce pont fonctionnant avec n fédérations peut en particulier être utilisé avec simplement 2 fédérations, et donc pour une interfédération linéaire : chaque liaison entre fédérations est réalisée par une instance d'un tel pont ($n-1$ ponts pour n fédérations). Comme nous l'avons vu, dans ces interfédérations acycliques, chaque pont sépare un certain nombre de sous-parties (fédérations ou interfédérations) et est le seul lien entre ces fédérations.

Le prototype actuel permet de réaliser certaines interfédérations cycliques : le pont peut filtrer les informations à transmettre. Plusieurs ponts fonctionnant ainsi, et configurés de manière à ce qu'aucune information ne soit

redondante ou oubliée, permettent de créer des interfédérations cycliques. Déterminer précisément quelles sont les informations que chaque pont doit transmettre est un travail de conception de l'interfédération, le pont ne fait aucune vérification.

Pour l'instant un seul critère a été mis en place : il concerne le nom des objets créés (nom donné en paramètre à *Register Object Instance*). Si le filtrage est activé, alors le pont considère comme seuls objets publics de la fédération les objets dont le nom commence par une chaîne de caractères donnée. Seuls ces objets seront reproduits dans les autres fédérations. Le préfixe utilisé peut être différent pour chaque fédération. D'autres critères sont à implémenter, mais celui-ci permet de mettre en place immédiatement les tests qui nous intéressent : interfédérations cycliques, et interfédérations avec des objets privés.

Les identifiants utilisés par les RTI diffèrent a priori dans chaque fédération. Chaque composant fédéré conserve donc une liste de toutes les traductions dont il peut avoir besoin : les éléments statiques sont initialisés dès qu'il y a connexion à une autre fédération, puis tous les éléments dynamiques sont traduits lors de leur découverte (ces traductions disparaissent en même temps que l'élément).

7.1.2 Détail des services traduits

Ce prototype ne propose pas de traduction à tous les mécanismes HLA. Les services pris en compte ont été déterminés en fonction de leur importance dans les fédérations et de leur support dans CERTI. Ces deux critères se recoupent largement puisque CERTI a été développé progressivement en privilégiant les services les plus essentiels.

Le prototype de pont actuel utilise bien sûr les services de *Federation Management* pour rejoindre des fédérations, mais ne propose pas de traduction pour la sauvegarde et la restauration.

Le pont utilise les services de *Declaration Management* afin de s'abonner à toutes les entités publiques de chaque fédération, et de publier ces mêmes entités. Ceci est tout d'abord réalisé au niveau du fédéré-pont, alors que chaque fédéré-pont n'a pas connaissance du contenu des autres fédérations. Le lien entre fédérés-ponts est ensuite établi, et chacun détermine la liste des traductions des identifiants.

Les services associés aux mécanismes directs de *Object Management* sont transmis au travers du pont. L'unité de temps est la même pour toutes les fédérations ; les identifiants d'objets, d'attributs et d'interactions sont traduits.

Require: $L_i \geq L_{min}$
  $T_i \Leftarrow$ *Query Federation Time*$/RTI_i$
  loop
    $LBTS_i \Leftarrow$ *Query LBTS*$/RTI_i$
    $GLBTS_i \leftarrow \min_{j \neq i}(LBTS_j \Leftarrow$ *Query LBTS*$/RTI_j)$
    if $L_i \neq L_{min}$ **then**
      $L_i \leftarrow L_{min}$
      $L_i \Rightarrow$ *Modify Lookahead*$/RTI_i$
    end if
    $L_i \Leftarrow$ *Query Lookahead*$/RTI_i$
    $request \leftarrow GLBTS_i - L_i$
    if $request > T_i$ **then**
      $request \Rightarrow$ *Time Advance Request*$/RTI_i$
      repeat
        $tick/RTI_i$
      until *Time Advance Grant* $\dagger/RTI_i$
      $T_i \Leftarrow$ *Query Federation Time*$/RTI_i$
    end if
  end loop

FIG. 7.1 – Algorithme du pont de fédérations

Si un élément (classe d'objet, d'interaction, etc.) est absent d'une fédération cible, alors le mécanisme est ignoré pour cette fédération. Par exemple, si un objet est créé avec *Register Object Instance* mais que la classe de cet objet n'est pas définie dans toutes les fédérations, alors le pont ne crée un objet représentant que dans les fédérations où cette classe existe.

Concernant *Time Management*, le pont est surtout utilisateur de ces services. Les estampilles des messages sont transmises avec les services de *Object Management*, et le pont, déclaré aussi bien contraint que régulateur (afin de pouvoir transmettre tous les messages TSO) n'envoie pas de messages qui lui sont propres, mais uniquement des messages reproduisant ceux qu'il reçoit. Le *lookahead* est ainsi déterminé à partir du LBTS dans la fédération source. Le détail de l'algorithme est indiqué sur la figure 7.1 (les flèches simples indiquent des affectations, les flèches doubles indiquent des affectations via un service HLA, le RTI concerné étant ensuite indiqué. Les variables L_i, T_i, $LBTS_i$ et L_{min} correspondent respectivement au lookahead, au temps logique et au LBTS d'un fédéré f_i, et à la valeur choisie comme lookahead minimal).

Actuellement le pont ne prend pas en compte les services de *Ownership Management* et *Data Distribution Management*.

7.1.3 Configuration

Le prototype de pont utilise un fichier de configuration pour déterminer les relations qu'il doit établir entre les fédérations. La figure 7.2 donne l'exemple d'un fichier de configuration utilisé pour une interconnexion de 2 fédérations. Il s'agit d'un format XML permettant de décrire chaque fédération par son nom, son fichier `.fed` associé, le nom que le pont doit utiliser en tant que fédéré, le nom de la machine sur laquelle se trouve le RTI de cette fédération, et éventuellement un préfixe à utiliser pour le filtrage des objets.

```
<?xml version="1.0"?>
<interfederation>
  <federation>
    <name>federation-01</name>
    <file>federation-01.fed</file>
    <surrogate>bridge1</surrogate>
    <host>cassis</host>
    <filter>a</filter>
  </federation>
  <federation>
    <name>federation-02</name>
    <file>federation-02.fed</file>
    <surrogate>bridge2</surrogate>
    <host>somport</host>
    <filter>b</filter>
  </federation>
</interfederation>
```

FIG. 7.2 – Exemple de fichier de configuration du pont

7.2 Résultats

7.2.1 Mise en œuvre d'interfédérations

Notre premier résultat est la réalisation d'un pont fonctionnel ayant permis de mettre en œuvre des interfédérations.

Bien que tous les services ne soient pas implémentés et que certains aspects de la configuration soient limités, plusieurs interfédérations ont été créées et exécutées afin de se placer dans des conditions représentatives des situations souhaitées :

- interfédérations équivalentes à une fédération initiale donnée, dans différentes formes (cycliques et acycliques) ;

– interfédérations avec des données privées ;

Le prototype de pont assure le filtrage configuré, et traite correctement les différentes formes de fédérations.

7.2.2 Performances

Nous avons également effectué des tests de performance de ces interfédérations. La table 7.1 présente ces résultats : nous avons comparé plusieurs simulations composées de 8 fédérés simulateurs d'un objet (les fédérés-ponts ne sont pas comptés). Comme précédemment, la performance est évaluée par le nombre de pas d'une simulation en temps coordonné à pas fixe et de durée déterminée. Tous les fédérés traitent un objet, et sont a priori intéressés par les objets des autres fédérés.

Les fédérations et interfédérations comparées sont les suivantes :

– une fédération classique à huit fédérés, sans pont ;
– deux fédérations de quatre fédérés, reliées par un pont – il s'agit de la simulation précédente, répartie sur deux fédérations ;
– deux fédérations de quatre fédérés, reliées par un pont, et avec des objets cachés : cas similaire au précédent, mais chaque fédération ne voit qu'un seul objet de l'autre (au lieu de quatre) ;
– interfédération en étoile : huit fédérations, n'ayant chacune qu'un fédéré, et toutes reliées par un unique pont ;
– une interfédération cyclique : avec deux fédérations de quatre fédérés, elle se rapproche du deuxième cas mais ces fédérations sont reliées par deux ponts, chacun traitant la moitié des objets.

type d'interfédération	nombre de pas
fédération classique	840
deux fédérations	562
deux fédérations, certains objets cachés	587
huit fédérations, en étoile	1191
deux fédérations, deux ponts	369

TAB. 7.1 – Comparaison des performances d'interfédérations de topologies différentes

Nous avons choisi ces interfédérations car elles correspondent à des simulations similaires, mais avec des topologies différentes, sans optimisation particulière (avec un seul type d'objet, la simulation ne s'y prête pas). Chaque fédéré, pont ou RTI est exécuté sur une machine différente, ce nombre de machines varie donc en fonction de la configuration testée.

Le deuxième test fait apparaître que, bien que chaque fédération n'ait plus que 5 fédérés (4 fédérés simulateurs et un pont), l'avancement de la simulation est plus lent que dans le cas de la fédération à 8 fédérés : le gain lié à cette charge moindre des RTI est compensé par le transfert des données par le pont.

Les résultats sont similaires lorsque chaque fédération présente des objets cachés à l'autre fédération : le rythme global semble avant tout imposé par les fédérés régulateurs. Un cas plus intéressant consisterait à employer un avancement orienté par les événements et une plus grande variété d'objets.

Concernant les optimisations possibles, il faut rappeler que celles-ci sont fondées sur l'idée d'isoler les événements qui ne concernent qu'une seule fédération. En particulier, le pont ne sera pas abonné aux classes d'objets ou d'interactions présentes dans une seule fédération, que ceci soit déterminé statiquement (description de la fédération) ou dynamiquement (absence effective d'objets au cours de l'exécution). Avec une seule classe d'objet, ce test est insuffisant pour mesurer l'effet de ces abonnements sélectifs, un autre jeu de comparaisons devrait être envisagé.

Par contre, lorsque le pont relie 8 fédérations ne contenant chacune qu'un seul fédéré, les performances sont améliorées. Ce résultat n'était pas attendu : dans cette configuration, le pont a un rôle central en communication avec 8 RTI, nous pensions le voir aussi surchargé que le RTI du cas nominal, où un RTI est connecté à 8 fédérés. Ce cas incite à effectuer des comparaisons plus nombreuses et complètes, en particulier dans le cas de fédérés gérant plusieurs objets, et plus généralement de simulations complexes. Enfin, sans considérer ce point comme une explication, il faut toutefois rappeler que cette configuration est celle qui dispose du plus grand nombre de machines (17 au total : 1 pont, 8 RTI et 8 fédérés simulateurs).

Enfin, le dernier test met en œuvre une interfédération cyclique : deux ponts relient deux fédérations, chaque pont ne gérant donc qu'une moitié des données à transmettre. Les performances sont encore moins bonnes qu'avec un seul pont, ce qui était attendu dans le cas présent : si un unique point ne compense pas la baisse de charge des RTI, on a ici deux ponts (donc un fédéré supplémentaire par RTI) avec certes moins de données à transmettre mais autant d'abonnements.

Si par de nombreux aspects cet ensemble de tests ne fait pas apparaître certains objectifs des ponts de fédérations, il met néanmoins en évidence le coût de leur utilisation dans différentes configurations à nombre de fédérés simulateurs identique. Le premier objectif, la réalisation proprement dite, a

permis de concevoir plusieurs interfédérations et de cerner leurs effets dans cette première mise en œuvre sans optimisation.

7.3 Perspectives

7.3.1 Services

Le prototype de pont traite un ensemble de mécanismes, principalement de *Object Management* et *Time Management*, mais pas l'ensemble des services HLA. Le pont sera d'autant plus générique que les services et mécanismes traduits seront nombreux. Cependant pour cela, la gestion du MOM est préférable et peut donc constituer un objectif préalable à la réalisation d'un pont plus complet.

7.3.2 Configuration, optimisation

Le prototype de pont transmet dans les fédérations cibles tout ce qui est observé dans une fédération source (aux restrictions de filtrage près). Mais si aucun fédéré n'est abonné à une classe donnée, dans l'ensemble des fédérations cibles, il est inutile pour le pont d'envoyer des informations de mise à jour. Il est même inutile qu'il soit abonné à ces informations dans la fédération source. Une optimisation consiste donc à ne pas produire de mise à jour ou à supprimer certains abonnements, lorsque ceux-ci ne sont plus pertinents. Des indications concernant ce type de situation peuvent provenir du RTI, mais on peut également envisager de les indiquer dans le fichier de configuration du pont si celles-ci sont permanentes et impossibles à déduire des messages du RTI. De la même manière, si le pont ne peut pas transmettre les éléments de *Data Distribution Management*, son rôle pourrait être réduit à certains espaces de routage (en particulier dans le cas des interfédérations cycliques).

7.3.3 Traduction de valeurs

La traduction d'identifiants d'une fédération à l'autre a été écrite au fur et à mesure des besoins : ainsi, les classes d'objet, les attributs, les classes d'interactions, les paramètres, et les objets sont traduits, puisque tous ces éléments sont utilisés par les mécanismes de *Object Management*, que le pont transmet. Certaines traductions ne sont pas implémentées, c'est par exemple le cas des labels utilisés pour les sauvegardes. Bien qu'il soit possible d'imposer que le contenu des labels de sauvegarde ne soit pas traduit (si on

souhaite y attacher fortement un sens), on peut imaginer des scénarios dans lesquels une interfédération ne serait constituée qu'après l'exécution pendant un certain temps des différentes fédérations qui la composent. En ayant des historiques différents, les labels de sauvegarde déjà utilisés par les fédérations pourrait être différents, ce qui laisse la place à un risque de conflit. La traduction de labels pourrait être utilisée dans de tels cas. D'une manière générale, les structures nécessaires à la traduction ont déjà été implémentées pour plusieurs types d'identifiants, toutes les traductions éventuellement nécessaires peuvent être implémentées de manière similaire.

Ces traductions ayant été nécessaires, l'ajout de quelques fonctionnalités profitant de ces structures pourrait être réalisé. En particulier on peut envisager comme rôle pour le pont, une traduction des noms du FOM, afin de relier des fédérations qui utilisent des noms différents pour les mêmes entités. Ces traductions pourraient concerner les noms utilisés dans la description de la fédération, mais aussi les valeurs, par exemple pour traiter à l'intérieur du pont des différences d'unités, plutôt que d'avoir à modifier les fédérations, et en particulier le code de certains fédérés.

7.3.4 Liaison avec les RTI

Le pont, en tant que fédéré observateur dans une fédération et simulateur dans une ou plusieurs autres fédérations, est avant tout une application utilisant les services du RTI. Comme toute application HLA, le pont est compilé en utilisant la bibliothèque fournie par le RTI, que celle-ci soit statique ou dynamique. Mais le pont est une application HLA très particulière, puisque dans certains cas, on souhaite l'utiliser pour relier des fédérations s'exécutant avec des RTI différents. Chaque fédéré présent dans le pont doit donc être associé au bon RTI. Ceci pose un problème pratique puisque lorsque le pont est constitué d'un seul processus, les libRTI auxquelles il devrait être lié utilisent exactement la même API pour être conformes à la norme HLA.

Plusieurs pistes sont envisagées pour résoudre ce problème. Si l'on se restreint à des bibliothèques statiques, l'édition des liens d'un programme utilisant plusieurs bibliothèques partageant la même API est impossible (un conflit apparaît et l'exécutable n'est pas créé). Dans le cas de bibliothèques dynamiques, compilation et édition de liens auront lieu, mais l'API commune ne pourra faire référence qu'à une bibliothèque à l'exécution (le lien est réalisé avec la première bibliothèque trouvée correspondant à l'API). Les solutions envisagées consistent donc à introduire une différence entre les API, ou à expliciter dans le programme l'implémentation à utiliser.

L'introduction d'une différence d'API ne peut se faire que dans le cas de CERTI, ou d'un RTI dont les sources sont disponibles et modifiables. L'API

HLA déclare un ensemble de types et de classes dans un espace de noms « RTI » (l'implémentation C++ est réalisée avec les espaces de noms de ce langage, ou en encapsulant toutes les déclarations dans une classe RTI). CERTI respecte cette norme, mais il est possible d'en créer une version particulière, utilisant un autre espace de noms. Dans ce cas, une édition de liens impliquant cette version de CERTI et un autre RTI peut être réalisée, et les services de chaque RTI peuvent être accédés à l'aide de leur espace de noms respectif.

Une autre solution, également basée sur une différenciation des API, consiste à laisser l'API du RTI modifiable strictement conforme au standard (c'est-à-dire, garder l'espace de noms « RTI »), et à ajouter des services spécifiques. Par exemple un service de création d'objet de classe *RTIambassador*, spécifique à CERTI, peut être ajouté. Une telle version de CERTI resterait conforme au standard HLA et proposerait simplement la possibilité de créer un *RTIambassador* de la libRTI de CERTI sans ambiguïté. Par contre, un conflit entre noms des deux libRTI restant possible, cette solution n'est envisageable que dans le cadre d'une édition de lien dynamique, ne donnant pas la priorité à CERTI (c'est en général l'ordre de déclaration des bibliothèques dans la commande de création de l'exécutable qui est utilisé).

Enfin, une dernière solution serait l'utilisation de bibliothèques *chargées dynamiquement*. Le mécanisme est différent de celui des *bibliothèques dynamiques*, et permet de charger une bibliothèque au cours de l'exécution d'un programme, puis d'obtenir des références aux fonctions qu'elle propose. Cependant, ce mécanisme est principalement prévu pour les bibliothèques en C. Le langage C++ n'est pas adapté à ce mécanisme en raison de l'absence d'ABI[1] C++ standard (l'ABI C utilise l'identifiant correspondant dans le code source, le C++ doit passer par une décoration des noms, laissée au choix du compilateur). La méthode habituellement employée pour résoudre ce problème consiste à exiger de la bibliothèque à charger qu'elle propose des constructeurs et destructeurs des classes à importer sous forme de fonctions C. L'annexe C++ de la norme HLA ne propose pas de telles fonctions. Une solution serait donc d'employer d'une part une libRTI de manière habituelle (lien statique ou dynamique, et utilisation usuelle dans le corps du fédéré), et d'autre part un mécanisme de chargement dynamique proposé par un autre RTI (ceci peut être ajouté simplement à CERTI).

[1] Appication Binary Interface : conventions utilisées pour l'édition des liens

7.3.5 Extension à plusieurs processus

Le pont est actuellement composé d'un seul processus. Tous les composants fédérés sont les instances d'une classe *Federate* et appartiennent à la même instance de programme. Ceci pose quelques problèmes qui pourraient être réglés avec un pont constitué de plusieurs processus, gérant chacun un fédéré. En particulier, le problème de la compilation avec des RTI différents serait résolu, chaque composant n'ayant à être lié qu'à un seul RTI. L'intérêt est plus général car une telle structure de pont permet de mettre en œuvre les opérations de filtrage localement, même si la simulation est distribuée sur plusieurs sites. Cette architecture n'a finalement pas été envisagée pour le prototype en raison des problèmes de communication et de synchronisation entre processus que cela pose.

Conclusion et perspectives

Conclusion

Nos travaux ont porté sur l'introduction de la notion de domaine dans la simulation distribuée à événements discrets, en particulier dans le cadre de l'architecture HLA. Cette notion générale, bien que présente dans les objectifs de modularité, d'interopérabilité et de réutilisation de cette architecture, est au centre de nombreuses perspectives visant à renforcer ces caractéristiques de HLA.

Nous avons tout d'abord présenté la simulation distribuée et ses architectures, ses normes, puis c'est dans le cadre plus spécifique de HLA et parfois des outils développés à l'ONERA que nous avons envisagé une nouvelle introduction de ce concept de domaine. Nous avons envisagé différentes perspectives, souvent indépendantes les unes des autres, et intervenant à différents niveaux d'abstraction. Ainsi au plus haut niveau, les interfédérations, et dans une certaine mesure la multirésolution sont utilisatrices de HLA et proposent des concepts qui n'y sont pas nativement. Puis au niveau-même d'abstraction proposé par HLA, nous avons considéré les services de gestion de la distribution des données. Au plus bas niveau, l'emploi du multicast et une conception plus distribuée du RTI sont des propositions qui exploitent cette notion de domaine.

Dans la seconde partie nous avons présenté nos contributions, qui peuvent être regroupées autour de trois points : la transition de CERTI d'un logiciel interne à un logiciel libre, le développement des services de gestion de la distribution des données, et la réalisation d'un prototype de pont de fédérations.

Si le premier point n'est pas directement lié au sujet de recherche de cette thèse, le développement collaboratif proposé par les logiciels libres présente de nombreux avantages, intervenant plus globalement sur le projet, et finalement bénéfiques à la plupart des travaux associés à CERTI. Ainsi de nombreuses contributions extérieures ont été reçues, allant du simple correctif jusqu'à l'implémentation de services complets de HLA, en passant aussi,

de manière plus inattendue, par de la documentation du code source. Plus récemment, plusieurs travaux de portage de CERTI vers d'autres architectures ont été reportés. En plus d'améliorer la qualité du code, les ports permettent d'atteindre un plus grand nombre d'utilisateurs et de contributeurs potentiels.

En dehors de cette contribution qui a concerné CERTI dans son ensemble, nous avons proposé une implémentation de services de gestion de la distribution de données, cependant ces travaux ne sont pas encore assez avancés pour être vérifiés ni donner lieu à des résultats.

Enfin, notre principale réalisation a été le développement d'un prototype de pont de fédérations. Les interfédérations, et notamment les ponts, sont particulièrement intéressants vis-à-vis de la notion de domaine, car leur caractère générique ne les limite pas à un seul objectif ou intérêt. Nous avons vu que leur utilisation était envisagée pour des questions de sécurité, de performance, d'interopérabilité. Si des travaux supplémentaires seraient nécessaires dans le cas des perspectives de performance, le prototype actuel est néanmoins fonctionnel et permet de mieux cerner les interfédérations à base de fédérés-ponts. Sans qu'elle soit généralisée à tous les types d'événements, nous avons implémenté un cas d'objets locaux à une seule fédération, ce qui rejoint aussi bien des objectifs de sécurité que de conception. Enfin concernant l'interopérabilité, les mécanismes de traduction déjà présents peuvent être étendus à la traduction de valeurs ou de concepts qui seraient présents dans plusieurs fédérations, mais incompatibles.

Perspectives

Ces derniers points constituent l'une des perspectives de ce travail de thèse. Ainsi concernant le prototype de pont, à la manière du développement de CERTI, seuls les principaux services ont été pris en compte dans un premier temps. Pour le développement de nouveaux services, deux directions sont possibles : s'il s'agit de créer un pont vraiment complet et générique, il nous semble que l'existence de services du MOM doit être considérée comme prérequis, auquel cas ce développement doit être réalisé sur CERTI. Par contre, si l'utilisation du pont est plus modérée, et vise par exemple une interopérabilité immédiate entre fédérations simples, des ajouts ponctuels de traductions, et la prise en considération de contraintes pour compenser l'absence du MOM devraient être suffisants.

Un pont propose une solution assez générique à un ensemble de problèmes concernant l'interopérabilité, la sécurité, la conception, et éventuellement les

performances. Si cet aspect générique présente un intérêt pratique évident, puisqu'une seule réalisation est envisagée pour répondre à plusieurs problèmes, cette solution présente justement l'inconvénient de ne pas être assez spécifique à chaque problème donné. Un pont ne saurait être la seule réponse à des objectifs d'optimisation. De même, s'il peut proposer une interopérabilité immédiate, là où une modification des fédérés aurait été nécessaire, une telle utilisation peut fortement diminuer les performances. L'optimisation et l'interopérabilité nous semblent vraiment nécessiter des études plus spécifiques, du point de vue de l'implémentation de HLA mais aussi par la proposition d'évolutions de la norme.

Enfin, CERTI a été au cœur de ce travail et nous pouvons évoquer quelques perspectives concernant son développement général. La poursuite d'une politique impliquant le choix et le développement de logiciels libres nous semble particulièrement adaptée à la simulation distribuée et à ses objectifs d'interopérabilité. Les contributions extérieures, particulièrement variées, viennent soutenir ce choix.

Concernant la conception de CERTI proprement dite, si l'architecture en processus communicants du RTI nous paraît tout aussi pertinente qu'à ses débuts (la plupart des fournisseurs de RTI ont suivi cette voie), le développement incrémental de CERTI a fait apparaître l'existence d'éléments qui ne sont pas assez factorisés. Ainsi les services sont gérés d'une manière procédurale, alors qu'ils feraient d'excellents candidats à une conception objet facilitant l'ajout et la maintenance de services HLA. Les premières versions de CERTI ne proposaient qu'un ensemble minimal de services, tandis qu'ils sont maintenant très nombreux. C'est pourquoi si une telle conception n'était pas essentielle au début, ses avantages sont aujourd'hui beaucoup plus convaincants. De plus, une amélioration de certains aspects de conception aurait pour conséquence de faciliter non seulement les travaux de l'ONERA, mais aussi l'accès au projet, en réduisant l'effort initial nécessaire aux contributeurs extérieurs.

Annexes

Annexe A

Contribution à CERTI

Nos travaux concernant CERTI peuvent être regroupés en contributions relatives à la mise à disposition de CERTI en logiciel libre, et d'autre part au développement du groupe de services *Data Distribution Management*.

A.1 Portabilité et maintenance

Fonctionnant initialement de manière stable sur systèmes Solaris (machines Ultrasparc), nous avons effectué le portage de CERTI sur système GNU/Linux, machines PC. Ce portage n'a pas nécessité de changements importants de CERTI, qui utilisait principalement des bibliothèques standard C, C++ et Unix. Les principales difficultés ont concerné l'existence de codes correspondant à des comportements indéfinis ou non spécifiés de la norme C++ [ISO98], fonctionnant de manière attendue sur machines Sun, mais de manière différente sur machines PC (malgré le même compilateur, GCC[1]).

Ce portage a mis en évidence le besoin d'utiliser des outils de compilation permettant de prendre en compte les spécificités des différentes plate-formes cibles, tout en affectant le moins possible le code source principal.

A.1.1 Système de compilation

Le précédent système de compilation de CERTI était uniquement basé sur l'outil *make*. Il permet de compiler tout un projet à partir de fichiers de description du processus de compilation, les *makefiles*. *Make* et les *makefiles* sont des outils qui ont surtout l'avantage d'être présents sur toutes les plate-formes Unix, mais qui présentent certains inconvénients :

[1]GNU Compiler Collection, http ://gcc.gnu.org

- ils doivent être modifiés dès qu'il y a modification de la structure du projet ;
- différents formats de *makefiles* existent, avec certaines incompatibilités ;
- leur contenu est une description de bas niveau, qui nécessite de préciser toutes les dépendances et une partie des commandes de compilation ;
- s'il est possible de prévoir des traitements différents pour la compilation sur différentes plates-formes, cette gestion n'est pas automatisée.

Les outils GNU *Autoconf*[2] et *Automake*[3] de la *Free Software Foundation* (FSF) ont pour but de générer des scripts et des *makefiles* à partir de descriptions du projet de plus haut niveau.

Autoconf permet de générer un script shell qui déterminera, juste avant la compilation, les spécificités de l'environnement de compilation, et passera au compilateur des options adaptées, quelle que soit cette plate-forme. *Autoconf* crée ce script à partir d'une description de haut niveau des tests à effectuer. Cet outil étant très utilisé, la plupart des tests nécessaires existent dans la bibliothèque fournie. On en trouve également sur des sites web dédiés.

Automake reprend la même idée pour les *makefiles* : à partir d'une description de haut niveau (indication des fichiers sources, des exécutables et bibliothèques à créer, etc.), *Automake* génère les *makefiles* associés, compatibles avec toutes les variantes de *Make*.

Ce système de compilation est dans un premier temps un peu plus compliqué à mettre en place que les *makefiles*, à cause du fichier de description à fournir à *Autoconf*. Mais une fois cette première étape réalisée, les modifications à effectuer dans les équivalents des *makefiles* sont plus rares, et toujours plus simples. De plus, *Autoconf* facilite fortement le portage de CERTI sur différentes architectures.

CERTI a ainsi été compilé et testé avec succès sur les plates-formes
- Sun/Solaris (avec les compilateurs Sun CC aussi bien que GCC) ;
- PC/GNU/Linux (distributions *Debian GNU/Linux* et *Linux Red Hat*) ;
- Windows, à l'aide de Cygwin qui propose un environnement de type Unix.
- SGI/Irix
- HP/HP-UX

A.2 Aspects coopératifs du projet

Pour la diffusion en logiciel libre, mais aussi comme bonnes pratiques de développement, quelques outils ont été introduits.

[2]Site web du projet : http ://sources.redhat.com/autoconf/
[3]Site web du projet : http ://sources.redhat.com/automake/

A.2.1 Contrôle de version

Au cours de son développement, CERTI a été placé sous contrôle de version avec le logiciel CVS[4], mais pas de manière systématique. Certaines évolutions ont été réalisées sans cet outil, et le développement incrémental n'ayant pas entraîné de conflits entre les versions développées, les versions 2.x n'étaient en pratique plus contrôlées. Nous avons donc réintroduit le contrôle de version par CVS.

Le choix du logiciel CVS s'est imposé en raison de sa fiabilité et du niveau de développement encore trop peu avancé des alternatives visant à combler ses lacunes (comme par exemple *Subversion*[5], qui était alors encore en version de développement). Depuis, les alternatives ont gagné en stabilité, cependant CERTI a été placé sur un site d'hébergement de projets libres et CVS y est toujours le seul outil disponible.

CVS permet tout d'abord de sauvegarder un projet à différents instants de son évolution. Il ne s'agit pas seulement de sauvegardes régulières, mais de sauvegardes de toute modification du projet, et il est ainsi possible de récupérer d'anciennes versions, que ce soit globalement pour le projet ou individuellement pour un fichier.

Dans ce cadre, CVS ne se limite pas à un développement linéaire du projet, mais permet la création de branches de développement. Il est possible de récupérer une ancienne version, d'effectuer des modifications sur celle-ci et ainsi de partir dans une « direction différente ». Un cas très fréquent d'une telle utilisation concerne l'apport de correctifs sur une version existante (une version 1.0 par exemple). Il est alors possible de travailler à la fois sur une branche 1.x améliorant la version 1.0, et sur une branche de développement menant à une version 2.0. Ces développements s'effectuent en parallèle, sans interférences, et l'historique de chaque branche est sauvegardé.

CVS permet alors la fusion (complète ou partielle) de branches : il s'agit d'appliquer certaines modifications effectuées sur une branche à une autre. CVS automatise cette tâche, en fusionnant les différentes versions des fichiers : les modifications effectuées de part et d'autre se retrouvent dans la version résultante. Seul le cas de la modification, de part et d'autre, d'une même portion d'un même fichier ne peut être traité automatiquement et reste à effectuer manuellement.

Les branches peuvent être utilisées dans de nombreuses situations, et pas uniquement pour le cas classique d'une branche principale de développement et de branches de correctifs d'anciennes versions. Tout développement im-

[4] Concurrent Versions System, http ://www.cvshome.org

[5] http ://subversion.tigris.org

portant peut être effectué sur une branche différente afin de ne revenir sur la branche principale que sous certaines conditions (stabilité, interfaces finalisées, etc.)

Enfin, CVS permet à plusieurs personnes de travailler sur un même projet sans avoir à bloquer les parties concernées. Le principe de fusion de fichiers est également appliqué lorsque plusieurs développeurs modifient le même projet, et en particulier le même fichier. Pour cela, CVS définit une *base* (ou *dépôt*) contenant le projet et son historique. Chaque developpeur crée une ou plusieurs copies de travail à partir de cette base, effectue des modifications sur cette copie, et met régulièrement à jour ces modifications dans la base.

CVS possède quelques inconvénients, corrigés désormais par la plupart de ses concurrents. Toutefois, les limitations de CVS en question ne sont pas gênantes pour CERTI, et concernent surtout des projets dans lesquels des changements de structure ont lieu fréquemment. C'est pourquoi CVS convient pour l'instant à CERTI et est employé pour la gestion des sources.

A.2.2 Nommage

Résultant du travail de plusieurs développeurs n'ayant pas utilisé une norme d'écriture commune, le style du code de CERTI était donc très hétérogène. Le respect d'un unique style de code étant fortement recommandé pour des questions de lisibilité, l'application d'un style a été envisagée suite à la publication de CERTI.

Non seulement un style de code facilite la lecture, mais en plus il diminue les risques de voir des différences entre versions qui ne correspondraient pas à un changement de fond mais de forme. Ainsi les historiques sauvegardés par le logiciel de gestion de versions ne contiennent que des modifications de fond, et les différences entre versions ne correspondent qu'à des changement effectifs, ce qui simplifie les comparaisons.

Le style de code intervient à la fois sur le nommage des identifiants et sur le positionnement des instructions et déclarations. Nous avons retenu les règles de nommage et de casse généralement employés en programmation orientée objet (celles que l'on retrouve par exemple avec les recommandations de Sun Microsystems pour le code Java). Les détails non liés au nommage ont été choisis en évaluant les arguments qui accompagnent divers documents de style de code.

A.2.3 Documentation

CERTI propose un ensemble de services destinés à implémenter la norme HLA, pour l'instant dans le cadre de sa version 1.3. En ce sens, la documentation des services est celle de HLA, avec les compléments de documentation que l'on peut trouver dans le RTI-NG (puisque cette implémentation du DoD a représenté le standard de fait pour les détails absents de la norme.) Le principal besoin de CERTI concerne donc, pour la partie utilisateur, l'installation et la configuration du logiciel, la connexion de fédérés à ce RTI, et le détail des services implémentés. La distribution actuelle de CERTI fournit un document d'installation, un fichier résumant les variables d'environnement utiles, et un programme de test. La liste des services non implémentés n'existe pas en tant que document, mais apparaît à la lecture du fichier source regroupant l'API.

CERTI évoluant régulièrement et de manière significative, le projet est confronté au problème de la mise à jour des documentations : celles-ci doivent idéalement être à jour et correspondre au code actuel, et lorsque ceci n'est pas possible ou n'a pas été réalisé, le caractère obsolète doit être mis en évidence.

Nous avons considéré l'utilisation d'un logiciel d'aide à la documentation, et destiné à documenter les fonctions d'un code, c'est-à-dire principalement leur interface. Si ceci ne représente pas l'ensemble de la documentation nécessaire, il s'agit de documentation pénible à mettre à jour si l'on ne force pas le lien entre code et documentation. Ce type de logiciel a pour but d'aider à la documentation des fonctions et à la description des relations existant dans le code (par exemple en C++, relations de hiérarchie ou d'utilisation de classes, espaces de noms, etc.) L'incitation à rendre la documentation à jour est réalisée en la plaçant directement dans les commentaires du code. On retrouve très largement cette idée dans la documentation des programmes Java. À l'aide de conventions et de balises, certains commentaires sont extraits pour fournir la base de la documentation des fonctions associées. Ceci est accompagné par une analyse du programme permettant de reconnaître classes, fonctions, etc.

Ceci ne concerne pas seulement la documentation utilisateur. Avec la distribution de CERTI en logiciel libre, la création d'une documentation d'implémentation devient très importante pour d'éventuels contributeurs découvrant le code source. Et ce point pose également le problème de la documentation des contributions. Dans un projet libre, pouvant recevoir des contributions extérieures, la maintenance de la documentation est une réelle difficulté (par exemple dans la communauté du logiciel libre, de nombreux contributeurs proposent des patches, mais rarement de la documentation — souvent les

rédacteurs de documentation ne sont pas les mêmes personnes, et la documentation est souvent en retard sur l'avancement du logiciel.) Une des solutions les plus intéressantes consiste à intégrer dans le code source une partie de la documentation, et en particulier la spécification des classes et des fonctions. À l'aide de balises particulières et de quelques conventions utilisées dans les commentaires du code source, des outils permettent de produire une documentation à partir de l'analyse du code source et de ces commentaires spéciaux. Écrite dans le code, ce type de documentation est la moins contraignante pour les contributeurs, puisqu'elle ne nécessite pas l'édition de fichiers de documentation séparés, et que son contenu est proche de ce que seraient des commentaires habituels. Le logiciel retenu, *Doxygen*[6], permet de créer ce type de documentation dans différents formats : PDF, HTML (avec coloration du code, mise en place d'hyperliens entre les méthodes et leurs définitions, dessins des associations ou des relations d'héritage, etc.), pages de manuel `man` Unix ou `info` (l'équivalent proposé par GNU). Il est utilisé dans de nombreux projets de grande envergure tels que l'environnement KDE.[7] Les détails de la génération peuvent être réglés dans un fichier de configuration (par exemple : présentation ou non dans la documentation des attributs de classe privés). Actuellement trois fichiers permettent de créer respectivement une documentation de développement, une documentation utilisateur, et la documentation de l'application de test. Bien sûr, l'état d'avancement de la documentation est directement lié aux commentaires rédigés dans le code source : les fonctions récentes ou modifiées sont documentées avec cet outil, tandis que le code plus ancien n'en profite pas encore.

A.3 Distribution en logiciel libre

Depuis octobre 2002, CERTI est distribué en tant que logiciel libre [BS02].

Un logiciel libre est un logiciel dont les termes de distribution assurent les propriétés suivantes :

- le logiciel peut être utilisé en toutes circonstances, par quiconque et sans limitations ;
- le logiciel peut être modifié librement, son code source étant fourni ;
- le logiciel peut être copié et distribué librement ;
- des versions modifiées ou des travaux dérivés peuvent être distribués librement.

Parmi les licences de logiciels libres, on peut distinguer deux grandes catégories : les licences *copyleft* et les licences non-*copyleft*. Le principe du

[6] http ://www.doxygen.org

[7] *K Desktop Environment* — http ://www.kde.org

copyleft est d'imposer, tout en respectant les propriétés du logiciel libre, que toute redistribution (d'une version originale, d'une version modifiée, ou d'un travail dérivé) conserve ces quatre libertés.

La principale licence copyleft est la GNU General Public Licence, ou GPL, qui n'autorise la distribution de travaux dérivés que s'ils sont eux-même sous licence GPL. La principale licence non-copyleft est la licence BSD, elle n'exige qu'une mention de copyright lors des redistributions (qui peuvent être propriétaires).

Pour CERTI, le choix d'une licence copyleft est préférable : ces licences imposent plus de contraintes pour la réutilisation, mais la transition vers une licence non-copyleft reste possible. À l'inverse, bien qu'il soit possible de passer d'une licence non-copyleft à une licence copyleft, l'inconvénient dans ce cas est que toute version distribuée sous licence non-copyleft peut être utilisée comme telle, même si les versions ultérieures utilisent une autre licence. Le choix d'une licence copyleft est donc le plus prudent.

Enfin, avec une licence copyleft, les travaux dérivés doivent être distribués sous la même licence, ce qui peut être contraignant pour la commercialisation de tels dérivés. Avec un principe de double licence, un même logiciel peut être proposé en version libre copyleft, et dans une version autorisant les travaux dérivés propriétaires. Certaines sociétés utilisent ce modèle afin de profiter des avantages d'un logiciel libre (développement ouvert, contributions), tout en proposant une autre version, celle-ci exploitée commercialement.

A.3.1 Développement ouvert

L'aspect coopératif du développement des logiciels libres a donné lieu à la mise en place de sites dédiés à leur hébergement à la fourniture de services adaptés.

CERTI utilise *Savannah*[8], le site de développement de la *Free Software Foundation*, à l'origine du projet GNU.[9] Le but de Savannah est de proposer aux projets libres un ensemble d'outils adaptés au modèle de développement du logiciel libre. Si la plupart de ces outils pourraient être mis en place de manière interne, Savannah a l'avantage de proposer un compte de projet dans lequel de nombreux outils sont déjà installés et fonctionnels.

Savannah permet de créer des comptes utilisateurs donnant accès aux nombreuses fonctionnalités du site. Un utilisateur peut en particulier demander la création d'un projet, et si celui-ci est approuvé, obtenir un compte de projet. Le demandeur obtient alors le statut *administrateur* sur ce projet,

[8] http://savannah.gnu.org
[9] http://www.gnu.org

et peut donner le statut de *développeur* ou *administrateur* à d'autres utilisateurs de Savannah. Les différentes actions possibles dépendent du statut de l'utilisateur par rapport à ce projet. Par exemple, l'administrateur peut modifier les statuts d'autres utilisateurs, activer ou non les différents outils disponibles, ce que ne peuvent pas faire les développeurs ou les utilisateurs ordinaires. Les développeurs ont un accès en écriture au code source, tandis que le reste des utilisateurs n'a d'accès qu'en lecture.

La page de développement du projet[10] propose des liens vers les différents outils et la page principale du projet[11], et permet de présenter clairement les dernières nouvelles.

A.3.2 Listes de diffusion

Savannah permet la création en ligne de listes de diffusion. Actuellement, une liste, *certi-announce*, est utilisée pour annoncer les nouvelles versions de CERTI, ou toute autre information importante. Une autre liste, *certi-devel*, est utilisée pour discuter du développement de CERTI. Ces listes sont publiques et archivées. L'abonnement à ces listes est possible directement à partir du site.

A.3.3 Rapports de bugs

Le gestionnaire de bugs permet de facilement proposer un rapport de bug. Il suffit de sélectionner parmi les choix proposés les caractéristiques du bug, et de rédiger une courte description. L'emploi d'un gestionnaire de bugs permet de centraliser les informations liées à des problèmes particuliers. Il est possible d'ajouter des informations et de discuter à l'aide d'un forum associé à chaque bug. Chaque utilisateur enregistré sur Savannah et qui soumet un bug a la possibilité de recevoir automatiquement par e-mail les informations de mise à jour concernant ce bug (changement de statut, nouveau message, etc.)

Un gestionnaire de bug a pour principal intérêt de centraliser tous les bugs recensés et d'éviter que certains d'entre eux ne soient oubliés. Ainsi ces informations sont correctement conservées même lorsque le développeur concerné par le bug rapporté quitte le projet. Cet outil permet également aux utilisateurs de voir si un problème rencontré a déjà été répertorié, s'il est corrigé par une version ultérieure, etc.

[10] la page du compte CERTI : http://savannah.nongnu.org/projects/certi/
[11] pour CERTI, http://www.cert.fr/CERTI/

A.3.4 Gestionnaire de patches

Le gestionnaire de patches permet de proposer des patches à appliquer au code source de CERTI. Le fonctionnement est assez similaire à celui du gestionnaire de bugs, l'utilisateur fournit simplement un patch et indique une catégorie. Là encore le statut du patch peut être suivi, et discuté dans un système de messages. L'intérêt réside de même dans le rattachement du patch au projet, et non à un développeur particulier.

A.3.5 Dépôt CVS

Savannah propose également les services du gestionnaire de versions CVS. On retrouve donc le fonctionnement classique de CVS, mais en utilisant un dépôt situé sur le serveur Savannah (auparavant CERTI utilisait un dépôt local). Le site web permet également de naviguer dans les sources, d'afficher des différences entre versions quelconques d'un même fichier, et d'afficher des arbres représentant l'évolution d'un fichier. Toutes les connexions pour modification sont effectuées de manière sécurisée par SSH. Les développeurs du projet ont un accès en écriture au dépôt CVS, les autres utilisateurs ont un accès anonyme en lecture seule.

A.3.6 Autres outils

Bien qu'ils ne soient pas utilisés sur le site de CERTI, Savannah propose d'autres outils :

- un gestionnaire de tâches aidant à la répartition entre développeurs des différents travaux à effectuer ;
- un gestionnaire de support technique, permettant aux utilisateurs de poster des demandes d'aide ;
- la possibilité de définir des sous-projets.

A.3.7 Contributions extérieures

Suite à la distribution en logiciel libre de CERTI, des contributions extérieures ont été proposées, principalement sous forme de patches. Celles-ci concernent des correctifs, l'utilisation des conteneurs de la bibliothèque standard C++ et l'ajout ou la traduction en anglais de commentaires dans les sources.

A.4 Régions de routage

Les services de gestion de la distribution des données constituent l'une des approches envisagées pour le renforcement de la notion de domaines dans les simulations basées sur CERTI. Nous avons réalisé l'essentiel de ces services mais l'ensemble doit encore être testé et finalisé.

A.4.1 Description dans le FOM

Les régions de routage doivent être définies dans le FOM de la fédération. CERTI utilisait un parseur de fichiers `.fed` correspondant aux éléments implémentés de la norme HLA 1.3, ce qui n'incluait pas la définition d'espaces de routage.

La première étape a été de réaliser un parseur de fichiers XML conformes aux descriptions de fédérations de HLA IEEE 1516. La principale raison de ce choix est la difficulté de maintenir le parseur de fichiers `.fed`, écrit en C++, puisque chaque ajout dans ce format représente une nouveauté grammaticale à gérer. En reprenant le format proposé dans HLA 1516, il est possible d'utiliser des bibliothèques libres de gestion de fichiers XML, et donc de minimiser les modifications liées aux fichiers de fédérations. En pratique, une bibliothèque XML proposant une approche DOM[12] permet de créer une structure de donnée représentative du contenu du fichier. Cette structure est celle d'un arbre et la bibliothèque propose des fonctions permettant de naviguer dans cette structure, d'un nœud à l'autre. Cette lecture simplifiée permet d'ajouter rapidement la reconnaissance de nouveaux éléments en phase de développement, par exemple pour tester des extensions. Un autre avantage de l'utilisation d'une bibliothèque de gestion de fichiers XML est que le parseur ne s'arrête pas sur les balises non connues : avec de telles bibliothèques, on cherche dans l'arborescence les éléments reconnus, sans tenir compte des autres, tandis que le parseur de fichiers `.fed` initial bloquait sur les éléments inconnus, ce qui nécessitait des adaptations lors du portage de fédérations vers CERTI.

Nous avons donc développé dans un premier temps un parseur de FOM décrit en XML. Il a été utilisé pour la lecture des informations relatives aux espaces de routage et le début du développement des services de *Data Distribution Management*. Différents problèmes de portabilité du parseur de fichiers `.fed` nous ont amené à réécrire celui-ci, non pas directement en C++ mais à l'aide des outils Lex et Yacc. La prise en compte des espaces de routage a été réalisée à cette occasion et CERTI dispose ainsi des formats de FOM

[12]Document Object Model

des deux principales versions de HLA : 1.3 et IEEE 1516.

Quel que soit le format, le contenu des fichiers de description est équivalent. Actuellement, CERTI peut utiliser indifféremment la version `.fed` ou la version XML d'une description. Par exemple, l'application de test (des déplacements de mobiles) peut être utilisée avec une description `.fed` (figure A.1) ou son équivalent XML (figure A.2).

```
(fed
 (objects
  (class "Bille"
         (attribute "PositionX" FED_RELIABLE FED_TIMESTAMP)
         (attribute "PositionY" FED_RELIABLE FED_TIMESTAMP)
         (class "Boule"
                (attribute "Color" FED_RELIABLE FED_TIMESTAMP)
                )
         )
  )
 (interactions
  (class "Bing" FED_RELIABLE FED_TIMESTAMP
         (parameter "BoulNum")
         (parameter "DX")
         (parameter "DY")
         )
  )
 )
```

FIG. A.1 – Description de la fédération *Billard* en HLA 1.3

A.4.2 Services implémentés

Les services ont été implémentés conformément à la norme, mais sans optimisations basées sur la réduction du nombre effectif de régions ou l'utilisation du multicast. En plus de la lecture de la description de la fédération, ces services ont donc nécessité la définition de types (régions, espaces de routages, etc.) et leur prise en compte dans les échanges de messages. Les services proprement dits mettent à jour les publications et souscriptions avec les régions correspondantes. Ces éléments sont déduits du fichier `.fed` d'une part avec la définition de sections décrivant les espaces de routage, et d'autre part avec une référence éventuelle à un espace de routage, dans les définitions des attributs et des classes d'interaction (cf. figure A.3).

La définition des régions (création, mise à jour, destruction) et les informations de publication et d'abonnement sont transmises jusqu'au au RTIA

```
<?xml version="1.0"?>
<objectModel>
        <objects>
                <objectClass name="Bille">
                        <attribute name="PositionX"
                                   transportation="HLAreliable"
                                   order="TimeStamp" />
                        <attribute name="PositionY"
                                   transportation="HLAreliable"
                                   order="TimeStamp" />
                        <objectClass name="Boule">
                                <attribute name="Color" />
                                           transportation="HLAreliable"
                                           order="TimeStamp" />
                        </objectClass>
                </objectClass>
        </objects>
        <interactions>
                <interactionClass name="Bing">
                                   transportation="HLAreliable"
                                   order="TimeStamp" />
                        <parameter name="BoulNum" />
                        <parameter name="DX" />
                        <parameter name="DY" />
                </interactionClass>
        </interactions>
</objectModel>
```

FIG. A.2 – Description de la fédération *Billard* en HLA IEEE 1516

puis au RTIG où sont effectués les tests de correspondance entre régions.

```
<?xml version="1.0"?>
<objectModel>
        <objects>
                <objectClass name="Bille">
                        <attribute name="PositionX"
                                   transportation="HLAreliable"
                                   order="TimeStamp"
                                   space="geo" />
                        <attribute name="PositionY"
                                   transportation="HLAreliable"
                                   order="TimeStamp"
                                   space="geo" />
                        <objectClass name="Boule">
                                <attribute name="Color" />
                                           transportation="HLAreliable"
                                           order="TimeStamp" />
                        </objectClass>
                </objectClass>
        </objects>
        <interactions>
                <interactionClass name="Bing">
                                   transportation="HLAreliable"
                                   order="TimeStamp"
                                   space="geo" />
                        <parameter name="BoulNum" />
                        <parameter name="DX" />
                        <parameter name="DY" />
                </interactionClass>
        </interactions>
        <routingSpace name="geo">
                <dimension name="x" />
                <dimension name="y" />
        </routingSpace>
</objectModel>
```

FIG. A.3 – Description de la fédération *Billard* avec un espace de routage

A.5 Perspectives

A.5.1 Normes

Si HLA 1.3 reste fortement utilisé en particulier aux États-Unis, le DMSO ne fournit plus aujourd'hui ni implémentation de référence gratuite, ni les

documents de la norme elle-même et renvoie l'utilisateur à la norme IEEE 1516. Par ailleurs on constate que si HLA a été moins pris en considération en Europe tant que la norme était définie par le DMSO, la publication en tant que norme civile IEEE a relancé son utilisation, mais dans cette version de l'IEEE. La prise en compte de la norme IEEE 1516 apparaît donc indispensable, celle-ci étant la seule norme activement maintenue.

D'autre part, si CERTI a été initialement développé à partir d'une partie seulement de C++, ce langage dispose d'une norme ISO depuis 1998 et les compilateurs « quasiment conformes » sont désormais nombreux. Certains éléments du code source, plus proches du C que du C++ pourraient profiter des possibilités actuelles des compilateurs C++ (conteneurs, aspects objet).

A.5.2 Liaison avec d'autres langages

La norme HLA propose trois versions de son API, chacune pour un langage : Ada, C++ et Java. CERTI propose actuellement une interface C++, une amélioration intéressante consiste donc à ajouter des interfaces Java et Ada. Ceci peut être réalisé en écrivant deux nouvelles versions de la libRTI, dans chacun de ces langages. En effet, la libRTI ne contient que la gestion des messages à envoyer au RTIA. Cependant elle fait appel à une partie de la bibliothèque CERTI commune à tous ses composants (libRTI, RTIA, RTIG), qui devrait alors être réécrite ou utilisée depuis un langage différent.

Une autre approche consiste à utiliser des outils de génération automatique de code de liaison tels que SWIG.[13] Le principal avantage est de conserver une bibliothèque libRTI de référence, qui sera utilisée par les autres langages. Une modification de celle-ci sera répercutée automatiquement sur le code généré, et l'indépendance entre langages est assurée : l'utilisation d'un ensemble de services donnera le même comportement, quel que soit le langage utilisé, tandis que des libRTI différentes font apparaître d'éventuels problèmes de synchronisation entre versions.

Un tel choix permet également d'envisager d'autres langages que C++, Java et Ada pour l'écriture de fédérés, en particulier des langages tels que Python qui présente des avantages :

- il est plus simple, et plus abordable pour le développeur de fédéré qui n'est pas nécessairement développeur de métier ;
- son caractère simple est aussi un avantage lors de la phase d'apprentissage de HLA, évitant certains obstacles liés au langage utilisé ;
- il peut être utilisé pour du prototypage, avant de passer à la mise en œuvre définitive avec un langage compilé.

[13] *Simplified Wrapper and Interface Generator*, http ://www.swig.org

Bien qu'il ne soit pas encore supporté par SWIG, un autre langage particulièrement intéressant dans ce cadre est Lua [IdFC96] : langage de script employé pour la recherche et dans l'industrie, orienté événement, il a notamment été utilisé dans le cas d'applications parallèles , avec CORBA [CRI97] et dans le cas d'extensions d'applications ou de systèmes distribués [RUI+96]. De nombreuses références sont disponibles sur le site web[14] consacré à ce langage.

A.5.3 Extensions

La disponibilité des sources et le placement sous licence de logiciel libre permet d'envisager plus facilement l'utilisation de CERTI comme base de travaux mettant en œuvre des extensions de HLA. Par exemple [Rau03, NRR03] présente l'utilisation de CERTI pour tester une proposition d'extension de HLA visant à rendre dynamique le modèle objet.

Cependant ces travaux ne sont possibles qu'en réalisant des travaux dérivés fondés sur CERTI (des versions dérivées, des modifications). En effet CERTI ne propose aucun mécanisme permettant aux développeurs d'extensions de fournir une bibliothèque séparée de CERTI. Un véritable mécanisme d'extension permettrait la prise en compte dynamique d'une telle bibliothèque sans modifier CERTI, avec un principe de greffons (*plug-ins*).

Le développement d'extensions est également un cas fréquent d'utilisation de langages de scripts : si le noyau d'une application est écrit en langage compilé, des extensions peuvent être écrites à l'aide de langages plus simples (sans gestion de mémoire) et de plus haut niveau.

Cette perspective peut donner lieu à une solution commune pour les extensions hors HLA et l'ajout de services HLA, qu'il s'agisse de services non implémentés ou de la version IEEE 1516.

[14] http ://www.lua.org/papers.html

Annexe B

Publications

Publications relatives à ce mémoire de thèse.

- Conférences internationales avec actes et comité de sélection
  - CERTI : Evolutions of the ONERA RTI Prototype [BS02]
  - Design and Implementation of a HLA Inter-federation Bridge [BS03b]
- Communications internationales sans actes et avec comité de sélection
  - Bridges in HLA Distributed Simulations [BS03a]
- Poster
  - An HLA Distributed Virtual Reality Application on a Linux Cluster [TBS+03]

Annexe C

Sigles et acronymes

ALSP *Aggregate Level Simulation Protocol.* Architecture de simulation distribuée.

CERT *Centre d'Études et de Recherches de Toulouse.* ONERA centre de Toulouse

CERTI CERT-*RTI.* RTI développé comme logiciel interne au centre de Toulouse de l'ONERA puis comme logiciel libre.

DIS *Distributed Interactive Simulation.* Architecture de simulation distribuée.

FOM *Federation Object Model.* Modèle objet d'une fédération : description conforme à l'OMT HLA de l'ensemble des données impliquées dans des échanges entre fédérés.

HLA *High-Level Architecture.* Architecture de simulation distribuée destinée à faciliter la mise en œuvre, l'interopérabilité et la réutilisation de simulations complexes. Voir § 2.3.

IEEE 1516 *Standard 1516 définit par l'IEEE.* Standardisation civile de HLA par l'IEEE. Initialement conçue et normalisée par le DMSO, HLA est une norme civile depuis son acceptation comme norme IEEE. Pratiquement HLA 1.3 et IEEE 1516 sont deux normes différentes et incompatibles.

MOM *Management Object Model.* Modèle objet de gestion : ensemble de services du RTI correspondant aux échanges ayant lieu entre RTI et fédérés, et destinés à un fédéré privilégié, gestionnaire de la fédération.

OMT *Object Model Template.* Modèle objet HLA : formalisme destiné à décrire les données d'un simulateur, ou les données susceptibles d'être échangées entre simulateurs.

ONERA *Office National d'Études et de Recherches Aérospatiales.*

RTI *Run-Time Infrastructure*. Infrastructure d'exécution. Le RTI désigne en particulier le noyau logiciel d'une simulation HLA, chargé de gérer la communication entre les simulateurs individuels.

SIMNET *Simulation Network*. Architecture de communication entre simulateurs créée dans les années 1980 par le DARPA.

SOM *Simulation Object Model*. Modèle objet d'un simulateur : description conforme à l'OMT HLA des données publiques d'un fédéré HLA.

Bibliographie

[Ade99] M. Adelantado. Experimenting the HLA framework for the ONERA Project Airport of the Future. In *Proceedings of the 1999 Fall Simulation Interoperability Workshop*, 1999.

[AR92] R. Ayani and H. Rajaei. Parallel simulation using conservative time windows. In *Proceedings of the 1992 Winter Simulation Conference*, pages 709–717, Arlington, Virginia, USA, 1992.

[AS00] M. Adelantado and P. Siron. Air-Ground Combat Simulation through the ONERA HLA Run-Time Infrastructure. In *Proceedings of the Advanced Simulation Technologies Conference*, April 2000.

[AS01] M. Adelantado and P. Siron. Multiresolution Modeling and Simulation of an Air-Ground Combat Application. In *Proceedings of the 2001 Spring Simulation Interoperability Workshop*, 2001.

[BAM82] K. R. Blashfield, S. M. Aldenderfer, and C. L. Morey. Cluster analysis software. In *Handbook of Statistics, vol 2.* North-Holland Publishing Company, 1982.

[BBSd98] A. Berrached, M. Beheshti, O. Sirisaengtaksin, and A. deKorvin. Alternative Approaches to Multicast Group Allocation in HLA Data Distribution. In *Proceedings of the 1998 Spring Simulation Interoperability Workshop*, 1998.

[BCSZ98] P. Bieber, J. Cazin, P. Siron, and G. Zanon. Security Extensions to ONERA HLA RTI Prototype. In *Proceedings of the 1998 Fall Simulation Interoperability Workshop*, 1998.

[BL97] W. Braudaway and R. Little. The High-Level Architecture's Bridge Federate. In *Proceedings of the 1997 Fall Simulation Interoperability Workshop*, 1997.

[Bry77] R. E. Bryant. Simulation of packet communications architecture computer systems. Massachussets Institute of Technology, MIT-LTC-TR-188, 1977.

[BS02] B. Bréholée and P. Siron. CERTI : Evolutions of the ONERA RTI Prototype. In *Proceedings of the 2002 Fall Simulation Interoperability Workshop*, 2002.

[BS03a] B. Bréholée and P. Siron. Bridges in HLA Distributed Simulations. In *International Industrial Simulation Conference 2003*, 2003.

[BS03b] B. Bréholée and P. Siron. Design and Implementation of a HLA Inter-federation Bridge. In *Proceedings of the 2003 European Simulation Interoperability Workshop*, 2003.

[Can99] D. Canazzi. yaRTI, a Ada 95 HLA Run-Time Infrastructure. In *Proceedings of the International Conference on Reliable Software Technologies – "Ada-Europe'99"*, Santander, Spain, June 1999.

[CM79] K. M. Chandy and J. Misra. Distributed Simulation : A Case Study In Design and Verification of Distributed Programs. In *IEEE Transactions on Software Engineering*, 1979.

[CM81] K. M. Chandy and J. Misra. Asynchronous Distributed Simulation via a Sequence of Parallel Computations. *Communications of the ACM*, 24(11), April 1981.

[Coh97] Danny Cohen. DDM Scenarios. In *Proceedings of the 1997 Fall Simulation Interoperability Workshop*. Perceptronics, 1997.

[Com94] DIS Steering Commitee. *The DIS Vision, A Map to the Future of Distributed Simulation*. Institute for Simulation and Training, Orlando, Florida, 1994.

[CRI97] Renato Cerqueira, Noemi Rodriguez, and Roberto Ierusalimschy. Binding an interpreted language to CORBA. In *Anais do II Simpósio Brasileiro de Linguagens de Programação*, pages 23–36, Campinas, Belo Horizonte, 1997.

[CTG98] Wentong Cai, Stephen J. Turner, and Boon Ping Gan. Hierarchical Federations : An Architecture for Information Hiding. In *Proceedings of the 1998 Fall Simulation Interoperability Workshop*, 1998.

[DGD02] J. Dingel, D. Garlan, and C. Damon. Bridging the HLA : Problems and Solutions. In *Sixth IEEE International Workshop on Distributed Simulation And Real Time Applications*, 2002.

[DMA+98] C. Damon, R. Melton, R. Allen, E. Bigelow, J. Ivers, and D. Garlan. Formalizing a specification for analysis : the HLA ownership properties. Technical report, CMU-CS-98-149, Department of Computer Science, Carnegie Mellon University, 1998.

[DoD96] US DoD. *HLA Time Management Design Document v1.0*, August 1996.

[DoD98a] US DoD. *High Level Architecture Interface Specification Version 1.3*, 1998.

[DoD98b] US DoD. *High-Level Architecture Object Model Template Specification Version 1.3*, 1998.

[DoD98c] US DoD. *High-Level Architecture Rules Version 1.3*, 1998.

[Fuj90] R. Fujimoto. Parallel discrete event simulation. *Communications of the ACM*, 33(10), 1990.

[Fuj98] R. Fujimoto. Time management in the High Level Architecture, 1998.

[Gaf88] A. Gafni. Rollback mechanisms for optimistic distributed simulation systems. In *Proceedings of the SCS Multiconference on Distributed Simulation*, pages 61–67, 1988.

[HRC96] D. J. Van Hook, S. J. Rak, and J. O. Calvin. Approaches to RTI implementation of HLA Data Distribution Management services. septembre 1996.

[HW01] M. Hyett and R. Wuerfel. Implementation of the data distribution management services in the RTI-NG. In *Proceedings of the 2001 Fall Simulation Interoperability Workshop*, 2001.

[IdFC96] R. Ierusalimschy, L. H. de Figueiredo, and W. Celes. Lua - an extensible extension language. *Software : Practice and Experience*, 26(6) :635–652, 1996.

[IEE00a] IEEE. *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture – Federate Interface Specification, IEEE Std 1516.1-2000*, 2000.

[IEE00b] IEEE. *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture – Framework and Rules, IEEE Std 1516-2000*, 2000.

[IEE00c] IEEE. *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture – Object Model Template (OMT), IEEE Std 1516.2-2000*, 2000.

[IS01] J.-L. Igarza and C. Sautereau. Distribution, use and reuse : Questioning the cost effectiveness of re-using simulations with and without HLA. 2001.

[ISO98] ISO/IEC. *Programming Language C++ Standard 14882 :1998.* 1998.

[JD88] K. A. Jain and C. R. Dubes. *Algorithms for Clustering Data.* Prentice-Hall, 1988.

[Jef85] D. R. Jefferson. Virtual time. *ACM Transaction on Programming Languages and Systems*, 7(3) :405–425, july 1985.

[Kap03] A Kapolka. The Extensible Run-Time Infrastructure (XRTI) : An emerging middleware platform for interoperable networked virtual environments. In *Proceedings of the Lake Tahoe Workshop on Collaborative Virtual Reality and Visualization*, october 2003.

[KWD99] F. Kuhl, R. Weatherly, and J. Dahmann. *Creating Computer Simulation Systems – An Introduction to the High Level Architecture.* Prentice Hall, 1999.

[Lak98] T. Lake. Time Management Over Inter-Federation Bridges. In *Proceedings of the 1998 Fall Simulation Interoperability Workshop*, 1998.

[Lin97] J. Linn. *Generic Security Service Application Programming Interface.* Internet RFC 2078, January 1997.

[LK91] Averill M. Law and W. David Kelton. *Simulation Modeling and Analysis.* McGraw-Hill, 1991.

[LLL00] Gary M. Lightner, Robert Lutz, and Reed Little. The IEEE HLA standards : The evolution from U.S. DoD HLA v1.3. In *Proceedings of the 2000 Fall Simulation Interoperability Workshop*, 2000.

[Lub88] B. D. Lubachevsky. The bounded lag distributed discrete event simulation. In *Proceedings of the 1988 SCS Western Multiconference on Distributed Simulation*, pages 183–191, February 1988.

[Lub89] B. Lubachesvsky. Efficient Distributed Event Driven Simulations of Multiple Loop Networks. *Communications of the ACM*, 1989.

[Lut] R. Lutz. A comparison of HLA object modeling principles with traditional object-oriented modeling concepts.

[MBD99] K. L. Morse, L. Bic, and M. Dillencourt. Characterizing scenarios for DDM performance and benchmarking RTIs. In *Proceedings of the 1999 Spring Simulation Interoperability Workshop.* Science Applications International Corporation, 1999.

[MBT99] K. L. Morse, L. Bic, and K. Tsai. Multicast grouping for dynamic data distribution management. In *Proceedings of the 31st Society for Computer Simulation Conference (SCSC '99)*, 1999.

[MS97] K. L. Morse and J. S. Steinman. Data distribution management in the HLA : Multidimensional regions and physically correct

filtering. In *Proceedings of the 1997 Spring Simulation Interoperability Workshop*, 1997.

[NF94] David M. Nicol and Richard M. Fujimoto. Parallel simulation today. *Annals of Operations Research*, (53) :249–285, 1994.

[NRR03] A. Nédélec, V. Raulet, and V. Rodin. A new API to the HLA declaration and object management services for runtime modifications. In *Proceedings of the 2003 European Simulation Interoperability Workshop*, 2003.

[oDfAT95] Under Secretary of Defense for Acquisition and Technology. Department of Defense Modeling and Simulation Master Plan, DoD 5000.59-p, October 1995.

[OMG02] OMG. *Distributed Simulation Systems Specification, version 2.0*, November 2002.

[PHS00] Daniel J. Paterson, Erik S. Houghland, and Juan J. Sanmiguel. A Gateway/Middleware HLA Implementation and the extra Services that can be provided to the Simulation. 2000.

[Pop89] A. Pope. SIMNET Network Protocols, July 1989.

[Pul99] J. Mark Pullen. Reliable Multicast Network Transport for Distributed Virtual Simulation. In *Proceedings of the 3rd IEEE International Workshop on Distributed Interactive Simulation and Real-Time Applications*, 1999.

[Rau03] V. Raulet. *Prototypage interactif et collaboratif – Vers une architecture de communication pour une interactivité coopérante dynamique dans les environnements virtuels distribués.* PhD thesis, Université de Bretagne Occidentale, 2003.

[RUI+96] Noemi Rodriguez, Cristina Ururahy, Roberto Ierusalimschy, , and Renato Cerqueira. The use of interpreted languages for implementing parallel algorithms on distributed systems. In A. Mignotte L. Bougé P. Fraigniaud and Y. Robert, editors, *Proceedings of Euro-Par'96 Parallel Processing*, volume 1, pages 597–600, Lyon, France, August 1996. Springer-Verlag, (LNCS 1123).

[SG96] M. Shaw and D. Garlan. *Software Architecture : Perspectives on an Emerging Discipline.* Prentice Hall, 1996.

[SHB00] T. Seiger, G. Holm, and U. Bergsten. Aggregation/disaggregation modeling in HLA-based multi resolution simulations. In *Proceedings of the 2000 Fall Simulation Interoperability Workshop*, 2000.

[Sir98] P. Siron. Design and Implementation of a HLA RTI Prototype at ONERA. In *Proceedings of the 1998 Fall Simulation Interoperability Workshop*, 1998.

[TBS+03] Patrice Torguet, Benoît Bréholée, Pierre Siron, Jean-Pierre Jessel, and Bernard Lecussan. An HLA distributed virtual reality application on a Linux cluster. Poster at CCGrid2003, May 2003.

[WW94] A. L. Wilson and R. M. Weatherly. The Aggregate Level Simulation Protocol : An Evolving System. In *Proceedings of the 1994 Winter Simulation Conference*, pages 781–787, 1994.

[ZPK00] Bernard P. Zeigler, Herbert Praehofer, and Tag Gon Kim. *Theory of Modeling and Simulation.* Academic Press, second edition, 2000.

Résumé

Cette thèse s'inscrit dans le contexte de HLA, une architecture de simulation distribuée dont l'objectif est de permettre l'interopérabilité et la réutilisation d'applications de simulation. L'ONERA a développé une plate-forme de simulation distribuée compatible HLA (appelée CERTI). L'objectif de cette thèse est d'étendre le travail entrepris sur CERTI en se basant sur l'introduction de la notion classique de domaine.

Nous étudions différentes techniques impliquant cette notion et dont les objectifs concernent l'interopérabilité, la sécurité, la réutilisation ainsi que l'amélioration des performances.

Notre contribution tient d'une part dans la réalisation de simulateurs-ponts permettant l'interconnexion de plusieurs simulations et d'autre part dans le développement de services de CERTI destinés à optimiser les échanges de données entre domaines.

Mots-clés Simulation distribuée, HLA, RTI, interopérabilité, réutilisation, extensibilité, interfédérations, fédérés-ponts, domaines.

Summary

Interconnection of HLA Distributed Simulations

This thesis focuses on HLA (High-Level Architecture), a distributed simulation architecture for simulation reuse and interoperability. A HLA-compliant distributed simulation platform (called CERTI) has been developed at ONERA. This thesis aims at extending the works involving CERTI by introducing the classic concept of domain.

We study several methods involving this notion and dealing with a wide range of purposes : interoperability, security, reuse and performance improvement.

Our main contribution is the design and implementation of bridge-simulators allowing the interconnection of several HLA-based simulations. We have also developed "Data Distribution Management" services for CERTI, whose purpose is to optimize inter-domain data transfers.

Keywords Distributed simulation, HLA, RTI, interoperability, reuse, scalability, interfederations, bridge-federates, domains.

***

10, avenue Édouard-Belin · BP 54032 · 31055 Toulouse cedex 4 · France
Tél : 33 (0)5 62 17 80 80 · Fax : 33 0(5) 62 17 83 30 · http://www.supaero.fr
Établissement public à caractère administratif sous tutelle du Ministère de la Défense