

# NOTE TO USERS

This reproduction is the best copy available.

**UMI<sup>®</sup>**



ÉCOLE POLYTECHNIQUE DE MONTRÉAL

ARCHITECTURE RÉSEAU DYNAMIQUEMENT MODIFIABLE POUR  
L'UTILISATION DANS DES ENVIRONNEMENTS VIRTUELS DE  
COLLABORATION

ISSAM KHALIL  
DÉPARTEMENT DE GÉNIE INFORMATIQUE  
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION  
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES  
(GÉNIE INFORMATIQUE)  
NOVEMBRE 2004



Library and  
Archives Canada

Bibliothèque et  
Archives Canada

Published Heritage  
Branch

Direction du  
Patrimoine de l'édition

395 Wellington Street  
Ottawa ON K1A 0N4  
Canada

395, rue Wellington  
Ottawa ON K1A 0N4  
Canada

*Your file    Votre référence*

*ISBN: 0-494-01344-3*

*Our file    Notre référence*

*ISBN: 0-494-01344-3*

#### NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

#### AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

---

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

ARCHITECTURE RÉSEAU DYNAMIQUEMENT MODIFIABLE POUR  
L'UTILISATION DANS DES ENVIRONNEMENTS VIRTUELS DE  
COLLABORATION

présenté par: KHALIL Issam

en vue de l'obtention du diplôme de: Maîtrise ès science appliquées

a été dûment accepté par le jury d'examen constitué de:

Mme. CHERIET Farida, Ph.D., présidente

M. OZELL Benoit, Ph.D., membre et directeur de recherche

M. BILODEAU Guillaume-Alexandre, Ph.D., membre

## REMERCIEMENTS

Ce mémoire n'aurait pas été possible sans le soutien et l'aide de plusieurs personnes.

Tout d'abord, je voudrais remercier mon directeur de recherche, Benoît Ozell, pour m'avoir donné l'opportunité de travailler avec lui, pour m'avoir guidé et soutenu dans le choix de ma recherche et pour l'aide financière qui me fut apportée. Je voudrais aussi remercier Chantal Pic pour son soutien et son aide pour la correction de ce rapport, sachant tous les maux de tête que ça a dû engendrer ! Il aurait été difficile de passer au travers de ces années d'études supérieures sans mes nombreux amis qui m'ont supporté lors des périodes difficiles, et qui étaient tous là pour célébrer les bons moments. Je voudrais remercier tout particulièrement pour cela François Chardavoine et Nicolas Belhoste.

Enfin, je voudrais remercier de tout mon coeur les personnes sans qui rien n'aurait été possible et qui m'ont toujours apporté leur soutien et leur amour : mon père Ahmed Khalil, ma mère Yamama Khalil et mes frères Hussein et Hayssam Khalil. Je ne pourrai jamais assez vous remercier.

## RÉSUMÉ

Les environnements virtuels de collaboration (EVC) permettent aux participants situés dans des lieux géographiquement distants de collaborer et travailler dans un monde virtuel en leur donnant l'illusion d'être tous physiquement présents au même endroit. Plusieurs EVC furent développés, permettant des collaborations dans des domaines variés tels que des simulations militaires ou des environnements de chirurgies virtuelles.

L'objectif principal de ce mémoire est de définir exactement ce qu'est un EVC, quel rôle doit remplir un tel environnement, quelles méthodes et techniques peuvent être utilisées pour remplir ce rôle et d'élaborer un concept de développement d'un nouvel EVC.

Nous étudions d'abord le concept même d'un EVC et ce que cela entraîne pour le développement d'un EVC. Nous proposons ensuite une nouvelle méthode de conception d'un tel environnement, après être arrivé à la conclusion que les méthodes actuellement utilisées ne permettent pas de créer un environnement remplissant réellement le rôle d'un EVC.

Notre méthode consiste à supposer qu'un EVC doit donc être vu comme un système de gestion d'applications, ceci étant la seule manière qu'un tel environnement puisse remplir son rôle. Pour cela nous proposons une séparation en trois couches : la couche réseautique, le gestionnaire et l'ensemble des applications. Nous nous concentrons ensuite sur la couche réseautique, et démontrons la nécessité de devoir mettre en place un système d'adaptation dynamique de la topologie du réseau, en montrant qu'aucune topologie ne résout tous les problèmes de communications d'un EVC, mais que pour chaque type de problème, il existe au moins une topologie offrant de bons résultats. La conception d'un prototype nous permet ensuite de démontrer d'une manière pratique les gains obtenus par l'utilisation d'une telle méthode.

## ABSTRACT

Collaborative virtual environments (CVE) allow geographically distant users to collaborate and work in a virtual environment by giving them the illusion of being all physically present in the same place. Many CVEs have been developed, allowing for collaborations in various areas, such as military simulations or virtual surgery.

This masters' main objective is to precisely define the nature and purpose of a CVE, as well as the methods and techniques that are used to fulfill that purpose, in order to then elaborate on a new development method for a CVE.

We first study the concept of a CVE and what it entails for its development. We then propose a new method for conceiving such an environment, after reaching the conclusion that the current methods are not satisfactory to build an environment that truly fulfills the purpose of a CVE.

Our method assumes that a CVE must be seen as an application management system, as this is the only way that such an environment can fulfill its purpose. To that end, we propose a 3-layer separation : networking, management and applications. We then concentrate on the network layer and demonstrate the need to have a dynamic network topology, as we will show that no single topology can solve all communication problems related to CVEs, but that for each type of problem there exists at least one optimal topology. A developmental prototype allows us to demonstrate in a practical way the gains that can be had by using such a method.

## TABLE DES MATIÈRES

|                                                                                |      |
|--------------------------------------------------------------------------------|------|
| REMERCIEMENTS . . . . .                                                        | iv   |
| RÉSUMÉ . . . . .                                                               | v    |
| ABSTRACT . . . . .                                                             | vi   |
| TABLE DES MATIÈRES . . . . .                                                   | vii  |
| LISTE DES TABLEAUX . . . . .                                                   | xi   |
| LISTE DES FIGURES . . . . .                                                    | xiii |
| LISTE DES SIGLES ET ABRÉVIATIONS . . . . .                                     | xv   |
| LISTE DES ANNEXES . . . . .                                                    | xvi  |
| CHAPITRE 1      INTRODUCTION . . . . .                                         | 1    |
| 1.1    Les environnements virtuels de collaboration (EVC) . . . . .            | 1    |
| 1.2    Présentation des objectifs du mémoire . . . . .                         | 2    |
| 1.3    Environnements virtuels de collaboration . . . . .                      | 2    |
| 1.3.1    Caractéristiques et composantes d'un EVC . . . . .                    | 3    |
| 1.3.2    Connaissances requises . . . . .                                      | 3    |
| 1.3.3    Élaboration du rôle d'un EVC (concepts d'espace et de lieu) . . . . . | 5    |
| 1.3.3.1    Concept d'espace . . . . .                                          | 6    |
| 1.3.3.2    Concept de lieu . . . . .                                           | 8    |
| 1.3.4    Applications distinctes . . . . .                                     | 11   |
| 1.4    Résumé du mémoire . . . . .                                             | 14   |

|            |                                                                                            |    |
|------------|--------------------------------------------------------------------------------------------|----|
| CHAPITRE 2 | REVUE DE LA LITTÉRATURE ET DES TRAVAUX RELIÉS                                              | 17 |
| 2.1        | Description des topologies réseautiques de base . . . . .                                  | 17 |
| 2.1.1      | Point à point simple . . . . .                                                             | 17 |
| 2.1.2      | Topologie cyclique . . . . .                                                               | 18 |
| 2.1.3      | Serveur-clients élémentaire . . . . .                                                      | 19 |
| 2.1.4      | Serveur-clients avec serveurs distribués . . . . .                                         | 19 |
| 2.1.5      | Serveur-clients avec serveur principal . . . . .                                           | 20 |
| 2.1.6      | Système BitTorrent . . . . .                                                               | 21 |
| 2.2        | Six défis majeurs . . . . .                                                                | 22 |
| 2.2.1      | Bande passante . . . . .                                                                   | 22 |
| 2.2.2      | Diversité . . . . .                                                                        | 23 |
| 2.2.3      | Architecture distribuée et interaction . . . . .                                           | 25 |
| 2.2.4      | Design d'un système temps réel . . . . .                                                   | 26 |
| 2.2.5      | Gestions des ratés . . . . .                                                               | 26 |
| 2.2.6      | Mise à l'échelle . . . . .                                                                 | 27 |
| 2.3        | Environnements virtuels de collaboration actuels . . . . .                                 | 28 |
| 2.3.1      | DIVE . . . . .                                                                             | 30 |
| 2.3.2      | MASSIVE . . . . .                                                                          | 32 |
| 2.3.3      | NPSNET-V . . . . .                                                                         | 34 |
| 2.3.4      | Jeux commerciaux . . . . .                                                                 | 35 |
| 2.4        | Limite et hypothèse . . . . .                                                              | 37 |
| 2.4.1      | Décomposition d'un EVC en trois couches (réseau - gestionnaire<br>- application) . . . . . | 39 |
| 2.4.2      | Concept d'adaptation dynamique de la couche de communica-<br>tion par réseau . . . . .     | 40 |
| 2.4.3      | Résumé de l'hypothèse . . . . .                                                            | 41 |
| CHAPITRE 3 | LA COUCHE DE COMMUNICATION PAR RÉSEAU ET                                                   |    |

|                                                                  |    |
|------------------------------------------------------------------|----|
| PROTOCOLES EXPÉRIMENTAUX . . . . .                               | 43 |
| 3.1 Rôle de la couche . . . . .                                  | 43 |
| 3.2 Information sur les données . . . . .                        | 45 |
| 3.2.1 Catégorisation des données à transmettre . . . . .         | 46 |
| 3.2.1.1 Type d'information du point de vue réseautique . . . .   | 47 |
| 3.2.1.2 Type d'information du point de vue de l'application .    | 49 |
| 3.2.2 Liens entre les types d'informations . . . . .             | 51 |
| 3.3 Valeurs à mesurer . . . . .                                  | 52 |
| 3.3.1 Délai . . . . .                                            | 52 |
| 3.3.2 Délai intermédiaire . . . . .                              | 52 |
| 3.3.3 Variation du délai . . . . .                               | 54 |
| 3.3.4 Perte de données . . . . .                                 | 54 |
| 3.3.5 Bande passante . . . . .                                   | 55 |
| 3.3.6 Mise à l'échelle . . . . .                                 | 55 |
| 3.4 Description des tests et méthodes d'implémentation . . . . . | 56 |
| 3.4.1 Méthodes de test . . . . .                                 | 56 |
| 3.4.2 Parc informatique utilisé . . . . .                        | 59 |
| 3.4.3 Programme de test . . . . .                                | 60 |
| 3.4.3.1 Programme émetteur . . . . .                             | 62 |
| 3.4.3.2 Programme récepteur . . . . .                            | 63 |
| 3.4.3.3 Programme de redistribution ("proxy") . . . . .          | 64 |
| 3.4.4 Protocole de test . . . . .                                | 66 |
| CHAPITRE 4 CLASSIFICATION DES ARCHITECTURES . . . . .            | 69 |
| 4.1 Présentation des architectures . . . . .                     | 69 |
| 4.1.1 Point à point simple . . . . .                             | 69 |
| 4.1.2 Topologie cyclique . . . . .                               | 71 |
| 4.1.3 Serveur-clients élémentaire . . . . .                      | 72 |

|            |                                                                      |     |
|------------|----------------------------------------------------------------------|-----|
| 4.1.4      | Serveur-clients avec serveurs distribués . . . . .                   | 74  |
| 4.1.5      | Serveur-clients avec serveur principal . . . . .                     | 76  |
| 4.1.6      | Système BitTorrent . . . . .                                         | 77  |
| 4.2        | Classification des architectures selon les tests effectués . . . . . | 78  |
| CHAPITRE 5 | L'ADAPTATION DYNAMIQUE . . . . .                                     | 83  |
| 5.1        | Concept d'adaptation dynamique et problème créé . . . . .            | 83  |
| 5.2        | Méthode d'implémentation de changement d'architecture . . . . .      | 85  |
| 5.2.1      | Topologies duales . . . . .                                          | 85  |
| 5.2.2      | Décharge de la topologie . . . . .                                   | 87  |
| 5.3        | Méthode de sélection d'architecture . . . . .                        | 88  |
| 5.4        | Analyse de l'utilisation d'un prototype de base . . . . .            | 90  |
| CHAPITRE 6 | CONCLUSION . . . . .                                                 | 93  |
| 6.1        | Résumé . . . . .                                                     | 93  |
| 6.2        | Vérification de l'hypothèse . . . . .                                | 94  |
| 6.3        | Prochaines étapes . . . . .                                          | 95  |
| RÉFÉRENCES | . . . . .                                                            | 97  |
| ANNEXE     | . . . . .                                                            | 102 |

## LISTE DES TABLEAUX

|             |                                                                                                                     |    |
|-------------|---------------------------------------------------------------------------------------------------------------------|----|
| Tableau 3.1 | Information du point de vue réseautique . . . . .                                                                   | 49 |
| Tableau 3.2 | Lien entre les types de catégories . . . . .                                                                        | 51 |
| Tableau 3.3 | Exigence des différents critères de transmission . . . . .                                                          | 53 |
| Tableau 3.4 | Étape de fonctionnement pour le "proxy" . . . . .                                                                   | 65 |
| Tableau 3.5 | Catégorisation du nombre de machines . . . . .                                                                      | 66 |
| Tableau 4.1 | Évaluation topologie point à point simple ( <i>selon les mesures définies à la Figure 3.1</i> ) . . . . .           | 70 |
| Tableau 4.2 | Évaluation topologie cyclique ( <i>selon les mesures définies à la Figure 3.1</i> ) . . . . .                       | 72 |
| Tableau 4.3 | Évaluation serveur-clients élémentaire ( <i>selon les mesures définies à la Figure 3.1</i> ) . . . . .              | 73 |
| Tableau 4.4 | Évaluation serveur-clients avec serveurs distribués ( <i>selon les mesures définies à la Figure 3.1</i> ) . . . . . | 74 |
| Tableau 4.5 | Évaluation serveur-clients avec serveur principal ( <i>selon les mesures définies à la Figure 3.1</i> ) . . . . .   | 77 |
| Tableau 4.6 | Évaluation Bittorrent ( <i>selon les mesures définies à la Figure 3.1</i> ) . . . . .                               | 78 |
| Tableau 4.7 | Résumé évaluation : 1 à 5 machines ( <i>selon les mesures définies à la Figure 3.1</i> ) . . . . .                  | 79 |

|              |                                                                                                         |    |
|--------------|---------------------------------------------------------------------------------------------------------|----|
| Tableau 4.8  | Résumé d'évaluation : 5 à 50 machines ( <i>selon les mesures définies à la Figure 3.1</i> ) . . . . .   | 79 |
| Tableau 4.9  | Résumé d'évaluation : 50 à 100 machines ( <i>selon les mesures définies à la Figure 3.1</i> ) . . . . . | 80 |
| Tableau 4.10 | Catégorisation des architectures (1) . . . . .                                                          | 81 |
| Tableau 4.11 | Catégorisation des architectures (2) . . . . .                                                          | 82 |
| Tableau 5.1  | Paramètres permettant une prise de décision . . . . .                                                   | 90 |

## LISTE DES FIGURES

|             |                                                                                                    |    |
|-------------|----------------------------------------------------------------------------------------------------|----|
| Figure 2.1  | Point à point simple. . . . .                                                                      | 17 |
| Figure 2.2  | Topologie cyclique. . . . .                                                                        | 18 |
| Figure 2.3  | Topologie serveur-clients élémentaire. . . . .                                                     | 19 |
| Figure 2.4  | Topologie serveur-clients avec serveurs distribués . . . . .                                       | 20 |
| Figure 2.5  | Topologie serveur-clients avec serveur principal . . . . .                                         | 21 |
| Figure 2.6  | Topologie BitTorrent . . . . .                                                                     | 21 |
| Figure 2.7  | DIVE (Distributed Interactive Virtual Environments) . . . . .                                      | 31 |
| Figure 2.8  | MASSIVE (Model, Architecture and System for Spatial Interaction in Virtual Environments) . . . . . | 32 |
| Figure 2.9  | NPSNET-IV . . . . .                                                                                | 35 |
| Figure 2.10 | Dark Age of Camelot . . . . .                                                                      | 36 |
| Figure 2.11 | World of Warcraft . . . . .                                                                        | 37 |
| Figure 3.1  | Gradients de mesures . . . . .                                                                     | 68 |
| Figure 4.1  | Point à point simple. . . . .                                                                      | 70 |
| Figure 4.2  | Topologie cyclique. . . . .                                                                        | 71 |
| Figure 4.3  | Topologie serveur-clients élémentaire. . . . .                                                     | 73 |

|            |                                                             |    |
|------------|-------------------------------------------------------------|----|
| Figure 4.4 | Topologie serveur-clients avec serveurs distribués. . . . . | 75 |
| Figure 4.5 | Topologie serveur-clients avec serveur principal. . . . .   | 76 |
| Figure 4.6 | Topologie BitTorrent . . . . .                              | 77 |
| Figure 5.1 | Méthode topologies duales . . . . .                         | 86 |
| Figure 5.2 | Exemple de la méthode de décharge . . . . .                 | 88 |

## LISTE DES SIGLES ET ABRÉVIATIONS

|        |                                                |
|--------|------------------------------------------------|
| CVE    | Collaborative Virtual Environnement            |
| CERCA  | Centre de recherche en calcul appliqué         |
| CAVE   | Cave Automatic Virtual Environment             |
| CSCW   | Computer-Supported Cooperative Work            |
| MMORPG | Massively Multiplayer Online Role Playing Game |
| P2P    | Point à Point (Peer-to-Peer)                   |
| TCP    | Transmission Control Protocol                  |
| UDP    | User Datagram Protocol                         |

**LISTE DES ANNEXES**

|          |                              |     |
|----------|------------------------------|-----|
| ANNEXE I | PROTOCOLES RÉSEAUX . . . . . | 102 |
| I.1      | TCP . . . . .                | 102 |
| I.2      | UDP . . . . .                | 103 |

## CHAPITRE 1

### INTRODUCTION

#### 1.1 Les environnements virtuels de collaboration (EVC)

Tout travail nécessite la collaboration entre plusieurs personnes : que cette collaboration se fasse entre chercheurs, ou entre un vendeur et un client, le besoin de collaborer pour réaliser quelque chose est toujours présent. L'évolution de la technologie, de l'invention du téléphone jusqu'à celle de l'Internet, ou par l'utilisation de la vidéo-conférence, a permis une collaboration non liée à la présence physique des personnes dans le même lieu. Par contre, toutes ces technologies permettent la collaboration à distance seulement dans certains cas ; dans d'autres cas où la présence physique des personnes est nécessaire, ces technologies ne peuvent pas être utilisées.

L'utilisation d'environnements de collaboration a pour but d'augmenter le nombre de cas où cette présence physique peut être remplacée, de permettre aux usagers de se rencontrer dans un monde virtuel, physiquement réel ou non, leur donnant l'illusion d'être présents dans un même lieu et ainsi de pouvoir mieux collaborer. L'élaboration d'un tel environnement consiste à mettre à la disposition des usagers les toutes dernières techniques multimédia et a donc pour conséquence de nécessiter une connaissance approfondie de différents domaines (Pekkola *et al.* (2000), Bendford *et al.* (2001), M *et al.* (2000), Jeffay *et al.* (2001)).

## **1.2 Présentation des objectifs du mémoire**

L'objectif principal de ce mémoire est de permettre de définir ce qu'est un environnement virtuel de collaboration, du rôle que doit remplir un tel environnement, des méthodes et techniques pouvant être utilisées pour remplir ce rôle et de l'élaboration d'un concept de développement d'un nouvel EVC.

L'objectif secondaire est l'élaboration d'une nouvelle technique de communication réseautique pour ces types d'environnement, permettant de répondre aux différents besoins de communication et s'ajustant à l'évolution de ces besoins.

La réalisation de ces deux objectifs a pour but principal de permettre le développement futur d'un environnement virtuel de collaboration pouvant surpasser les environnements existants en offrant une véritable collaboration, non pas en fixant la manière dont cette collaboration doit se faire, mais en permettant aux utilisateurs de créer leur propre méthode de collaboration. Le développement d'un tel environnement est un objectif à long terme, ce mémoire permettra donc de poser les premières pierres de sa construction.

## **1.3 Environnements virtuels de collaboration**

Un environnement virtuel de collaboration (EVC) consiste en l'utilisation d'un système de réalité virtuelle mis en réseau et permettant ainsi à plusieurs personnes d'interagir entre elles quel que soit leur emplacement physique à travers le monde. Généralement, l'environnement virtuel utilisé vise à donner un sens de réalisme et une impression d'immersion, mais a pour but principal de permettre la collaboration selon le contexte d'utilisation par les usagers, pour des utilisations allant du domaine médical au domaine ludique (Bowers *et al.* (1996), Smith *et al.* (2000)). La littérature présente plusieurs environnements virtuels décrits comme des EVC, mais ayant des rôles et des utilisations

souvent différents. Une présentation des caractéristiques et composantes d'un EVC est donc nécessaire pour permettre une différenciation entre environnements.

### **1.3.1 Caractéristiques et composantes d'un EVC**

Selon plusieurs auteurs, quel que soit le domaine d'utilisation, les cinq caractéristiques suivantes sont les bases de tout EVC (Singhal et Zyda (1999), Bendford *et al.* (2001), Jeffay *et al.* (2001)).

Un sentiment d'espace partagé

Un sentiment de présence partagé

Une impression de temps commun

Une manière de communiquer

Une manière de partager

On trouve chez d'autres auteurs (Frécon *et al.* (1999), Singhal et Zyda (1999), Leung et Chen (2001), Bowers *et al.* (1996)) qu'un EVC est composé des quatre composantes suivantes, qui permettent l'implémentation des caractéristiques de base :

Moteur graphique et affichage

Interface de contrôle et de communication

Système de gestion du temps de calcul

Réseau informatique

### **1.3.2 Connaissances requises**

Les développeurs d'environnements virtuels de collaboration doivent posséder de nombreuses connaissances dans des domaines différents généralement considérés distants,

mais intimement reliés lors du design et du développement d'un EVC. En effet, un EVC est en fait une fusion d'un système distribué, d'une ou plusieurs applications graphiques et surtout aussi d'une application interactive. Dès le départ, la quantité de connaissances techniques nécessaires est très élevée, mais de plus un EVC doit généralement s'interfacer avec d'autres systèmes et services, ce qui nécessite encore d'autres connaissances dépendantes de l'utilisation de l'EVC, telle que la capacité d'utilisation de systèmes externes. Prenons l'exemple d'un EVC qui aurait pour but de remplacer un centre commercial. Deux types de système externe sont utilisés : il est nécessaire d'avoir une intégration avec une base de données pour gérer l'inventaire physique des produits vendus, en plus d'une interaction avec des systèmes de commerce et de transactions bancaires. Or, l'intégration d'un système avec une base de données n'est ni intuitif ni automatique, de même pour celle avec les systèmes de transactions bancaires.

Les connaissances requises par les développeurs se divisent en deux catégories : technique et sociale. Selon Deriggi *et al.* (1998), D. Franco (1999), il Ko et Choi (1999), Jeffay *et al.* (2001) et Zhang *et al.* (2001), la catégorie technique contient, comme son nom l'indique, tous les défis techniques relatifs à la création d'un EVC, allant de la connaissance réseautique jusqu'au design de système en temps réel, en passant par les techniques de gestion des défaillances. Selon Bowers *et al.* (1996), Kirner *et al.* (2001) et Harrison et Dourish (1996)), la catégorie sociale contient les défis relatifs à l'utilisation de l'EVC par les usagers, non pas du côté technique, mais du côté du choix du type d'interface nécessaire et du type et nombre d'interactions nécessaires pour que l'EVC remplisse son but.

Ces deux catégories sont reliées entre elles puisque tout choix fait du côté technique est bien évidemment dépendant du choix fait du côté social. Supposons qu'il fut décidé que, pour le bon fonctionnement de l'EVC, le participant puisse utiliser de nombreuses interfaces pour se connecter au monde virtuel, allant d'un ordinateur de bureau à l'utilisation

d'un Palm ou même peut-être d'un téléphone cellulaire. Dans ce cas, le choix d'une solution technique est dépendant de la vision du type d'interactions voulues. De même, tout choix fait du côté social est aussi relié au côté technique : nous ne pouvons pas fixer un but s'il est techniquement impossible de le réaliser. Dans l'exemple précédent, on peut permettre à l'utilisateur de se connecter à l'aide de différentes interfaces, mais il est utopique de vouloir que la sensation d'immersion et le nombre d'interactions possibles soient les mêmes entre une voûte d'immersion (CAVE) et un cellulaire.

### **1.3.3 Élaboration du rôle d'un EVC (concepts d'espace et de lieu)**

Nous avons décrit jusqu'ici les caractéristiques et composantes majeures d'un EVC que l'on retrouve dans la littérature. Par contre, nous n'avons pas encore expliqué ce qu'un environnement virtuel représente pour ses utilisateurs. Nous ne parlons pas ici de savoir si un environnement virtuel de collaboration sert à remplacer un centre commercial ou être un jeu interactif englobant des milliers de personnes, mais plutôt si l'EVC a été développé selon l'un des deux concepts suivants : espace ou lieu. Selon plusieurs auteurs (Leung et Chen (2001), M *et al.* (2000), Harrison et Dourish (1996), Bowers *et al.* (1996)), ces deux concepts, utilisés autant en architecture qu'en design urbain, servent à définir tout type d'environnement et cela même si l'environnement est virtuel, tel que pour un EVC.

Nous allons donner un aperçu général permettant de comprendre et de distinguer les deux concepts, mais surtout d'avoir une idée de leurs besoins en termes techniques. Le principe suivant donne déjà une idée de base de la distinction entre les deux concepts : "Space is the opportunity, place is the understood reality" (Harrison et Dourish (1996)).

### 1.3.3.1 Concept d'espace

Le concept d'espace, comme son nom l'indique, amène à l'interprétation que la création d'un environnement et son utilisation dépendent de son espace. Cet espace est la structure du monde représenté, c'est l'environnement tridimensionnel dans lequel des objets et des événements se situent et dans lequel ils ont des positions et directions relatives.

En fait, un EVC utilisant le concept d'espace est un EVC ayant un modèle d'interaction spatiale (Harrison et Dourish (1996)) : un système dans lequel la conscience des participants les uns envers les autres et leurs interactions sont obtenues au travers d'une représentation spatiale de leur présence dans le monde virtuel. En somme, c'est la représentation physique des usagers (avatar) dans un environnement partagé, où les positions et directions sont relatives à l'utilisateur, qui permet une collaboration. Pour cela, Bowers *et al.* (1996), Singhal et Zyda (1999) et Hartling *et al.* (2001) montrent que plusieurs caractéristiques du monde réel sont utilisées et peuvent être regroupées selon les types suivants :

**Orientation relationnelle** L'organisation spatiale du monde est la même pour tous. Les concepts de haut, bas, devant et derrière sont donc identiques quelle que soit la personne et quelle que soit sa position dans le monde. Cette orientation commune est un atout majeur pour notre capacité d'interprétation et de présentation de nos activités et comportements. En effet, une des raisons pour lesquelles nous faisons des gestes pour communiquer est que nous sommes sûrs que l'environnement dans lequel nous sommes est identique pour toutes les autres personnes. Lorsque par exemple, dans un cours, le professeur pointe au tableau, il n'a aucun doute que tous les élèves voient qu'il pointe au tableau puisque tous partagent le même environnement physique avec une même organisation spatiale. Il en est aussi ainsi lorsque nous parlons par référence : "la porte devant nous" ou "la molécule à gauche".

**Proximité et actions** Dans le monde réel, la distance est un facteur non négligeable sur notre comportement. Lorsque l'on agit, on est généralement à proximité de la cible de notre action. Si l'on aperçoit une personne avec qui l'on veut discuter, on s'approche d'abord d'elle afin d'être à proximité. De la même manière, on approche de soi un objet que l'on veut examiner. Lorsque l'on voit des gens discuter autour d'une table, il est évident pour nous qu'une autre personne se trouvant à une certaine distance de cette table ne fait pas partie de la conversation. La distance des objets et des personnes entre eux et par rapport à nous affecte nos actions et nos interprétations.

**Partitionnement** Comme la distance est un facteur non négligeable et que plus elle est grande, plus le nombre d'interactions possibles est faible, il est possible de partitionner l'espace en fonction de la distance. Les seuls usagers ayant une interaction sont en général ceux qui sont proches les uns des autres. De même, les seuls objets avec lesquels on peut interagir sont ceux autour de nous.

**Conscience des autres** Être conscient des autres permet d'organiser nos activités et surtout de mieux collaborer entre nous. Par exemple, on est au courant si la personne avec qui l'on travaille quitte la table temporairement, ce qui nous évite de continuer à parler dans le vide. On sait que la personne a quitté la table car on l'a vue la quitter, on est au courant de ce que les gens font autour de nous sans que l'on ait à nous le dire. De plus, il y a des signes d'activité d'autres personnes tout autour de nous, ce qui fait que l'on sait que le monde n'est pas désert.

Toutes ces caractéristiques nous permettent d'organiser nos comportements en nous donnant plusieurs indices critiques. En utilisant ce concept d'espace, les concepteurs d'environnements virtuels de collaborations essaient d'utiliser notre familiarité avec l'organisation spatiale de notre environnement physique. Selon Harrison et Dourish (1996), ils veulent ainsi utiliser la manière dont notre environnement physique structure et organise

nos activités et interactions. En créant virtuellement, par exemple, une salle de réunion, la plus semblable possible d'une vraie salle de réunion, ils créent un environnement qui va nous amener à tenir des réunions et donc à collaborer. En somme, ce qu'un environnement utilisant le concept d'espace essaie de faire, c'est organiser nos activités dans un contexte de collaboration. L'idée est que s'il est concevable d'imiter le plus possible l'organisation spatiale du monde réel, on aura les conditions nécessaires pour que les usagers aient des comportements, des actions et des interactions identiques à ceux qu'ils ont dans le monde réel. En d'autres mots : "spatially-organised systems will support spatially-managed behaviours" (Harrison et Dourish (1996)).

Or, le concept de lieu entre en contradiction avec cette affirmation.

### **1.3.3.2 Concept de lieu**

L'idée principale de ce concept, et la raison majeure de son existence, est que les caractéristiques utilisées par le concept d'espace ne sont pas suffisantes pour recréer nos comportements et activités dans un monde virtuel. La raison étant que, même dans le monde réel, ces caractéristiques n'amènent pas automatiquement le type de comportement désiré. Le fait que nous soyons dans une salle ayant toutes les caractéristiques d'une salle de réunion ne va pas nous amener obligatoirement à faire une réunion et/ou à collaborer. Nos comportements et nos actions ne sont pas dictés par notre environnement physique mais par notre sentiment d'existence du lieu et notre sentiment de présence dans ce lieu.

Un lieu est un espace physique imprégné d'un type de comportement approprié. Un exemple de Harrison et Dourish (1996) illustre parfaitement ce point :

"A conference hall and a theatre share many similar spatial features (such as lighting and orientation); and yet we rarely sing or dance when presenting conference papers,

and to do so would be regarded as at least slightly odd (or would need to be explained). We wouldn't describe this behavior as "out of space"; but it would certainly be "out of place"; and this feeling is so strong that we might try quite hard to interpret a song or a dance as part of a presentation, if faced with it suddenly. It is a sense of place, not space, which makes it appropriate to dance at a Grateful Dead concert, but not at a Cambridge college high table; to be naked in a bedroom, but not in the street; and to sit at our windows, peering out, rather than at other people's windows, peering in. Place, not space, frames appropriate behaviour."

Alors comment pouvons-nous créer, dans un monde virtuel, le concept de lieu ? Comment pouvons-nous transformer une représentation réaliste d'un environnement physique en un lieu de collaboration ? La réponse, comme nous allons le voir plus loin, est que nous ne pouvons pas le faire en tant que développeur (Kirner *et al.* (2001), Bowers *et al.* (1996), Harrison et Dourish (1996)). Un lieu ne se crée pas, c'est l'utilisation par les usagers de l'environnement qui détermine le type de lieu existant.

Prenons comme exemple la représentation virtuelle d'une salle contenant une table ronde et des chaises. Cet environnement physique représente différents lieux pour différentes personnes. Pour certains, c'est un lieu de travail où une réunion peut être organisée. Rien de mieux qu'une table ronde pour être face à face lors d'une discussion. Pour d'autres, c'est une salle de jeux parfaite pour une séance de jeux de cartes ou de jeux de société. Rien de mieux qu'une table ronde pour être face à face lors de ces types de jeux. Nous avons donc le même environnement physique, mais deux comportements possibles. Ce n'est pas l'espace physique qui a dicté le type de comportement, mais l'utilisation faite par les usagers de cet espace, leur transformation de cet environnement en un lieu de travail ou en un lieu de jeu. Par contre, c'est cet environnement qui a offert la possibilité d'existence de ce type de lieu.

La citation suivante illustre ce point : "Those of us who are concerned with buildings

tend to forget too easily that all the life and soul of a place, all of our experiences there, depend not simply on the physical environment, but on the pattern of events which we experienced there" (Alexander (1979)).

Jusqu'à présent nous avons donc l'impression que le concept de lieu est lié à celui d'espace. S'il est vrai qu'un espace physique n'est pas suffisant pour créer des comportements de collaboration ou tout autre comportement désiré, il est par contre nécessaire, pour les obtenir, de transformer cet espace en un lieu propre à chaque groupe d'utilisateurs. Cela provient du fait que nous lions ces concepts à notre monde physique et il est vrai que, dans notre monde tridimensionnel, tout lieu est situé dans un espace physique. Lorsque transposé dans un monde virtuel, l'expérience démontre que des lieux peuvent se créer sans aucune correspondance à un espace physique.

Prenons l'exemple des groupes de nouvelles ("newsgroups" sur Internet) : ceux-ci n'ont pas pour but de recréer un espace physique, mais seulement d'offrir un nouvel outil de communication. Or, une simple visite de différents groupes de nouvelles montre que quelque chose de spectaculaire s'est créé : des communautés d'utilisateurs. Ces communautés, en utilisant les mêmes outils, ont créé leur propre lieu d'interaction, un monde avec leurs règles propres où des discussions et collaborations de différents types ont lieu et cela sans aucune représentation d'un espace physique. Visiter un groupe de nouvelles portant sur les jeux-vidéos ou sur la physique montrent bien que nous avons affaire à deux lieux différents où les types de discussions et de collaborations suivent des règles différentes. Pour le premier, les discussions sont plus énergiques et les guerres de mots ("flames wars") sont courantes, généralement acceptées et encouragées, alors que pour le deuxième les discussions sont plus logiques, moins énergiques, plus scientifiques.

En somme, la création d'un lieu est due à l'adaptation, l'utilisation et l'appropriation d'outils techniques par une communauté d'utilisateurs. La représentation d'un environnement physique (concept d'espace) n'est donc qu'un outil de plus offert pour la création

d'un lieu de travail, mais c'est un outil qui n'est pas négligeable.

#### 1.3.4 Applications distinctes

Comme nous l'avons vu auparavant, un EVC est le mélange d'un système distribué, d'une application graphique et d'une application interactive (Bowers *et al.* (1996), Singhal et Zyda (1999)). Chacune de ces parties oblige certains fonctionnements et nécessite de surmonter plusieurs défis :

**Système distribué** Entraîne la nécessité de faire la gestion des ressources réseau, la mise en place d'une procédure permettant le fonctionnement continu lors des pertes de données, des erreurs et des pannes de réseau, en plus de la création d'une technique de communication ordonnée entre les différentes parties du système.

**Application graphique** Entraîne la nécessité de garder une fluidité de l'affichage (un nombre d'images par seconde constant), tout en étant un système en temps réel. Oblige donc une allocation contrôlée du temps de CPU entre le rendu et les autres tâches du système.

**Application interactive** Entraîne la nécessité d'offrir aux usagers une interaction en temps réel. En aucun temps ceux-ci ne doivent attendre pour obtenir un résultat d'une de leurs actions, que ce résultat soit graphique ou non (communication audio par exemple). Ceci entraîne le fait que l'EVC doit se comporter du point de vue de l'utilisateur comme une application locale, alors que par définition un EVC est un système distribué.

À la lecture de ces défis, il nous apparaît que le développement d'un EVC nécessite que pour toute prise de décision pour l'implémentation d'une de ces applications, nous devons tenir compte des répercussions sur les autres. Il est donc nécessaire en plus d'avoir

un certain équilibre des fonctionnalités de l'EVC.

Par exemple, supposons que pour l'implémentation d'un système distribué, on décide d'utiliser des agents mobiles pour la collecte d'informations sur chaque noeud du réseau. Or, pour minimiser la charge sur le réseau et donc avoir un système distribué très fonctionnel, on choisit d'utiliser des agents mobiles ayant une petite taille. Ceci est généralement compensé par une utilisation plus grande du CPU sur chaque noeud (reconstruction du code de l'agent, compilation, décompression et compression de l'information). Or, ceci risque d'entraîner une dégradation des performances des applications graphiques et interactives puisque la répartition du temps CPU entre ces applications vient d'être modifiée. Comme il est impossible de prévoir lors du développement de ces applications à quel moment un agent risque d'être sur un noeud, cela équivaut à considérer une utilisation aléatoire du CPU. Pour éviter cela, il est nécessaire de supposer que le pourcentage du CPU utilisé par l'agent l'est toujours, qu'un agent soit sur le noeud ou pas.

En plus de la difficulté de fusionner trois types d'applications distinctes en une, le développement d'un EVC est encore plus complexe du fait qu'il est nécessaire d'interagir avec différents services existants externes à l'EVC mais nécessaires tant à son fonctionnement que, et surtout, à son utilité. Par exemple, voici selon plusieurs auteurs (Leigh *et al.* (2001), Campbell et Coulson (1996), Kirner *et al.* (2001), Pekkola *et al.* (2000)) quelques applications externes pouvant être nécessaires :

**Système de base de données** Tel que vu plus haut, il se peut que, pour le bon fonctionnement d'un EVC, des interactions avec une base de données se fassent. Ces interactions ne sont pas reliées au fonctionnement technique de l'environnement virtuel, mais à son utilité. Dans le cas de l'exemple du centre commercial, la base de données de l'inventaire des produits physiques permet de donner une utilité à l'EVC, l'achat est réel. Sans cette interaction, l'environnement fonctionne en

ayant toutes les caractéristiques d'un EVC, mais n'a aucune utilité.

**Interactions avec des systèmes de transactions** Prenons l'exemple d'un EVC commercial, où l'utilisation d'un EVC est dépendant d'un paiement mensuel. Il est donc évident que pour remplir son rôle commercial, l'EVC doit vérifier si un utilisateur a payé avant de le laisser entrer dans l'environnement virtuel, ce qui se fait en interagissant avec un système de vérification d'utilisateur. Si de plus, une fois à l'intérieur, l'utilisateur peut acheter quelque chose, il est nécessaire que l'EVC puisse interagir avec des systèmes établis de commerce et de transactions monétaires et cela sans briser le sens de l'immersion de l'utilisateur (on ne peut ainsi lui dire de sortir de l'environnement et d'utiliser une interface web pour le paiement).

**Enregistrement d'événements en temps réel sur un système de stockage** Un exemple parfait de ce type d'interaction peut se voir dans les EVC commerciaux ludiques, les jeux de rôles à plusieurs sur Internet (MMORPG, "massively multiplayer online role playing game"). Ces jeux permettent à des milliers de personnes de jouer dans le même monde virtuel en même temps. Or tout jeu à plusieurs entraîne un sens de compétition et donc invariablement amène des problèmes de tricherie. Pour remédier à ce problème, toutes les actions et interactions faites par un utilisateur, en plus de ses biens virtuels, sont stockés par la compagnie responsable du jeu. Or, un système de stockage n'est pas distribué, mais fixe et on peut voir les difficultés pouvant se produire lorsqu'un système (l'EVC) essaie de stocker tous les événements dans un système statique.

Il nous apparaît donc que la construction d'un environnement virtuel de collaboration offre un défi majeur, qui est l'intégration et le bon fonctionnement de domaines différents dans une seule application. Ceci requiert, plus qu'une prouesse technique, une capacité à faire des choix basés sur le fonctionnement général de toutes les parties de l'EVC. Rien n'est plus vrai que la citation suivante : "Net-VE development is a difficult balancing act

of engineering trade offs" (Singhal et Zyda (1999)).

Actuellement, une personne désirant implémenter un EVC a deux choix (Bendford *et al.* (2001)) :

**Tout implémenter soi même :** Nécessite donc une expertise dans tous les domaines vus précédemment. Malheureusement, le niveau des EVC actuels étant la barrière d'entrée (comme nous le verrons plus loin), cela implique qu'il devient impossible d'utiliser cette option pour implémenter un EVC ayant les caractéristiques requises en un temps normal, à moins de posséder une équipe de travail considérable. Par contre, ironiquement, les meilleurs EVC sont ceux qui ont été développés en choisissant cette option.

**Utiliser des outils de développement d'EVC existants :** Il existe actuellement des EVC modifiables, telle que DIVE ou MASSIVE (Greenhalgh *et al.* (2000)), permettant de créer un EVC sans posséder d'immenses capacités techniques. En utilisant les outils offerts, sous la forme de scripts tcl/tk pour DIVE par exemple, il est possible d'ajouter ses propres modifications à ces EVC. Par contre, ces EVC ne sont modifiables qu'en surface et imposent donc leur vision sur la structure d'un EVC du côté technique, limitant ainsi considérablement le type de modifications possibles.

## 1.4 Résumé du mémoire

La première partie de ce mémoire consiste en une revue et analyse de la littérature et des travaux sur le sujet, permettant l'élaboration d'une hypothèse présentant une nouvelle méthode de conception d'un environnement virtuel de collaboration. La deuxième partie permet de démontrer la validité de notre hypothèse en montrant d'abord qu'elle répond théoriquement aux défis du développement d'un EVC, et ensuite que des tests et analyses

sur les différentes parties de l'hypothèse démontrent non seulement sa validité mais aussi sa nécessité. Les chapitres composants ce mémoire sont les suivants :

Le chapitre 2 présente une revue de la littérature et des travaux reliés. Premièrement, une définition d'un EVC est fournie, en plus de présenter un résumé des différentes caractéristiques et composantes d'un tel environnement tiré de plusieurs publications sur le sujet. Deuxièmement, une présentation de deux concepts du fonctionnement d'un EVC ayant chacun un impact majeur sur son développement est faite. Une description de différents environnement virtuels existants est ensuite fournie, en décrivant leur fonctionnement, mais aussi le concept utilisé, en plus du type de topologie réseau sur lequel chaque environnement est basé, pour après cela présenter différentes topologies réseautiques utilisables. Enfin, ce chapitre conclut par la présentation de notre hypothèse : la décomposition d'un EVC en trois couches et l'adaptation dynamique de la topologie réseau utilisé.

Le chapitre 3 présente la synthèse des différents défis reliés à la conception d'un EVC et identifie six catégories de défis majeurs. Pour chaque catégorie, l'utilité de notre hypothèse est démontrée pour la résolution de ces défis.

Le chapitre 4 présente le rôle de la couche réseautique, les différents types de données et d'informations que devra transmettre une telle couche de même que leurs critères de transmission. Le chapitre 5, quant à lui, présente nos méthodes d'évaluation des différentes topologies réseautiques de base, en détaillant les valeurs à mesurer et les méthodes de mesure de ces valeurs. Le chapitre 6 montre ensuite les valeurs obtenues pour chacune des topologies de base et montre clairement qu'aucune topologie ne peut combler, à elle seule, les besoins d'un environnement virtuel de collaboration. Le chapitre 7 se base sur les résultats précédents pour présenter le concept d'adaptation dynamique, en plus de montrer les différentes observations faites lors de notre utilisation d'un prototype de base permettant d'évaluer cette adaptation. Enfin, le chapitre 8 conclut et présente des

pistes de développements futurs.

## CHAPITRE 2

### REVUE DE LA LITTÉRATURE ET DES TRAVAUX RELIÉS

#### 2.1 Description des topologies réseautiques de base

Les topologies suivantes représentent les éléments de base avec lesquelles nous pouvons construire diverses autres topologies. Nos tests d'évaluation seront ainsi effectués en combinant ces topologies élémentaires.

##### 2.1.1 Point à point simple

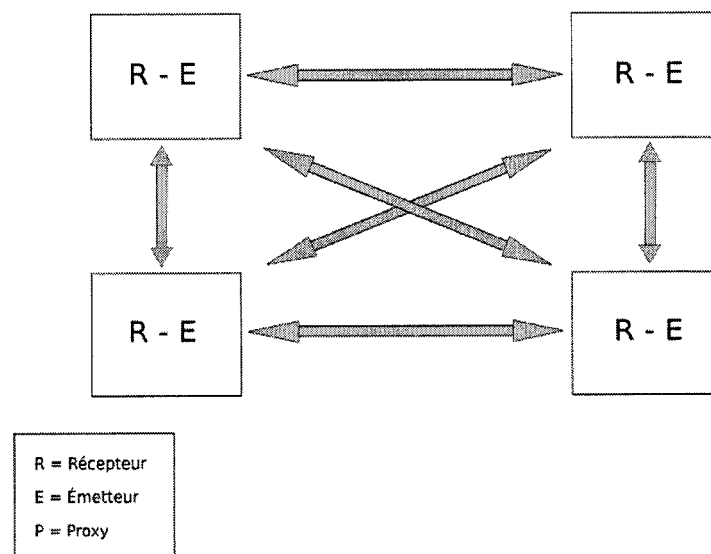


Figure 2.1 Point à point simple.

La figure 2.1 illustre la topologie point à point. Ce type d'architecture est la plus élémentaire possible, puisqu'elle correspond à une livraison directe du message entre émetteur

et récepteur, il n'y a aucun noeud, ni aucune manière d'améliorer la transmission. Ceci a pour effet que cette architecture n'est d'aucune utilité dès qu'un certain nombre de machines communiquent ensemble. Par contre, elle est très utile comme base d'évaluation pour les autres architectures.

### 2.1.2 Topologie cyclique

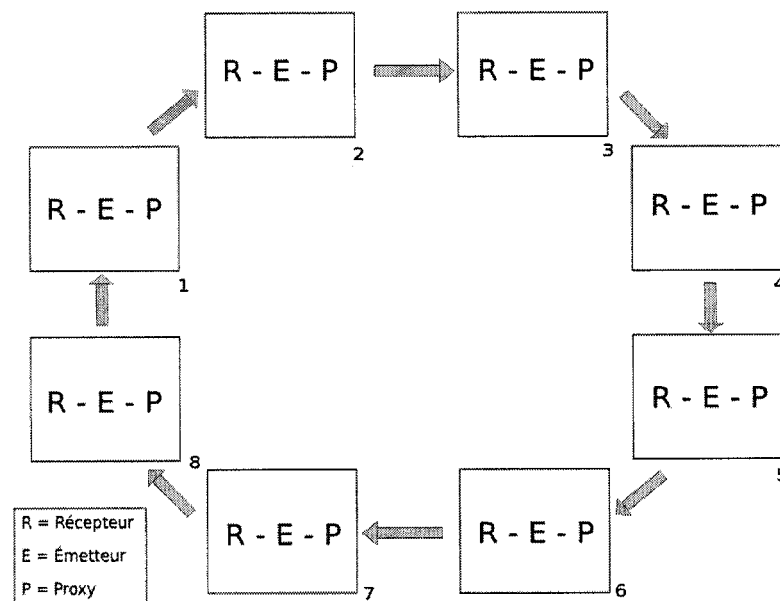


Figure 2.2 Topologie cyclique.

La figure 2.2 illustre le concept de la topologie cyclique. Contrairement à la topologie précédente, la communication ne se fait pas de manière directe mais est toujours considérée comme une communication P2P. Cette architecture consiste à faire un lien entre les machines en les mettant toutes en série les unes après les autres, ce qui signifie qu'un message entre deux machines doit transiter entre toutes les autres machines intermédiaires. Par exemple, dans la figure 2.2, un message devant être émis par la machine 2 et reçu par la machine 6 doit passer d'abord par les machines 3, 4 et 5.

### 2.1.3 Serveur-clients élémentaire

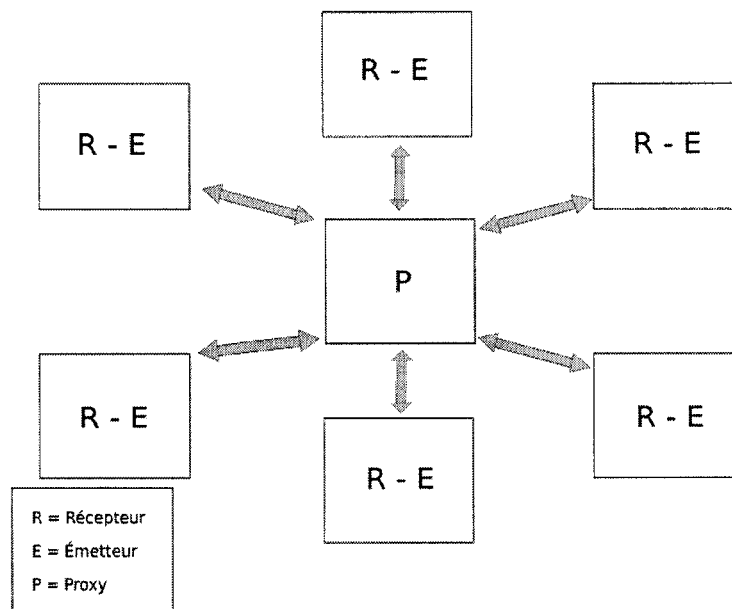


Figure 2.3 Topologie serveur-clients élémentaire.

La figure 2.3 illustre le concept de la topologie serveur-clients. Cette topologie est la plus souvent utilisée grâce à sa facilité de mise en place et son concept assez simple. Chaque message est envoyé à une machine, le serveur, et celui-ci a pour fonction de retransmettre ce message à toutes les autres machines, les clients.

### 2.1.4 Serveur-clients avec serveurs distribués

La figure 2.4 illustre le concept d'une des deux topologies serveur-clients hybride. Cette topologie essaie de diminuer les défauts du concept serveur-clients, en créant plusieurs serveurs reliés entre eux, les clients étant reliés au serveur leur offrant la meilleure connexion. Un message émis passe donc par trois étapes, premièrement il est envoyé au serveur le plus proche, ensuite il est distribué à tous les autres serveurs selon la topologie point à point simple, pour être finalement retransmis à tous les clients par les

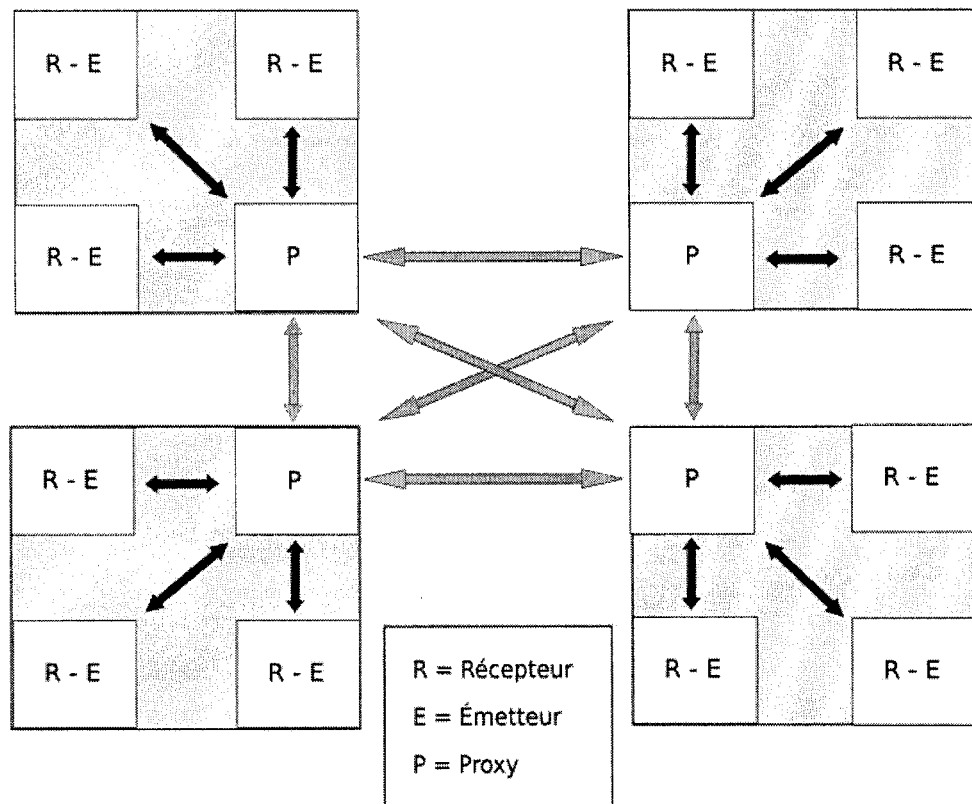


Figure 2.4 Topologie serveur-clients avec serveurs distribués

serveurs.

### 2.1.5 Serveur-clients avec serveur principal

La figure 2.5 illustre le concept de la deuxième topologie serveur-clients hybride qui peut être utilisée. Cette topologie est différente de la précédente dû au fait que la communication entre serveurs ne se fait plus d'une manière point à point, mais en utilisant un serveur (serveur principal). La raison majeure de l'utilisation d'une telle topologie est d'essayer de garder les caractéristiques précédentes, tout en permettant d'obtenir une technique de synchronisation des messages, ce que nous avons perdu avec l'architecture

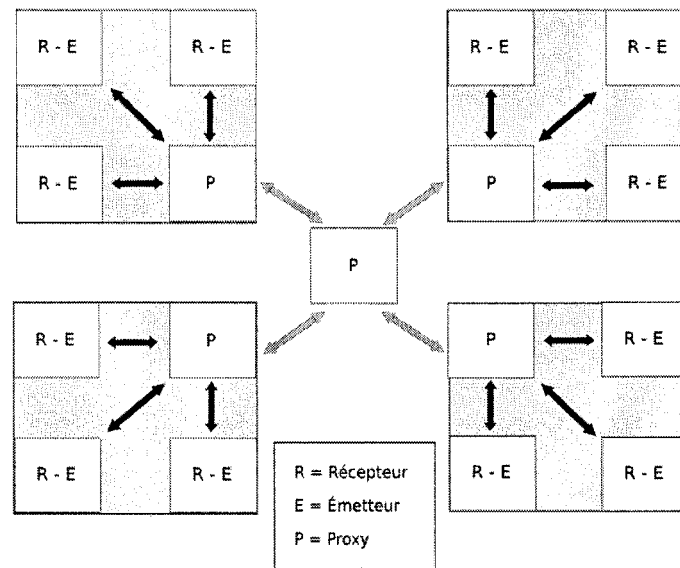


Figure 2.5 Topologie serveur-clients avec serveur principal

précédente.

### 2.1.6 Système BitTorrent

La figure 2.6 illustre le concept de la dernière topologie qui peut être mise en place. La capacité première d'une topologie utilisant le protocole Bittorrent est le fait que lorsque plusieurs clients téléchargent (download) le même fichier, ils se le téléchargeent (upload) aussi entre eux, amenant ainsi un partage de la bande passante nécessaire.

## 2.2 Six défis majeurs

Une analyse de la littérature nous a amené à classifier les problèmes reliés à la conception d'un EVC en six défis majeurs à résoudre. Pour chacun d'entre eux, nous verrons comment notre hypothèse constitue une solution.

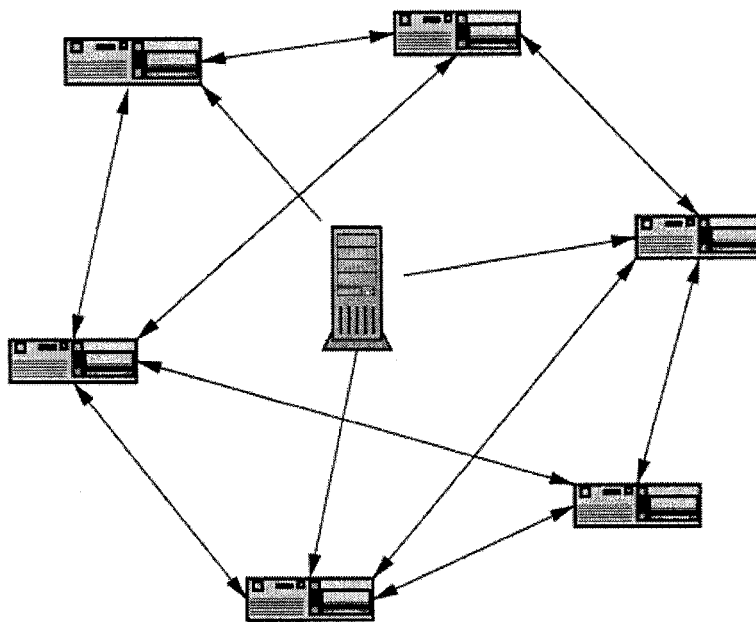


Figure 2.6 Topologie BitTorrent

### 2.2.1 Bande passante

Un environnement virtuel de collaboration est un système distribué, ce qui implique que la base du système est le réseau qui est utilisé pour la communication. Plusieurs défis sont reliés au fonctionnement du réseau, dont le type d'architecture à choisir et la mise à l'échelle du réseau, deux défis sur lesquels nous reviendrons, mais le défi de base est la maîtrise de la capacité du réseau, sa bande passante.

En effet, cette caractéristique du réseau est une ressource finie, elle ne peut être modifiée par le développeur de l'EVC. Cela nécessite donc d'avoir une gestion de cette ressource. Or, le premier défi provient du fait que, de par sa nature, toute action faite dans un EVC nécessite d'être communiquée à tous les participants. Il est donc nécessaire de limiter le nombre d'actions possibles en fonction de la bande passante disponible et de développer des techniques de communications nécessitant le moins de ressources possible (Haas *et al.* (2000), Sato *et al.* (2000), Chawathe *et al.* (2003)).

Le deuxième défi majeur provient de la distribution inégale des capacités réseaux entre participants (Pettifer *et al.* (2000), Park *et al.* (2000)). En effet, les premiers EVC furent développés dans des environnements universitaires et militaires, ce qui avait pour effet qu'ils étaient basés sur des réseaux locaux (LAN) où chaque participant possédait la même bande passante. Or, dès que l'on sort de ces environnements fermés, ceci n'est plus le cas. Même si on suppose une utilisation d'une connexion haute vitesse par tous les participants (supposition actuellement utopique), ceux-ci ont des fournisseurs de services différents, certains offrant des vitesses de téléchargement beaucoup plus rapides et des vitesses de télédéchargement beaucoup plus petites, tandis que d'autres font l'inverse. Si le premier défi consistait à utiliser le moins de bande passante et à gérer l'utilisation, le deuxième défi consiste à rendre les solutions du premier adaptables aux changements des capacités réseau par participant (Junginger et Lee (2004), Cui et Nahrstedt (2003), Kalnis *et al.* (2002)).

Notre hypothèse offre la possibilité de résoudre ce problème en séparant les défis : la couche réseautique ayant une capacité adaptative pour résoudre le problème du changement des capacités réseaux, et la couche de gestion s'occupant de s'assurer que ne soit allouée qu'une certaine quantité de bande passante aux applications de l'EVC.

### 2.2.2 Diversité

Nous venons de voir le problème lié à la distribution inégale des capacités réseau entre participants et du défi ainsi créé. Or, ce type d'inégalité n'est pas limité aux capacités réseau, mais s'applique aussi à tous les types d'équipements utilisés par les participants. Nous vivons dans un monde hétérogène et un des grands défis lors du développement d'un EVC est de définir une manière de fonctionner en tenant compte de cela.

Une des solutions rapides est d'obliger tous les participants à avoir le même type d'équi-

pements. Cela aura bien sûr pour effet de limiter énormément le nombre de participants, sauf si l'utilisation de l'EVC est pour un environnement fermé. "Visual eyes" (Smith *et al.* (2000)) est un EVC développé chez la compagnie GM pour relier ses groupes de recherche et développement distribués à travers le monde. Or, comme l'utilisation est limitée à cette compagnie, il est parfaitement possible que tous les usagers aient le même type d'équipements.

Une autre possibilité est de "cacher" la différence entre les participants, en réduisant l'EVC au plus petit dénominateur commun. Mais cela peut être désastreux si un des participants dispose d'un équipement beaucoup moins performant que la majorité des autres participants, ceux-ci étant obligés d'avoir à leur disposition des outils moins puissants pour collaborer à cause d'une seule personne. Généralement, cette technique est donc utilisée si les différences entre équipements ne sont pas trop grandes, mais justement, la nouvelle caractéristique prévue pour les EVC futurs est la possibilité de se connecter à partir de n'importe quel lieu et à l'aide du plus grand nombre d'équipements possible (Pekkola *et al.* (2000)). Il deviendrait ainsi possible de se connecter à l'aide de son ordinateur de poche ("handheld") alors que d'autres participants utilisent des environnements d'immersion (CAVE), ce qui empêche bien évidemment le choix de cette technique comme solution.

La dernière possibilité est donc d'adapter d'une manière dynamique le type d'interactions et d'actions disponibles aux participants en fonction du type d'équipement disponible. S'il est possible de le faire actuellement, le défi de cette solution provient du fait que nous devons l'implémenter tout en ne détruisant pas le concept de l'EVC. Supposons que l'environnement virtuel est utilisé par des scientifiques pour analyser une molécule, il est nécessaire de distinguer le type d'information qui peut être réduit (avatar par exemple) du type d'information nécessaire à toute bonne collaboration (l'affichage de la molécule dans ce cas). Un autre exemple est lorsqu'un EVC est utilisé pour un

jeu ou un entraînement entre 2 équipes. Si l'EVC, pour accommoder l'utilisateur ayant un système d'affichage moins puissant, décide de ne pas afficher certains éléments de l'environnement, telle que la végétation ou des habitations, alors ironiquement celui-ci sera avantagé puisque sa vision sera moins gênée, ce qui rend le jeu ou l'entraînement inutile.

### **2.2.3 Architecture distribuée et interaction**

Un EVC doit permettre à ses usagers d'interagir en temps réel pour permettre une collaboration et un EVC est une application distribuée avec des participants situés à des endroits géographiques différents. Permettre une interaction en temps réel dans ce cas-là est un autre défi majeur (Park *et al.* (2000)).

Cela peut être vu comme donner l'illusion à chaque participant que l'environnement est situé sur sa machine et que ses actions ont un effet direct. Bien entendu, nous parlons d'illusion puisqu'il est actuellement impossible que les actions aient un effet direct, le délai de transmission dû au réseau étant non négligeable. Si différents types d'architectures permettent d'avoir des délais différents selon le type d'actions et donc le type de communication nécessaire, aucun ne les éliminent. Il est donc nécessaire de tenir compte de ce délai, ce qui est généralement possible en interpolant le résultat jusqu'à ce qu'une confirmation arrive.

Actuellement, ce défi est immense et aucune solution générale n'a encore été trouvée. Encore une fois, la solution est dépendante du type de réseaux utilisé et devra être adaptative dans le cas d'un EVC hétérogène, ce que notre hypothèse offre.

#### **2.2.4 Design d'un système temps réel**

Comme nous l'avons vu à plusieurs reprises, un EVC est une application fonctionnant en temps réel. S'il est évident que faire fonctionner un système en temps réel est un problème analysé depuis longtemps et que plusieurs techniques et solutions fonctionnelles existent déjà, le cas d'un EVC peut être vu comme un problème extrême.

En effet, nous avons vu qu'un environnement virtuel de collaboration est constitué de plusieurs applications, chacune ayant des problèmes et solutions spécifiques, mais toutes ayant un point commun : l'obligation de fonctionner en temps réel. Le nombre de processus utilisé fait qu'il est nécessaire, si ce n'est obligatoire, d'employer les techniques les plus récentes pour le développement et le fonctionnement d'un système temps réel.

#### **2.2.5 Gestions des ratés**

Que ce soit volontaire ou pas, il peut arriver qu'un noeud "disparaisse" du réseau, en somme, que pour une raison ou une autre, une machine quitte brusquement l'EVC. Selon le type d'architecture réseau utilisé, cela peut avoir un effet dévastateur ou anodin. Si, par exemple, l'EVC fonctionne selon le modèle serveur-clients et que l'ordinateur qui a "disparu" est le serveur, alors le système arrête de fonctionner complètement, ce qui est bien évidemment mis dans la catégorie désastreuse. Par contre, si l'ordinateur n'était qu'un client, le système n'est en général pas affecté. Pour un système point à point, comme nous le verrons plus loin, cela n'est pas désastreux mais peut affecter le délai de communication pendant un certain temps et amener une perte d'information. Il est donc nécessaire que l'EVC puisse agir de la meilleure manière possible lorsque cela arrive, ce qui selon le type d'architecture amène des solutions différentes. Là encore nous avons à faire face à un défi nécessitant des solutions adaptatives dans le cas où plusieurs architectures réseau peuvent être utilisées.

### 2.2.6 Mise à l'échelle

La mise à l'échelle d'un EVC, tout en gardant son bon fonctionnement, est généralement considérée comme l'un des plus grands défis à résoudre lors de la conception d'un environnement virtuel (Singhal et Zyda (1999), Pekkola *et al.* (2000), Bendford *et al.* (2001), Greenhalgh *et al.* (2000), Hartling *et al.* (2001)).

Par mise à l'échelle nous voulons parler du comportement de l'EVC lorsqu'une ou plusieurs des variables de l'environnement changent d'ordre de grandeur, par exemple lorsque le nombre d'utilisateurs de l'EVC passe d'une centaine à des milliers. Présentement, le problème le plus important se présente lors d'un changement majeur du nombre d'utilisateurs, aucun des EVC actuels n'ayant un bon mécanisme de mise à l'échelle pour supporter un tel changement. La raison en est simple : les ressources disponibles pour le fonctionnement de l'EVC sont finies. Une de ces ressources les plus affectées est la capacité réseau du système.

En effet, plus le nombre d'utilisateurs augmente, plus le nombre de communications réseautiques nécessaires, dues aux actions des utilisateurs, augmente de manière exponentielle, chaque action d'un participant devant être transmise à tous les autres participants. Le type d'architecture réseau choisi affecte la capacité de résistance à ce changement, mais aucun n'offre la possibilité de bien fonctionner lors d'un changement massif du nombre d'utilisateurs.

Une autre des ressources finies affectant la capacité de mise à l'échelle du nombre d'utilisateurs est la puissance de calcul des machines de chacun des clients du système. Même si les capacités du réseau étaient infinies, il faudrait que chaque action d'un participant soit analysée pour produire l'effet désiré, or il est évident que plus le nombre d'utilisateurs augmente, plus le temps d'analyse d'une action pour définir comment elle affecte les autres participants augmente. Dès lors, selon la puissance de calcul disponible, à partir

d'un certain nombre d'utilisateurs le système ne peut plus être considéré en temps réel, ce qui par définition veut dire que l'EVC n'est plus fonctionnel.

Une des manières les plus utilisées actuellement pour contrer ce problème de mise à l'échelle du nombre d'utilisateurs est la partition du monde virtuel. Le principe général est le suivant : création de zones dans le monde virtuel, les participants dans une zone ne pouvant affecter ceux dans une autre, tout en offrant la possibilité aux utilisateurs de changer de zone. En général, ces zones ne sont qu'une partie de l'espace de l'environnement virtuel et notre position dans cet espace définit la zone dans laquelle nous sommes. Si cette technique de partition permet de résister à l'augmentation du nombre d'utilisateurs, elle n'élimine pas le problème puisque celui-ci est ramené dans chaque zone plutôt que dans le monde au complet (Bendford *et al.* (2001), Frécon *et al.* (1999), Bowers *et al.* (1996)).

Ce problème de mise à l'échelle n'est pas seulement relié au nombre d'utilisateurs, mais aussi à plusieurs autres facteurs. Par exemple, la taille du monde virtuel et le nombre de détails disponibles dans ce monde sont d'autres facteurs affectant les EVC, ainsi que le nombre d'actions possibles offertes aux utilisateurs. Mais le problème reste le même : les ressources étant finies, chaque augmentation des demandes de ressources, due à une augmentation de certains paramètres de l'environnement virtuel, risque de rendre l'EVC inutilisable.

### **2.3 Environnements virtuels de collaboration actuels**

Les environnements virtuels de collaboration peuvent être divisés en trois catégories : académique, militaire et commerciale.

La catégorie académique correspond aux EVC en développement dans les institutions

académiques et qui ont surtout pour but principal d'être des environnements d'expérimentation. Ces environnements n'ont en général pas d'utilisation pratique, ne fonctionnent généralement que sur des réseaux à haute vitesse et n'offrent pas des environnements très beaux. Par contre, ce sont sur eux que l'on retrouve toutes les dernières techniques développées et publiées pour la création d'EVC, et en général ils offrent des outils permettant d'analyser les comportements des usagers du système. De plus, cette catégorie correspond, et de loin, au plus grand nombre d'EVC que nous avons observé dans la littérature

La catégorie militaire correspond aux EVC servant généralement dans les différents échelons militaires pour diverses simulations, allant de l'entraînement des soldats dans des environnements urbains aux simulations de guerre internationale. Ces EVC fonctionnent en général en offrant un très grand nombre de simulations, comme par exemple la simulation d'une armée au complet ou celle d'un bataillon dans une ville, et ont pour but d'offrir aux participants des oppositions réalistes.

La catégorie commerciale correspond aux EVC actuellement sur le marché et qui offrent juste une utilité, que ce soit la visualisation d'une auto par des ingénieurs d'une compagnie internationale, ou juste un but de divertissement. Ces EVC se concentrent par dessus tout à offrir juste ce qu'ils ont besoin pour remplir leur fonction, et en général n'utilisent que des techniques de conception ayant été prouvées fonctionnelles.

Voici quelques noms et références d'EVC que nous avons observés (selon leur catégorie) :

**Académique :**

- DIVE (Carlsson et Hagsand (1993))
- MASSIVE (Greenhalgh *et al.* (2000))
- BrickNet (Singh *et al.* (1994))

- PARADISE (<http://www.dsg.stanford.edu/paradise.html>)
- AVIARY (Snowdon et West (1994))
- Avocado (Tramberend (1999))
- Bamboo (Watsen et Zyda (1998))
- NICE (Johnson *et al.* (1998))

#### **Commerciale :**

- Visual eyes (Leigh *et al.* (1999))
- World of warcraft (<http://www.worldofwarcraft.com>)
- Dark Age of Camelot (<http://www.darkageofcamelot.com>)

#### **Militaire :**

- SIMNET (Calvin *et al.* (1993))
- NPSNET

Étant donnée le nombre très élevé d'EVC, nous allons présenter en détail seulement quatre d'entre eux qui représentent de façon générale l'ensemble des EVC observés.

### **2.3.1 DIVE**

DIVE (Distributed Interactive Virtual Environments) est un prototype développé à l'Institut d'informatique de Suède (SICS, Swedish Institute of Computer Science) qui permet à des participants de naviguer dans un monde 3D et d'interagir entre eux, et aussi d'utiliser des applications fonctionnant dans l'environnement virtuel. (Carlsson et Hagsand (1993), Frécon *et al.* (1999), Carlsson (1993))

DIVE permet l'utilisation de scripts tcl/tk pour permettre aux usagers de créer leur propre application et de définir les comportements que doivent avoir certains objets. Si ces scripts ne peuvent pas dévier de ce qui est permis par le système, ils offrent quand même la possibilité de modifier d'une certaine façon l'environnement virtuel, et sont donc une manière d'implémenter un concept de lieu. Par contre, en général, cet EVC représente

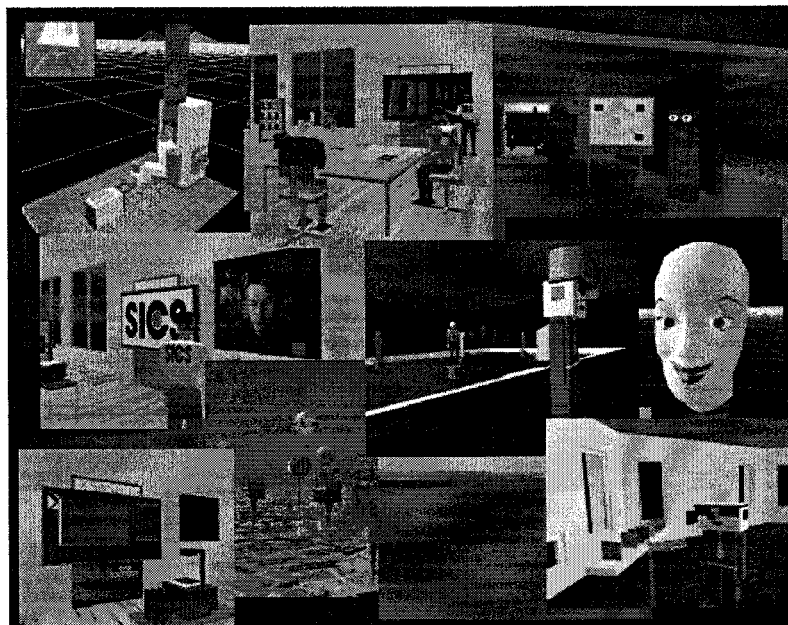


Figure 2.7 DIVE (Distributed Interactive Virtual Environments)

bien les environnements utilisant le concept d'espace.

Une base de données distribuée et partagée est implémentée pour la description de l'environnement. Pour diminuer la charge sur le réseau, cette base de données est répliquée partiellement sur les machines de chaque usager, permettant que les modifications soient faites sur la copie locale avant d'être propagées sur le réseau. Cette technique permet de diminuer le nombre de communications nécessaires pour la synchronisation des événements, ce qui diminue la charge du réseau et permet d'avoir un plus grand nombre de personnes en même temps dans l'environnement virtuel. Par contre, cette technique a le désavantage de créer une copie du monde virtuel qui n'est plus obligatoirement une copie exacte, même si cela est pour un court laps de temps.

Les communications réseaux se font en utilisant des protocoles de transfert multi-point ("multicast"). Pour les changements sur la base de données, une implémentation d'un système de multicast fiable, utilisant un système de confirmation basé sur le protocole

TCP, est utilisée. Pour tous les autres messages, le transfert par multicast normal est utilisé. Un arbre hiérarchique est utilisé pour la sélection du groupe multicast utilisé pour l'envoi de messages, avec au sommet, le groupe multicast utilisé pour l'envoi d'évènements affectant l'environnement virtuel au complet.

### 2.3.2 MASSIVE

MASSIVE (Model, Architecture and System for Spatial Interaction in Virtual Environments) est développé au département d'informatique de l'université de Nottingham, et est actuellement à sa troisième version (Greenhalgh *et al.* (2000)).

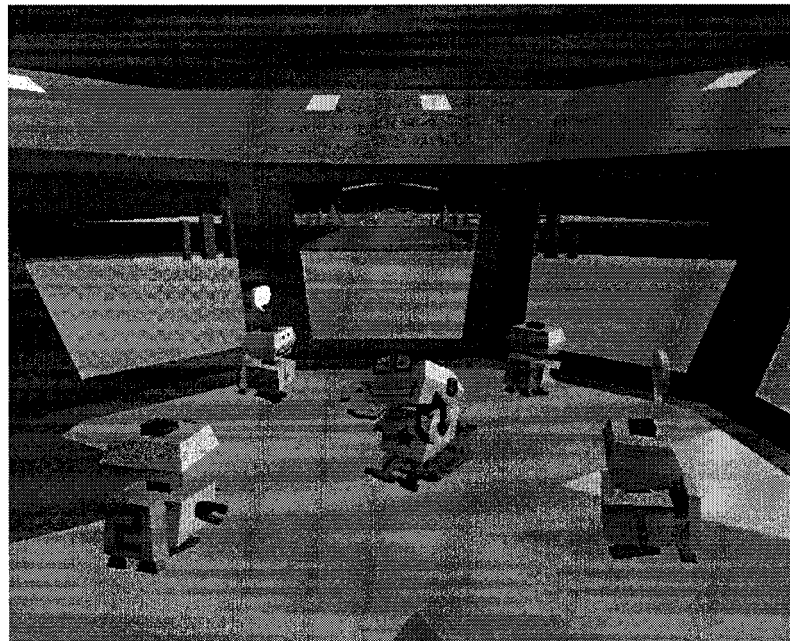


Figure 2.8 MASSIVE (Model, Architecture and System for Spatial Interaction in Virtual Environments)

Tout comme DIVE, une base de données distribuée est utilisée pour la description de l'environnement, par contre elle ne peut être modifiée localement. Chaque modification de la base de données est effectuée en créant un évènement approprié qui est propagé

dans le système. La priorité est donnée à la synchronisation de l'environnement plutôt qu'à la diminution de la quantité de bande passante utilisée. Si cela empêche d'avoir un grand nombre d'utilisateurs, il est possible en contrepartie de créer des environnements virtuels où la synchronisation est très importante.

L'objectif de diminution de l'utilisation de la bande passante est atteint en utilisant un concept de subdivision du monde virtuel, qui consiste à séparer le monde en plusieurs parties, appelées chacune un local, les participants situés dans un local ne pouvant communiquer qu'avec les autres participants situés dans le même local. En somme, la synchronisation du monde virtuel oblige la diminution du nombre de personnes pouvant interagir en même temps.

MASSIVE offre la possibilité aux utilisateurs de créer leur propre application en fournissant une API permettant d'avoir accès à plusieurs méthodes du système. Ici aussi, le développement de l'application est limité par ce qui est fourni, mais ouvre quand même la voie à l'utilisation du concept de lieu en permettant un certain contrôle sur l'environnement, par exemple en modifiant le fonctionnement d'objets dans le monde. Par contre la limitation de l'API implique que nous pouvons implémenter seulement les comportements que l'API peut supporter. Contrairement à l'EVC précédent, cet environnement utilise plus un mélange des deux concepts, lieu et espace.

La communication réseautique se fait en utilisant un système de multicast logique, qui consiste en fait en un système serveur-clients, mais qui simule le concept de multicast. La raison de l'utilisation d'un système serveur-clients est la possibilité de pouvoir mieux gérer la synchronisation de l'environnement et d'être sûr que cela fonctionnera sur tout l'Internet (la gestion du multicast n'étant pas encore étendue sur tous les routeurs). La congestion du serveur est gérée par l'utilisation de plusieurs serveurs, chacun s'occupant de certains clients, et communiquant entre eux en utilisant une topologie point à point.

### 2.3.3 NPSNET-V

NPSNET-V est développé au département d'informatique de l'École supérieure navale américaine. Cette nouvelle version est actuellement en développement, mais a pour but précis d'offrir une plateforme permettant la réalisation du "cyberspace" ou "métaverse" de la science fiction : un monde virtuel persistant qui n'est jamais éteint même pour une modification ou une maintenance. Les cinq critères suivants permettent d'arriver à ce but :

**Extensibilité** La possibilité d'étendre le système dynamique pendant qu'il est en marche pour introduire de nouvelles fonctionnalités d'une manière transparente pour l'utilisateur.

**Combinabilité** La possibilité de combiner les éléments de différentes applications et environnements d'une manière simple, avec des résultats logiques et intuitifs.

**Interopérabilité** La définition et l'utilisation de standards pour permettre à des systèmes hétérogènes d'interagir de manière fiable.

**Mise à l'échelle** La possibilité de grandir d'un simple système en un système plus complexe, et de supporter des environnements avec un nombre élevé d'entités et d'interactions.

**Sécurité** L'utilisation de mécanismes pour garantir la sécurité de la machine hôte lorsque des systèmes essaient d'intégrer des nouvelles fonctionnalités, maintenir la cohérence du monde virtuel en réponse à de possibles tentatives de corruption et protéger les informations partagées entre clients.

Le but du développement de cet environnement est donc d'incorporer pour la première fois le concept de lieu dans un environnement virtuel de collaboration. Même si cet environnement n'en est qu'à ses débuts, il est d'une certaine manière la confirmation de

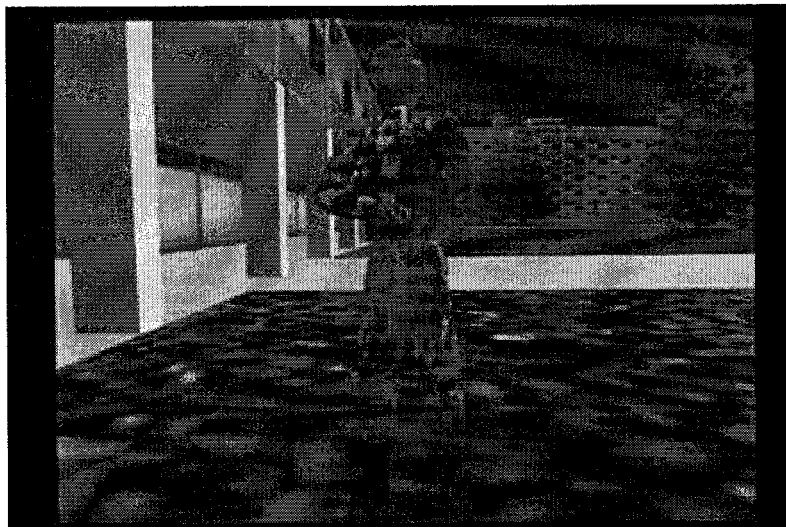


Figure 2.9 NPSNET-IV

l'idée qu'un nouvel EVC ne peut suivre le concept d'espace tel que tous les anciens EVC le font.

#### 2.3.4 Jeux commerciaux

Les rares EVC commerciaux actuels sont en grande majorité des jeux. Ces jeux englobent de nombreux joueurs et la collaboration consiste à leur permettre de s'amuser ensemble dans un monde virtuel en suivant les règles du jeu.

Plusieurs problèmes doivent être résolus pour la création de ce type de jeux, les deux plus importants étant la synchronisation de l'environnement entre les participants tout en gardant une interactivité logique et la gestion de la bande passante.

Prenons l'exemple d'un jeu de combat où les participants doivent échanger des coups de feu. Il est évident qu'un système de détection de collisions doit être implémenté dans ce jeu, ce qui est facile localement, mais très difficile dans un environnement distribué. Supposons qu'un des participants tire sur un deuxième : pour confirmer l'impact, le système



Figure 2.10 Dark Age of Camelot

doit savoir la position de la deuxième personne à tout moment lors du trajet de la balle. Pour limiter la capacité réseau nécessaire, la position du deuxième participant est reçue régulièrement, à un intervalle fixe de temps, et la position entre ces réceptions est interpolée. Il peut donc arriver que, lorsque le tir a lieu, l'interpolation de la position permette de conclure à un impact, ce qui est signalé à l'utilisateur. Toutefois, si la deuxième personne a brusquement modifié sa position, l'interpolation est erronée et pour le deuxième participant il n'y a pas eu d'impact ! En somme, pour garder l'illusion d'un système temps réel, une différence fut créée entre la perception du monde par les utilisateurs. Ceci est bien entendu inacceptable dans un EVC.

Les jeux actuels résolvent ce problème en se basant énormément sur des architectures de type serveur-clients, toute action étant synchronisée au niveau du serveur. Mais cela engendre bien entendu une plus grande lenteur et nécessite de plus que la compagnie responsable du jeu ait, pour chacun de ses serveurs, une quantité de bande passante adéquate pour supporter le nombre d'utilisateurs.



Figure 2.11 World of Warcraft

Les nouveaux jeux, tel que "World of Warcraft", essaient de diminuer justement cette dépendance envers les serveurs en utilisant d'autres types de topologies pour le transfert de données non pertinentes à la synchronisation de l'environnement, comme par exemple l'utilisation du système Bittorrent pour la distribution de fichiers.

## 2.4 Limite et hypothèse

Selon nous, une nouvelle manière de concevoir des EVC est nécessaire, puisqu'un EVC ne doit plus être vu comme une application, mais un environnement général offrant de nombreux outils. Un EVC doit pouvoir permettre aux usagers de faire des choses auxquelles le développeur n'a pas pensé, mais où les outils nécessaires leur ont été fournis. Tout ce que nous avons vu jusqu'à maintenant avait pour but de montrer la très grande difficulté de la tâche. En effet, en plus des défis majeurs, nous devons maintenant créer des outils qui permettraient aux usagers de combler leurs propres besoins, sans que

l'on sache quels sont ces besoins. Notre hypothèse sera que cela est en fait possible, mais qu'une modification de la méthode de conception d'un EVC doit être envisagée.

Actuellement, un EVC est vu comme une application, complexe et monolithique. Certains EVC permettent à l'utilisateur de modifier le comportement de celui-ci, mais de manière contrôlée et jamais en permettant des modifications assez importantes. Cette manière de faire ne peut en aucun cas permettre l'élaboration d'un concept de lieu.

Un exemple parfait de l'illustration des concepts de lieu dans un environnement peut être vu dans notre quotidien. Les systèmes d'exploitation que nous utilisons permettent de créer un environnement virtuel, un environnement qui est totalement modifiable puisque les usagers peuvent créer leurs propres programmes. Par exemple, un environnement de traitement de texte est modifié par l'utilisation d'une autre application qui nous permet d'écouter de la musique en écrivant notre rapport, ou une application de vidéoconférence nous permettant de voir notre interlocuteur tout en naviguant sur Internet à l'aide d'une autre application. C'est l'utilisation de ces applications ensemble qui crée un environnement offrant un concept de lieu ; leur utilisation permettant à chaque usager de modifier leur environnement de la manière qui leur convient.

Il est de notre avis que cela doit être également le cas pour un EVC. Ce n'est pas une application qui crée un environnement, mais l'utilisation de plusieurs applications ensemble qui permet de créer un environnement et qui offre à l'utilisateur la possibilité de vraiment collaborer d'une manière qui lui convient.

La solution que nous proposons est de redéfinir la nature d'un EVC, non pas comme une application, mais comme un système de gestion d'applications. Permettre aux usagers de modifier complètement le comportement de l'EVC ne peut se faire qu'en leur donnant une liberté de création de leur propre programme.

### **2.4.1 Décomposition d'un EVC en trois couches (réseau - gestionnaire - application)**

Un EVC doit donc être vu comme un système de gestion d'applications, comme ce que nous utilisons actuellement sur nos ordinateurs, à la différence que le gestionnaire doit gérer des applications dans un système distribué, où l'environnement n'est pas peuplé d'une seule personne mais est peuplé de plusieurs personnes situées dans des lieux différents.

Les trois couches que nous voyons sont : la couche de communication par réseau (couche réseau), le gestionnaire et l'ensemble des applications. L'ensemble des applications représente l'environnement et est la partie modifiable permettant la conception d'un environnement collaboratif. Le gestionnaire représente la couche responsable du bon fonctionnement du système et offre différentes options aux applications pour leur permettre de remplir leur rôle. La couche de communication par réseau est la base du système et permet de rendre l'environnement distribué en permettant aux différentes applications de communiquer d'une manière transparente. Cette couche est aussi responsable d'enlever de la conception de chaque application l'obligation d'avoir une connaissance réseautique avancée.

Il est important ici de bien voir que la couche de communication par réseau ne correspond pas à ce qui est actuellement offert par les systèmes d'exploitation, mais peut être vue comme une couche plus générale, rendant les connections logique transparente à l'utilisateur. Pour cela elle se base sur les couches réseautique du système d'exploitation pour fonctionner, mais du point de vue de l'utilisateur, cette couche est la seule nécessaire pour l'utilisation des capacités réseaux du système.

Cette représentation en couches nous permet aussi de subdiviser le travail nécessaire pour le développement d'un EVC. Nous nous concentrerons sur la couche de communication

par réseau (couche réseau), puisqu'elle représente la base du système sur lequel nous pourrons plus tard bâtir la couche de gestion des applications.

#### **2.4.2 Concept d'adaptation dynamique de la couche de communication par réseau**

Nous avons jusqu'à présent décrit qu'un EVC offre de nombreuses difficultés, mais la plus importante, et celle qui se retrouve quasiment dans tous les problèmes à résoudre, est le fait que les solutions de ces problèmes sont nombreuses mais en général dépendent de l'utilisation finale de l'EVC.

Dans le cas de la couche de communication par réseau (couche réseau), cela veut dire que si nous voulons permettre un EVC offrant le concept de lieu, donc une utilisation dépendante de l'utilisateur, alors il est nécessaire de pouvoir offrir le plus d'architectures réseau possibles pouvant permettre de combler les différents besoins possibles. Or, dans ce cas-là, un défi majeur apparaît : il devient nécessaire de pouvoir modifier le type d'architecture utilisé par la couche réseautique pour la transmission selon le type de données à transmettre, c'est-à-dire qu'il devient nécessaire de permettre une adaptation dynamique de la couche réseautique.

Cette adaptation dynamique est la clé du bon fonctionnement de la couche de communication par réseau (couche réseau), puisque, comme nous allons le voir, aucune architecture réseautique n'est acceptable pour un grand nombre de types de données à transmettre, ce qui oblige une modification fréquente.

Il est de notre avis que pour chaque message à transmettre et quels que soient les critères associés, il existe au moins une architecture offrant de très bons résultats pour cette transmission. Ceci veut dire que pour le développement d'un EVC fonctionnel, il est né-

cessaire que la couche réseautique puisse offrir plusieurs topologies différentes selon les différents cas. Or, étant donné l'interdépendance des topologies et des ressources disponibles, il est très difficile d'avoir un grand nombre de topologies fonctionnelles en même temps. Il est donc obligatoire selon nous d'avoir une couche réseautique s'ajustant dynamiquement aux besoins de l'EVC, en créant et utilisant la meilleure topologie possible selon les ressources disponibles et les critères de transmission voulus.

### 2.4.3 Résumé de l'hypothèse

Notre hypothèse peut être résumée en plusieurs points :

- Un EVC doit s'inspirer du concept de lieu
- Une nouvelle manière de concevoir un EVC doit être créée
- Nous proposons une séparation en trois couches pour la conception d'un EVC
- Quelles que soient les conditions de transmission, une topologie existe permettant cette transmission
- L'adaptation dynamique de la couche réseautique est nécessaire

Ce mémoire a donc pour but de spécifier une méthode d'adaptation dynamique de la couche de communication par réseau (couche réseau), ce qui nous amènera tout d'abord à argumenter de la validité de notre hypothèse pour la résolution des défis reliés à un EVC. Ceci va nous amener ensuite à catégoriser différentes architectures selon leur capacité à répondre aux besoins de différents types de données utilisés par les EVC. Pour cela, une énumération de ces types de données sera faite, en plus de leurs besoins respectifs, ce qui nous permettra de définir une méthode de test à utiliser pour l'analyse des différentes architectures et nous amènera à une catégorisation de ces architectures.

Enfin, nous présenterons notre méthode de fonctionnement de l'adaptation dynamique et les résultats obtenus.

## CHAPITRE 3

### LA COUCHE DE COMMUNICATION PAR RÉSEAU ET PROTOCOLES EXPÉRIMENTAUX

Dans ce chapitre, nous allons présenter plus en détails le rôle de la couche de communication par réseau pour définir les différents types de données qu'elle aura à transmettre et classifier ces données en différentes catégories.

#### 3.1 Rôle de la couche

Notre hypothèse décompose un EVC en trois couches : réseautique, gestion et application. Nous nous intéressons maintenant à la couche réseautique et il est donc nécessaire de clarifier notre vision du rôle de cette couche avant de présenter les différents résultats obtenus.

La couche de communication par réseau (couche réseau) peut être considérée comme la base d'un EVC, elle est en fait la pièce qui permet de mettre en place la caractéristique la plus importante de tout environnement virtuel de collaboration : une application distribuée. Même si un environnement virtuel offre de nombreuses fonctionnalités, toutes ont comme caractéristique commune d'être distribuables. C'est un concept de base de tout EVC et c'est la raison d'être de la couche réseautique.

L'importance de la partie réseau dans un EVC est d'ailleurs un des défis majeurs pour leur conception puisque cela nécessite que les développeurs de l'environnement aient des compétences élevées dans le domaine. La couche réseautique a pour rôle premier d'enlever cette barrière à l'entrée dans le domaine du développement de EVC en prenant

en charge une des tâches les plus difficiles et en rendant son utilisation la plus simple possible.

Présentement, l'utilisation de la partie réseautique de n'importe quel système d'exploitation courant remplit déjà ce rôle, mais seulement pour une connexion point à point. En effet, lors du développement d'un programme, nous n'avons qu'à utiliser une interface offerte par le système pour l'envoi d'informations à travers le réseau en donnant juste quelques informations (communication par TCP ou UDP) et la méthode d'envoi (unicast, multicast ou broadcast avec leur information spécifique nécessaire) et le système d'exploitation se charge du reste. Par contre, toute implémentation d'une architecture doit être faite par le programme lui-même. Ainsi, si nous voulons envoyer le même message à plusieurs destinations en utilisant une architecture de type client-serveur, nous devons implémenter nous-mêmes le serveur, tout comme si nous voulons utiliser des architectures point à point plus avancées. Si initialement ce n'était pas très grave, pour un concepteur de EVC c'est énorme et c'est la raison majeure pour laquelle généralement un EVC ne possède qu'une seule architecture réseautique pour la livraison de ses messages : l'implémentation d'une seule bonne architecture nécessite déjà énormément de temps et de compétence réseautique.

Notre concept de couche de communication par réseau (couche réseau) veut donc combler ce problème, en offrant justement le même type de facilité que l'utilisation de la partie réseautique d'un système d'exploitation mais en offrant en plus la prise en charge du déploiement d'architecture réseau pour la livraison de messages. Une bonne analogie serait le système de poste actuel. En effet, lorsque nous voulons envoyer une lettre ou un paquet par la poste, nous n'avons généralement qu'à donner la destination et le délai de livraison qui nous intéressent, et le service postal s'occupe de l'envoi. Nous n'avons pas à savoir quel camion ou quel avion sera utilisé, quel sera le trajet ou toute autre information sur la méthode de livraison, la seule chose qui nous intéresse est la livraison

elle même. En somme, toute l'architecture mise en place pour la livraison nous est invisible, pour nous une lettre est envoyée et arrive à destination après un certain délai. C'est cette invisibilité de l'architecture utilisée que notre concept de couche réseautique veut amener, ceci permettant un développement plus facile d'un environnement virtuel de collaboration en éliminant, ou tout du moins en diminuant énormément, les compétences et connaissances requises sur la partie réseautique.

Par contre, de la même manière que le système de poste nous permet de suivre le cheminement de notre livraison et d'obtenir de l'information sur son état, de même notre couche réseautique devra permettre aux développeurs intéressés un accès aux informations qui les intéressent sur l'architecture utilisée et une possibilité de modification de l'architecture.

Pour pouvoir faire cette couche, il est donc nécessaire avant cela de classer les différentes architectures réseau utilisables par un EVC, nous permettant ensuite de définir une manière de les implémenter de façon invisible pour le développeur. Pour cela, il est nécessaire tout d'abord de faire une liste des différents types de données qui pourront être transmises par la couche et de clarifier leurs besoins. Ensuite nous pourrions classer les architectures selon leur capacité à transmettre ces différents types de données.

### **3.2 Information sur les données**

Tel que vu précédemment, un EVC possède plusieurs données différentes à transmettre à travers un réseau, certaines ayant des besoins communs, d'autres ayant des besoins quelquefois opposés. Nous allons faire ici une énumération des différents types d'information que doit transmettre le réseau et les classer selon leurs besoins. Cela nous permettra d'avoir un aperçu des besoins que la couche réseau doit en général combler pour un EVC, nous amenant ensuite à définir une manière d'évaluer différents types

d'architecture réseau.

Faire une classification des types de données pouvant être transmises par un EVC revient à faire une classification des différents types de données transmissibles tout court. En effet, tel que discuté précédemment, un EVC fonctionnel est un EVC permettant aux usagers de le modifier et d'utiliser l'environnement de la manière qui leur convient. Par contre, il est important que dans notre cas, nous détaillons aussi l'évolution des besoins des informations à transmettre selon le nombre d'utilisateurs concernés, en somme la capacité de mise à l'échelle des besoins.

### **3.2.1 Catégorisation des données à transmettre**

Les informations à transmettre peuvent être catégorisées de deux manières, selon que nous les regardons du point de vue de l'application ou du point de vue de la couche réseau. Du point de vue réseautique, le type d'information à transmettre est généralement séparé de la façon suivante :

1. Information de petite taille
2. Information de taille élevée
3. Information de temps réel

Du point de vue de l'application, nous pouvons catégoriser les informations selon les six types suivants :

1. Vidéo et audio en temps réel
2. Information descriptive sur l'environnement
3. Information de type travail collaboratif (CSCW, computer-supported cooperative work)

4. Information de contrôle
5. Message d'actualisation simple
6. Message d'actualisation collaborative

Dans les sections suivantes, nous allons détailler chaque type et ses besoins et montrer comment chaque type d'information d'un EVC est représenté par un ou plusieurs types du point de vue réseautique.

### 3.2.1.1 Type d'information du point de vue réseautique

Les informations sont séparées selon leurs besoins plutôt que selon les informations transmises. Si cela est logique d'un point de vue réseautique, nous verrons plus loin que cela a pour effet d'obliger le concepteur d'un EVC à connaître les concepts de la réseautique.

**Information de petite taille** Si la taille d'une information à transmettre à travers le réseau est inférieure à la taille d'un paquet de transmission, alors cette information fait partie de cette catégorie. La caractéristique principale de cette catégorie est le fait qu'étant donnée la faible taille, la meilleure architecture pour l'envoi de ce type d'information est généralement un lien direct entre les machines concernées.

**Information de taille élevée** Cette catégorie correspond aux données à transmettre qui, à cause de leur taille imposante en terme du nombre de paquets à transmettre, nécessitent en général des architectures réseaux plus avancées qu'une simple liaison directe entre les machines concernées. Les différents clients point à point utilisés actuellement pour l'échange de fichiers illustrent très bien le nombre de méthodes différentes utilisables pour la transmission de données de cette catégorie.

**Information de temps réel** Dans cette catégorie, la caractéristique première des données concernées est la nécessité que la réception de l'information se fasse en continu, avec une vitesse minimum de réception, en plus d'éviter à tout prix une interruption de la réception. De la même manière que la catégorie précédente, selon les critères de transmission voulus, de nombreuses architectures existent, mais dans les cas les plus performants elles ne se prêtent pas à la livraison des données des autres catégories.

Les critères de transmission suivants sont généralement associés à une ou plusieurs catégories de données et permettent ainsi de finaliser le choix du type d'architecture à utiliser pour la transmission des données.

**Réception en continu ("*streaming*")** Si ce critère est le défaut pour la transmission en temps réel, il peut aussi être existant pour la transmission de données n'ayant pas une obligation de réception en temps réel, mais quand même en continu. En général, ceci est dû au fait qu'il n'y a pas de vitesse minimum prédéfinie nécessaire à maintenir pour la transmission, comme c'est le cas pour les données en temps réel, mais qu'il est nécessaire, une fois la réception commencée, de garder une vitesse constante.

**Perte des données** Selon le type de transmission et l'architecture réseau choisis, il peut arriver assez souvent que certaines données transmises n'arrivent pas à destination ou arrivent incomplètes. Ce critère permet de définir si, pour l'information à transmettre, le risque de perte d'information est négligeable comparé aux autres gains (vitesse, réception continue, etc.) pouvant être engendrés par l'utilisation de certaines architectures.

**Délai de transmission** Le temps entre l'envoi d'un message par une application et sa réception par une autre correspond au délai de transmission. Ce critère est très important, puisque qu'il sert à définir la vitesse de transmission et donc varie selon

le type de message et ses besoins.

**Variation du délai de transmission ("Jitter")** Depuis l'introduction assez récente de la communication en temps réel, un autre critère est apparu comme très important pour la transmission de messages, c'est celui de la variation du délai de transmission. Cette variation est appelé le "delay jitter" ou généralement "jitter", et est un des paramètres les plus importants pour la transmission de contenus multimédias en temps réel.

Dans le tableau 3.1, on montre quel critère peut être associé avec quelle catégorie.

Tableau 3.1 Information du point de vue réseautique

|               | Réception continue | Perte d'information | Délai | Variation |
|---------------|--------------------|---------------------|-------|-----------|
| Petite taille | ✓                  | ✓                   | ✓     |           |
| Taille élevée |                    | ✓                   | ✓     |           |
| Temps réel    | ✓                  | ✓                   | ✓     | ✓         |

### 3.2.1.2 Type d'information du point de vue de l'application

Du point de vue de l'application, les informations sont séparées selon leur utilité plutôt que selon leur besoin réseautique. Cela est plus logique pour le concepteur de l'EVC, mais a comme désavantage que le lien avec le côté réseautique n'est plus aussi simple.

**Vidéo et audio en temps réel** Généralement, cette catégorie peut-être vue comme celle des informations d'application de type vidéo-conférence. Les informations doivent être reçues de manière continue, avec des vitesses minimums dépendantes du protocole utilisé par l'application, en offrant généralement une assez bonne tolérance à la perte de données.

**Information descriptive sur l'environnement** Voici quelques exemple du type de données correspondants à cette catégorie : des données représentant un avatar ou tout autre modèle 3D de l'environnement, des informations sur le climat de l'environnement, des données décrivant la maladie d'un patient dans un EVC médical. Contrairement à la catégorie précédente, il est moins facile de définir des besoins réseautiques généraux pour cette catégorie, chaque donnée pouvant avoir des besoins totalement différents l'une par rapport à l'autre.

**Information de type travail collaboratif** Un EVC peut et généralement contient des sous-applications du type CSCW (computer-supported cooperative work - travail collaboratif), qui sont des applications 2D dont l'affichage est partagé entre plusieurs participants. Le partage des tableaux blancs en est un exemple. Toutes les données nécessaires pour ce type de partage entrent dans cette catégorie, qui encore une fois peut difficilement être reliée à des besoins réseautiques spécifiques.

**Information de contrôle** Ce sont des données utilisées par l'application pour s'assurer de la bonne cohérence de l'environnement, de la gestion des différents services offerts et de toute autre information relative au fonctionnement du système. Ce type d'information ne supporte pas la perte ou la corruption de données, et nécessite une transmission rapide. Par contre, en général, la fréquence de transmission de ce type de données est de beaucoup moindre que celle des autres catégories.

**Message d'actualisation simple** Les données transmises pour informer les autres participants du mouvement d'un avatar ou d'un véhicule quelconque sont un bon exemple du type d'information caractérisant cette catégorie. En général, la taille des données est très petite, la tolérance aux pertes d'informations assez grande, mais la fréquence de transmission est la plus élevée comparée à toutes les autres catégories.

**Message d'actualisation collaborative** La dernière catégorie correspond aux données contenant l'information nécessaire pour permettre le déroulement d'une tâche col-

laborative. Les messages envoyés permettent au système de gérer le déroulement de la tâche, comme par exemple en permettant de définir quel usager a le contrôle d'une application lorsque plusieurs personnes l'utilisent en même temps. Les besoins réseautiques de cette catégorie sont assez variés, mais nécessitent le plus souvent une transmission la plus rapide possible et une manière de définir un ordre d'envoi et de réception.

### 3.2.2 Liens entre les types d'informations

Tableau 3.2 Lien entre les types de catégories

| -                                                  | Petite taille | Taille élevée | Temps réel |
|----------------------------------------------------|---------------|---------------|------------|
| <b>Vidéo et audio en temps réel</b>                | -             | -             | ✓          |
| <b>Information descriptive sur l'environnement</b> | Rarement      | ✓             | -          |
| <b>Information de type collaboratif</b>            | ✓             | ✓             | -          |
| <b>Information de contrôle</b>                     | ✓             | ✓             | -          |
| <b>Message d'actualisation simple</b>              | ✓             | Rarement      | -          |
| <b>Message d'actualisation collaborative</b>       | -             | ✓             | ✓          |

Le tableau 3.2 nous permet de résumer les liens entre les catégories de données vues par l'application et celles vues par le réseau. Nous avons aussi montré dans le tableau 3.1 les différents critères associés avec les catégories réseaux. Ces critères ont des niveaux d'exigence différents selon le type de données transportées, par exemple la tolérance à la perte d'information peut être nulle pour certaines données alors qu'elle peut être flexible pour d'autres. De la même manière, le critère du délai peut être très strict pour certaines informations ou être très flexible pour d'autres. Le tableau 3.3 donne un résumé des différentes exigences des critères selon le type d'information transmise, en plus de la fréquence générale de transmission de ce type de données pour une application de type

EVC.

### **3.3 Valeurs à mesurer**

Nous définissons les cinq valeurs suivantes qui doivent être mesurées pour chaque architecture :

#### **3.3.1 Délai**

C'est le temps entre l'envoi d'un paquet et sa réception. Pour pouvoir obtenir cette valeur, il est nécessaire de synchroniser les horloges entre les machines réceptrice et émettrice. Or, étant donné que le délai est généralement petit, la synchronisation des horloges doit être presque parfaite, ce qui actuellement est très difficile à faire. Une méthode généralement utilisée est de calculer l'aller-retour, qui est le temps entre l'aller et le retour d'un paquet, et de diviser la valeur obtenue par deux. Le temps ainsi obtenu n'est pas parfait, mais la moyenne obtenue après l'aller-retour de plusieurs paquets donne une bonne approximation de la valeur réelle. L'unité de mesure est la seconde.

#### **3.3.2 Délai intermédiaire**

C'est le temps entre la réception de deux paquets consécutifs d'un même message. Étant donné que cette mesure ne se fait que du côté de la machine réceptrice, la synchronisation du temps entre les machines n'est donc pas nécessaire. Cette valeur est différente du délai d'envoi entre deux paquets, dû à la variation du délai provoqué par plusieurs facteurs, tels que la congestion et le rythme de rejet des paquets par les routeurs. La valeur obtenue est donc par définition variable entre différents paquets, et nous mesurerons donc le délai

Tableau 3.3 Exigence des différents critères de transmission

| -                                                  | Réception continue | Perte    | Délai    | Variation | Fréquence  |
|----------------------------------------------------|--------------------|----------|----------|-----------|------------|
| <b>Vidéo et audio en temps réel</b>                |                    |          |          |           |            |
| Temps réel                                         | toujours           | flexible | stricte  | stricte   | faible     |
| <b>Information descriptive sur l'environnement</b> |                    |          |          |           |            |
| Petite taille                                      | quelquefois        | stricte  | flexible | -         | moyenne    |
| Taille élevée                                      | -                  | stricte  | flexible | -         | moyenne    |
| <b>Information de type CSCW</b>                    |                    |          |          |           |            |
| Petite Taille                                      | quelquefois        | flexible | stricte  | -         | importante |
| Taille élevée                                      | -                  | stricte  | flexible | -         | faible     |
| <b>Information de contrôle</b>                     |                    |          |          |           |            |
| Petite taille                                      | rarement           | stricte  | stricte  | -         | moyenne    |
| Taille élevée                                      | -                  | stricte  | flexible | -         | faible     |
| <b>Message d'actualisation simple</b>              |                    |          |          |           |            |
| Petite taille                                      | fréquente          | flexible | stricte  | -         | importante |
| Taille élevée                                      | -                  | stricte  | stricte  | -         | faible     |
| <b>Message d'actualisation collaborative</b>       |                    |          |          |           |            |
| Taille élevée                                      | -                  | stricte  | stricte  | -         | moyenne    |
| Temps réel                                         | toujours           | stricte  | stricte  | stricte   | importante |

maximum et minimum observé. L'unité de mesure est la seconde.

### **3.3.3 Variation du délai**

Cette valeur permet d'exprimer la variation du temps d'arrivée entre les paquets d'un message. Contrairement aux autres valeurs, elle peut être exprimée de plusieurs manières dans la littérature, puisque, dû à son introduction récente dans l'étude des réseaux, il n'existe pas encore de définition mathématique acceptée par l'ensemble de la communauté. Par contre, elle est généralement considérée comme étant la variance de la distribution du temps d'arrivée, et c'est cette méthode de mesure que nous utilisons pour notre étude. L'unité de mesure est la seconde.

### **3.3.4 Perte de données**

La perte de données correspond au pourcentage de paquets qui ne se rendent pas à destination, et ce pour diverses raisons. Tout comme pour la bande passante, la qualité des connexions réseaux et le nombre d'applications utilisant le réseau sont un facteur important de la perte de données, deux raisons que nous ne pouvons contrôler, nous empêchant ici aussi de garantir des conditions identiques lors de la mesure des valeurs. Il est possible par contre de détecter pour une architecture réseau les noeuds pouvant engendrer la perte d'information due à la congestion provoquée par le nombre de messages devant passer en ce point en même temps. Il est donc possible de caractériser les architectures par leur capacité à éviter la formation d'un de ces points de congestion, ou plutôt par le risque de formation d'une telle congestion. La mesure se fera en pourcentage de perte.

### 3.3.5 Bande passante

La bande passante représente la quantité de données pouvant être transmise à travers le réseau par seconde. La mesure de cette valeur est compliquée par le fait que le réseau est utilisé par plusieurs applications en même temps, ce qui signifie que l'application que nous utiliserons pour mesurer la bande passante mesurera en fait la bande passante *disponible* lors de la mesure. Étant donné que nous ne pouvons contrôler le réseau universitaire sur lequel plusieurs de nos tests seront effectués, il est donc impossible de garantir qu'une méthode de mesure de la bande passante puisse être effectuée dans des conditions identiques pour chaque architecture. Par contre, il est possible de mesurer la quantité de bande passante utilisée par le réseau pour le passage d'un type de message, et donc de comparer les architectures par la quantité de bande passante utilisée pour la transmission d'un message, ce qui est une valeur dépendante de la topologie du réseau plutôt que du réseau lui-même. L'unité de mesure est l'octet (byte) par seconde (B/s)

### 3.3.6 Mise à l'échelle

Les quatre premières valeurs représentent les critères de transmission de messages tels que vus au chapitre précédent. Elles permettent donc d'évaluer la capacité de l'architecture étudiée à répondre à ces différents critères. La mesure de la bande passante quant à elle ne sert pas à dire si le réseau peut transmettre ou non un message, mais plutôt s'il y a une bonne gestion de la bande passante disponible, et donc de pouvoir avoir une idée de la quantité de messages d'un certain type que la topologie du réseau utilisée permet de transmettre en même temps.

Nous avons présenté six défis majeurs à surmonter pour la conception d'un EVC :

- Bande passante
- Diversité

- Architecture distribuée et interaction
- Design d'un système temps réel
- Gestions des ratés
- Mise à l'échelle

De ces six défis, deux d'entre eux peuvent servir à classifier les différentes architectures réseaux que nous allons utiliser. En effet, si le défi de gérer la bande passante est déjà évalué lors de la mesure de la bande passante, le problème de la mise à l'échelle est un autre défi ayant une influence sur la performance de l'architecture. Il est donc possible de classifier les architectures selon leur capacité à résister à ce cas en évaluant la modification des valeurs des cinq mesures précédentes, comme par exemple la variation du pourcentage de paquets perdus selon le nombre de clients dans le réseau communiquant en même temps.

### **3.4 Description des tests et méthodes d'implémentation**

#### **3.4.1 Méthodes de test**

Maintenant que nous avons identifié les mesures que nous voulons faire pour nous permettre de classifier les différentes architectures, il est nécessaire de montrer de quelle manière ces mesures seront prises. En effet, étant donné le nombre d'architectures que nous avons à tester, il est nécessaire d'avoir une méthode de test uniforme permettant d'obtenir des valeurs utilisables pour notre classification. Pour les différentes valeurs nous utiliserons donc les méthodes suivantes :

**Délai** Le délai est une mesure qui varie énormément selon l'état du réseau, un même test effectué sur la même architecture donnera invariablement des résultats différents, étant donné notre impossibilité de contrôler le réseau utilisé. Pour contour-

ner ce problème, il est possible de prendre le délai d'une architecture comme une moyenne plutôt que comme une valeur fixe. Notre méthode de mesure consiste donc à envoyer plusieurs paquets et à faire la moyenne de leur temps de livraison. Nous pourrions donc générer plusieurs petits paquets et les envoyer un par un, ce qui permettrait d'obtenir notre valeur, mais il est aussi possible d'envoyer un message de taille considérable et de mesurer seulement le temps de livraison de ce message au travers du réseau. En effet, lors de l'envoi d'un gros message, celui-ci est automatiquement décomposé en plusieurs paquets par la couche réseautique du système d'exploitation, ce qui nous évite de gérer nous-mêmes l'envoi de plusieurs paquets et nous permet de n'envoyer qu'un seul message. Son temps d'arrivée divisé par le nombre de paquets nécessaire pour l'envoyer permet d'obtenir une valeur représentant assez bien le délai de transmission d'une architecture donnée. Finalement, étant donnée la différence entre une transmission par UDP et TCP sur le nombre de sous-messages transmis pour la livraison d'un paquet, il est nécessaire d'évaluer le délai de transmission selon le type de protocole utilisé.

Le type de message que nous utiliserons pour prendre la mesure de cette valeur est donc un message de type lourd, ce qui pour un environnement virtuel de collaboration peut correspondre, selon le tableau 3.2, à une information descriptive sur l'environnement. Nous utiliserons donc pour cela un fichier descriptif d'un modèle 3D donné qui représente ce type d'information.

**Délai intermédiaire** Cette mesure est très importante pour la classification des architectures selon leur capacité à offrir une certaine qualité de réception en continu. La mesure se fait, tel que vu lors de la description de cette valeur, en mesurant le délai maximum et le délai minimum obtenus entre la réception de paquets consécutifs d'un même message. Pour la même raison que la mesure précédente, il est nécessaire ici aussi de faire plusieurs mesures consécutives et d'utiliser les moyennes de ces mesures comme les valeurs représentant l'architecture évaluée.

Le type de message que nous devons utiliser pour cette mesure se doit d'avoir comme caractéristique majeure le critère de réception continue, ce qui, selon les tableaux 3.1 et 3.2, correspond à des messages de vidéo et audio en temps réel. Étant donné que ce type de message varie selon la vidéo et l'audio transmises, il est donc impossible de transmettre réellement une vidéo et une bande audio captées en temps réel lors de la transmission, puisque cela sera différent lors de différents tests. Pour contrer ce problème, nous allons donc utiliser une méthode proposée par Khamayseh (1998) pour une génération de messages représentant une vidéo-conférence, plus spécifiquement une vidéo-conférence utilisant le protocole H.263 pour la transmission des messages. Cela nous permet donc d'avoir le même type de message pour chacun de nos tests.

**Variation du délai** Cette mesure étant elle aussi très importante pour la classification des architectures selon leur capacité à offrir une certaine qualité de réception en continu, le même type de message sera donc utilisé pour notre évaluation. La mesure consistera à calculer la variance du délai d'arrivée entre la réception de paquets consécutifs d'un même message, et donc pourra se faire en même temps que la mesure de ce type de délai.

**Perte de données** Cette mesure consiste, tel que vu précédemment, à calculer le risque de congestion dans une architecture. Ce risque est donc dépendant de l'envoi fréquent de messages à travers le réseau. Selon le tableau 3.3, cela correspond souvent à des messages de type léger. Pour la mesure, nous ferons des tests à plusieurs fréquences d'envoi de ce type de message et nous détecterons à partir de quel moment il y aura apparition de perte de données. Cette détection sera assez facile, puisque les messages envoyés correspondront à une valeur qui est incrémentée à chaque envoi, la détection consistant juste à observer un trou dans les valeurs reçues à chaque noeud du réseau.

**Bande passante** La bande passante utilisée pour le transfert d'un message peut être

obtenue par la division de la taille du message par le délai de transmission de ce message, en supposant que la transmission est uniforme. Cela convient donc à mesurer le délai de transmission, puisque nous utilisons le même réseau pour chacune de nos architectures. Comme vu auparavant, l'effet de la topologie du réseau sur la bande passante peut se voir plutôt sur la répartition à travers le réseau de la quantité utilisée lorsqu'un message est transmis à plusieurs usagers. En somme, nous pouvons classer les architectures selon la quantité maximum de bande passante entre deux noeuds utilisés pour la transmission d'un même message.

Nous utiliserons pour cette mesure des messages de taille élevée, qui est le type de message utilisé pour la mesure du délai de transmission, nous permettant l'envoi d'un seul message pour faire deux mesures en même temps. Le message correspond à un fichier descriptif d'un modèle 3D donné.

Enfin, pour classer les architectures selon leur comportement envers le défi de mise à l'échelle, il suffit d'augmenter le nombre de messages envoyés en même temps pour chaque mesure, et d'observer la variation des différentes valeurs évaluées.

Maintenant que nous avons décrit nos méthodes théoriques de mesure, il est nécessaire de décrire les techniques utilisées pour une mise en pratique de ces méthodes, et donc pour les prises des différentes mesures. Nous regarderons d'abord les différentes machines utilisées pour nos tests, nous présenterons ensuite les différents programmes utilisés et leur fonctionnement, et nous conclurons cette partie en décrivant la mesure des différentes valeurs obtenues pour deux architectures réseaux différentes.

### **3.4.2 Parc informatique utilisé**

Nous avons pu effectuer nos tests sur plusieurs machines situées dans des réseaux performants. Nous nous sommes par contre seulement concentrés sur des PC utilisant le

système d'exploitation Linux. La liste des machines utilisées est la suivante, chaque groupe étant soit dans un réseau séparé, soit dans un sous-groupe différent :

- Six ordinateurs appartenant au CERCA, ayant des performances allant de 700MHz à 1.7GHz, basés sur un réseau à haute vitesse
- Cinq ordinateurs situés sous le réseau Étoile de l'École polytechnique de Montréal, les performances étant d'environ 2.5GHz
- Environ 90 ordinateurs situés dans des locaux de l'École polytechnique de Montréal, ayant des performances allant de 1GHz à 1.6GHz (situés dans 3 laboratoires Linux différents)
- Cinq ordinateurs personnels situés dans différents lieux à Montréal, ayant des performances réseautiques allant de la connexion téléphonique à une connexion haute vitesse commerciale

De ces machines, nous avons l'accès root pour toutes celles situées sous le réseau Étoile, de trois machines situées au CERCA et des cinq ordinateurs personnels situés dans différents lieux à Montréal, l'accès root étant nécessaire pour l'utilisation du programme TCPDUMP utilisé pour l'analyse des communications et des performances des topologies.

### **3.4.3 Programme de test**

Étant donné que nous avons à tester plusieurs topologies ayant des formes quelquefois énormément différentes entre elles, il fut nécessaire de trouver une manière générale de pouvoir construire une topologie, sans avoir à réécrire une application différente à chaque fois.

Une analyse des topologies existantes nous permet d'affirmer que chaque topologie peut

être séparée en trois parties, chacune ayant un rôle spécifique :

1. Partie émettrice
2. Partie réceptrice
3. Partie de redistribution

Si les parties émettrice et réceptrice représentent le début et la fin d'une topologie, la partie de redistribution peut être vue comme le milieu de la topologie, celle qui reçoit un message de la partie émettrice et qui le transmet aux différents récepteurs en utilisant un parcours dépendant de la topologie. Par exemple, pour une topologie serveur-clients de base, le serveur peut être vu comme la partie de redistribution et les clients comme les deux autres parties.

Pour la simulation d'une topologie, trois programmes furent donc créés, chacun représentant une des parties d'une topologie. Les sections qui suivent feront un résumé du fonctionnement de ces trois applications.

Chacune de ces applications devant permettre des connexions multiples, un choix entre l'utilisation de fork ou de thread a dû être fait. Étant donné que chaque application doit utiliser le moins de ressources possibles pour ne pas être un facteur dans l'évaluation des topologies, nos expériences nous ont amenés à utiliser des threads pour chacune des applications.

Enfin, chaque application ne fonctionne qu'en mode console et ne dispose d'aucun GUI, par contre chacune offre la possibilité d'être contrôlée à distance, en ayant un port toujours ouvert pour la réception de commandes.

### 3.4.3.1 Programme émetteur

Ce programme permet d'émettre des messages spécifiques à des destinations données, il peut donc être considéré comme la source de l'information circulant sur le réseau.

Il est possible d'utiliser cette application pour transmettre des messages selon trois types de méthodes :

- Unicast TCP
- Unicast UDP
- Multicast

Il fut décidé que nous n'utiliserions pas la méthode du broadcast du message sur le réseau, pour éviter de polluer celui-ci, le réseau utilisé étant celui de l'université.

Les types de messages pouvant être transmis à partir de cette application sont les suivants :

- Transfert vidéo temps réel (liaison avec l'application VIC)
- Transfert de fichiers
- Paquet généré par une simulation de vidéo-conférence
- Paquet numéroté envoyé à une certaine fréquence précise

Les trois derniers types de message nous servent à faire notre évaluation des différentes topologies, puisqu'ils correspondent à ceux que nous avons définis comme étant nécessaires pour la mesure de chaque type de valeur. Le premier type de message sert à nous assurer que notre réseau est fonctionnel, puisqu'il nous permet facilement de voir si en faisant nos modifications de la topologie nous perdons une connexion, puisque cela équivaut à perdre la vidéo.

Il est possible d'émettre plusieurs messages à la fois, ce qui est nécessaire pour nos tests de mise à l'échelle. Pour ce faire, nous avons introduit dans le programme le concept de groupe, chaque groupe étant juste un nouveau thread qui s'occupe d'un seul type

de message. Les étapes de fonctionnement illustrent l'ordre habituel d'utilisation du programme :

1. Création d'un groupe pour la transmission d'un message
2. Définition des destinations d'un message
3. Définition des méthodes de transmission (TCP,UDP, Multi-diffusion)
4. Définition de l'heure d'envoi du message

Ces étapes sont répétées pour chaque message que nous voulons transmettre.

Enfin, le programme offre deux manières de le contrôler, soit par l'utilisation d'un script comportant les commandes pré-écrites, soit par l'envoi de commandes directement par le réseau puisqu'un port est ouvert pour permettre la réception de commandes.

#### **3.4.3.2 Programme récepteur**

Ce programme est le complémentaire du précédent puisqu'il a pour but de recevoir des messages spécifiques émis sur des ports donnés. Mais son importance provient du fait que c'est ce programme qui est responsable de la prise des mesures, puisque comme nous l'avons vu lors de la description des méthodes de test, les valeurs sont toutes calculées à la réception des messages.

Les messages pouvant être reçus et les mesures ou actions prises sont les suivants :

- Transfert vidéo ⇒ redirection vers une application vic
- Fichier ⇒ calcul du délai et de la bande passante utilisée
- Paquet de simulation vidéo ⇒ utilisation du programme TCPDUMP pour le calcul du délai intermédiaire et la variation du délai
- Paquet numéroté ⇒ détection et calcul du pourcentage de perte d'information

L'utilisation de l'application est très simple, puisque quasiment toutes les prises de mesures sont automatisées. L'utilisateur a donc seulement à spécifier la création d'un groupe de réception, c'est à dire un port, le type de protocole de transmission et le type de message.

De la même manière que l'application précédente, il est ici aussi possible d'utiliser un script pour la création des groupes lors du lancement de l'application, ou de créer les groupes interactivement au travers du réseau.

#### **3.4.3.3 Programme de redistribution ("proxy")**

Si les deux derniers programmes peuvent être considérés comme le début et la fin de la topologie du réseau, alors ce programme doit être considéré comme la partie centrale de la topologie, puisqu'il permet justement, dépendant de sa configuration, d'obtenir différentes architectures.

Le but du programme est très simple : chaque message reçu est retransmis à plusieurs destinataires donnés selon le même protocole de transmission ou avec un autre protocole. En général, ce programme peut être considéré comme les différents noeuds de la topologie, permettant par exemple de simuler un serveur qui reçoit un message et le retransmet à tous ses clients ou un noeud de sortie, qui reçoit des messages de plusieurs clients et les redirige vers une machine spécifique.

Ce programme permet de spécifier exactement, pour un message donné, comment et à qui nous voulons le retransmettre. Par exemple, supposons que nous voulons recréer une topologie de type serveur-clients, où chaque message d'un client qui doit être transmis à toutes les autres machines (y compris la machine émettrice) doit passer par un serveur pour s'assurer de la synchronisation des messages (principe du premier arrivé, premier

servi). Dans ce cas-là, en supposant par exemple que le message est une vidéo continue, il est possible d'utiliser ce programme pour simuler le serveur, en suivant les étapes présentées dans le tableau 3.4.

Tableau 3.4 Étape de fonctionnement pour le "proxy"

| Étapes                                          | Exemple                                                      |
|-------------------------------------------------|--------------------------------------------------------------|
| Création d'un groupe                            | "create Groupe1"                                             |
| Définition du port et protocole d'arrivée       | "set Groupe1 portArrival 10020 protocolArrival TCP"          |
| Définition du type de message                   | "set Groupe1 message VideoStream"                            |
| Ajout des destinataires                         | "set Groupe1 add 132.204.345.21 add 134.205.34.21<br>..... " |
| Définition du port et protocole de transmission | "set Groupe1 portSend 10020 protocolSend UDP"                |
| Définition de l'heure de démarrage              | "set Groupe1 start now"                                      |

Une des options précédentes dans le programme est la possibilité de prendre certaines mesures, de la même manière que le programme de réception. En effet, tel que vu lors de la présentation des mesures, l'évaluation de la bande passante et du pourcentage de perte de données doit se faire à chaque noeud de la topologie. Or, comme chacun des noeuds est simulé avec cette application, il est intéressant de mettre la prise de ces valeurs directement dans l'application, plutôt que de faire une interaction avec le programme de réception pour la prise de ces mesures.

Plusieurs autres options existent pour spécifier des comportements spécifiques nécessaires pour simuler une topologie donnée, nous en verrons quelques-uns lors de la présentation des exemples de mesures prises sur certaines topologies.

### 3.4.4 Protocole de test

Nos tests seront effectués avec trois nombres différents de machines permettant ainsi d'obtenir une idée de l'évolution du comportement de la topologie en fonction du nombre de machines. Le tableau 3.5 donne les différents nombres de machines utilisés selon le niveau de mise à l'échelle à tester.

Tableau 3.5 Catégorisation du nombre de machines

| Niveau de mise à l'échelle | Nb d'émetteurs | Nb de récepteurs |
|----------------------------|----------------|------------------|
| Faible                     | 2              | 5                |
| Moyenne                    | 25             | 50               |
| Large                      | 50             | 100              |

Pour chaque niveau, un seul émetteur et un seul récepteur sont utilisés pour les mesures, les autres servent à créer du trafic sur la topologie pour simuler une utilisation du réseau. Il est important de signaler que notre définition de large ne représente pas du tout le nombre de machines représentant un large EVC, ce nombre pouvant atteindre un millier de machines dans certains environnements ludiques, mais est quand même une bonne indication du comportement de la topologie lorsque l'on s'approche d'un grand nombre de machines.

Pour chaque niveau, chaque mesure est prise une dizaine de fois, et la moyenne donne le résultat en écartant les valeurs extrêmes s'il y en a, ce qui serait dû à des conditions hors de notre contrôle.

Les valeurs à mesurer variant selon la condition du réseau, on ne peut pas comparer deux valeurs directement puisqu'elles peuvent être assez proches pour représenter le même comportement de la topologie. Par exemple, obtenir un délai de 10ms et un délai de 20ms ne veut pas dire que la première topologie est meilleure, étant donné que la fluc-

tuation des conditions du réseau permet d'affirmer quand les deux topologies peuvent être considérées comme ayant le même comportement. Cela nous oblige donc à utiliser un système de comparaison différent qui nous permet de comparer directement les architectures entre elles ; la figure 3.1 montre les différents gradients utilisés.

Le choix des valeurs extrêmes fût pris en faisant observer, par sept personnes, plusieurs expérimentations effectuées et notant leurs observations selon les valeurs mesurées. Ceci fut fait deux fois pour chaque valeur mesurée d'une topologie, et une analyse des différentes observations nous a permis d'obtenir les valeurs extrêmes.

DÉLAI

|            |       |        |         |              |             |
|------------|-------|--------|---------|--------------|-------------|
| 10 ms et - | 30 ms | 100 ms | 250 ms  | 400 ms       | 600 ms et + |
| Très bon   | Bon   | Normal | Mauvais | Très Mauvais |             |

INTER MESSAGE DÉLAI

|            |       |        |         |              |             |
|------------|-------|--------|---------|--------------|-------------|
| 10 ms et - | 20 ms | 40 ms  | 60 ms   | 100 ms       | 150 ms et + |
| Très bon   | Bon   | Normal | Mauvais | Très Mauvais |             |

JITTER

|           |       |        |         |              |             |
|-----------|-------|--------|---------|--------------|-------------|
| 5 ms et - | 15 ms | 30 ms  | 50 ms   | 100 ms       | 200 ms et + |
| Très bon  | Bon   | Normal | Mauvais | Très Mauvais |             |

PERTE D'INFORMATION

|        |       |       |
|--------|-------|-------|
| 0 %    | 20 %  | 100 % |
| Faible | Élevé |       |

BANDE PASSANTE (Pour N clients)

|            |                |                |            |
|------------|----------------|----------------|------------|
| 1 x Taille | N / 4 x Taille | N / 2 x Taille | N x Taille |
| Faible     | Bon            | Maximum        |            |

Figure 3.1 Gradients de mesures

## CHAPITRE 4

### CLASSIFICATION DES ARCHITECTURES

Dans ce chapitre, nous présenterons les différentes architectures sur lesquelles des tests furent effectués, un résumé des mesures prises pour chaque topologie, ainsi qu'une classification des architectures selon leur capacité à transmettre les différents types de messages d'un EVC. Nous terminerons par une analyse des résultats.

#### 4.1 Présentation des architectures

Nous avons choisi six topologies sur lesquelles nous ferons nos tests. Ces topologies permettent d'obtenir une majorité des autres topologies existantes en les fusionnant entre elles, ce qui nous permet d'affirmer qu'elles peuvent être considérées comme les topologies de base.

Pour chaque topologie, nous présenterons une description du fonctionnement, suivie des techniques que nous avons utilisées pour créer cette topologie et enfin d'un tableau résumant les mesures prises selon le nombre de machines utilisées.

##### 4.1.1 Point à point simple

Pour créer cette topologie, nous n'avons eu besoin que des programmes récepteur et émetteur tels que montré à la figure 4.1. Pour gérer le démarrage et la configuration des différents programmes, un script fut utilisé pour le lancement des applications en se connectant aux différentes machines par ssh et en exécutant des commandes spécifiques.

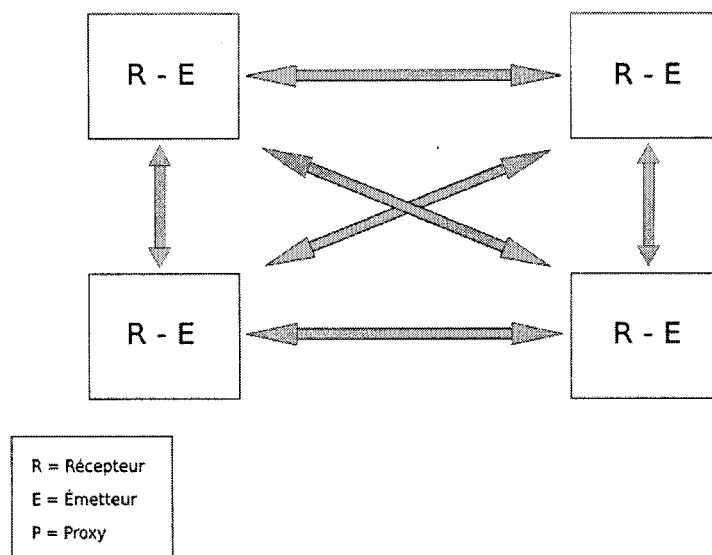


Figure 4.1 Point à point simple.

Tableau 4.1 Évaluation topologie point à point simple (selon les mesures définies à la Figure 3.1)

| Mise à l'échelle     | Délai        | Délai I.     | Variation | Bande P. | Perte  |
|----------------------|--------------|--------------|-----------|----------|--------|
| Faible (1 à 5 R-S)   | Très bon     | Très bon     | Très bon  | Maximum  | Aucune |
| Moyenne (5 à 50 R-S) | Mauvais      | Mauvais      | Très bon  | Maximum  | Aucune |
| Large (50 à 100 R-S) | Très Mauvais | Très Mauvais | Très bon  | Maximum  | Aucune |

Le tableau 4.1 présente une évaluation du comportement de la topologie selon le nombre de machines présentes. Il est important de noter deux points. Premièrement, notre définition de large avec 100 machines correspond au nombre maximum de machines que nous pouvons utiliser pour nos tests, mais ne correspond pas au nombre de machines pouvant être présentes en même temps dans un EVC, certains EVC ayant des milliers de personnes en même temps. Deuxièmement, une bonne valeur de la variation du dé-

lai intermédiaire ici ne signifie pas que la topologie est utilisable. En effet, dans ce cas par exemple, le délai intermédiaire devient mauvais très rapidement ce qui signifie une impossibilité d'utiliser cette topologie, quelque que soit la variation du délai, pour la transmission en temps réel par exemple.

#### 4.1.2 Topologie cyclique

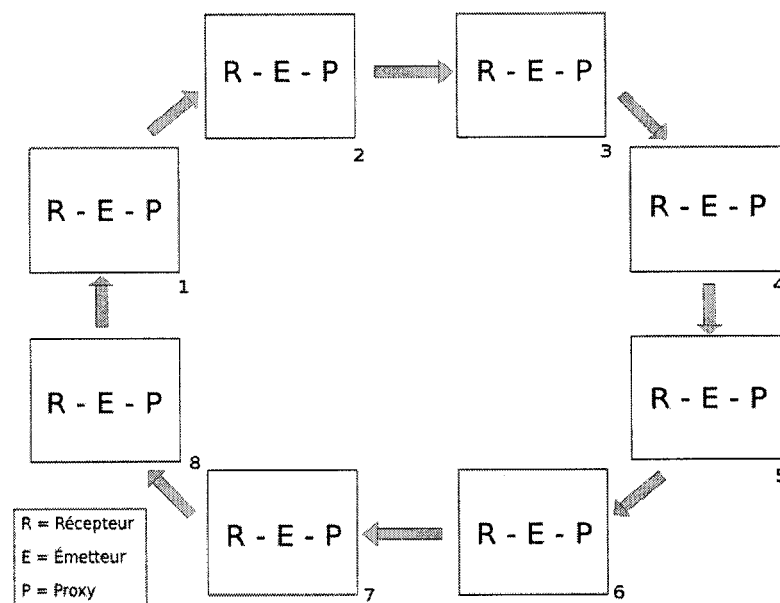


Figure 4.2 Topologie cyclique.

La création de cette topologie nécessite d'utiliser les trois programmes à notre disposition. En effet, pour simuler cette mise en série des différentes machines, la meilleure manière consiste à faire que chacun des noeuds de la machine consiste en un programme proxy qui remet ce qu'il reçoit à une autre machine prédéfinie, correspondant à elle-même si la machine concernée est celle qui doit contenir le récepteur.

Le tableau 4.2 présente une évaluation du comportement de la topologie selon le nombre de machines présentes. On remarque que le gain majeur d'une telle architecture est la

Tableau 4.2 Évaluation topologie cyclique (*selon les mesures définies à la Figure 3.1*)

| Mise à l'échelle     | Délai        | Délai I.     | Variation | Bande P. | Perte  |
|----------------------|--------------|--------------|-----------|----------|--------|
| Faible (1 à 5 R-S)   | Bon          | Bon          | Très bon  | Faible   | Aucune |
| Moyenne (5 à 50 R-S) | Mauvais      | Mauvais      | Très bon  | Faible   | Aucune |
| Large (50 à 100 R-S) | Très mauvais | Très mauvais | Très bon  | Faible   | Aucune |

constance des valeurs, puisqu'en effet, qu'un message doive être envoyé à une autre machine ou à toutes les autres machines, le trajet est le même, puisque par définition ce message se doit de passer par toutes les machines. Cette topologie permet donc, en diminuant la qualité des transmissions, de garder les capacités assez constantes quel que soit le nombre de machines. Par contre, une fois le nombre de machines devenu trop élevé, la dégradation est trop importante pour que la garantie de constance ait une importance.

#### 4.1.3 Serveur-clients élémentaire

La mise en place d'une telle topologie nécessite l'utilisation des trois programmes à notre disposition avec des configurations assez simples. Le serveur correspond à un programme proxy qui est configuré pour retransmettre tous les messages reçus à toutes les autres machines, celles-ci ayant des récepteurs, certaines machines ayant aussi des émetteurs envoyant leur message au proxy. La machine émettrice reçoit elle aussi le message de la part du proxy, puisque généralement ce type d'architecture est utilisé par les EVC pour gérer la synchronisation des messages. Finalement, pour bien simuler l'utilisation d'une telle architecture, la machine utilisée pour faire le serveur est une des plus perfor-

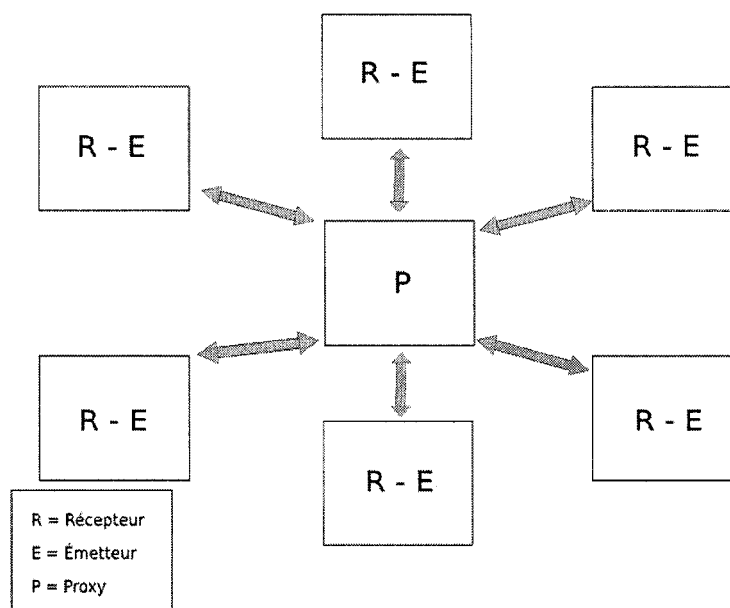


Figure 4.3 Topologie serveur-clients élémentaire.

mantes à notre disposition.

Tableau 4.3 Évaluation serveur-clients élémentaire (selon les mesures définies à la Figure 3.1)

| Mise à l'échelle     | Délai   | Délai I. | Variation | Bande P. | Perte  |
|----------------------|---------|----------|-----------|----------|--------|
| Faible (1 à 5 R-S)   | Bon     | Bon      | Très bon  | Maximum  | Aucune |
| Moyenne (5 à 50 R-S) | Normal  | Mauvais  | Mauvais   | Maximum  | Faible |
| Large (50 à 100 R-S) | Mauvais | Mauvais  | Mauvais   | Maximum  | Élevée |

Le tableau 4.3 présente une évaluation du comportement de la topologie selon le nombre de machines présentes. On remarque que le délai est toujours considéré comme bon même lorsque le nombre de machines est élevé. Par contre, le délai intermédiaire et la variation diminuent de qualité lors de cette augmentation de machines, puisque chaque

message doit passer par le serveur selon le principe du premier arrivé premier servi, donc deux parties d'un même message peuvent être séparées par plusieurs autres messages si le serveur reçoit ces autres messages avant la deuxième partie par exemple. Quand à l'utilisation de la bande passante, elle est considérée comme maximum, puisque pour chaque message à envoyer, l'émetteur ne l'envoie qu'une seule fois quel que soit le nombre de récepteurs, mais le serveur doit renvoyer ce message à tous ces récepteurs, donc une utilisation maximum de la bande passante, puisque nous nous retrouvons dans ce cas dans une topologie point à point simple. Finalement, la perte de données se trouve au niveau du serveur lors de la création de congestion ; cette perte est surtout due au conflit entre paquets UDP et TCP.

#### 4.1.4 Serveur-clients avec serveurs distribués

La mise en place de cette architecture fut possible en utilisant pour chaque serveur deux proxy, le premier recevant un message d'un client et le retransmettant à tous les serveurs, le deuxième recevant un message d'un serveur et le retransmettant aux clients du serveur simulé. Les clients sont ici aussi simulés par les programmes récepteur et émetteur.

Tableau 4.4 Évaluation serveur-clients avec serveurs distribués (*selon les mesures définies à la Figure 3.1*)

| Mise à l'échelle     | Délai  | Délai I. | Variation | Bande P. | Perte  |
|----------------------|--------|----------|-----------|----------|--------|
| Faible (1 à 5 R-S)   | Bon    | Bon      | Très bon  | Bon      | Aucune |
| Moyenne (5 à 50 R-S) | Normal | Normal   | Normal    | Bon      | Faible |
| Large (50 à 100 R-S) | Normal | Mauvais  | Mauvais   | Bon      | Faible |

Le tableau 4.4 présente une évaluation du comportement de cette topologie hybride selon

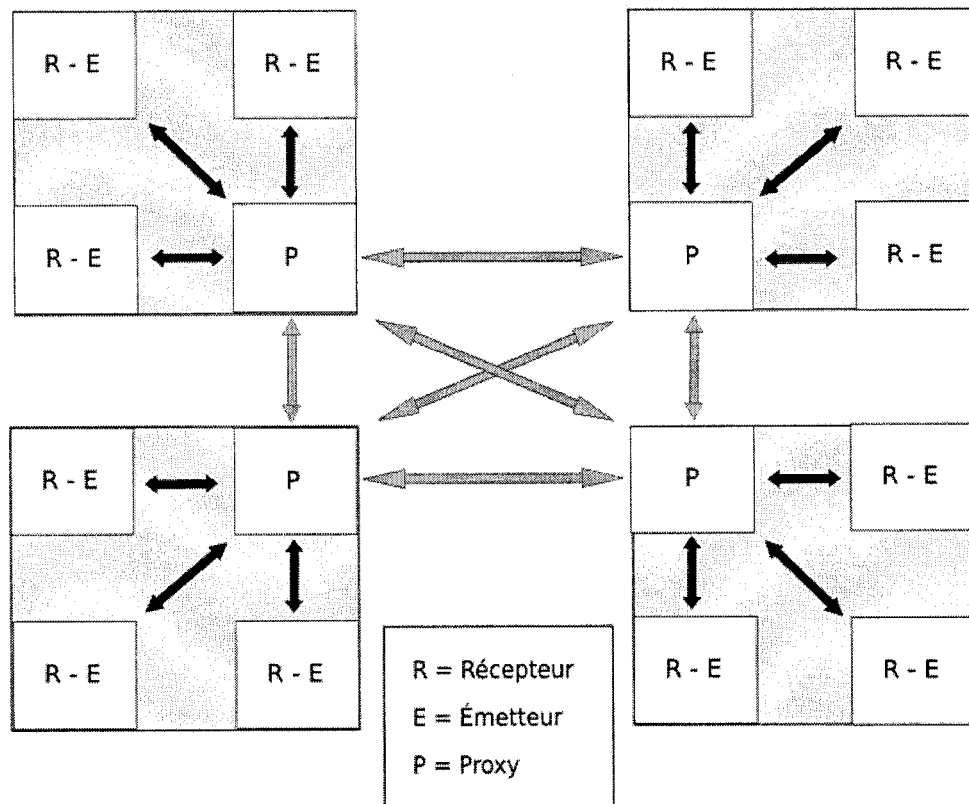


Figure 4.4 Topologie serveur-clients avec serveurs distribués.

le nombre de machines présentes. Il est intéressant de remarquer que la bande passante utilisée est considérée comme bonne. Cela est dû au fait que, selon le nombre de serveurs utilisés, nous diminuons la bande passante utilisée pour l'envoi d'un message du serveur vers ses clients. Par contre nous augmentons la bande passante nécessaire pour l'envoi vers les autres serveurs, puisque dans ce cas c'est une topologie point à point qui est utilisée. Or, les résultats démontrent que cette augmentation est négligeable comparée au gain obtenu pour la distribution aux clients. Enfin, en augmentant le nombre de serveurs, on diminue la vitesse à laquelle l'augmentation de perte de données progresse.

#### 4.1.5 Serveur-clients avec serveur principal

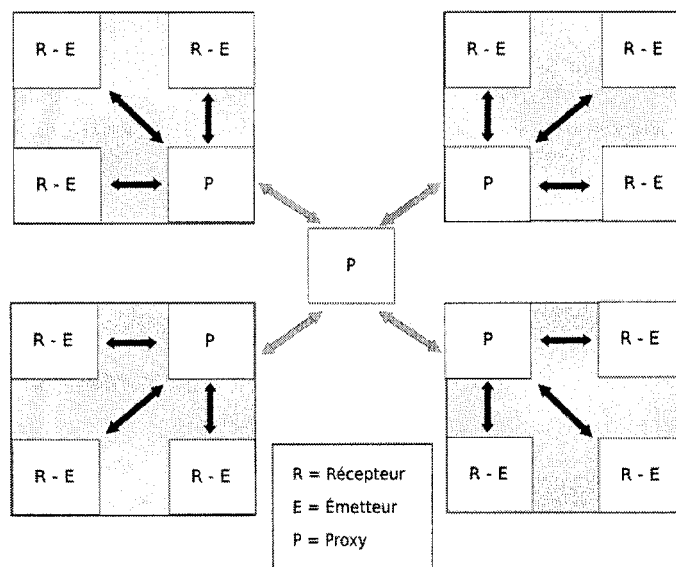


Figure 4.5 Topologie serveur-clients avec serveur principal.

Pour chaque serveur, nous avons utilisé deux proxy, un qui reçoit un message d'un client et le retransmet au serveur principal, et un autre qui reçoit un message du serveur principal et le retransmet à tous les clients. Pour le serveur principal, un seul proxy fut utilisé, celui-ci reçoit un message et le retransmet à tous les serveurs. Les clients sont ici aussi simulés par les programmes récepteur et émetteur.

Le tableau 4.5 présente une évaluation du comportement de cette topologie hybride selon le nombre de machines présentes. On remarque une dégradation des performances comparée à la topologie précédente montrant clairement que le gain que nous avons effectué au niveau de la synchronisation a eu un prix à payer non négligeable.

Tableau 4.5 Évaluation serveur-clients avec serveur principal (selon les mesures définies à la Figure 3.1)

| Mise à l'échelle     | Délai   | Délai I.     | Variation    | Bande P. | Perte  |
|----------------------|---------|--------------|--------------|----------|--------|
| Faible (1 à 5 R-S)   | Normal  | Normal       | Très bon     | Bon      | Aucune |
| Moyenne (5 à 50 R-S) | Normal  | Mauvais      | Mauvais      | Bon      | Faible |
| Large (50 à 100 R-S) | Mauvais | Très mauvais | Très mauvais | Bon      | Élevée |

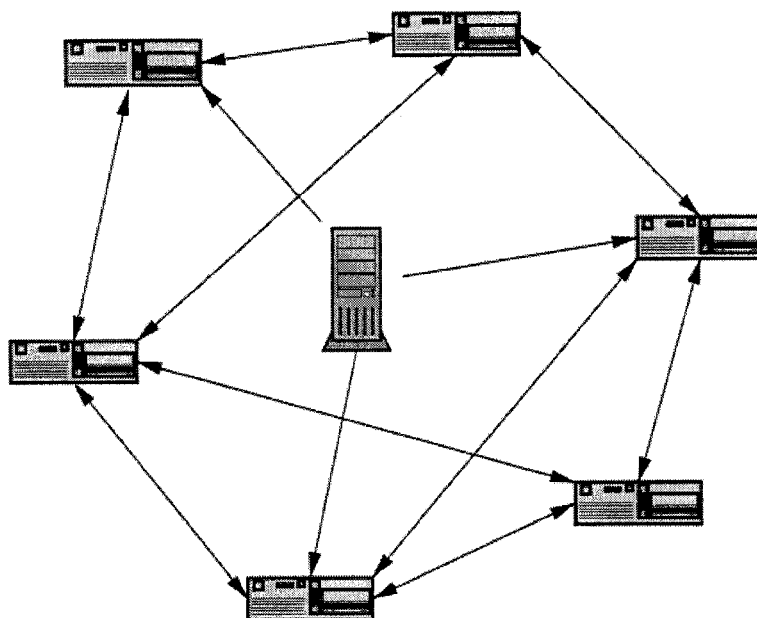


Figure 4.6 Topologie BitTorrent

#### 4.1.6 Système BitTorrent

Étant donné que ce protocole n'est utilisé que pour la distribution de fichiers, nous avons seulement mesuré le délai et la bande passante utilisée. Pour cela nous avons utilisé un client existant actuellement au lieu de simuler la topologie.

Tableau 4.6 Évaluation Bittorrent (*selon les mesures définies à la Figure 3.1*)

| Mise à l'échelle     | Délai  | Délai I. | Variation | Bande P. | Perte |
|----------------------|--------|----------|-----------|----------|-------|
| Faible (1 à 5 R-S)   | Bon    | -        | -         | Bon      | -     |
| Moyenne (5 à 50 R-S) | Normal | -        | -         | Bon      | -     |
| Large (50 à 100 R-S) | Normal | -        | -         | Bon      | -     |

Le tableau 4.6 présente une évaluation du comportement de cette topologie. Lors des tests, il fut remarqué que la bande passante est distribuée sur tous les clients. Pour l'envoi d'un message à tous les clients, en général l'émetteur n'envoie pas plus de deux à trois fois la taille du fichier quel que soit le nombre de clients disponibles. Si cette architecture offre de bonnes qualités pour la distribution de fichier, la technique utilisée ne permet pas d'utiliser le fichier avant son arrivée complète, puisque les bouts du fichier arrivent dans un ordre pouvant être considéré comme aléatoire. De plus, cette technique de livraison devient encombrante lorsque le nombre de machines connectées est faible, puisque ici une connexion directe est très fonctionnelle.

## 4.2 Classification des architectures selon les tests effectués

Les tableaux 4.7, 4.8 et 4.9 donnent un résumé des différents résultats obtenus selon le nombre de machines utilisées. On peut y voir très bien que les différentes architectures donnent de bons résultats quand le nombre de machines est faible, mais que plus ce nombre augmente, plus certaines architectures se différencient des autres, sans toutefois qu'aucune n'offre de bons résultats pour tous les critères en même temps.

Les différents résultats obtenus, nos catégorisations des différents messages, en plus de

Tableau 4.7 Résumé évaluation : 1 à 5 machines (*selon les mesures définies à la Figure 3.1*)

| Mise à l'échelle                    | Délai    | Délai I. | Variation | Bande P. | Perte  |
|-------------------------------------|----------|----------|-----------|----------|--------|
| <b>P2P-Simple</b>                   | Très bon | Très bon | Très bon  | Maximum  | Aucune |
| <b>Cyclique</b>                     | Bon      | Bon      | Très bon  | Faible   | Aucune |
| <b>S-C élémentaire</b>              | Bon      | Bon      | Très bon  | Maximum  | Aucune |
| <b>S-C avec serveurs distribués</b> | Bon      | Bon      | Très bon  | Bon      | Aucune |
| <b>S-C avec serveur principale</b>  | Normal   | Normal   | Très bon  | Bon      | Aucune |
| <b>Bittorrent</b>                   | Bon      | -        | -         | Bon      | -      |

Tableau 4.8 Résumé d'évaluation : 5 à 50 machines (*selon les mesures définies à la Figure 3.1*)

| Mise à l'échelle                    | Délai   | Délai I. | Variation | Bande P. | Perte  |
|-------------------------------------|---------|----------|-----------|----------|--------|
| <b>P2P-Simple</b>                   | Mauvais | Mauvais  | Très bon  | Maximum  | Aucune |
| <b>Cyclique</b>                     | Mauvais | Mauvais  | Très bon  | Faible   | Aucune |
| <b>S-C élémentaire</b>              | Normal  | Mauvais  | Mauvais   | Maximum  | Faible |
| <b>S-C avec serveurs distribués</b> | Normal  | Normal   | Normal    | Bon      | Faible |
| <b>S-C avec serveur principale</b>  | Normal  | Mauvais  | Mauvais   | Bon      | Faible |
| <b>Bittorrent</b>                   | Normal  | -        | -         | Bon      | -      |

certaines tests effectués pour confirmer le tout, nous permettent de présenter dans les tableaux 4.10 et 4.11 quand utiliser certaines architectures pour la livraison de messages de type EVC.

Par exemple, pour la livraison de vidéo en temps réel, nous arrivons à la conclusion que la topologie point à point ne peut être utilisée que dans le cas où le nombre de machines est faible. En effet, si nous regardons les tableaux 4.7, 4.8 et 4.9, nous pouvons voir que cette topologie offre de mauvais délais initial et intermédiaire dès lors que le nombre de

Tableau 4.9 Résumé d'évaluation : 50 à 100 machines (*selon les mesures définies à la Figure 3.1*)

| Mise à l'échelle                    | Délai        | Délai I.     | Variation    | Bande P. | Perte  |
|-------------------------------------|--------------|--------------|--------------|----------|--------|
| <b>P2P-Simple</b>                   | Très mauvais | Très mauvais | Très bon     | Maximum  | Aucune |
| <b>Cyclique</b>                     | Très mauvais | Très mauvais | Très bon     | Faible   | Aucune |
| <b>S-C élémentaire</b>              | Mauvais      | Mauvais      | Mauvais      | Maximum  | Élevée |
| <b>S-C avec serveurs distribués</b> | Normal       | Mauvais      | Mauvais      | Bon      | Faible |
| <b>S-C avec serveur principale</b>  | Mauvais      | Très mauvais | Très mauvais | Bon      | Élevée |
| <b>Bittorrent</b>                   | Normal       | -            | -            | Bon      | -      |

machines commence à être élevé, étant donné que ces deux critères sont très importants pour la transmission de vidéo en temps réel.

De même, pour la livraison de message d'information de contrôle, qui nécessite un délai et des pertes faibles, une topologie serveur-clients élémentaire offre de bons résultats pour un nombre moyen de machines, par contre une topologie serveur-clients distribuée offre de bons résultats et ce même pour un nombre élevé de machines. La seule raison d'utiliser une topologie serveur-clients élémentaire revient donc, non pas à avoir besoin de ces qualités de transfert, mais à une raison de fonctionnement de l'EVC. Dans ce dernier cas, le besoin de synchronisation des messages de contrôle est plus facilement réalisable par l'utilisation d'une topologie avec serveur central que d'une topologie distribuée.

Tableau 4.10 Catégorisation des architectures (1)

|                                                     | <b>P2P simple</b>                                                      | <b>Cyclique</b>                                                          | <b>S-C élém.</b>                                             | <b>S-C avec serveurs distribués</b>                                                  | <b>S-C avec serveur principal</b>                                                | <b>Bittorent</b>                                                                                  |
|-----------------------------------------------------|------------------------------------------------------------------------|--------------------------------------------------------------------------|--------------------------------------------------------------|--------------------------------------------------------------------------------------|----------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|
| <b>Vidéo et audio en temps réel</b>                 | Nombre faible de machines                                              | Nombre faible de machines et synchronisation des arrivées non nécessaire | Nombre faible de machines, avec nécessité de synchronisation | Nombre moyen de machines                                                             | Non utilisable pour ce type de message en général                                | Jamais                                                                                            |
| <b>Information descriptrice sur l'environnement</b> | Nombre faible de machines et synchronisation de message non nécessaire | Nombre moyen de machines et synchronisation de message non nécessaire    | Nombre moyen de machines avec synchronisation des messages   | Nombre élevé de machines avec envoi de messages à des clients spécifiques            | Nombre élevé de machines avec synchronisation des messages nécessaire            | Taille élevée du message avec un nombre assez élevé de machines et synchronisation non nécessaire |
| <b>Information de type travail collaboratif</b>     | Nombre faible de machines avec délai court                             | Envoi à toutes les machines sans synchronisation                         | Nombre moyen de machines avec besoin de synchronisation      | Nombre élevé de machines, envoi à certains clients et synchronisation non nécessaire | Nombre élevé de machines, envoi à certains clients et synchronisation nécessaire | Taille élevée du message avec un nombre de machines assez élevé et synchronisation non nécessaire |

Tableau 4.11 Catégorisation des architectures (2)

|                                              | <b>P2P simple</b>                          | <b>Cyclique</b>                                  | <b>S-C élém.</b>                                                       | <b>S-C avec serveurs distribués</b>                                                  | <b>S-C avec serveur principal</b>                                                | <b>Bittorent</b>                                                                         |
|----------------------------------------------|--------------------------------------------|--------------------------------------------------|------------------------------------------------------------------------|--------------------------------------------------------------------------------------|----------------------------------------------------------------------------------|------------------------------------------------------------------------------------------|
| <b>Information de contrôle</b>               | Nombre faible de machines avec délai court | Envoi à toutes les machines sans synchronisation | Nombre moyen de machines avec besoin de synchronisation                | Nombre élevé de machines, envoi à certains clients et synchronisation non nécessaire | Nombre élevé de machines, envoi à certains clients et synchronisation nécessaire | Taille élevée du message avec nombre élevé de machines et synchronisation non nécessaire |
| <b>Message d'actualisation simple</b>        | Nombre faible de machines                  | Synchronisation non nécessaire                   | Nombre moyen de machines avec envoi des messages à toutes les machines | Nombre élevé de machines avec envoi de messages à des clients spécifique             | Nombre élevé de machines avec synchronisation des messages nécessaire            | Jamais                                                                                   |
| <b>Message d'actualisation collaborative</b> | Nombre très faible de machines             | Jamais                                           | Nombre moyen de machines                                               | Jamais                                                                               | Nombre moyen de machines avec délai flexible                                     | Jamais                                                                                   |

## CHAPITRE 5

### L'ADAPTATION DYNAMIQUE

Jusqu'à présent, nous avons démontré qu'un EVC se doit de permettre le transfert de plusieurs types de messages, que nous ne pouvons prédire les messages qui seront transmis et que la topologie réseautique à utiliser est dépendante du type de messages à transmettre, du nombre de personnes devant recevoir ce message et des critères associés au message.

Dans ce chapitre nous présenterons plus en détail notre concept d'adaptation dynamique de la couche réseautique en plus de faire une présentation des différentes méthodes que nous avons étudiées pour son implémentation.

Premièrement, nous décrirons notre concept d'adaptation dynamique et présenterons deux problèmes majeurs reliés à cette adaptation, nous présenterons ensuite nos méthodes de résolution de ces problèmes, pour terminer par une analyse de nos observations lors du développement et de l'utilisation d'un prototype permettant une adaptation de la couche réseautique.

#### 5.1 Concept d'adaptation dynamique et problème créé

Nous avons vu qu'aucune architecture de base ne permet de répondre parfaitement aux besoins d'un EVC. Or, au des résultats obtenus lors de nos classifications des topologies de base, il est aussi vrai que pour chaque message à transmettre et quels que soient les critères associés, il existe au moins une architecture offrant de très bons résultats pour cette

transmission. Ceci veut dire que pour le développement d'un EVC fonctionnel, il est nécessaire que la couche réseautique puisse offrir plusieurs topologies différentes selon les différents cas. Toutefois, étant donné l'interdépendance des topologies et des ressources disponibles, il est impossible d'avoir un grand nombre de topologies fonctionnelles en même temps. Il est donc nécessaire d'avoir une couche réseautique s'ajustant dynamiquement aux besoins de l'EVC, en créant et utilisant la meilleure topologie possible selon les ressources disponibles et les critères de transmission voulus. Cette allocation dynamique nécessite donc une couche réseautique ayant une intelligence artificielle de base pour la prise de décisions.

En somme, pour chaque message transmis dans un EVC, la meilleure topologie possible au moment de la transmission doit être utilisée. Or, les différents tests que nous avons effectués lors de la création d'un prototype de la couche réseautique ont fait ressortir deux problèmes majeurs de fonctionnement dont nous devons tenir compte :

**Création de topologie contrôlée** Il est impossible de créer ou modifier une topologie existante pour chaque message que nous voulons transmettre, les ressources nécessaires pour cela étant trop élevées. Le fonctionnement doit se faire en choisissant des topologies offrant de bonnes conditions pour différents messages, et la création ou la modification d'une topologie ne doit se faire que quand le gain effectué par cette modification en vaut le coût en ressources. Il est donc nécessaire que la couche réseautique puisse avoir une méthode d'évaluation d'un gain possible, ce qui équivaut à avoir une certaine compréhension des liens entre les types de messages transmis pour un EVC et les critères de transmission de messages à travers le réseau. Ces liens furent présentés dans le tableau 3.3.

**Période de transition** Lors de la modification d'une topologie existante, il existe un moment pendant lequel la transmission ne fonctionne plus, il y a une perte de données ou il y a une augmentation du délai. En somme, la modification de la

topologie affecte les données transmises lors de cet évènement. Or, ceci entraîne donc des problèmes pour le fonctionnement d l'EVC, pouvant même amener à une perte d'immersion des usagers. Il est nécessaire que lors de la modification des topologies, une méthode permettant une transition en douceur soit utilisée, pour éviter ces différents problèmes.

## **5.2 Méthode d'implémentation de changement d'architecture**

Comme nous l'avons vu plus haut, il est nécessaire lors du changement de topologie que nous tenions compte de la méthode de transition que nous allons utiliser. Ce problème survient lorsque nous voulons remplacer une topologie utilisée, pas lorsque nous voulons seulement créer une nouvelle topologie.

Nous avons deux méthodes permettant de faire ce changement, la première étant utilisée lorsque les ressources disponibles permettent la création d'une nouvelle topologie en parallèle de celle que nous voulons, la deuxième étant quand cela n'est pas le cas.

### **5.2.1 Topologies duales**

Cette première méthode consiste à créer la nouvelle topologie en parallèle de celle qui doit être modifiée et lorsque la nouvelle topologie est implémentée, les transmissions utilisent celle-ci au lieu de l'ancienne. La figure 5.1 illustre ce concept.

L'implémentation de cette méthode nous a permis de découvrir que l'utilisation de la nouvelle topologie directement n'est pas désirable pour tous les messages transmis à l'aide de l'ancienne. Certains types de messages, tels ceux d'une vidéo-conférence par exemple, sont très susceptibles à un changement, et celui de la modification de la topolo-

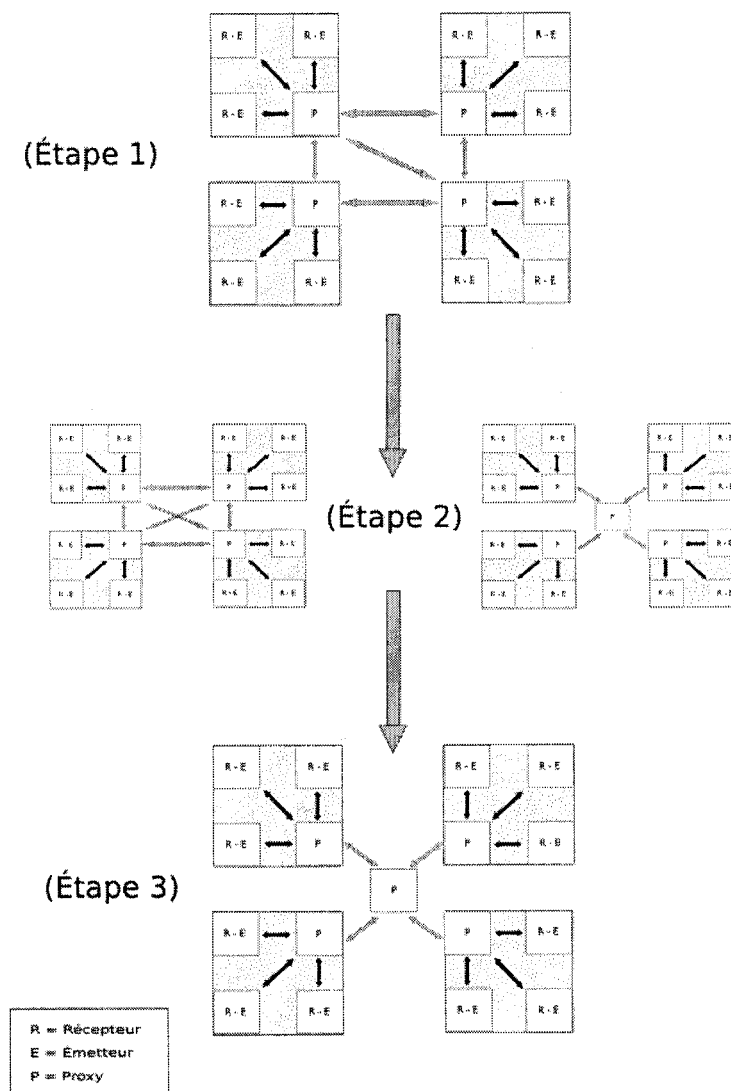


Figure 5.1 Méthode topologies duales

gie de transmission entraîne par défaut une modification de la performance de livraison des messages qui est remarquée par l'utilisateur et est non désiré dans un EVC. Il est donc nécessaire que l'ancienne topologie soit en place jusqu'à que ces types de messages aient finis d'être transmis ; les autres, tout comme les nouveaux messages, utilisant la nouvelle topologie.

### 5.2.2 Décharge de la topologie

Cette deuxième méthode permet de modifier une topologie lorsque les ressources disponibles ne permettent pas de créer la nouvelle topologie en parallèle avec l'ancienne, ce qui en général est dû à une surcharge d'utilisation d'un noeud. La modification ne peut se faire directement, en arrêtant la livraison des messages, modifiant la topologie et en retransmettant les messages à l'aide de la nouvelle architecture, puisque le temps d'arrêt nécessaire pour cela crée automatiquement un arrêt de fonctionnement de l'EVC, ce qui n'est pas acceptable. Il est donc nécessaire d'avoir une méthode qui permet de modifier l'architecture tout en permettant de continuer à livrer les messages. Par contre, nous ne pouvons comme pour l'autre méthode garantir que certains types de messages ne seront pas perturbés lors de la transmission.

La technique que nous utilisons consiste à diminuer la charge d'un noeud suffisamment pour pouvoir ensuite le modifier de façon à l'intégrer dans la nouvelle topologie. La diminution de la charge consiste à modifier la manière dont certains messages sont transmis en utilisant une topologie nécessitant moins de ressources (tel que P2P), bien que cela entraîne alors une diminution de la performance pour la transmission de ce type de messages. Pour faire cette modification, nous utilisons la possibilité de donner de nouvelles commandes à nos trois programmes utilisés pour la simulation d'un EVC, en modifiant interactivement le lieu de transmission et de réception des différents messages, ce qui nous permet de les reconfigurer d'une manière à simuler une autre topologie. La figure 5.2 illustre un exemple d'application de cette méthode pour le passage d'une topologie serveur-clients hybride avec un serveur principal à une topologie serveur-clients hybride avec communication cyclique entre les serveurs.

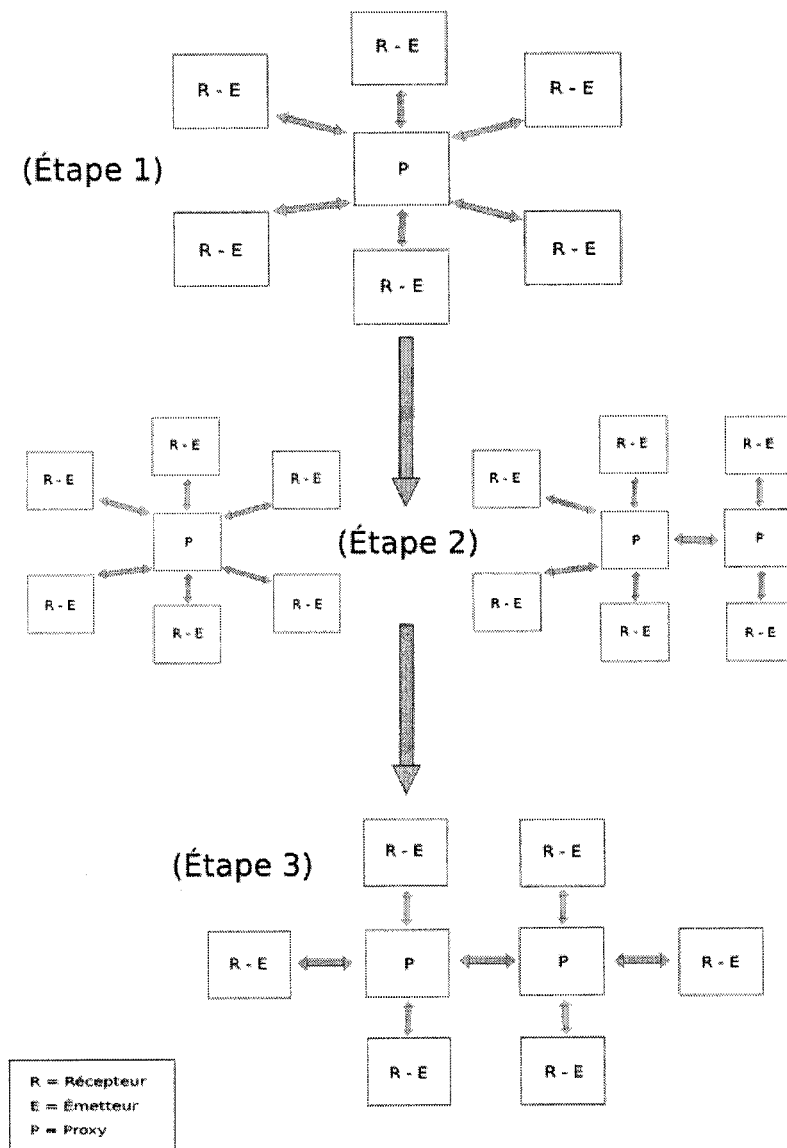


Figure 5.2 Exemple de la méthode de décharge

### 5.3 Méthode de sélection d'architecture

Le choix d'utiliser une nouvelle architecture pour la transmission des messages ne peut se faire aussi facilement que de choisir la meilleure architecture pour le type de messages que nous devons transmettre, puisque à cause du grand nombre de messages différents

à transmettre en même temps pour un EVC, cela nécessiterait d'avoir plusieurs topologies différentes utilisées en parallèle, ce qui implique une trop grosse utilisation des ressources disponibles, que ce soit de la bande passante ou de la puissance de calcul à certain noeuds.

Nos tests d'implémentation d'un prototype nous ont permis de définir certains paramètres qui permettront de développer une technique de choix, ces paramètres étant les différentes valeurs qui seront justement affectées par un changement d'architecture ou qui limiteront ce changement.

- Capacité de calcul disponible à chaque machine
- Bande passante disponible entre chaque machine
- Arbre de connexion possible entre machine
- Délai de transmission entre chaque machine
- Calcul de la charge des machines
- Type de messages transmis et leur critère de transmission

Ces différents paramètres permettent tous de prendre certaines décisions, le tableau 5.1 donnant les différentes décisions affectées par chaque paramètre.

Si l'arbre de connexion ne varie pas beaucoup, et le type de messages et leurs critères sont connus, il est beaucoup plus difficile d'obtenir les autres valeurs, puisqu'elles changent souvent selon la charge du réseau utilisé. Il est donc nécessaire de prendre les mesures le plus souvent possible, mais dans ce cas-là, les méthodes de mesure ne peuvent utiliser que de faibles ressources au risque de saturer rapidement le réseau. L'utilisation d'agents fut la seule solution trouvée qui permettait ces prises de mesure tout en utilisant le moins de ressources possibles. Quelques utilisations d'un prototype basé sur la maîtrise de Khamayseh (1998) a permis de confirmer la viabilité de cette option, en nous montrant que nous pouvions obtenir de l'information sur les différentes composantes du

Tableau 5.1 Paramètres permettant une prise de décision

| Paramètre                          | Décision                                                                                                                                                                                       |
|------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Capacité de calcul</b>          | Permet de définir quelles machines pourront être utilisées comme noeuds                                                                                                                        |
| <b>Bande passante</b>              | Permet de définir le nombre de messages pouvant transiter par chaque machine                                                                                                                   |
| <b>Arbre de connexion</b>          | Permet de définir les différentes connexions entre les machines et d'avoir une liste de plusieurs chemins possibles entre différentes machines                                                 |
| <b>Délai de transmission</b>       | Permet de définir les types de messages pouvant être transmis entre chaque machine                                                                                                             |
| <b>Calcul des charges</b>          | Permet de savoir si une machine peut supporter un rôle plus important dans la topologie, ou si elle peut être utilisée comme un noeud dans différentes architectures fonctionnant en parallèle |
| <b>Type de message et critères</b> | Nécessaire pour savoir si la topologie actuelle est suffisante, si un changement d'architecture offrira un meilleur gain et si il est possible d'avoir une période de transition acceptable    |

réseau en utilisant assez peu de ressources.

#### 5.4 Analyse de l'utilisation d'un prototype de base

Nous avons implémenté un prototype nous permettant de vérifier si l'adaptation dynamique offre des gains d'un côté pratique. Le prototype que nous avons créé ne comportait pas d'intelligence artificielle mais pouvait être contrôlé à n'importe quel moment par un opérateur et permettait, en plus de créer ou modifier une topologie, d'obtenir des informations sur les différents noeuds formant l'architecture.

La création d'une topologie se fait en donnant aux programmes les machines la formant

et leur rôle, en plus de spécifier une des topologies connues du prototype. La modification d'une topologie par contre doit se faire plus manuellement par l'opérateur en spécifiant le comportement de chaque machine du réseau pendant les étapes de modification de la topologie. Ceci a pour effet qu'il est seulement possible de tester l'adaptation dynamique sur un réseau comportant peu de machines, sinon la tâche deviendrait beaucoup trop ardue.

De l'utilisation de notre prototype nous avons obtenu la confirmation qu'un changement dynamique était acceptable, puisqu'en général il fut possible d'améliorer la connexion en modifiant la topologie selon le type de messages actuellement transmis, les gains observés étant assez souvent non négligeables et quelquefois assez importants. Pour ce prototype, le changement de topologie fut décidé manuellement en analysant les capacités du réseau, et en comparant les besoins en critères de transmission des messages à ces capacités. L'utilisation d'une intelligence artificielle pouvant faire ces choix ne put être mise au point pour ce prototype, mais l'expérience que nous avons eue lors de son utilisation nous permet de croire qu'une intelligence artificielle pourra être mise au point, les choix étant toujours dictés par des comparaisons directes entre les capacités du réseau et les besoins des messages.

Notre utilisation du prototype nous a aussi permis de mieux définir certaines tâches que devra effectuer la couche réseautique pour implémenter une adaptation dynamique efficace :

**Gestion du flux de données** Il sera nécessaire qu'il y ait une gestion des données transmises, pour aider à la prise de décision mais aussi pour éviter qu'une transmission non gérée déstabilise le réseau. Dans nos tests, nous avons observé que par exemple une transmission d'un fichier en P2P direct pouvait affecter énormément la transmission d'une vidéo-conférence utilisant une autre topologie en modifiant

les ressources disponibles. La couche réseautique doit donc s'assurer d'être au courant de tous les messages transmis ou du moins se donner une assez bonne marge de manoeuvre pour éviter toute déstabilisation.

**Multi-diffusion** L'utilisation de la multi-diffusion le plus possible permet de diminuer énormément les besoins pour la transmission des messages à plusieurs clients, la couche réseautique doit donc être capable d'implémenter une solution logiciel (ou du moins en utiliser une existante) permettant l'utilisation de multi-diffusion si cela est non supporté par le matériel du réseau.

**Transparence** Les connaissances réseautiques nécessaires pour l'utilisation de plusieurs topologies en même temps et la modification de topologie en plus du travail de monitoring nécessaire tout le long de l'utilisation, font que la couche réseautique se doit d'être la plus automatisée possible, permettant d'offrir une transparence aux développeurs de l'EVC, ceux-ci ne devant pas avoir à faire des modifications pour faire fonctionner la couche.

**Collecte d'information** La couche réseautique doit toujours avoir de l'information sur l'état du réseau pour pouvoir permettre une meilleure analyse de la situation. L'utilisation d'agents est donc nécessaire et le prototype que nous avons utilisé donnait de bons résultats et nous permet d'affirmer que leur utilisation est très utile pour cette collecte.

**Intelligence artificielle** L'utilisation d'une intelligence artificielle est nécessaire pour pouvoir permettre une transparence du mécanisme de fonctionnement de la couche. Cette intelligence se doit d'être capable de prendre des décisions sur le choix de topologie à utiliser en se basant sur les informations à sa disposition, mais doit aussi être en mesure de pouvoir faire une analyse des besoins futurs pour prévoir la quantité de ressources à conserver.

## CHAPITRE 6

### CONCLUSION

Dans ce mémoire, nous avons défini le concept d'environnement virtuel de collaboration, en détaillant les problèmes et les défis techniques mais aussi le côté social d'un tel environnement. Ceci nous a permis de démontrer que la conception d'un EVC fonctionnel permettant une bonne collaboration est une tâche énorme à réaliser et qu'une nouvelle manière de concevoir un EVC devait être envisagée.

#### 6.1 Résumé

La revue de littérature nous a permis de décrire en détails les environnements de collaboration virtuels, et de voir les difficultés liées à la conception de tels environnements.

Cela nous a amené à proposer l'hypothèse de diviser un EVC en une application formée de trois couches :

1. Couche de communication par réseau
2. Gestionnaire d'applications
3. Applications

Une catégorisation des défis reliés au développement d'un EVC nous a permis de démontrer comment notre hypothèse permet de résoudre ou du moins de diminuer les difficultés liées à la conception d'un EVC.

Pour continuer à démontrer la validité de notre hypothèse, nous nous sommes ensuite concentrés sur la couche réseautique, et nous avons défini le rôle que nous envisageons pour cette couche, la manière dont elle pourra remplir ce rôle et les techniques et problèmes rencontrés. Nous avons montré que les différentes topologies de bases ne répondent pas aux besoins de la couche et qu'il est nécessaire d'avoir une adaptation dynamique.

Pour cela, une catégorisation des différents types de données pouvant transiger sur le réseau fut faite, ce qui nous a permis de clarifier les différents critères selon lesquels une topologie pourra être évaluée. Nous avons ensuite défini les différentes mesures que nous avons prises et les méthodes que nous avons utilisées pour ces mesures. Une classification des topologies de base fut ensuite faite, montrant clairement qu'aucune architecture ne peut à elle seule être responsable de la communication pour un environnement virtuel de collaboration.

Ceci nous a finalement amené à introduire notre concept d'adaptation dynamique, d'expliquer de quelle manière à notre avis cela permettra la création d'une couche réseautique pouvant répondre à son rôle. Nous avons ensuite présenté les deux grands problèmes qui attendent toute implémentation d'une adaptation dynamique, pour ensuite conclure en résumant ce que nous avons appris lors de l'implémentation d'un premier prototype.

## **6.2 Vérification de l'hypothèse**

Ce mémoire a permis de mieux cerner ce que représente un environnement virtuel de collaboration en faisant une analyse approfondie de la littérature sur le sujet, mais a en plus permis de proposer une nouvelle méthode de développement d'un EVC. Enfin, il a permis de décrire plus en détails le fonctionnement de la couche réseautique, partie qui est la base de notre nouvelle méthode de développement, et il a aussi introduit une solu-

tion en proposant une adaptation dynamique permettant de détourner un des problèmes majeurs de la conception d'un réseau servant un EVC.

L'analyse des différentes topologies et de leur comportement selon les types de messages à transmettre et les conditions du réseau nous a permis de démontrer l'utilité d'avoir un système proposant plusieurs topologies. La création d'un prototype nous a permis d'identifier les besoins pour la création d'un tel système dynamique et de démontrer son utilité.

Ce mémoire aura premièrement proposé qu'un environnement de collaboration ne doit pas être vu comme une seule application, mais comme un ensemble d'applications fonctionnant ensemble, ces multiples applications communiquant entre elles et créant un environnement virtuel où les usagers peuvent créer un lieu de collaboration sur un sujet qui leur est propre.

Deuxièmement, ce mémoire aura démontré que la conception de la couche réseautique est nécessaire, en plus d'être possible. Les tests et résultats effectués montrent que l'idée d'une couche offrant plusieurs topologies selon les besoins du système est fonctionnelle.

### **6.3 Prochaines étapes**

Une continuation des travaux de ce mémoire peut se faire en deux étapes, soit l'élaboration du concept de gestionnaire d'application que nous avons présenté comme la couche médiane d'un EVC, et l'implémentation complète de la couche réseautique.

L'élaboration du concept de gestionnaire consistera à faire la même chose que nous avons fait pour la couche réseautique dans ce mémoire, ce qui veut dire une définition approfondie du rôle de cette couche, des différentes techniques permettant de remplir ce rôle et des tests préliminaires nécessaires pour confirmer ce rôle et définir les différents

problèmes à résoudre. Une étude d'un tel gestionnaire pourra se baser sur les différents environnements 3D actuellement en développement, telle que Berlin Seefeld et Hoare (2000), qui permettent d'observer les différentes méthodes de gestion d'interface 3D, mais qui présentement ne permettent pas la gestion d'un environnement distribué.

L'implémentation de la couche réseautique aura pour but de mettre au point l'adaptation dynamique présentée dans ce mémoire, en réalisant les différentes tâches à remplir que nous avons présentées. Les deux difficultés majeures de cette implémentation seront la création d'une intelligence artificielle pouvant prendre des décisions et la collecte de données nécessaires pour cette prise de décision.

Une fois ces deux étapes réalisées, il sera possible de commencer à pouvoir créer un EVC, en commençant l'implémentation du gestionnaire d'applications pouvant utiliser la couche réseautique mise au point.

## RÉFÉRENCES

- ALEXANDER (1979). *The timeless way of building*. Oxford University Press.
- BENDFORD, S., GREENHALGH, C., RODDEN, T. ET PYCOCK, J. (2001). Collaborative virtual environments. *Communication of the ACM*.
- BOWERS, J., O'BRIEN, J. ET PYCOCK, J. (1996). Practically accomplishing immersion : Cooperation in and for virtual environments. *ACM CSCW'96*.
- CALVIN, J., DICKENS, A., GAINES, B., METZGER, P., MILLER, M. ET OWEN, D. (1993). The simnet virtual world architecture. *Proceedings of IEEE VRAIS '93*.
- CAMPBELL, A. ET COULSON, G. (1996). A qos adaptive multimedia transport system : design, implementation and experiences. *ACM Multimedia*.
- CARLSSON, C. ET HAGSAND, O. (1993). Dive-a multi-user virtual reality system. *Proceedings of the IEEE Virtual Reality Annual International Symposium*.
- CARLSSON, K. (1993). The process group approach to reliable distributed computing. *Communications of the ACM*.
- CHAWATHE, Y., RATNASAMY, S., BRESLAU, L., LANHAM, N. ET SHENKER, S. (2003). Making gnutella-like p2p systems scalable. *ACM SIGCOMM'03*.
- CUI, Y. ET NAHRSTEDT, K. (2003). Layered peer-to-peer streaming. *Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*.

D. FRANCO, I. GARCÉS, E. L. (1999). A new method to make communication latency uniform : distributed routing balancing. *Proceedings of the 13th international conference on Supercomputing*.

DERIGGI, F. V., KUBO, M. M., SEMENTILLE, A. C., BREGA, J. R. F., DOS SANTOS, S. G. ET KIRNER, C. (1998). Corba platform as support for distributed virtual environments. *IEEE Virtual Reality*.

FRÉCON, E., GREENHALGH, C. ET STENIUS, M. (1999). The divebone - an application-level network architecture for internet-based cves. *VRST'99 - Symposium on Virtual Reality Software and Technology*.

GREENHALGH, C., PURBRICK, J. ET SNOWDON, D. (2000). Inside massive 3. *Proceedings of the third international conference on collaborative virtual environments*.

HAAS, Z., HALPERN, J. Y., LI, L. ET WICKER, S. B. (2000). A decision-theoretic approach to resource allocation in wireless multimedia networks. *Proceedings of the 4th international workshop on Discrete algorithms and methods for mobile computing and communications*.

HARRISON, S. ET DOURISH, P. (1996). Re-place-ing space : The roles of place and space in collaborative systems. *Proceedings of CSCW'96*.

HARTLING, P., JUST, C. ET CRUZ-NEIRA, C. (2001). Distributed virtual reality using octopus. *Proceedings of the Virtual Reality 2001 Conference (VR'01)*, 53.

IL KO, D. ET CHOI, Y. (1999). Design of network protocols for distributed virtual environments. *IEEE ICMCS*.

JEFFAY, K., HUDSON, T. ET PARRIS, M. (2001). Beyond audio l& video : Multimedia networking support for distributed, immersive virtual environments. *IEEE EURO-MICRO*.

JOHNSON, A., ROUSSOS, M. ET LEIGH, J. (1998). The nice project : Learning together in a virtual world. *Proceedings of VRAIS '98*.

JUNGINGER, M. O. ET LEE, Y. (2004). A self-organizing publish/subscribe middleware for dynamic peer-to-peer networks. *IEEE Network*.

KALNIS, P., NG, W. S., OOI, B. C., PAPADIAS, D. ET TAN, K.-L. (2002). An adaptive peer-to-peer network for distributed caching of olap results. *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, 25 – 36.

KHAMAYSEH, Y. M. (1998). *Evaluation of Campus Network for Video Conferencing Traffic*. Mémoire de maîtrise, The University of New Brunswick.

KIRNER, T. G., KIRNER, C., KAWAMOTO, A. L., CANTAO, J., PINTO, A. ET WAZLAWICK, R. S. (2001). Development of a collaborative virtual environment for educational applications. *ACM WEB3D*.

LEIGH, J., JOHNSON, A. ET DEFANTI, T. (1999). A review of tele-immersive applications in the cave researc network. *Proceedings of IEEE VR '99*.

LEIGH, J., YU, O., SCHONFELD, D., ANSARI, R., HE, E., NAYAK, A., GE, J., KRISHNAPRASAD, N., PARK, K., JOO CHO, Y., HU, L., FANG, R., VERLO, A., WINKLER, L. ET DEFANTI, T. A. (2001). Adaptive networking for tele-immersion. *Proceedings of the Immersive Projection Technology*.

LEUNG, W. H. ET CHEN, T. (2001). Creating a multiuser 3-d virtual environment.

*IEEE Signal Processing Magazine*, 18, 9–16.

M, B., CHRISTENSEN, M., GRØNBÆK, K., KROGH, P., MOGENSEN, P., ØRBÆK, P. ET SHAPIRO, D. (2000). Collaborative augmented reality environments : Integrating vr, working materials, and distributed work spaces. *Proc. CVE2000 : The Third International Conference on Collaborative Virtual Environments*, 47–56.

PARK, K. S., CHO, Y. J., KRISHNAPRASAD, N. K., SCHARVER, C., LEWIS, M. J., LEIGH, J. ET JOHNSON, A. E. (2000). Cavernsoft g2 : A toolkit for high performance tele-immersive collaboration. *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*.

PEKKOLA, S., ROBINSON, M., SAARINEN, M.-J. O., KORHONEN, J., HUJALA, S. ET TOIVONEN, T. (2000). Collaborative virtual environments in the year of the dragon. *Proceedings of the third international conference on Collaborative virtual environments*, 11–18.

PETTIFER, S., COOK, J., MARSH, J. ET WEST, A. (2000). Deva3 : architecture for a large-scale distributed virtual reality system. *Proceedings of the ACM symposium on Virtual reality software and technology*, 33–40.

SATO, F., MINAMIHATA, K. ET MIZUNO, T. (2000). A totally ordered and secure multicast protocol for distributed virtual environment. *Seventh International Conference on Parallel and Distributed Systems : Workshops*.

SEEFELD, S. ET HOARE, G. (2000). Berlin : a structured user interface. *Ottawa Linux Symposium 2000, Proceedings*.

SINGH, G., SERRA, L. ET PRG, W. (1994). Bricknet : Sharing object behaviors on the net. *Proceedings of the Virtual Reality Annual International Symposium*.

SINGHAL, S. ET ZYDA, M. (1999). *Networked Virtual Environments, design and implementation*. Addison-Wesley Professional.

SMITH, R. C., PAWLICKI, R. R., LEIGH, J. ET BROWN, D. A. (2000). Collaborative visualeyes. *Proceedings of IPT 2000 : Immersive Projection Technology Workshop*.

SNOWDON, D. ET WEST, A. (1994). The aviary distributed virtual environment. *Proceedings of the 2nd UK VR, SIG conference*.

TRAMBEREND, H. (1999). Avocado - a distributed virtual environment. *Proceedings of IEEE Virtual Reality 1999*.

WATSEN, K. ET ZYDA, M. (1998). Bamboo-a portable system for dynamically extensible, networked, real-time, virtual environments. *Proceedings of the 1998 Virtual Reality Annual International Symposium*.

ZHANG, Q., ZHU, W. ET ZHANG, Y.-Q. (2001). Resource allocation for multimedia streaming over the internet. *IEEE Trans. Multimedia*, 3, 339–355.

## ANNEXE I

### Protocoles réseaux

#### I.1 TCP

Le protocole TCP ("Transmission Control Protocol") est grandement utilisé par plusieurs applications utilisant l'Internet. Il offre une communication garantie entre deux machines, en utilisant les caractéristiques suivantes :

- Le protocole s'occupe de transmettre les messages d'accusé de réception et retransmet les messages si besoin est.
- Une procédure de contrôle de flux existe entre les deux machines, permettant d'éviter que l'application envoyant les messages ne déborde la capacité de réception de la seconde application.
- L'ordre de transmission des paquets est garanti, c'est-à-dire que le message original est livré dans l'état d'envoi.

La garantie de transmission est très avantageuse pour de nombreuses applications, par contre l'envers de la médaille est l'utilisation d'une bande passante plus grande que celle nécessaire pour la livraison du message seul, dû à tous les autres messages permettant d'offrir cette garantie. De même, la garantie de livraison renvoie un message tant que celui-ci n'est pas reçu, ce qui pour certains messages souvent modifiés n'est pas nécessaire.

## I.2 UDP

Le protocole UDP (User Datagram Protocol) peut être vu comme l'opposé du protocole TCP du fait de sa simplicité mais aussi parce que, contrairement au TCP, aucune garantie n'est offerte sur la transmission, tant en ce qui concerne la réception que l'ordre d'arrivée ou l'intégrité du message. Le protocole UDP est défini comme un système offrant un système de "meilleur-effort".

Le gain obtenu par ce système est l'utilisation de beaucoup moins de bande passante pour la livraison d'un message comparativement au TCP. Si cela est utilisable pour certains types de messages, en général le besoin d'un certain niveau de garantie est requis et l'utilisation du protocole UDP permet aux concepteurs de l'application de créer leur propre méthode de garantie couvrant leurs besoins. Par exemple, la détection de messages perdus peut se faire en ajoutant un nombre séquentiel à chaque message envoyé ; lors de la réception, si on détecte une suite de nombres non consécutifs, une demande de retransmission peut être envoyée à l'application émettrice.