N° d'ordre : 2536

THESE

présentée

pour obtenir

LE TITRE DE DOCTEUR DE L'UNIVERSITE DE TOULOUSE
Délivré par l'INPT

École doctorale : Mathématiques, Informatique et Télécommunications de Toulouse

Spécialité : Réseaux et Télécommunications

Par M. Hervé THALMENSY

Titre de la thèse **Emulation de réseaux au niveau IP pour l'expérimentation de services et protocoles de communication. Application aux réseaux satellites.**

Soutenue le 09 Novembre 2007 devant le jury composé de :

M.	FRABOUL Christian	Président
M.	DIAZ Michel	Directeur de thèse
M.	DAIRAINE Laurent	Co-encadrant de thèse
M.	PUJOLLE Guy	Rapporteur
M.	CASTANET Richard	Rapporteur
M.	FLEURY Eric	Membre
M.	SENAC Patrick	Invité

Remerciements

Je tiens à remercier tout particulièrement mon directeur de thèse M. Michel Diaz pour avoir cru en moi et pour m'avoir guidé dans ce travail. Merci pour sa patience et son encouragement à finir un travail commencé il y a longtemps. Son œil critique m'a été très précieux pour structurer le travail et pour améliorer la qualité des différentes sections. Merci également à M. Laurent Dairaine pour son encadrement, son aide ainsi que ses nombreux conseils.

Pendant ces cinq années j'ai été accueilli au Département Mathématiques et Informatique de l'ENSICA dirigé par M. Patrick Senac. Je tiens ici à lui exprimer toute ma reconnaissance pour m'avoir permis de travailler dans de bonnes conditions, pour son soutien et également pour avoir accepté de participer à mon jury.

Ce jury était également composé des professeurs Guy Pujolle et Richard Castanet (rapporteurs) et je tiens aussi à les remercier pour la rapidité avec laquelle ils ont lu mon manuscrit et l'intérêt qu'ils ont porté à mon travail. Merci également aux autres membres du jury qui ont accepté de juger ce travail : les professeurs Eric Fleury et Christian Fraboul.

Ces travaux ont été menés dans une excellente ambiance au DMI de l'ENSICA grâce d'une part aux enseignants chercheurs (Tanguy, Fabrice, Jérôme, Pierre, Emmanuel) mais également au personnel administratif (René : notre chevalier de la légion d'honneur aujourd'hui à la retraite, et Bernard sans qui la soutenance n'aurait pas pu se dérouler correctement). Je leur adresse un merci tout particulier. Je tiens également à remercier M. Yves Caumel pour ses conseils et son aide très précieuse dans le domaine des probabilités, ainsi que M. Yohan Garcia, chercheur invité de l'Université de Karlstad (Suède) pour les différents échanges que nous avons pu avoir.

Je salue également les membres du groupe OLC du LAAS (les doctorants, les chercheurs et Gina) pour leur accueil chaleureux.

Je n'oublie pas non plus les doctorants actuels et anciens de l'ENSICA que j'ai pu côtoyer pendant ces années et avec qui j'ai parfois partagé le bureau : Florestan, Laurent, Ernesto, Amine, Juan, Lei, Alexandre, Thomas. Merci tout particulier à Ahlem, Ali, Benjamin, Tarek et Jérôme qui ont contribué grandement à l'excellent souvenir que je garderai de ces années au DMI. Je tiens également à remercier tout particulièrement Mathieu pour l'aide très précieuse qu'il m'a apportée et dont j'ai un peu abusé notamment sur la fin de la rédaction du manuscrit.

Quant à Emmanuel que je connais depuis le DEA et avec qui j'ai partagé mon bureau, je souhaite ici le remercier pour m'avoir supporté pendant ces années et pour les bons moments que nous avons pu passer.

Merci également à tous ceux et celles qui m'ont soutenu de près ou d'un peu plus loin pendant cette thèse : Catherine et Gilles, Bérangère pour avoir été là notamment à la fin de cette thèse, Jean-Charles, Célia, Samia, Béatrice, Nathalie et Dominique, Anaïs, Cédric, Sophie ...

Enfin, et surtout, je tiens à remercier ma famille. Merci à mes parents de m'avoir toujours encouragé dans les moments de doutes et soutenu dans les moments un peu plus difficiles, je sais tout ce que je leur dois. Mes deux frères Eddy et Gilles ont été d'un énorme soutien et je les en remercie. C'est d'une importance capitale de ne jamais se sentir seul.

Merci donc à tous, et même ceux qui n'ont pas été cités car sans eux rien de tout cela n'aurait été possible.

Table des matières

1 Introduction **13**

2 L'expérimentation réseau **17**
2.1 Introduction 17
2.2 Les différentes approches d'expérimentation 18
2.2.1 Le test en environnement réel 18
2.2.2 La simulation réseau 20
2.3 L'émulation 22
2.3.1 Présentation de l'émulation 22
2.3.2 Niveaux d'émulation 23
Émulation de niveau physique 24
Émulation de niveau liaison 25
Émulation de niveau réseau 25
2.3.3 Besoins 26
2.3.4 Modèle d'architecture d'émulation 27
Type d'émulation 27
Niveau conditionnement 32
Niveau modèle 35
2.3.5 Implémentation des différentes approches 38
Niveau hardware 39
Niveau noyau 39
Niveau utilisateur 39
2.3.6 Comparaison entre les différentes approches d'émulation . . . 39
2.4 Conclusion 41

3 Méthodologie pour l'émulation **43**
3.1 Fonctionnalités nécessaires à l'émulation 43
3.1.1 Reproduction d'expérience 44
3.1.2 Précision des conditions 45
3.2 Architecture fonctionnelle d'émulation 46
3.2.1 Niveau d'émulation retenu 46
3.2.2 Présentation de l'architecture 46
3.3 Une méthodologie pour l'émulation 48

3.3.1 Présentation . . . 48
3.3.2 Phase d'analyse . . . 48
3.3.3 Phase de modélisation . . . 49
Modélisation de la technologie . . . 50
Modélisation de l'architecture d'émulation . . . 51
3.3.4 Phase de vérification . . . 54
3.3.5 Phase d'implémentation . . . 56
3.4 Conclusion . . . 57

4 Emulation de comportements **59**
4.1 Comment modéliser un comportement ? . . . 59
4.1.1 Présentation des différents types de modèles de comportement 60
4.1.2 Les modèles statiques . . . 60
4.1.3 Les modèles dynamiques . . . 61
Scénarios . . . 61
Rejeu de traces . . . 66
Utilisation de simulateur hors ligne . . . 66
4.1.4 Conclusion . . . 67
4.2 L'émulation active . . . 67
4.2.1 Présentation du concept . . . 68
4.2.2 Architecture d'émulation active . . . 68
Description de l'architecture d'émulation . . . 68
Le module de récupération des informations . . . 70
Le module actif de décision . . . 70
4.2.3 Modèles de comportement actifs . . . 71
Modèles basés sur des chaînes de Markov . . . 71
Modèles basés sur des machines à état ou Finite State Machine (FSM) . . . 72
4.2.4 Conclusion . . . 73
4.3 Le trafic concurrent . . . 73
4.3.1 Trafic réel . . . 74
4.3.2 Trafic virtuel . . . 74
4.3.3 Trafic généré à base de traces . . . 75
4.3.4 Discussion . . . 75
4.3.5 Les modèles basés sur des équations de comportement . . . 76
4.4 Conclusion . . . 79

5 Cas d'étude : émulation satellite dans le projet EuQoS **81**
5.1 Présentation du projet EuQoS . . . 81
5.1.1 Objectifs . . . 82
5.1.2 Architecture EuQoS de bout en bout . . . 82
5.1.3 La Qualité de Service de bout en bout dans EuQoS . . . 83

Les différentes classes de service EuQoS 83
5.1.4 Test du protocole Enhanced Transport Protocol (ETP) 83
Présentation du protocole ETP 84
Définition du besoin d'émulation 84
5.1.5 La plate-forme d'expérimentation générique 85
Architecture générique 85
Architecture de la plate-forme utilisée 86
Implémentation 86
5.1.6 Résultats d'expériences 91
Conclusion 98
5.2 Présentation du contexte satellite 98
5.2.1 Architecture considérée 99
5.2.2 La norme Digital Video Broadcasting (DVB) 99
Le MPEG-2 100
Le DVB-S 100
Le DVB-RCS 101
5.2.3 Principe de fonctionnement de l'allocation en DAMA 104
Cas 1 105
Cas 2 106
Cas 3 106
Cas 4 107
5.3 Intégration de l'accès satellite à EuQoS 107
5.3.1 Objectif 107
5.3.2 Intégration du système satellitaire émulé dans l'architecture EuQoS 108
5.3.3 Modélisation et implémentation de l'émulation satellite 110
5.3.4 Résultats 112
Description de l'expérience 116
Résultats 116
5.3.5 Conclusion 117

6 Conclusion **119**

Bibliographie **125**

Table des figures

2.1 Expérimentation en environnement réel ou émulé 23
2.2 Différents niveaux d'émulation réseau 24
2.3 Classification d'émulation réseau 28
2.4 Architecture d'émulation centralisée 31
2.5 Fonctionnement de Dummynet 33
2.6 Emulation virtuelle 36
2.7 Architecture émulateur par simulation temps réel 37
2.8 Récapitulatif des différentes approches 40

3.1 Architecture classique d'un émulateur 47
3.2 Méthode d'émulation 48
3.3 Modèle de technologie d'une liaison satellite 50
3.4 Modèle UML d'émulation 51
3.5 Diagramme de classes du canal d'émulation 52
3.6 Composite Structure Diagram du canal d'émulation 53
3.7 Exemple de modèle d'émulation de délai 54
3.8 Principe de vérification 56
3.9 Description du comportement du processeur 57
3.10 Comportements du modèle et de l'observateur 57

4.1 Evolution de la fiabilité de la transmission selon différentes conditions de propagation 62
4.2 Exemple de schéma RELAX NG compact associé à un fichier XML . 63
4.3 Extrait d'un scénario d'émulation pour la partie scénario 65
4.4 Architecture d'émulation active 69
4.5 Exemple de chaîne de Markov modélisant des pertes probabilistes . . 71
4.6 Exemple de machine à état 72
4.7 Génération de trafic réel 74
4.8 Génération de trafic virtuel 75
4.9 Deux flux TCP se partageant un canal 77
4.10 Exemple de modèle de saturation / dé-saturation 78
4.11 Courbe de saturation 79

5.1 Architecture EuQoS de bout en bout 83

5.2 Vision globale des mécanismes du protocole ETP 84
5.3 Architecture de la plate-forme d'émulation NINE 85
5.4 Architecture de la plate-forme d'émulation 86
5.5 Abstraction réseau EuQoS 87
5.6 Exemple de description du domaine 88
5.7 Extrait de description d'un chemin de QoS 89
5.8 Exemple de scénario d'émulation 90
5.9 Débit au niveau applicatif pour le cas non streamé en utilisant UDP 92
5.10 Taux de pertes pour une application non streamée utilisant UDP . . 92
5.11 Débit au niveau applicatif pour le cas non streamé en utilisant TCP 92
5.12 Taux de pertes pour une application non streamée utilisant TCP . . 93
5.13 Débit au niveau applicatif pour le cas non streamé en utilisant ETP (CdS RT Interactive) 93
5.14 Taux de pertes pour un profil d'application non streamée utilisant ETP (CdS RT Interactive) 93
5.15 Débit au niveau applicatif pour le cas non streamé en utilisant ETP (CdS Standard) 94
5.16 Taux de pertes pour un profil d'application non streamée utilisant ETP (Cds Standard) 94
5.17 Profil de trafic considéré 95
5.18 Débit au niveau applicatif pour le cas streamé en utilisant UDP . . . 96
5.19 Taux de pertes pour une application streamée utilisant UDP 96
5.20 Débit au niveau applicatif pour le cas streamé en utilisant TCP . . . 96
5.21 Taux de pertes pour une application streamée utilisant TCP 97
5.22 Débit au niveau applicatif pour le cas streamé en utilisant ETP (CdS RT Interactive) 97
5.23 Taux de pertes pour un profil d'application streamée utilisant ETP (CdS RT Interactive) 97
5.24 Architecture satellite étudiée 100
5.25 Format d'une supertrame DVB-RCS 103
5.26 Principe de requête / allocation du DAMA 106
5.27 Liaison satellite dans le réseau EuQoS 108
5.28 Intégration de l'architecture satellite à EuQoS 109
5.29 Modèle de technologie d'un accès satellite 110
5.30 Diagramme de cas d'utilisation de l'accès satellite 111
5.31 Modélisation du VBDC 112
5.32 Plate-forme d'expérimentation 113
5.33 Délai des paquets en VBDC 114
5.34 Evolution du délai des paquets dans le cas d'une classe VBDC à 128K 115
5.35 Délai des paquets en CRA et VBDC 115
5.36 Scénario de flux concurrents 116
5.37 Comparaison des résultats obtenus par émulation de niveau 2 et 3 pour le cas du débit du flux TCP 117

5.38 Evolution du délai de bout en bout du trafic CBR dans le cadre d'application de VoIP 118

Chapitre 1

Introduction

Avec l'avènement d'Internet et l'augmentation des capacités des matériels informatiques, de nombreux protocoles et applications ont vu le jour. Ces applications ou protocoles proposent de nouveaux services multimédia aux utilisateurs, comme l'interactivité, en profitant de l'augmentation des ressources disponibles. Ainsi, la phase d'expérimentation tient une place importante dans le processus de recherche scientifique en général, et particulièrement dans le contexte des réseaux et protocoles de communications. Elle permet de vérifier que le protocole ou l'application en phase de développement respecte bien les propriétés spécifiées et également que le comportement observé correspond aux attentes de l'utilisateur. La validation fonctionnelle permet aussi de réduire le risque d'erreur et donc de garantir des logiciels de qualité. C'est une technique de validation, très répandue dans le milieu industriel qui, de plus, est présente à toutes les étapes du cycle de développement logiciel.

Que ce soit pour un protocole ou une application, le développeur est confronté aux mêmes problèmes de test et d'évaluation. En effet, lors du développement d'un protocole ou d'une application répartie, il est nécessaire d'évaluer le comportement des différents mécanismes proposés, dans le but de valider leur fonctionnement et par la suite de pouvoir établir des comparaisons avec les mécanismes existants. Cette phase de test permet également de configurer les différents mécanismes en déterminant les valeurs des paramètres dans des conditions limites. Pour évaluer un nouveau protocole de communication ou une nouvelle application, il existe trois grandes approches :

– L'expérimentation en environnement réel qui utilise un support d'infrastructure réelle afin de mener des expérimentations. Cette approche assure implicitement un excellent niveau de réalisme. Nous prendrons comme définition du réalisme, celle donnée dans [6] comme étant *« la capacité offerte à un chercheur réseau de soumettre un protocole ou une architecture réseau à des conditions réseau qui ressemblent le plus fidèlement possible à celles rencontrées lors d'un dé-*

ploiement réel ». L'expérimentation en environnement réel se trouve cependant limitée par le manque de contrôle et de reproductibilité proposés à l'expérimentateur. En effet, il est très difficile, avec ce type d'approche, d'offrir un contrôle sur l'ensemble des paramètres de l'environnement. Prenons pour exemple une expérience dans un contexte satellite où on se rend compte que de nombreux paramètres sont hors de contrôle comme les conditions climatiques ou alors des perturbations électromagnétiques. De plus, il est quasiment impossible de pouvoir reproduire deux fois de suite la même expérience.

- La seconde solution, la simulation, propose une modélisation complète de l'environnement ainsi que du protocole ou de l'application à évaluer dans un environnement synthétique et parfaitement contrôlable. Cependant, la limitation forte de la simulation provient de la nécessité de modéliser aussi bien l'environnement que l'application ou le protocole pour mener une expérimentation. La question importante qui se pose et qui est valable pour tous les modèles, est la suivante : comment s'assurer de la validité du modèle ? De plus, la simulation travaille en temps logique, ce qui empêche l'évaluation d'applications interactives par exemple.
- Au carrefour de ces deux approches, l'émulation est une méthode d'évaluation dont le principe consiste à faire fonctionner et évaluer des implémentations réelles de protocoles ou d'applications distribuées dans un contexte où, une partie de l'architecture de communication est simulée en temps réel. Le service ainsi offert par l'émulateur propose une interface d'accès au service équivalente au réseau réel reproduit. De ce fait, le système d'émulation peut être utilisé par des implémentations réelles d'applications ou de protocoles sans aucune modification.

Nous avons choisi de nous intéresser à l'émulation même si elle impose certaines contraintes. En effet, étant donné que l'émulation fournit un service en « temps réel », les modèles des couches basses ne devront pas être trop complexes sous peine de ne pas pouvoir tenir les contraintes temps réel. Tout le concept d'émulation repose sur cette faculté à pouvoir tenir les contraintes imposées par le temps réel. Ainsi, de nombreux outils d'émulation actuels permettent la mise en oeuvre de l'émulation, parfois au détriment du réalisme de l'émulation rendue, pour être en mesure de garantir ces contraintes. En effet, les ressources matérielles nécessaires sont liées à la complexité des modèles. Plus les modèles seront complexes, plus les ressources à mettre en oeuvre seront importantes en termes de calcul mais aussi de matériel. Cette complexité a également un impact sur le réalisme de l'émulation. Pour améliorer les performances, des solutions existent : l'utilisation de grilles de calcul ou des clusters par exemple. Mais ce choix de parallélisation de l'émulation introduit des délais et de nouvelles contraintes au niveau de la gestion du système d'émulation.

De plus, face au nombre important d'outils existants, pour un utilisateur novice dans le domaine de l'émulation, il se peut que le choix d'un outil s'avère compliqué du fait du nombre important de solutions potentielles.

L'objectif des travaux présentés dans cette thèse est de définir une méthode permettant de modéliser le réseau réel (cible) à l'échelle d'un laboratoire, ceci dans le but de déduire un émulateur correspondant aux besoins tout en essayant de préserver un maximum de précision. La précision des modèles d'émulation sera abordée avec la proposition d'une approche d'émulation dont le but est d'étendre les modèles d'émulation dynamiques existants. Nous utiliserons la plate-forme NINE pour mener nos expérimentations. Elle se comporte comme une boîte noire rendant un service aux applications ou protocoles équivalent à ceux d'une réseau réel. Il est possible de connecter des machines hétérogènes sur lesquelles les applications ou protocoles seront déployés. Ceci garantit un niveau de flexibilité de la plate-forme.

Ce manuscrit s'articule autour de quatre chapitres de la façon suivante :

Le premier chapitre présente un état de l'art des différentes approches de tests existantes en présentant leurs avantages et leurs limites. Nous présenterons quelques outils permettant la mise en oeuvre de ces approches. Nous nous intéresserons plus particulièrement à l'approche d'émulation et présenterons les besoins, les différents niveaux d'émulation et une classification des différents outils.

Le chapitre 2 insistera sur les besoins qui nous paraissent essentiels à l'émulation. Après avoir présenté l'architecture fonctionnelle d'un système d'émulation, nous proposerons une méthode d'émulation avec ses différentes phases que nous détaillerons.

Le chapitre 3 se consacre à la modélisation de comportements. Il présente les différents types de modèles possibles d'émulation selon les besoins. Pour répondre à l'objectif de précision et d'une certaine façon à celui de réalisme, nous proposerons d'étendre les modèles d'émulation dynamiques avec les modèles actifs. L'architecture d'émulation active sera présentée ainsi que les approches de modélisation selon la vision du problème (probabilistique, événementiel, temporel). Nous nous intéresserons également à la modélisation des événements extérieurs se produisant sur un réseau et nous prendrons pour exemple le trafic concurrent. Nous proposerons une approche d'émulation active basée sur des équations de comportement.

Le chapitre 4 propose deux cas d'étude basés sur le projet européen EuQoS incluant des évaluations expérimentales des solutions. Le premier permet de montrer l'utilisation de la méthode d'émulation et le second présente les résultats obtenus avec l'émulation active pour la modélisation d'un accès à une ressource partagée dans le cadre d'une liaison satellite. La modélisation d'un DAMA sera présentée ainsi que les résultats obtenus.

Enfin, nous conclurons ce manuscrit en présentant un bilan des contributions de cette étude et dégagerons les différentes pistes possibles de travail à explorer dans le futur.

Chapitre 2

L'expérimentation réseau

Dans ce chapitre, j'aborderai les points suivants :

- ⋆ les différents environnements d'expérimentation ;
- ⋆ le test en environnement réel ;
- ⋆ le test en environnement simulé ;
- ⋆ le test en environnement émulé ;
- ⋆ une classification des approches d'émulation ;

2.1 INTRODUCTION

De nos jours, la phase d'expérimentation occupe une place importante dans le cadre de la recherche en systèmes informatiques et particulièrement en ce qui concerne les études portant sur le domaine des réseaux et protocoles de communication. Outre l'évaluation de la faisabilité d'une solution, l'expérimentation vise à établir qu'un système vérifie les propriétés exigées par sa spécification. En somme, elle doit permettre de réduire le risque d'erreurs de conception et d'implémentation et ainsi garantir la qualité des logiciels. Quelle que soit la méthode de conception d'un nouveau protocole de communication, il est impératif de valider son comportement théorique grâce à des mesures expérimentales. Dans ce contexte d'ingénierie des protocoles, l'expérimentation est donc utilisée pour des besoins divers liés à la validation de ces protocoles tels que :

* *la validation fonctionnelle d'applications ou de protocoles :* une plate-forme d'expérimentation peut être utilisée pour réaliser la validation fonctionnelle d'une application ou d'un protocole (dans diverses conditions réseau) et ainsi de mettre en évidence des erreurs de conception, des problèmes d'implémentation et donc permettre l'amélioration du logiciel.

* *l'analyse de performance de protocoles :* les performances d'un protocole peuvent être testées dans des conditions réseau spécifiques (par exemple, réseau congestionné, diverses technologies sous-jacentes, conditions limites) et comparées à d'autres protocoles afin d'analyser et de comparer les performances.

* *la démonstration :* une plate-forme d'expérimentation peut également être utilisée dans le cadre d'une démonstration d'une application ou d'un protocole de communication dans des conditions particulières du réseau.

Plusieurs environnements permettent de réaliser des expérimentations : l'environnement réel (une infrastructure opérationnelle est utilisée pour tester une application ou un protocole réseau donné), la simulation (un simulateur est utilisé dans un environnement synthétique permettant de modéliser l'application ou le protocole à tester) et enfin, l'émulation (alternative aux précédents environnements proposant de modéliser uniquement le réseau étudié et d'utiliser des implémentations réelles des protocoles ou applications).

L'objectif de ce chapitre est de présenter les trois approches possibles pour l'expérimentation réseau tout en établissant une comparaison basée à la fois sur des capacités fonctionnelles et non fonctionnelles des approches. Dans une seconde partie, ce chapitre présentera de manière plus précise l'expérimentation en environnement émulé. Enfin, différents outils d'émulation seront présentés avec leurs caractéristiques, dans le but d'établir une classification.

2.2 LES DIFFÉRENTES APPROCHES D'EXPÉRIMENTATION

Dans le contexte de développement d'applications et de protocoles, la phase d'analyse est généralement suivie d'un développement de maquettes ou de prototype. Cette démarche permet de raffiner les besoins et les problèmes de manière plus réaliste et d'évaluer les gains des solutions proposées. Ainsi dans le contexte d'expérimentation, plusieurs types d'environnements sont traditionnellement utilisés pour le test de propositions de protocoles : l'environnement réel, l'environnement simulé et l'environnement émulé. Ces trois environnements vont maintenant être présentés.

2.2.1 Le test en environnement réel

Dans un objectif de recherche de réalisme et de précision, le test sur une infrastructure réelle ou opérationnelle, est de loin celui qui offre de meilleurs résultats, dans le sens où ce mode d'expérimentation utilisera un réseau opérationnel (on entend par opérationnel, un réseau basé sur une technologie réellement déployée) ainsi que des implémentations réelles d'applications ou de protocoles.

L'expérimentation en environnement réel consiste donc à utiliser le support d'infrastructures réseaux, par exemple, un réseau local Ethernet, un réseau métropolitain ou encore l'Internet. Ce réseau est alors utilisé comme support de communication pour y évaluer les architectures de communications développées. Ce type de démarche assure implicitement le réalisme des mesures et le bon fonctionnement des architectures dans l'environnement. Cependant, ce type d'expérimentation peut

s'avérer coûteux à mettre en oeuvre et ne permet pas toujours un contrôle précis des conditions d'expérimentation et ce, plus particulièrement, dans un contexte de réseaux basés sur des technologies spécifiques. Ainsi, par exemple, le test dans le cadre d'un réseau local offrira naturellement une bande passante élevée et un taux de perte quasiment nul, alors que l'utilisation d'Internet induira une qualité de service de communication dépendant de paramètres difficilement contrôlables (état de congestion du réseau lors de l'expérience, topologie du réseau, choix de la localisation des hôtes, etc.).

Prenons pour exemple le projet DIPCAST [17] dont l'objectif était d'étudier un service IP Multicast sur un support réseau satellite géostationnaire transparent supportant des services IP Multicast. Afin de développer les protocoles utilisant cette technologie, il aurait été nécessaire de disposer d'un accès à cette ressource satellite. Or, cette ressource s'est révélée être très coûteuse à mettre en oeuvre car elle n'existait pas ou du moins n'était pas déployée lors de ce projet. Ainsi, il apparaissait clairement qu'il faille disposer d'un autre moyen permettant la mise en oeuvre de l'expérimentation.

De plus, dans ce cadre d'expérimentation, l'expérimentateur n'a pas de contrôle sur les conditions réseau et de ce fait, la reproduction d'une même expérience s'avère être quasiment impossible. Reprenons le cas de notre liaison satellite géostationnaire du projet DIPCAST. Dans ce cadre, indéniablement, l'expérimentateur n'a pas de contrôle sur les conditions météorologiques. Ceci peut s'avérer être problématique lors du développement d'applications ou de protocoles. En effet, un développeur de protocole peut avoir besoin de modifier des paramètres de son protocole et il lui est donc nécessaire de pouvoir reproduire une même expérience pour identifier le paramétrage optimal. Cette reproductibilité est également nécessaire pour pouvoir comparer le protocole en développement avec les protocoles existants. Or, pour que ces comparaisons soient fiables, il faut que les conditions soient exactement les mêmes ce qui ne peut être assuré dans un test en environnement réel puisque les conditions peuvent changer très fortement entre deux tests.

On peut également noter que ce type d'expérimentation ne permet pas de mettre en oeuvre le test aux limites c'est-à-dire dans des conditions qui sortent de l'ordinaire, qui ne sont peut être pas forcément réalistes, mais qui peuvent néanmoins arriver (fortes perturbations électromagnétiques, conditions météo extrêmes).

Cela dit, il existe certains projets tels que Planetlab [39] reposent sur cette approche. Ainsi, ce projet est basé sur le principe de l'utilisation de noeuds situés un peu partout à travers le monde et reliés entre eux grâce à l'Internet. Étant donné que ce projet inclut de vrais routeurs et de vrais protocoles réseau, l'émulation est transparente pour l'utilisateur. Planetlab se résume en fait, à une plate-forme distribuée proposant un nombre important de machines hôtes. Cependant, il souffre des mêmes inconvénients liés à l'utilisation de ce cadre pour l'expérimentation : manque de contrôle sur les conditions de test, impossibilité de reproduire deux fois la même expérience.

En conclusion, l'expérimentation en environnement réel permet de fait, d'obtenir les meilleurs résultats en termes de précision et de réalisme au niveau des conditions d'expérimentation mais à un coût pouvant être important, avec un faible niveau de contrôle et de reproductibilité des conditions de test.

2.2.2 La simulation réseau

Pour pallier principalement les difficultés de mise en oeuvre d'expérimentation en environnement réel (manque de contrôle au niveau des paramètres, quasi impossibilité de reproduire une même expérience, etc.), un moyen classique permettant l'évaluation de protocole est la simulation de réseau. Ce type d'approche d'expérimentation propose d'utiliser un simulateur dans un environnement synthétique et contrôlable. Les simulateurs de réseau sont des outils logiciels généralement utilisés pour le développement, le test ou encore le "debugage" d'applications ou de protocoles réseau. La simulation offre une flexibilité maximale de part l'aisance de la mise en oeuvre d'architectures dont les règles de fonctionnement sont entièrement contrôlées. Il est égalment possible, grâce à des modèles, de reproduire un comportement ou une condition réseau donnée. Cette approche d'expérimentation implique une modélisation complète de l'environnement et de l'architecture considérée en utilisant un formalisme et des outils liés au système de simulation utilisé : file d'attente, réseaux de Pétri, etc. De nombreux protocoles sont actuellement modélisés et bon nombre de ces modèles sont reconnus par la communauté scientifique. De ce fait, la simulation est souvent utilisée par les chercheurs pour mener leurs expérimentations.

Des différents modes de simulation réseau existants, la simulation à événements discrets est la plus utilisée par la communauté scientifique. Elle permet de modéliser le système à étudier sous forme d'une séquence d'événements qui dirigent l'évolution discrète du système. Un simulateur à événements discrets se caractérise par le fait que les changements d'états dans le réseau simulé (événements) se produisent à des instants (sans durée) répartis de manière discrète sur l'axe des temps. Ceci permet donc, dans le cas de réseaux complexes, c'est-à-dire composés d'un nombre d'événements importants, de pouvoir traiter les événements sans être soumis à la contrainte du temps réel. Ainsi par exemple, deux événements normalement espacés de 10 secondes dans la réalité pourront être traités sans tenir compte de cette contrainte de temps réel et, suivant la charge du simulateur et la complexité du modèle envisagé, dans un temps logique variable pouvant être proche ou très éloigné du temps réel. Il existe de nombreux outils permettant de mettre en oeuvre la simulation réseau (ns2 [8], OPNET [47], QualNet [51], GloMoSim [67]).

ns2 fait partie des outils de simulation les plus utilisés. Il s'agit d'un simulateur réseau développé par un conglomérat de centres de recherche comprenant AT&T research institute at Berckeley (ACIRI), Xerox PARC et Sun MicroSystems. Il s'agit d'un simulateur à événements discrets orienté objets qui s'inscrit dans le cadre du projet VINT. Ce projet a pour mission de faciliter l'étude des comportements réseaux et l'interaction entre différents protocoles. Le simulateur propose un environnement permettant de créer et d'étudier des algorithmes de routage, des protocoles de trans-

port, des applications, des politiques de files d'attente etc. A l'origine, il avait été conçu pour étudier en particulier le protocole TCP mais grâce au support de la communauté scientifique, de nombreux protocoles ont été développés. Ce simulateur est un logiciel libre à code source ouvert.

OPNET [47] est également un simulateur réseau à événements discrets qui permet la modélisation et la simulation de réseaux de communication grâce à ses bibliothèques conséquentes de modèles (routeurs, commutateurs, stations de travail, serveurs etc.) et de protocoles (TCP/IP, FTP, FDDI, Ethernet, ATM etc.). OPNET et ns2 sont deux outils de simulation qui proposent de base un grand nombre de modèles.

OPNET est une solution commerciale alors que ns2 est gratuit. Les buts de ces deux outils sont relativement proches et ils permettent de reproduire globalement les mêmes types de comportements. Cependant, l'évaluation de performances réalisée dans [25] a permis de mettre en comparaison les résultats obtenus sur un environnement réel pour des trafics CBR et FTP avec les résultats obtenus par simulation. Leurs conclusions montrent qu'un effort doit être fait pour qu'il y ait une adéquation entre les simulateurs et les bancs d'essai.

Il existe également la simulation distribuée qui consiste à faire coopérer plusieurs processus de simulation pour atteindre un objectif commun. La distribution peut être alors considérée comme un moyen permettant d'accroître sensiblement la puissance de calcul. Ainsi, il s'agit de partager les ressources de calcul par des moyens traditionnels de la programmation parallèle et du calcul distribué afin d'inter opérer des simulateurs propriétaires pour qu'ils puissent participer à une simulation de plus grande envergure, les simulateurs élémentaires pouvant être géographiquement distants.

Avec ce mode d'expérimentation, la phase de modélisation s'avère complexe. En effet, la validité et le niveau de précision de l'expérience restent à démontrer [49]. Prenons pour exemple les travaux présentés dans [50] qui ont montré à quel point il est difficile de simuler des réseaux complexes et à fortiori l'Internet. En outre, la mise en oeuvre réelle peut parfois être relativement éloignée du modèle de départ. Autre inconvénient qui au demeurant peut être également un avantage, le fait que la simulation soit basée sur un temps logique ou simulé c'est-à-dire que le temps effectif de simulation varie en fonction de la complexité des modèles considérés. Ainsi, cette approche d'expérimentation qui implique la modélisation du réseau cible et celle de l'application considérée, ne permet pas d'utiliser une implémentation réelle du protocole ou de l'application à tester.

En conclusion, la simulation réseau permet de mettre en oeuvre des expérimentations dans un environnement contrôlable, et facilement reproductible en modélisant à la fois le support réseau considéré mais également les applications ou protocoles à tester. Mais elle ne permet pas l'intégration d'implémentations réelles du fait qu'elle fonctionne en temps logique. Cette modélisation implique parfois des hypothèses ou des simplifications qui peuvent éloigner le comportement simulé du comportement réel et ainsi fausser les résultats de mesure.

2.3 L'ÉMULATION

2.3.1 Présentation de l'émulation

L'émulation peut être vue comme étant une alternative à l'expérimentation en environnement réel et à la simulation c'est-à-dire qu'elle permet de reproduire le comportement d'un système donné sur un autre système. Pour cela, le système d'émulation doit être surdimensioné par rapport au système cible. Dans le pire des cas, il doit offrir des caractéristiques au moins équivalentes à celles du système dont on veut reproduire le comportement. Ainsi, avec l'accroissement des performances des équipements informatiques et la diminution sensible de leurs coûts observés depuis quelques années, le développement des outils d'émulation a connu une évolution importante. Mais tout d'abord, que signifie émuler un système ? La définition du terme émulation est la suivante : « *émulation : qui cherche à imiter, à égaler.* »

Le concept d'émulation introduit donc l'idée de reproduire un comportement ou une situation particulière d'un système donné. D'une manière générale, un émulateur reproduit les fonctionnalités offertes par un système cible sur un autre système pouvant être totalement différent du système cible (on entend par système cible, le système dont on veut reproduire le comportement). Le système obtenu aura en fin de compte, un comportement similaire au système cible.

Dans le cadre de l'émulation réseau de réseaux informatiques, le système considéré est un réseau informatique offrant différents services de communication, basé sur un ensemble de protocoles de communication. Le principal objectif de l'émulation réseau est de proposer un environnement d'expérimentation permettant de tester en temps réel des implémentations réelles de protocoles ou d'applications, de façon à pouvoir évaluer leurs propriétés fonctionnelles (le protocole ou l'application fonctionnent-ils ?) et non fonctionnelles (comment le protocole ou l'application se comportent ils dans une situation réseau particulière). Une même application pourra être exécutée de la même manière et sans modifications sur le système réel et dans l'environnement émulé. Sur la figure 2.1 on constate que les services rendus par le système réel et par le système émulé sont équivalents pour les applications ou les protocoles.

Un émulateur « parfait » devrait donc offrir les mêmes services (propriétés fonctionnelles) et le même niveau de performances (propriétés non fonctionnelles) que le système reproduit. L'émulation de réseau permet également de reproduire un comportement réseau donné sur un environnement contrôlable. Les comportements réseaux peuvent être relativement variés et basés sur des technologies réelles (lien satellite, ADSL, sans fil WiFi etc.). L'émulation permet d'avoir un contrôle équivalent à celui de la simulation sur les paramètres du réseau mais la différence avec la simulation se situe dans la possibilité d'utiliser des implémentations réelles d'applications ou de protocoles. Ainsi, elle permet de tester un protocole aux conditions limites, ce qui est très difficile à faire dans le cadre de l'expérimentation en environnement réel, et surtout de reproduire une même expérience plusieurs fois.

Il y a cependant quelques inconvénients liés à l'utilisation de l'émulation. Tout d'abord, il faut disposer d'un support réseau sur-dimensionné par rapport au réseau

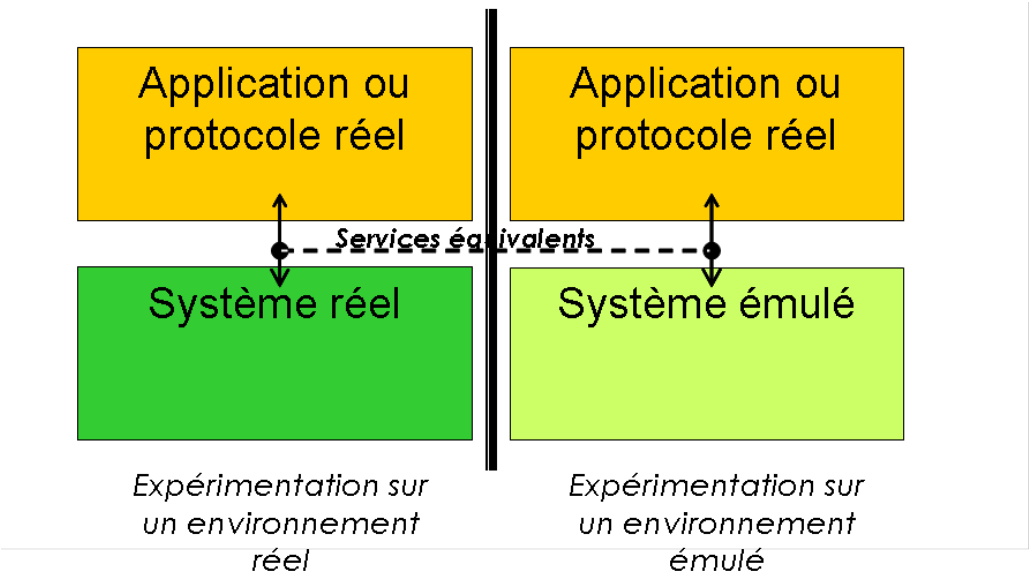


Fig. 2.1 Expérimentation en environnement réel ou émulé

à émuler. Ainsi, il ne sera jamais possible d'émuler un réseau de coeur par exemple sur un réseau de type LAN Ethernet. Ensuite, étant donné que l'émulation doit permettre l'expérimentation de protocoles ou d'applications réelles, elle doit fournir un service aux applications ou aux protocoles respectant les contraintes du temps réel. Ainsi pour pouvoir tenir ces contraintes, il ne faut pas que les modèles décrivant le fonctionnement des couches basses du réseau à émuler soient trop complexes. On entend par modèles complexes, des modèles de réseaux dont le nombre d'équipements est important et qui demandent beaucoup en ressources. Si les modèles sont trop complexes et qu'ils induisent des temps de calcul trop importants pour fournir les résultats, les contraintes temps réel risquent de ne plus être respectées. Ainsi le niveau de réalisme de l'émulation rendue se verrait affecté. A l'inverse, si les modèles sont trop simplistes, les contraintes temps réel seront respectées, mais dans ce cas également le niveau de réalisme obtenu risque d'être touché. En effet, un modèle trop simple pourrait ne pas représenter exactement la réalité. La principale difficulté sera donc de trouver un compromis entre complexité et réalisme des modèles.

2.3.2 Niveaux d'émulation

L'un des objectifs de l'émulation de réseau est de faciliter le développement et le test d'applications ou de protocoles. En fonction des besoins, il sera possible avec l'émulation de choisir le niveau d'abstraction (au sens OSI [34]du terme) où l'on veut se placer pour modéliser le réseau ou la condition réseau que l'on souhaite évaluer. Ainsi il sera donc possible de définir le niveau d'agrégation des différentes fonctions ou services réseau à émuler et permettre une mise en oeuvre en fonction des besoins au sein de l'outil d'émulation.

Supposons que l'on veuille évaluer une application ou un protocole de routage : le niveau d'émulation, et donc le niveau d'abstraction qui va être nécessaire, ne sera pas le même. Ainsi, pour évaluer un protocole de routage il faut modéliser et reproduire en temps réel le comportement des couches de niveau inférieur c'est-à-dire liaison et physique. L'émulateur doit donc rendre un service de niveau liaison. De même, pour évaluer une application, il est possible pour l'émulateur de rendre là encore un service de niveau liaison. Cependant, il faut disposer des implémentations réelles des protocoles de niveau transport et de niveau réseau (routage par exemple) qui sont utilisés dans le réseau à émuler, or, ce n'est pas toujours le cas.

Ainsi, si les implémentations réelles des protocoles de niveau réseau ne sont pas disponibles, il est préférable d'utiliser un émulateur rendant un service de niveau réseau, c'est-à-dire un émulateur qui va reproduire en temps réel le comportement des couches jusqu'au niveau de la couche réseau. La figure 2.2 présente cette notion de niveau d'émulation et de service rendu par un émulateur.

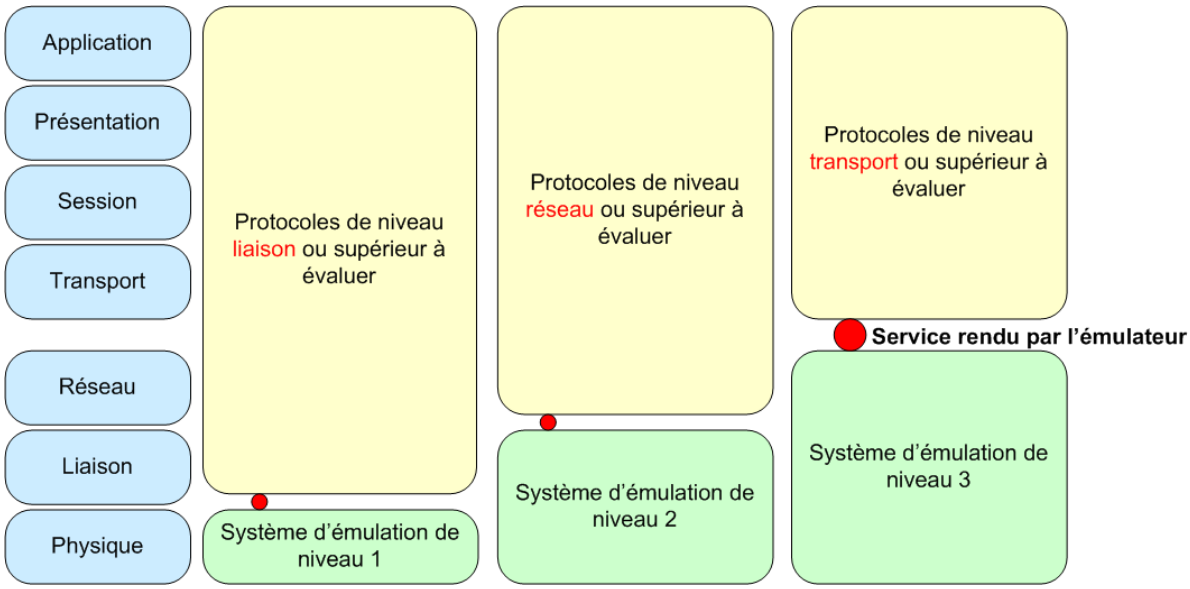


Fig. 2.2 Différents niveaux d'émulation réseau

Ainsi, l'émulation réseau peut être mise en oeuvre à différents niveaux et ce, en fonction des besoins par rapport à l'émulation. Ces besoins permettront de définir le niveau d'abstraction. De ce niveau d'abstraction dépendra la mise en oeuvre de mécanismes ou de protocoles réseau. Moins l'abstraction sera importante, plus le nombre de protocoles à déployer réellement sera important ainsi que le nombre de paramètres à manipuler.

Émulation de niveau physique

Une émulation de niveau physique, ou de niveau 1, consiste à reproduire en temps réel les services et la qualité de service de la couche physique. Il s'agirait de reproduire, dans le cas de réseaux sans fil, les effets du signal reçu au niveau des cartes

réseau. Ceci reviendrait à équiper les cartes réseau d'atténuateurs. Les travaux présentés dans [52][2] ou encore dans [57] proposent des solutions d'émulation utilisant une grille (dans le cas de ORBIT), des atténuateurs de signal ainsi que des générateurs de bruits parasites, pour tenter d'émuler des réseaux sans fils (802.11, satellite, etc.). Cette approche d'émulation permet d'obtenir des résultats proches de ceux obtenus lors du test réel. Cependant on pourrait se demander si les résultats obtenus sont réalistes par rapport au niveau de contrôle sur les communications sans fil entre deux noeuds. Etant donné que les communications passeront dans l'air, il se peut qu'il y ait des perturbations extérieures qui peuvent influer sur la qualité de l'émulation rendue. Les travaux de [24] vont dans ce sens, en montrant que même en paramétrant correctement les interfaces, la reproduction exacte d'une même expérience ne peut être assurée.

Donc à moins de ne reproduire que des réseaux filaires, l'émulation de niveau physique est compliquée à mettre en oeuvre. En effet, les couches physiques radio et Ethernet sont très différentes d'où la difficulté de les reproduire. Il faut noter qu'il existe très peu de plate-formes permettant de mettre en oeuvre ce niveau d'émulation.

Émulation de niveau liaison

Comme indiqué sur la figure 2.2, l'émulation de liaison de données s'attache à reproduire les mécanismes des niveaux physique et liaison de données, et fournit aux applications ou protocoles un service de niveau liaison de données. Ce type d'émulation est réaliste mais la contrepartie se situe au niveau des ressources (en terme d'équipements) qui peuvent être relativement importantes pour la mettre en oeuvre. En effet, supposons que l'on veuille évaluer le comportement d'un protocole de routage sur un réseau ADSL par exemple. La mise en oeuvre d'une émulation d'une liaison ADSL au niveau 2 demandera le déploiement d'équipements tels que : un DSLAM, un BAS, un splitter, un DNS, des modems ADSL (un par utilisateur), etc. Il apparaît clairement que cette approche sera précise mais que la maintenance et le coût induits peuvent être assez importants. De plus, il faut noter que les différents messages échangés entre les différents équipements devront également être mis en oeuvre, ce qui, de ce fait, rend complexe cette approche d'émulation.

Ce niveau d'émulation apporte donc un certain niveau de réalisme et de précision mais est également associé à une complexité de mise en oeuvre ainsi qu'à un besoin en ressources.

Émulation de niveau réseau

Au niveau réseau, l'émulation rend un service de niveau IP et reproduit les mécanismes des couches physique, liaison de données et réseau. Elle permet l'expérimentation de protocoles de niveau transport ou supérieurs. Cette approche d'émulation propose une émulation à un coût moins important qu'une émulation de niveau 2. Elle ne reproduit pas tout le détail des mécanismes d'une émulation de niveau liai-

son mais elle prend en compte le comportement sur le trafic des technologies sous jacentes. Ainsi seuls les effets des mécanismes de bas niveau seront répercutés au niveau IP. En effet, si on reprend l'exemple de la liaison ADSL, il ne sera pas nécessaire de déployer tous les équipements cités précédemment et ce type d'émulation peut être mis en oeuvre de façon centralisée c'est-à-dire sur une seule machine. De plus, à ce niveau d'émulation, peu de paramètres sont nécessaires pour reproduire une condition réseau donnée et ce, de bout en bout. Ces paramètres sont les délais, les débits et les pertes d'informations.

Notre objectif dans ce mémoire est un objectif de performance plutôt qu'une recherche de réalisme ou de précision tout en proposant une méthode d'émulation à moindre coût. Ainsi, la suite des travaux se placera à ce niveau d'émulation réseau. En effet, les résultats obtenus grâce aux mesures réalisées dans le cadre du projet DIPCAST[17] [32][31], ont montré que les valeurs obtenues avec un émulateur de niveau 2 (plate-forme avancée à Alcatel) sont proches de celles obtenues avec l'émulateur de niveau 3 (plate-forme de base ENSICA). Cela nous a donc amené à nous positionner à ce niveau d'abstraction et donc à l'enrichir afin de lui permettre d'obtenir un niveau de précision ou réalisme suffisant. La suite de ce manuscrit présentera un état de l'art des solutions permettant de mettre en oeuvre l'émulation de niveau IP puis fera ressortir les carences et proposera une méthodologie permettant d'émuler un réseau donné de façon optimale.

2.3.3 Besoins

Comme cela a été présenté précédemment, les expérimentations en environnement émulé permettent de tester à la fois les propriétés fonctionnelles et non fonctionnelles de protocoles ou d'applications. Du point de vue des propriétés non fonctionnelles, un émulateur réseau doit être capable d'introduire des modifications au niveau d'un canal de communication en termes de délai, perte de paquet, et erreurs bit à un flux de données et ce, en fonction de modèles préalablement définis par les utilisateurs. Ainsi, il se dégage deux cadres principaux d'utilisation de l'émulation : la production d'un comportement particulier et la reproduction d'une technologie cible. Un émulateur réseau propose un moyen de produire un comportement réseau spécifique, pas forcément réaliste, mais qui permet de mettre en oeuvre un test dans des conditions spécifiques. Par exemple, il peut s'agir d'évaluer le comportement d'une implémentation particulière du protocole TCP lorsqu'un paquet d'acquittement sur deux est perdu. La mise en oeuvre d'un comportement spécifique peut être liée à la reproduction du comportement d'une technologie donnée. Ceci permet par exemple d'évaluer une application ou un protocole dans un environnement très proche de la réalité tel qu'un réseau WiFi ou un réseau satellite dans un cadre où il maîtrise les paramètres.

D'une manière générale, les modèles conduisant l'émulation sont soumis à des besoins en termes de :

- *Reproduction* : les conditions d'une expérimentation produites pour une expérience donnée doivent pouvoir être réutilisables pour réaliser des comparaisons et ainsi réaliser une évaluation de performance équitable.
- *Contrôle* : le modèle doit proposer un ensemble de modifications qui permettent de reproduire aussi bien une architecture réseau réelle qu'une condition réseau particulière basée sur un scénario par exemple. Les modifications pouvant se produire sont le délai, la perte de paquet ou encore l'introduction d'erreur bit. Toutes ces modifications sont bien entendu réalisées sous le contrôle du modèle.
- *Précision* : les modifications (pertes ou erreurs bit) doivent affecter uniquement les paquets pour lesquelles elles ont été paramétrées. Les autres paquets doivent être transmis avec le délai adéquat si il y a lieu en accord avec la spécification du modèle.
- *Transparence* : l'utilisation d'un protocole ou d'une application doit pouvoir se faire sans, ou alors avec un minimum de modification sur l'émulateur.
- *Flexibilité* : l'émulateur doit proposer à l'utilisateur un choix important de méthodes pour traiter les paquets. Ces méthodes doivent être adaptées soit au développement de modèles d'expérimentation en relation avec une technologie réseau réelle, soit à la création de modèles d'émulation spécifiques dans le but de mettre en évidence le comportement d'un protocole dans un contexte précis.
- *Extensibilité* : l'émulateur doit permettre d'ajouter de nouveaux modèles d'émulation pour répondre à de nouveaux besoins.
- *Passage à l'échelle* : c'est un besoin important dans le cadre de test de protocoles à haut débit ou encore dans le cas du Multicast.
- *Dynamicité* : la qualité de service d'un canal de communication doit pouvoir évoluer au cours du temps et ce, avec différents niveaux de granularité dépendant de la technologie.

Il faut noter que tous ces besoins ne demandent pas d'être tous satisfaits en même temps.

2.3.4 Modèle d'architecture d'émulation

Afin d'établir une classification des systèmes d'émulation existants, nous avons construit un modèle. Ce modèle représenté sur la figure 2.3 propose trois niveaux de classification. Le type d'émulation est basé sur les ressources physiques mises en jeu pour l'émulation. On distinguera deux types : les systèmes centralisés et les systèmes distribués. Le niveau suivant correspond au type de conditionnement de trafic considéré. Enfin le dernier niveau, le niveau modèle, s'intéresse à la façon dont est modélisé le comportement du réseau cible.

Type d'émulation

L'émulation peut être réalisée de façon locale c'est-à-dire sur une plate-forme se trouvant dans la même pièce que l'expérimentateur. Par opposition on parle d'émulation à distance lorsque la plate-forme d'émulation se trouve déportée par

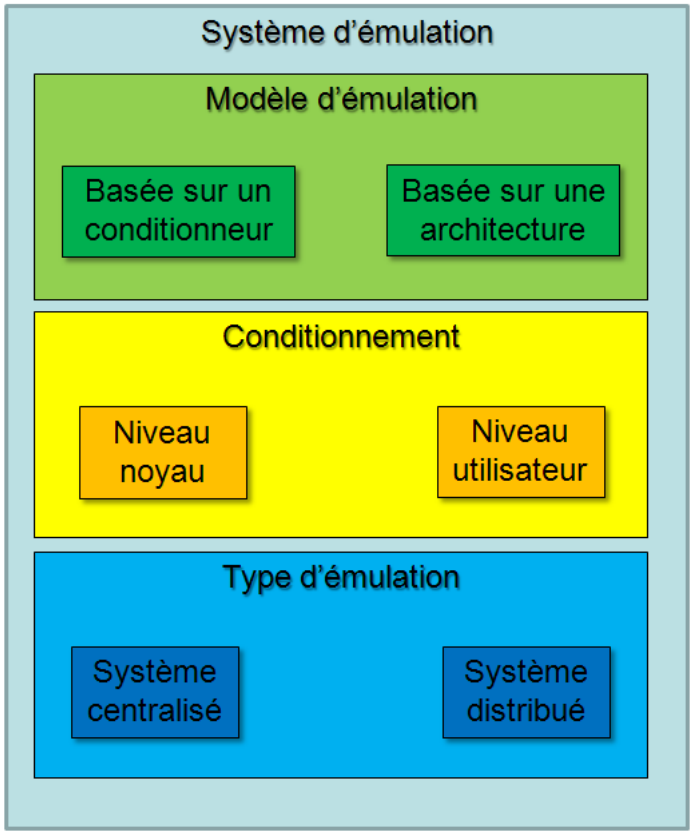


Fig. 2.3 Classification d'émulation réseau

rapport à l'utilisateur. L'émulation peut être également réalisée de manière réelle centralisée c'est-à-dire que les fonctionnalités d'émulation sont regroupées au sein d'un seul noeud d'émulation. Ainsi l'émulateur apparaît comme étant une « boite noire » entre deux machines hôtes. A l'inverse l'émulation peut être réelle distribuée : à ce moment plusieurs noeuds d'émulation (machines ou routeurs par exemple) sont en charge de l'émulation.

Nous allons distinguer deux grands types d'émulation. Ce niveau permet d'établir une classification entre les équipements physiques qui interviennent dans le système d'émulation.

Émulation distribuée

Une des approches possibles pour l'émulation consiste à représenter la topologie du réseau cible sur un support réseau distribué réel (plate-forme d'émulation distribuée). L'objectif ici est de pouvoir modéliser la topologie du réseau cible de façon précise. Cette approche implique de fait un niveau de réalisme et la possibilité de passer à l'échelle. Sur la plate-forme d'émulation distribuée, chaque composant de la topologie réelle (routeur, etc.) peut correspondre à un composant de la plate-forme.

Le principe de cette approche d'émulation consiste donc à partager l'émulation entre différents noeuds faisant partie de plate-forme d'émulation. Ce type de plate-forme est généralement divisé en deux parties : une servant à l'administration et l'autre aux expérimentations. Des sous réseaux distincts sont le plus souvent utilisés afin de séparer les trafics.

Il existe des outils mettant en oeuvre cette approche basés par exemple sur des clusters tels que le projet EMPOWER[68]. On entend par cluster un ensemble de noeuds répartis jouissant d'une proximité géographique. Cette proximité géographique permet un contrôle sur le matériel, sur les logiciels et sur la ou les politique(s) de sécurité. Ainsi, il est possible d'obtenir un environnement relativement homogène et qui permet de bénéficier d'un réseau de communication dédié entre les noeuds tout en ayant un haut débit et une latence faible.

EMPOWER est un projet d'émulation dans lequel chaque noeud est représenté grâce à un noeud virtuel qui correspond à un module du noyau Linux attaché à une interface réseau. Un noeud implémente un VDM (Virtual Device Module). Ce module est mappé sur un port réseau et reçoit les paquets qui sont récupérés par IP au niveau du système d'exploitation.

Les outils peuvent également être basés sur des grilles de calcul comme dans le cadre du projet Grid5000[14].

Dans le cas des grilles de calcul, la répartition géographique des noeuds de la grille est complètement différente. Elle est beaucoup plus vaste que pour les clusters. On se trouve donc dans un environnement qui a peu de chance d'être homogène. L'homogénéité est perdue aussi bien au niveau matériel et logiciel, mais aussi au niveau des politiques de sécurité qui sont spécifiques à chaque site. Dans le cadre du projet Grid5000, le but est d'interconnecter différents clusters se trouvant dans différentes villes au moyen de liens à très haut débit en utilisant le réseau français de la recherche RENATER[53]. La grille de calcul Grid'Explorer, fait partie du projet Grid5000, et permet d'émuler des conditions réseau grâce à un cluster d'environ 100 machines, une base de donnée de conditions expérimentales et un ensemble d'outils tels que des émulateurs et des simulateurs.

Un autre outil, NET[29, 28], est un émulateur de réseaux filaires et de réseaux sans fil qui utilise des scénarios pré-calculés. Ces scénarios sont décrits au format XML et servent à la fois à modéliser la topologie du réseau et à spécifier les modèles permettant de faire varier dynamiquement les paramètres au niveau du conditionneur de trafic. Ces modèles dynamiques peuvent agir soit au niveau temporel, c'est-à-dire qu'à une date prédéfinie on fait varier le paramètre désiré, soit au niveau paquet ce qui revient à faire varier le paramètre en fonction d'un paquet particulier (le 10ème paquet d'un flux par exemple). Ces modèles de variations dynamiques peuvent être basés sur des tables, sur des distributions gaussiennes (pour les délais essentiellement) ou sur des modèles markoviens (pour la bande passante et les pertes). Une fois analysés, les résultats obtenus à partir des scénarios sont joués sur NETShaper, un conditionneur de trafic mis au point spécialement pour les besoins de NET qui se présente sous la forme d'une couche d'émulation s'insérant entre la couche liaison

et la couche réseau. Chaque noeud composant la plate-forme NET embarque une instance du conditionneur de trafic NETShaper. Le noeud central situé sur le réseau d'administration se base donc sur le scénario pré-calculé pour simuler en temps réel les mouvements des noeuds et déterminer les communications qui sont possibles dans le réseau. Il envoie ensuite périodiquement les résultats de sa simulation aux instances de NETShaper qui vont être chargées de les reproduire.

ModelNet[5] est un environnement d'émulation qui est intégré à Netbed. Il utilise un noyau modifié de FreeBSD. ModelNet permet d'émuler des réseaux de grande taille.

Discussion

L'idée d'utiliser un support réseau distribué pour émuler un réseau est intéressante à deux points de vue. Elle permet d'obtenir un niveau de précision et de réalisme du fait de la représentation précise de la topologie et également au niveau du passage à l'échelle c'est-à-dire de la capacité à modéliser un réseau de taille importante.

Avec une émulation distribuée, il est possible de réaliser de l'émulation de niveau 2 grâce au principe de modélisation, d'où le niveau de précision et de réalisme.

Cependant, avec ce genre d'approche, les ressources nécessaires à sa mise en oeuvre peuvent être relativement importantes. De plus elle est assez intrusive dans la mesure où chaque noeud de la plate-forme participant à l'émulation embarque une partie de l'émulateur. Un des problèmes liés à l'utilisation de ce genre de plate-forme réside dans la difficulté de les administrer. L'utilisateur n'a pas de contrôle précis sur les conditions d'expérimentation. Avec la plate-forme Emulab par exemple, l'administration s'avère être relativement délicate. En effet, cette plate-forme peut être utilisée par plusieurs utilisateurs simultanément et en parallèle. De ce fait, il n'est pas possible de savoir si les ressources allouées permettront de mener à bien l'expérimentation. Un des désavantages majeurs de ce type d'expérimentation repose sur le fait que lorsque des expérimentations sont menées sur ces plate-formes, l'utilisateur n'a pas de connaissance sur l'état de charge. De ce fait, les résultats d'émulation peuvent être affectés.

Un autre inconvénient lié à l'utilisation de cette approche se situe au niveau du trafic réseau concurrent entre les différents équipements du réseau considéré. En effet, il doit être généré réellement ce qui implique une complexité de mise en oeuvre non négligeable.

En résumé, cette approche permet d'obtenir un niveau de réalisme au niveau de la modélisation de la topologie du réseau cible considéré. Il est possible de mettre en oeuvre une émulation de niveau 2 mais ceci a un coût pouvant être rapidement très élevé.

Émulation centralisée

Le principe de cette approche d'émulation est de pouvoir disposer d'une architecture comprenant un noeud central qui intègre les fonctionnalités d'émulation comme présenté sur la figure2.4. L'émulateur est alors vu comme une « boite noire » entre

deux machines hôtes. Il se contentera de rendre le service spécifié dans les modèles. L'émulation est donc transparente pour les utilisateurs.

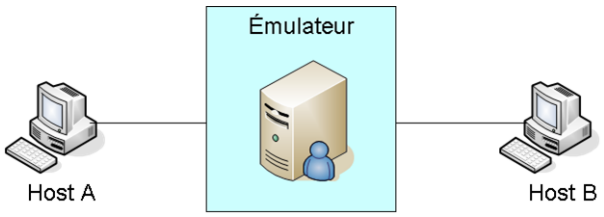


Fig. 2.4 Architecture d'émulation centralisée

La plupart des outils mettant en oeuvre l'émulation centralisée sont des conditionneurs de trafic. La prochaine section sur le niveau conditionnement présentera quelques uns d'entre eux.

Approche hybride (émulation à distance)

L'émulation à distance est plus un mode d'émulation plutôt qu'un type d'émulation. En effet, l'émulation à distance peut être aussi bien centralisée que distribuée. Ici l'idée consiste à utiliser une plate-forme d'émulation qui serait délocalisée géographiquement par rapport à l'utilisateur. Quelques exemples sont présentés dans la suite.

IREEL

La plate-forme IREEL[36] (Internet Remote Experiment Emulation Laboratory) est une plate-forme de niveau IP accessible à distance (via l'Internet) permettant de mener des expérimentations de réseaux et protocoles. Elle est basée sur une approche d'émulation centralisée à distance. Elle offre un moyen de réaliser des expériences avec de réelles applications distribuées et protocoles de communication. Elle sert pour l'instant principalement à l'enseignement. Cette plate-forme propose donc un ensemble de protocoles et d'applications prédéfinies qui peuvent ainsi être testés dans les conditions spécifiées par les utilisateurs.

Netbed (Emulab)

Il s'agit d'une plate-forme d'émulation réseau accessible à distance (Emulab) faisant partie du projet NetBed[65]. Ce projet a pour vocation de fournir un environnement d'expérimentation basé sur des outils existants pour intégrer la simulation temps réel (nse), l'émulation (Dummynet) et même le test réel. Cette plate-forme utilise le principe des grilles de calcul pour mettre en oeuvre l'émulation : elle compte plusieurs centaines de noeuds connectés par un réseau filaire. Ces noeuds peuvent

servir d'éléments terminaux où l'application à tester est déployée ou d'éléments de calcul, suivant qu'ils embarquent un émulateur ou un simulateur temps réel.

Pour décrire une topologie, Emulab utilise des fichiers ns. La plate-forme d'émulation est divisée en trois sous plate-formes qui ont toutes une vocation différente : une pour les réseaux sans fil mobiles, pour les réseaux sans fil 802.11, et une dite « classique » pour les autres expérimentations. Sur ces sous plate-formes il est possible de configurer et de contrôler les noeuds ainsi que les liens. Les paramètres modifiables sont : la bande passante, les latences, les taux de pertes et les tailles de queue.

Niveau conditionnement

Le conditionnement de trafic est un aspect important d'un système d'émulation. En effet, il va permettre d'appliquer aux flux traversant le système d'émulation les modifications au niveau des paramètres des paquets, afin de reproduire le comportement du réseau cible. Le conditionnement peut être réalisé à différents niveaux : au niveau noyau et au niveau utilisateur.

Niveau noyau

C'est probablement l'un des outils les plus utilisés pour l'émulation de réseau. Dummynet[54] est un outil logiciel développé par L. Rizzo et implémenté sous FreeBSD dont il fait partie intégrante du noyau. Il utilise le firewall IP de FreeBSD : ipfw pour intercepter les paquets et ce, en fonction de règles préalablement définies par l'utilisateur. Grâce à un principe de « pipes » qui peuvent être couplés les uns aux autres, Dummynet peut appliquer au trafic des modifications en termes de limitation de bande passante, de délais de propagation, de pertes de paquets, de limitations de queues. Chaque pipe peut être configuré séparément et les paquets qu'il reçoit sont sélectionnés par ipfw en fonction de caractéristiques telles que : l'adresse IP source, destination, numéro de port, protocole de transport, sens de communication (in / out), etc. Pour mettre en oeuvre ces effets sur les paquets sélectionnés, Dummynet utilise deux queues.

Maintenant voyons un peu plus en détail comment Dummynet met en oeuvre le conditionnement du trafic avec les pipes. Pour y parvenir, Dummynet utilise deux queues comme présentées sur la figure 2.5.

Lorsqu'un paquet arrive c'est à dire qu'il a été filtré en fonction des règles du firewall ipfw, il est inséré dans une première file d'attente (la R-queue). Cette file modélise les queues que l'on peut trouver en entrée d'une interface réseau par exemple. Les paquets sont ensuite extraits de cette file et insérés dans une autre file (la P-queue) à une vitesse correspondant au débit paramétré par l'utilisateur. Les paquets restent dans cette file pendant un temps qui correspond au délai défini par l'utilisateur. Ils sont ensuite remis au destinataire. Les pertes, si elles ont été paramétrées, sont appliquées lors du passage des paquets entre les files R et P. Ces pertes sont définies de façon probabiliste c'est à dire qu'il y a un tirage aléatoire pour chaque paquet. Si la valeur obtenue est inférieure à la valeur de PLR fixée par l'utilisateur, le paquet est perdu. Sinon il est passé à la file P.

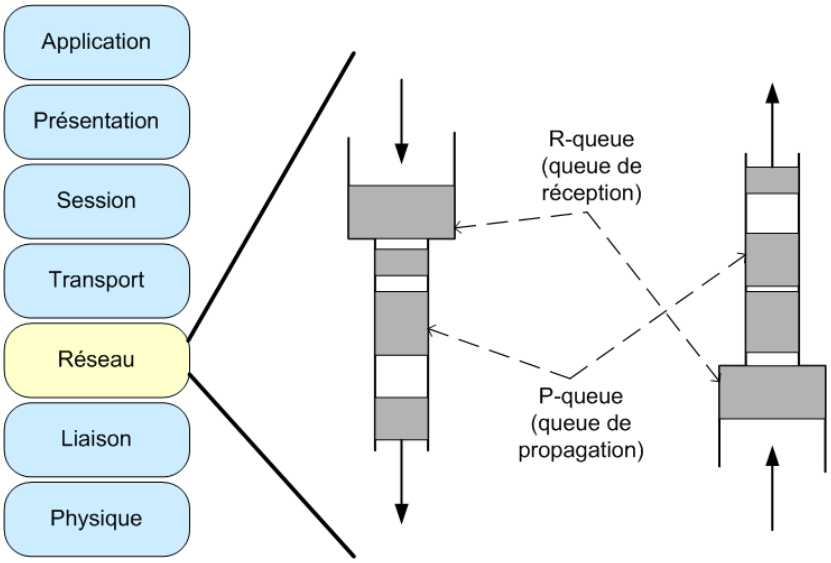


Fig. 2.5 Fonctionnement de Dummynet

En fonction des règles qui ont été paramétrées au niveau du firewall, un paquet peut être amené à subir plusieurs fois des modifications. De ce fait, Dummynet permet d'émuler un réseau de bout en bout composé de plusieurs sous réseaux.

Delayline[33] est un émulateur de réseau métropolitain, développé par D.Ingham et G.Parrington de l'université de Newcastle. Son but était de tester des applications réparties sur un réseau WAN. Delayline se présente comme une plate-forme d'émulation de niveau 3 capable de changer les conditions du réseau sous-jacent sur lequel sont testées les applications réparties. Il permet de reproduire le partitionnement d'un réseau et le comportement des éléments d'interconnexions des sous-réseaux ainsi partitionnés. Ceci permet d'émuler des pannes de routeurs et des pannes de passerelles entre sous réseaux.

Pour intercepter les paquets, Delayline nécessite une recompilation des applications à tester. En effet, Delayline utilise des sockets pour intercepter des paquets et doit donc être inclus sous forme de librairie dans le code des applications à tester. Grâce à ces sockets, il peut intercepter et manipuler les paquets en introduisant des délais mais également des pertes qui vont reproduire les congestions et les partitionnements du réseau. Le fait de devoir recompiler l'application à évaluer élimine toute possibilité de comparaison entre une application en cours de développement et une application commerciale dont les sources ne sont pas disponibles. Ce besoin de disposer du code source de l'application le rend donc difficilement utilisable. Enfin, il faut noter que les sockets utilisées nécessitent de tester l'application sur un système d'exploitation disposant de sockets Berkeley, ce qui élimine de fait toutes les applications développées sous Windows.

NistNet

NistNet[43] a été développé par un groupe de recherche du National Institute of Standards and Technology (NIST) appelé Internetworking Technology Group (ITG). Il est implémenté sous Linux (version 2.4) comme un module noyau et travaille au niveau IP. Il peut être assimilé à un routeur spécialisé qui serait capable de reproduire le comportement d'un réseau complexe sur une simple machine. Il capture les paquets au moyen d'un module noyau. NistNet permet de contraindre le trafic entre sous réseaux au niveau des délais, des bandes passantes, des taux de pertes. A la différence de Dummynet, le taux de duplication de paquets ainsi que le modèle de répartition des pertes est paramétrable. Mais il n'est pas possible d'enchaîner plusieurs queues ou pipes.

TC + Netem Les noyaux Linux (2.4, 2.6) possèdent un outil, le Traffic Control (tc[60]), qui met en oeuvre un ensemble de mécanismes afin de conditionner le trafic réseau. tc est un programme utilisateur permettant de créer et d'associer des files à une interface de sortie. C'est un tout en un, il est utilisé pour installer divers types de files, associer des classes à ces files, mettre en place les filtres de classification. Avec tc, il est possible de définir des queuing discipline (qdisc) qui permettent à l'utilisateur, grâce au mécanisme CBQ (Class Based Queuing) de séparer en queues les différents flux.

tc fait partie du package iproute2 contenant également d'autres utilitaires tel que ip permettant notamment d'effectuer de l'IP aliasing ou de l'IP tunneling. Le principal inconvénient de cet outil résidait dans le fait qu'il ne proposait que la limitation de bande passante mais pas de paramétrage de délai et ou de pertes. La qdisc netem, développée par S. Hemminger, résout ce problème. Netem[27] (fonctionnant avec les noyau Linux 2.6) reprend les idées de NistNet et permet d'émuler les propriétés de nombreux réseaux. La dernière version permet d'émuler des délais variables, des pertes, des duplications et des dé-séquencements de paquets

Cela dit, il n'est pas très évident d'enchaîner des qdisc pour faire une analogie avec les pipes dans le cas de Dummynet, mais il faut aussi noter que netem est toujours en développement.

Niveau utilisateur

ONE[42] est un émulateur issu d'un projet de recherche de l'Ohio University's Internetworking Research Group développé en collaboration avec la NASA. Il se présente sous la forme d'un routeur entre deux sous réseaux. Il permet d'émuler un réseau sur une machine sous Solaris. Son objectif premier était d'émuler un lien satellite en introduisant des délais de propagation variables. Mais il est également possible à l'utilisateur de contrôler différents paramètres du réseau tels que : le délai de propagation variable basé sur les orbites de satellites, ou encore la bande passante. L'utilisateur fourni à ONE des paramètres qui permettront d'appliquer, aux paquets qui transitent par l'outil, des retards qui seront calculés en fonction de leur taille. Le fonctionnement de ONE est basé sur la caractérisation du délai. Pour y arriver

il va tenir compte de trois aspects : le délai de transmission qui correspond à la bande passante du lien, le délai de file d'attente qui correspond lui au délai auquel sera soumis le paquet en traversant les routeurs du réseau émulé et enfin le délai de propagation qui correspond au temps que mettra un paquet pour traverser le canal de communication.

Le principal inconvénient de cet outil réside dans le fait qu'il ne permet d'émuler des limitations qu'entre deux sous réseaux. Pour mettre en oeuvre l'émulation de réseaux plus importants en termes de taille et de complexité, il faut déployer autant d'instances de ONE que de sous réseaux à émuler d'où un problème de passage à l'échelle.

Niveau modèle

Afin de contrôler les conditionneurs de trafic, il faut disposer de modèles. Ces modèles sont de différents types et ainsi permettent de proposer plusieurs niveaux de précision.

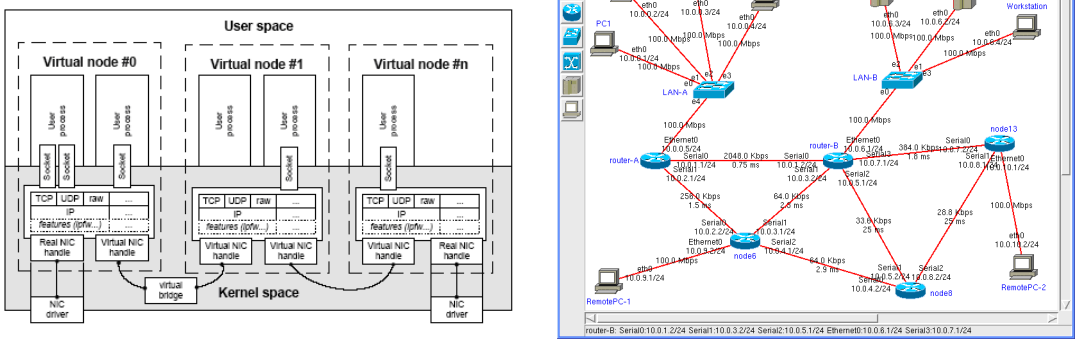
Emulation virtuelle

La figure 2.6a présente l'émulation virtuelle qui consiste à utiliser des noeuds et des liens virtuels afin de reproduire le comportement d'un réseau donné. Tous les noeuds constituant la topologie du réseau à émuler sont implémentés soit sur une seule machine (approche centralisée) soit sur plusieurs composants (approche distribuée). Ils sont connectés entre eux par des liens à haut débit. Des liens virtuels sont utilisés pour connecter les noeuds virtuels en fonction de la topologie du réseau cible. Les noeuds virtuels ont l'avantage de pouvoir « embarquer » des implémentations réelles de protocoles (par exemple IP ou des protocoles de routage).

Imunes[45] est un émulateur virtuel développé par M. Zec : c'est un outil basé sur une approche centralisée qui permet d'émuler un réseau en utilisant des partitions du noyau (noeuds virtuels) qui sont interconnectées par liens de niveau noyau pour former des topologies potentiellement complexes. Avec Imunes, chaque noeud virtuel dispose d'une pile protocolaire complète, ce qui permet de réaliser une émulation précise au niveau du comportement des routeurs par exemple. Il est également possible de faire communiquer des applications de niveau utilisateur avec les noeuds virtuels.

Au niveau utilisateur, Imunes propose une interface permettant de définir la topologie du réseau à émuler. La figure 2.6b présente cette interface. Il est possible de définir les noeuds virtuels, leurs applications et les liens avec les paramètres de QoS correspondants.

ENTRAPID[66] est un environnement d'émulation virtuel semblable à Imunes. Il travaille en espace utilisateur et est basé sur la notion de noyaux réseau virtuels multiples. Chaque noyau virtuel implémente exactement tous les services réseaux existants sur un noyau BSD 4.4. A haut niveau, ENTRAPID peut être vu comme



(a) Architectutre d'émulation par noeuds virtuels (b) Interface graphique d'Imunes

Fig. 2.6 Emulation virtuelle

un processus tournant entièrement en espace utilisateur, capable d'inter-agir avec d'autres processus et avec les interfaces réseau physiques.

ENTRAPID peut être vu comme étant une « boîte noire » à laquelle se connectent des machines hôtes. Cette boîte noire peut représenter un simple lien ou alors un réseau complexe, et ce grâce au principe de noyaux réseau virtuels. Cependant même si cette approche offre un haut niveau de réalisme, il s'agit d'une solution centralisée avec les inconvénients associés.

Emulation conduite par de la simulation temps réel

Une autre approche d'émulation consiste à utiliser un simulateur à événements discrets qui serait en charge de définir le modèle d'émulation. En y ajoutant un moyen de traiter les paquets comme des vrais paquets et non des événements, et en adaptant l'ordonnanceur afin qu'il puisse fonctionner en « temps réel », on obtient un émulateur avec un conditionneur de trafic et des modèles. Le simulateur fournit alors le modèle d'émulation et conduit le processeur d'émulation. La figure 2.7 présente l'architecture de ce type d'émulateurs.

Ns2 mode émulation (nse)

Le simulateur ns2 présenté précédemment en 2.2.2 utilise cette approche. Grâce à l'extension (nse[8]), il est désormais possible d'utiliser le simulateur ns2 comme un émulateur. Le simulateur se trouve donc capable de capturer de vrais paquets, de les traiter si besoin est, et de les re-injecter sur le réseau. Pour ce faire, nse propose deux modes de fonctionnement. Tout d'abord le mode opaque ; dans ce mode, le simulateur a un fonctionnement semblable à un conditionneur de trafic. En effet les paquets réels sont traités de façon « opaque » c'est-à-dire qu'ils ne sont pas interprétés par le simulateur. nse se contente d'appliquer les modifications des paramètres en terme de délai, bande passante et de pertes de paquets, aux flux qu'il reçoit. Il se comporte comme un pont entre deux extrémités du réseau.

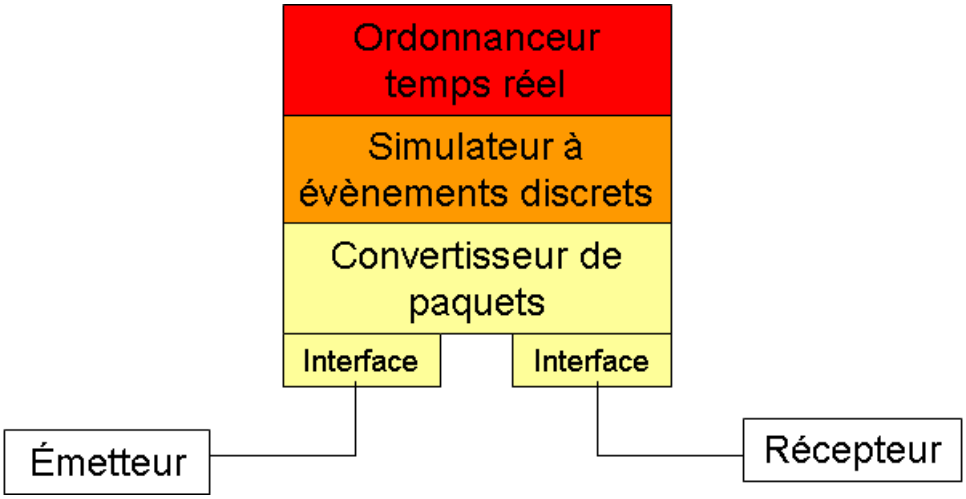


Fig. 2.7 Architecture émulateur par simulation temps réel

L'autre mode, le mode protocole, permet d'avoir accès au contenu des paquets. Les paquets réels seront interprétés, transformés en paquets simulés puis reformatés et enfin réinjectés en sortie du simulateur. De plus, le simulateur est capable d'interagir avec des éléments extérieurs.

Ce type d'approche pour l'émulation est intéressant dans le sens où il permet de modéliser la topologie du réseau dans un premier temps grâce au simulateur, mais également parce qu'il permet de définir les modèles d'émulation de façon relativement précise.

Cependant, cet outil souffre du fait qu'il doive utiliser un simulateur à événements discrets pour mettre en oeuvre l'émulation. En effet, le fonctionnement d'un simulateur est le suivant : lorsqu'un événement arrive, celui ci est traité puis le simulateur passe à l'événement suivant et ainsi de suite, en se basant sur l'arrivée d'un événement et non sur une horloge temporelle.

De plus cet outil est basé sur une approche centralisée. Le simulateur doit alors traiter en temps réel un nombre assez important de paramètres. Lorsque ce nombre devient trop important, il ne peut plus tenir les contraintes « temps réel » et de ce fait, le réalisme de l'émulation se trouve affecté. Dans [41], les auteurs ont montré que cet outil n'arrivait pas à tenir les contraintes temps réel lorsque le nombre de flux réels à traiter augmente et ce, même avec des modèles simples. Or, plus les topologies et les modèles à émuler seront complexes, plus le nombre d'événements à traiter sera important. Donc pour pouvoir tenir les contraintes « temps réel » la solution serait de distribuer la simulation de manière à paralléliser les traitements.

NCTUns

NCTUns [63][64] est également un simulateur réseau temps réel. Il a été développé par l'université de Harvard. Il s'agit d'un simulateur basé sur un fonctionnement distribué. Cependant il n'a pas été évalué pour émuler des réseaux complexes utilisant

plusieurs trafics réels ou utilisant des modèles de mobilité complexes dans le cas des réseaux sans fil par exemple. En effet, bien que le fait de distribuer la simulation puisse faire gagner du temps au niveau de la durée des calculs, l'administration d'un tel système s'avère beaucoup plus complexe. Ainsi, le noeud chargé de faire le conditionnement de traffic doit pouvoir traiter et synthétiser en temps réel les résultats qui lui sont envoyés par les différentes parties du simulateur. Cette phase de traduction peut donc être relativement lourde et le temps gagné en simulation peut de nouveau être perdu lors de cette collecte des résultats. De plus, les différentes parties du simulateur doivent être synchronisées les unes avec les autres pour maintenir la cohérence des résultats obtenus, ce qui n'est pas encore assuré actuellement.

Émulation conduite par des traces

Cette approche d'émulation permet en fait, d'obtenir un modèle de comportement du réseau cible. Ce modèle conduira par la suite le conditionneur de trafic. Dans ce cas, des traces sont récupérées, analysées puis rejouées. Ce type d'émulation repose donc sur trois phases. Tout d'abord une phase de collecte des traces, puis une phase de distillation qui correspond en fait à l'analyse des traces (à ce stade, les informations pertinentes sont extraites des traces et permettent de construire le scénario d'émulation) ; enfin une phase de modulation où les traces seront rejouées.

Cette approche a l'avantage de proposer un comportement émulation relativement proche du comportement du réseau cible. Mais en fait, ce type d'émulation est relativement figé dans le sens où les conditions reproduites par les traces correspondent à des mesures réalisées à des instants donnés sur le réseau cible. L'utilisateur n'a donc que peu de contrôle sur les paramètres : il ne peut pas les faire évoluer et de ce fait, ne peut pas tester des conditions particulières. De plus, il faut noter que la phase de distillation peut s'avérer être relativement délicate à mettre en oeuvre. En effet, on peut se poser la question de savoir quelles sont les informations pertinentes de toutes les valeurs récupérées, et par conséquent, comment être sûr de ne pas « manquer » une donnée importante.

Ce type d'approche permet d'émuler des topologies réseau complexes tout en proposant un bon niveau de réalisme. Elle permet de spécifier de manière précise le réseau cible et d'obtenir des comportements réalistes. Néanmoins, le problème du passage à l'échelle se pose. En effet, on pourrait se demander comment émuler un réseau de coeur par exemple sur une seule machine ? Comment pourrait-on produire des conditions réalistes de réseaux complexes avec une approche centralisée ?

2.3.5 Implémentation des différentes approches

Les approches d'émulation peuvent être implémentées à différents niveaux, du niveau hardware au niveau applicatif.

Niveau hardware

Ce type d'implémentation est certainement celui offrant le plus haut niveau de performance. En effet, dans ce cas, l'implémentation se fait directement au niveau matériel grâce à l'utilisation de circuits intégrés (FPGA par exemple). Ainsi, le temps de traitement des paquets se trouve réduit par rapport au temps nécessaire à une implémentation logicielle. Mais par rapport à une solution logicielle, ce type d'implémentation est moins flexible, et plus compliqué à développer et à programmer.

Niveau noyau

Il est également possible d'implémenter une solution d'émulation au niveau noyau. C'est d'ailleurs le cas des outils comme Dummynet ou NISTNet par exemple. Ce type d'implémentation offre l'avantage d'être totalement transparent pour les programmes utilisateur et d'introduire un overhead limité. En effet, les paquets ne subissent pas de copie lors de leur traitement. Cependant, ces solutions sont dépendantes des sytèmes d'exploitation sur lesquels elles sont développées. Elles se trouvent limitées par la granularité des systèmes c'est-à-dire par la précision de l'horloge système ainsi que par la charge éventuelle de ces systèmes. Ceci a un impact direct sur les performances des outils.

Pour résoudre le problème de la précision des horloges systèmes et de la charge, un solution consisterait à envisager l'implémentation d'un outil d'émulation sur un système d'exploitation temps réel.

Niveau utilisateur

Le développement d'émulateurs dans l'espace utilisateur permet de disposer d'un large choix d'outils et de librairies. Ainsi il est relativement aisé de construire des modèles d'émulation complexes. Pour mettre en oeuvre ce type d'approche il faut pouvoir récupérer les paquets. Il existe différentes solutions pour les remonter au niveau utilisateur : les raw socket, libpcap, et divert socket.

L'avantage de l'implémentation à ce niveau réside dans le niveau de flexibilité offert en comparaison avec le niveau noyau. Cependant, deux inconvénients liés à la performance sont à signaler. En effet, l'utilisation de ce type d'implémentation suppose que le paquet traité sera recopié dans l'espace utilisateur pour être traité.

Ensuite les processus utilisateurs peuvent être suspendus si une tâche de plus haute priorité arrive. Elle sera traitée en priorité et la tâche d'émulation attendra. D'où des performances dégradées.

2.3.6 Comparaison entre les différentes approches d'émulation

Le tableau de la figure 2.8 est une comparaison des diverses approches d'émulation présentées. Cette comparaison se base sur les critères suivants : les ressources nécessaires pour la mise en oeuvre de l'émulation considérée, le réalisme de l'émulation, sa flexibilité et sa mise à l'échelle.

	Ressources nécessaires	Réalisme	Flexibilité / Contrôlabilité	Mise à l'échelle
Réseau réel	+++	+++	--	-
Décentralisée (grilles)	++	++	+	+
Émulation virtuelle (Imunes)	--	-	+	-
Simulation temps réel (nse)	--	-	+	--
Rejeu de traces	--	++	--	--
Impairments brut (Dummynet)	--	--	++	-
Besoin	--	Mesurable	++	+

Fig. 2.8 Récapitulatif des différentes approches

La figure 2.8 propose une comparaison en prenant comme référence le test en environnement réel. Les paramètres rentrant en compte sont les ressources nécessaires à la mise en oeuvre de l'approche d'émulation, le réalisme, la flexibilité et le passage à l'échelle. L'approche d'émulation à distance n'apparaît pas dans ce tableau car il s'agit plus d'un mode plutôt qu'un type d'émulation.

Dans une approche décentralisée (des grilles par exemple), les ressources nécessaires pour mettre en oeuvre une émulation sont importantes. En effet, le principe même de la décentralisation ou distribution implique un besoin en ressources important. Le réalisme ainsi que la mise à l'échelle apportés par ce genre d'émulation sont assez difficiles à évaluer : ils dépendront du niveau d'émulation choisi. En effet, ces solutions peuvent émuler aussi bien du niveau 2 que du niveau 3. Mais d'une manière générale on peut dire que le niveau de réalisme obtenu est bon du fait de l'importance des ressources disponibles.

L'émulation virtuelle permet de mettre en oeuvre l'émulation avec très peu de ressources et ce, avec un bon niveau de réalisme. Elle permet également à l'utilisateur d'avoir un bon niveau de contrôle au niveau des paramètres d'émulation. Cela dit, étant donné que ce genre d'approche est le plus souvent basée sur une solution centralisée, le problème de mise à l'echelle apparaît.

L'émulation basée sur une approche de simulation temps réel, offre quasiment les mêmes caractéristiques que l'émulation virtuelle, mais son principal inconvénient réside dans le fait que ce type d'approche ne passe pas à l'échelle. Premièrement à cause de l'utilisation d'une approche centralisée mais également à cause des ordonnanceurs utilisés par les simulateurs. En effet, ceux-ci ne sont pas conçus à l'origine pour être capables de fonctionner en « temps réel » et de ce fait, ils ne tiennent pas la charge dès lors que la quantité de données à traiter devient importante.

L'émulation par rejeu de traces ne demande pas beaucoup de ressources pour être mise en oeuvre et elle offre un bon niveau de réalisme puisqu'il s'agit de traces réelles qui sont rejouées ; par contre cette approche n'est pas flexible. Il n'y a pas moyen d'évaluer des conditions particulières (comme par exemple un test en conditions particulières). C'est une approche relativement rigide. Elle a aussi l'inconvénient de ne pas passer à l'échelle principalement à cause du fait qu'elle est généralement basée sur une approche centralisée.

Les outils qui se contentent d'appliquer des modifications aux paquets (conditionneurs) ont en commun le fait qu'ils laissent le soin à l'utilisateur de paramétrer les modifications à effectuer. Ce sont des outils qui proposent une émulation de niveau IP et donc basée sur une approche centralisée. De ce fait, ils n'ont pas besoin d'énormément de ressources pour mettre en oeuvre l'émulation. De plus, ils offrent un très bon niveau de flexibilité. Par contre, ils n'offrent pas un bon niveau de réalisme. Cependant en les couplant à des modèles d'émulation « réalistes » on peut obtenir une émulation ayant un comportement proche de celui observé sur une technologie réelle. Ces outils permettent surtout de mettre en oeuvre une émulation basée sur l'objectif de reproduire des conditions réseau spécifiques indépendantes des technologies.

D'une manière générale, ces différentes approches permettent de mettre en oeuvre l'émulation à différents niveaux et elles offrent un bon niveau de flexibité ; par contre aussi bien au niveau des approches centralisées que décentralisées se pose le problème du passage à l'échelle. Mais il s'agit là d'un problème lié à l'émulation réseau en général. En effet, pour la mettre en oeuvre il faut que les ressources soient surdimensionnées.

Ainsi, de façon générale, on pourrait se demander de quoi a-t-on effectivement besoin dans le cadre d'une émulation de réseau ?

L'émulateur idéal serait donc un outil qui soit capable de réaliser une émulation de réseau et ce, avec un minimum de ressources, dont le niveau de réalisme soit mesurable, qui puisse passer à l'échelle et surtout qui soit flexible c'est à dire qu'il soit facile de le contrôler.

2.4 CONCLUSION

Comme nous l'avons vu, l'expérimentation dans le cadre de l'ingénierie des protocoles peut être mise en oeuvre grâce à trois grands types d'environnements :

- l'expérimentation en environnement réel qui permet d'obtenir une précision et un réalisme importants mais à un coût élevé et sans contrôle sur les paramètres d'expérimentation ;
- l'expérimentation en environnement simulé où le réseau considéré ainsi que les applications ou protocoles à évaluer sont modélisés dans un environnement facilement contrôlable à faible coût de revient, mais qui fonctionne en temps logique ou simulé ;
- l'expérimentation en environnement émulé qui permet de modéliser uniquement le réseau considéré tout en offrant la possibilité d'utiliser des applications ou protocoles réels.

De cette manière, l'expérimentateur dispose d'un environnement facilement contrôlable pour reproduire le comportement d'un réseau cible donné.

L'émulation de réseau peut être mise en oeuvre à différents niveaux au sens OSI, en fonction des besoins recherchés par les utilisateurs. En effet, plusieurs types d'émulation sont alors envisageables :

- concernant le conditionnement du trafic (solution centralisée ou décentralisée) ;
- concernant la modélisation du réseau à émuler : outils intégrant des modèles (émulation virtuelle, simulation temps réel, émulation conduite par des traces).

Cependant, la plupart des expérimentations en environnement émulé ne nécessitent pas le déploiement d'architectures d'émulation complexes. En fait, le plus souvent, le besoin se résume à la notion de canal de communication de bout en bout entre machines terminales. Ce canal de communication est capable de reproduire aussi bien des conditions réalistes (issues du comportement d'un réseau cible) ou alors spécifiques (donc pas forcement réalistes) pour mettre en évidence un comportement précis à un instant donné.

L'objectif ne sera donc pas un objectif de recherche de réalisme optimal, mais plutôt de montrer qu'il est possible avec une approche d'émulation se basant sur un canal de communication de bout en bout (niveau 3), de définir une méthodologie qui permet de répondre au mieux au besoin d'émulation. C'est ce qui sera présenté dans la suite de ce document.

Chapitre 3

Méthodologie pour l'émulation

Dans ce chapitre, les points suivants seront abordés :

- ⋆ les caractéristiques nécessaires à l'émulation de réseau de niveau IP ;
- ⋆ une architecture d'un système d'émulation ;
- ⋆ une méthode d'émulation qui comprend : d'abord une modélisation du comportement de l'émulateur et ensuite une vérification des contraintes comportementales et temporelles de ce qui sera implémenté et qui servira alors à la conduite de l'émulation ;

3.1 FONCTIONNALITÉS NÉCESSAIRES À L'ÉMULATION

Le chapitre précédent a permis de mettre en avant le fait qu'il existe un grand nombre d'outils d'émulation permettant de répondre à des besoins d'émulation variés. Tous ces outils permettent de mettre en oeuvre l'émulation de réseau, mais en fonction de l'objectif à atteindre, le choix d'un outil d'émulation adapté à des besoins spécifiques peut s'avérer difficile. Notre objectif n'est pas d'apporter une solution d'émulation universelle, mais plutôt de proposer une méthodologie générique pour émuler un grand nombre de besoins.

Ainsi, nous avons choisi de nous concentrer sur l'étude et l'amélioration de deux caractéristiques qui nous semblent essentielles : le besoin de reproduction et le besoin de précision des expériences. Un système d'émulation doit être capable de reproduire une même expérience plusieurs fois dans le temps, et ce avec un minimum de précision. En effet, dans le cadre de l'expérimentation de protocoles ou d'applications, il est aussi important de pouvoir répéter plusieurs fois la même expérience, de manière à établir des comparaisons fonctionnelles, que de disposer de comportements réalistes de l'émulateur. Le réalisme peut être obtenu en proposant des modèles d'émulation précis.

Notre objectif est également de savoir si le résultat final produit par le système d'émulation peut être considéré comme fiable. En fait, comme nous l'avons présenté précédemment, le but principal d'un système d'émulation est de reproduire le comportement d'un système cible donné sur un environnement synthétique différent de l'environnement initial. Une des interrogations majeures sera de savoir si le résultat obtenu quant à la reproduction du comportement est satisfaisant par rapport aux contraintes de l'utilisateur.

Dans ce chapitre, nous allons présenter les caractéristiques à satisfaire pour garantir une émulation fiable, c'est-à-dire qui propose un résultat d'émulation conforme aux attentes de l'utilisateur. Deux caractéristiques d'émulation seront détaillés, puis une méthode d'émulation sera proposée. Cette méthode basée sur l'utilisation de modèles, permet d'obtenir un résultat d'émulation conforme aux spécifications de l'utilisateur de l'émulation.

3.1.1 Reproduction d'expérience

Dans le contexte de l'expérimentation de protocoles ou d'applications réseau, la faculté de pouvoir reproduire plusieurs fois une même expérience est primordiale dans le sens où elle permet de mettre en évidence d'éventuelles erreurs de conception en établissant des évaluations de performances ou alors des comparaisons fonctionnelles. Un des objectifs principaux de l'émulation réseau consistera à pouvoir reproduire une même expérience plusieurs fois de suite.

Avec l'expérimentation en environnement émulé, il est intéressant de pouvoir, par exemple, étudier l'impact d'une modification ou d'une amélioration d'un mécanisme sur le comportement global du protocole ou de l'application. Ainsi, il est indispensable de pouvoir disposer d'un système capable de reproduire les mêmes conditions réseau en termes de délais, pertes et bande passante afin de réaliser des comparaisons. Reproduire une expérience consiste à fournir les mêmes conditions de bande passante, de délai et de taux de pertes à un instant donné.

Mais pourquoi a-t-on besoin de reproduire une expérience dans l'évaluation de protocoles ou d'applications ? Cela permet d'avoir des conditions d'expérimentation identiques, et spécifiques à des réseaux et des circonstances donnés (c'est-à-dire de « *fixer* » les paramètres de Qualité de Service (QdS) tels que le délai, le modèle de pertes, ou encore la bande passante), et ainsi de se concentrer sur le fonctionnement effectif des applications ou du protocole à évaluer dans ces conditions.

Prenons pour exemple, l'évaluation d'un mécanisme d'acquittement pour un protocole transport soumis à des délais importants (de l'ordre de 2 secondes) et un bande passante limitée (quelques Ko). En fixant les paramètres de QdS (délais et bande passante) l'expérimentateur pourra mettre en évidence le bon fonctionnement de son protocole à la condition que le système d'émulation fournisse plusieurs fois de suite le même contexte d'expérimentation.

Cependant, un des problèmes que pose la reproduction de comportement est lié à l'indéterminisme introduit par les modèles de pertes. En effet, la plupart des conditionneurs de trafic ne proposent que des modèles probabilistes et non reproduc-

tibles de pertes. Les pertes sont insérées de manière aléatoire et ne concernent pas les mêmes paquets si on reproduit plusieurs fois l'expérience. Ces outils proposent généralement un taux de perte moyen : ils assurent que si on reproduit plusieurs fois une même expérience on obtient en moyenne le même taux de perte. Mais ils n'assurent pas que les mêmes paquets seront affectés d'une expérience à l'autre.

De manière générale, on peut se contenter de ces outils d'émulation pour évaluer des applications réparties ; cependant, dès lors qu'il s'agit de l'évaluation de protocoles, il faut pouvoir contrôler l'introduction de pertes. En effet, si on veut montrer l'efficacité d'un mécanisme par rapport à un autre, il faut pouvoir le réaliser dans les mêmes conditions. Par exemple, supposons que l'on veuille évaluer le protocole TCP. Les résultats au niveau des délais seront différents dans le cas où les pertes de paquets concernent des paquets de données ou des paquets de signalisation.

Ainsi donc, dans le cadre de l'évaluation de protocoles, la position des pertes est importante. Il faut pouvoir les reproduire de manière fine d'une expérience à l'autre et de ce fait, permettre d'augmenter la fiabilité des résultats obtenus.

3.1.2 Précision des conditions

La précision de l'émulation, c'est-à-dire la capacité du système d'émulation à reproduire fidèlement une condition ou une situation réseau donnée est également un objectif principal d'émulation. En effet, lorsque l'on veut émuler un réseau ou une condition réseau, le challenge est d'être capable de fournir un résultat d'émulation le plus fidèle possible par rapport à la réalité, c'est-à-dire de proposer des conditions d'émulation semblables ou du moins les plus proches possibles des conditions du réseau cible. Pour y parvenir, il existe différents modèles que nous détaillerons dans le chapitre suivant comme des modèles basés sur des traces, des modèles mathématiques, ou encore calculés par simulation préalable, etc. . Afin d'obtenir un résultat le plus réaliste possible, il est nécessaire de faire évoluer les différents paramètres au cours de l'expérience. En effet, les conditions rencontrées sur un réseau donné sont rarement figées : elles évoluent en fonction de paramètres extérieurs comme des perturbations électromagnétiques, climatiques, à cause de congestions dues à un trafic concurrent important, ou encore en fonction du chemin qu'emprunte le trafic par exemple. Le niveau de précision est étroitement lié au niveau de réalisme final que souhaite obtenir l'utilisateur. De ce fait, pour disposer d'une émulation précise, il faut disposer de modèles d'émulation précis. Or, augmenter la précision et par la même le niveau de réalisme d'un modèle induit une augmentation de la complexité du modèle. La précision d'émulation s'avère donc être un élément très important dans le sens où elle a une influence significative sur la qualité de l'émulation rendue. Il faut donc trouver un compromis entre le besoin requis de réalisme et la complexité du modèle d'émulation à implémenter.

Au niveau de l'implémentation, des ordres d'émulation seront créés et permettront de reproduire l'évolution des conditions au niveau du conditionneur de trafic. Cependant, il y a un facteur important dont il faut tenir compte : l'intervalle de mise à jour des ordres d'évolution des paramètres. En effet, si cet intervalle est impor-

tant, le modèle d'émulation aura beau être complexe et précis, le résultat obtenu par émulation, lui, ne le sera pas. Ainsi on met en lumière l'une des caractéristiques importantes d'un système d'émulation. A ceci, il faut ajouter le fait qu'il faille disposer d'une solution surdimensionnée.

Cet objectif de précision peut être mis en oeuvre de diverses façons grâce à des modèles analytiques, de l'analyse de traces réelles ou encore de la simulation. Comme cela a été vu précédement, la complexité des modèles d'émulation sera fonction du niveau de précision recherché. L'impact de cette précision d'émulation est fortement visible dans le résultat de l'émulation. En effet, un modèle trop simple ne permettra pas de reproduire un comportement précis ou réaliste et fournira à l'utilisateur des résultats pouvant être erronés mais surtout non réalistes. Quant à la simplicité des modèles d'émulation, elle s'entend en terme de traitements à réaliser.

Ainsi nous avons vu que les besoins de précision et de reproduction sont importants. En fait, on constate que la précision de l'émulation obtenue peut dépendre de la façon dont la reproduction est mise en place. Intéressons-nous maintenant, plus précisément à la manière dont doit être mise en oeuvre cette reproduction.

3.2 ARCHITECTURE FONCTIONNELLE D'ÉMULATION

3.2.1 Niveau d'émulation retenu

En tenant compte des besoins ainsi que de différentes possibilités d'émulation offertes par les outils existants, nous avons choisi de nous positionner sur une émulation de niveau IP. En effet, l'émulation de niveau inférieur, bien qu'elle apporte une plus grande précision, s'avère complexe et coûteuse à mettre en oeuvre (en termes d'équipements, de ressources, protocoles à mettre en place) et à contrôler. Dans l'objectif de trouver le bon compromis entre réalisme et complexité, nous avons choisi de nous baser sur une approche centralisée qui ne soit pas trop complexe à administrer, puis de l'améliorer si le besoin se fait sentir.

L'avantage de l'utilisation d'une émulation de niveau IP se situe également dans le faible nombre de paramètres de QdS que l'utilisateur doit manipuler. Ces paramètres sont les délais, les modèles ou taux de pertes et la bande passante.

La section suivante présente une architecture possible d'émulation permettant la mise en oeuvre d'une émulation de niveau IP.

3.2.2 Présentation de l'architecture

L'émulation réseau va donc apporter un moyen de tester les propriétés non fonctionnelles des implémentations réelles de protocoles. Pour ce faire, une architecture permettant d'introduire des modifications au niveau des caractéristiques du trafic circulant dans l'émulateur, en termes de délai, bande passante, pertes de paquets, erreur bit etc.) a été définie. La figure 3.1 présente l'architecture classique d'un système d'émulation réseau.

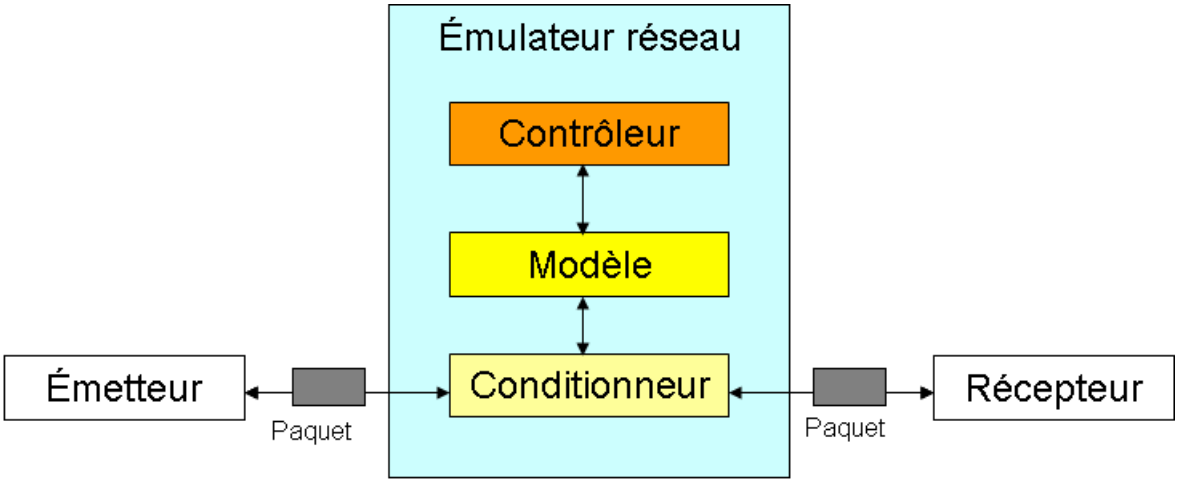


Fig. 3.1 Architecture classique d'un émulateur

L'émulation réseau est mise en oeuvre sur une plate-forme d'émulation. Un système d'émulation réseau ou émulateur réseau, se compose généralement de trois éléments : un conditionneur de trafic et un modèle d'émulation et un contrôleur d'émulation.

Le conditionneur de trafic ou processeur d'émulation, a pour mission de capturer les paquets arrivant au niveau des interfaces de l'émulateur et de les traiter en leur appliquant des modifications ou déformations en termes de délais, de taux de perte et de bande passante et ce, en temps réel. Le conditionneur est un composant de base de tout système d'émulation. En effet, sans conditionneur de trafic on ne peut pas vraiment parler d'émulation. On parlera alors de test en environnement réel. Le conditionneur de trafic est l'outil qui différencie l'expérimentation en environnement simulé de l'expérimentation en environnement émulé. Il va donc permettre d'appliquer au trafic réel arrivant au niveau des interfaces du système d'émulation les modifications au niveau des paramètres de QdS (latence, pertes d'informations, débits). Un conditionneur seul ne peut pas être considéré comme étant un émulateur car il lui faut un modèle d'émulation. Il est donc conduit par des modèles qui peuvent être plus ou moins complexes.

Ces modèles peuvent être intégrés au niveau du conditionneur (dans le cas de modèles de bas niveau) ou alors dépendre du contrôleur d'émulation si il existe (modèles de haut niveau). Les modèles permettent de reproduire plus ou moins précisément un comportement cible, incluant celui d'une technologie particulière ou des conditions artificielles spécifiques. En fait, la précision du système d'émulation dépendra surtout de la qualité des modèles. Ils ont donc un rôle majeur dans l'émulation au même titre que le conditionneur.

Le contrôleur d'émulation est en charge de l'application des modifications des paramètres de QdS en fonction des modèles. Il peut servir également dans le cadre de l'utilisation de modèles de haut niveau, à paramétrer la plate-forme d'émulation.

Cette configuration peut se faire au moyen de fichier XML par exemple. Une fois que le paramétrage de la plate-forme est réalisé, il peut alors contrôler le conditionneur en fonction des modèles.

3.3 UNE MÉTHODOLOGIE POUR L'ÉMULATION

Maintenant que les besoins vis à vis de l'émulation ont été présentés ainsi que l'architecture sur laquelle nous allons nous baser, nous allons nous intéresser à la manière de mettre en place l'émulation.

3.3.1 Présentation

De nos jours, lorsque l'on veut émuler un réseau, il y a un enchaînement de tâches qui sont réalisées implicitement par les chercheurs. Notre objectif est de proposer une méthode à suivre afin de formaliser ce raisonnement et d'arriver à un résultat conforme aux attentes des utilisateurs de l'émulation. La méthode proposée pour émuler un réseau est la suivante :

- une phase d'analyse qui permet d'identifier les caractéristiques du réseau cible ou de la condition à émuler,
- une phase de modélisation servant à modéliser ces caractéristiques, mais également servant à identifier le comportement du réseau cible et à déduire un modèle de comportement,
- une phase de vérification du modèle de comportement,
- et enfin une phase d'implémentation du modèle de comportement sur un outil d'émulation.

La figure 3.2 suivante présente la méthode d'émulation que nous avons proposée.

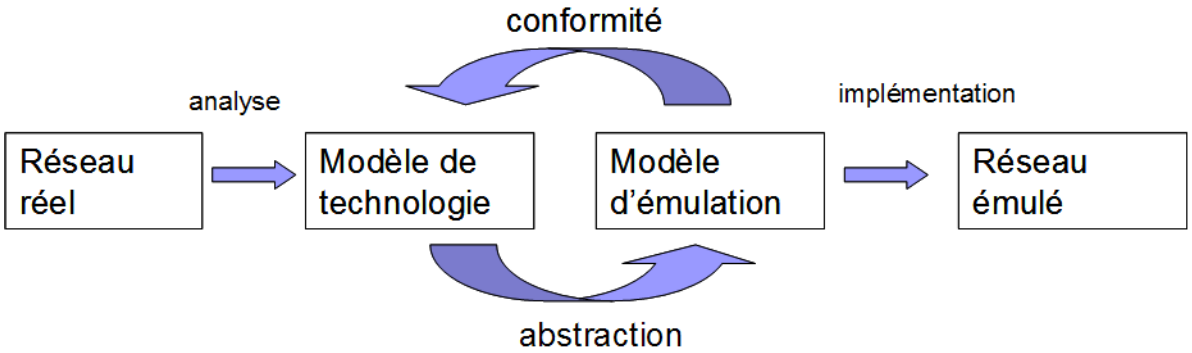


Fig. 3.2 Méthode d'émulation

3.3.2 Phase d'analyse

En vue d'établir un bilan précis de la condition ou de la situation considérée, il est impératif de réaliser une analyse de l'existant. Cette phase d'analyse a pour but de déterminer les besoins de l'utilisateur vis-à-vis du système d'émulation. L'objectif

final étant de définir le service que devra rendre le système d'émulation. Ainsi, l'analyse formelle des besoins permettra, si besoin est, de simplifier un problème ou une situation complexe et surtout de connaître de façon relativement précise les attentes de l'utilisateur de l'émulation par rapport au réseau cible.

Tout d'abord, il s'agira d'identifier les différents composants du réseau cible considéré, ceci dans le but d'établir un modèle de la topologie du réseau considéré. Puis, il s'agira de déterminer, en fonction des caractéristiques intrisèques des équipements faisant partie de la topologie (noeud, lien, etc.), les valeurs (maximales ou minimales selon les cas) des différents paramètres (en termes de délai, bande passante et taux de pertes). Ensuite, il s'agit d'étudier les différents échanges de données entre ces équipements à savoir, le nombre, la quantité, la fréquence, etc.

Après avoir défini les caratéristiques techniques de la condition ou du réseau à émuler, il s'agit de formaliser les besoins fonctionnels et non fonctionnels de l'utilisateur vis à vis de l'émulation.

Cette analyse a pour but de proposer un modèle de comportement qui soit proche de la réalité ou de la condition cible souhaitée. Elle est importante dans le sens où elle permet d'établir des modèles qui seront implémentés au sein du système d'émulation.

Les paramètres de QdS

Au niveau des paramètres à prendre en compte, nous allons considérer les délais, les taux de pertes et les bandes passantes. Plusieurs délais peuvent être pris en compte, mais ici l'objectif est de disposer du délai de propagation pour tous les équipements intervenant dans le réseau. Notons qu'il s'agit ici des valeurs maximales théoriques qui correspondent à des conditions théoriques où le réseau est non chargé. Il en est de même pour les taux de pertes ainsi que pour les bandes passantes moyennes. Ces valeurs permettront de définir les bornes maximales ou minimales et ainsi permettront de paramétrer le système d'émulation.

3.3.3 Phase de modélisation

La phase de modélisation comporte deux parties : une modélisation de bas niveau que nous appelerons modèle de technologie en référence à la technologie utilisée, et un modèle de plus haut niveau, la modèlisation de l'architecture d'émulation qui est issue d'abstractions du modèle de technologie.

Supposons que l'on soit dans un cas où on ne dispose ni d'implémentation réelle, ni de traces du réseau ou de la condition réseau à expérimenter soit parce que cette technologie n'est pas encore déployée, qu'elle est fortement peu probable, ou alors qu'il s'agit d'une solution propriétaire. Il faut pouvoir disposer d'un modèle sur lequel on pourra se référer afin d'établir des comparaisons et ainsi vérifier la cohérence des résultats.

La première partie consiste donc à réaliser une modélisation de niveau 2 au sens OSI du réseau ou de la condition réseau considéré (le modèle de technologie).

En se basant sur le niveau d'émulation choisi c'est-à-dire sur une émulation de niveau IP, la modélisation du système d'émulation offrira un canal de communica-

tion de bout en bout afin de réaliser le test d'applications ou de protocoles. Ce canal de communication permettra d'introduire des modifications au niveau des caractéristiques de QdS.

Modélisation de la technologie

La modélisation de la technologie est la première étape du processus de modélisation. Le modèle obtenu est issu des informations récupérées lors de la phase d'analyse et permet de représenter la topologie du réseau ainsi que les différents mécanismes de communication qui y interviennent.

Ainsi une première abstraction de la topologie du réseau cible, si il existe, est obtenue. Elle permet de mettre en évidence les différents acteurs ou mécanismes qui ont un impact significatif sur le comportement du trafic circulant sur le réseau.

Pour réaliser cette modélisation, nous avons choisi la méthodologie UML car elle offre un grand nombre de diagrammes, mais ce choix n'est pas limité à un outil ou une méthodologie de modélisation. L'objectif est de disposer de résultats obtenus grâce à la simulation du modèle.

En reprenant l'exemple cité précedemment, la figure 5.29 présente un modèle possible de technologie.

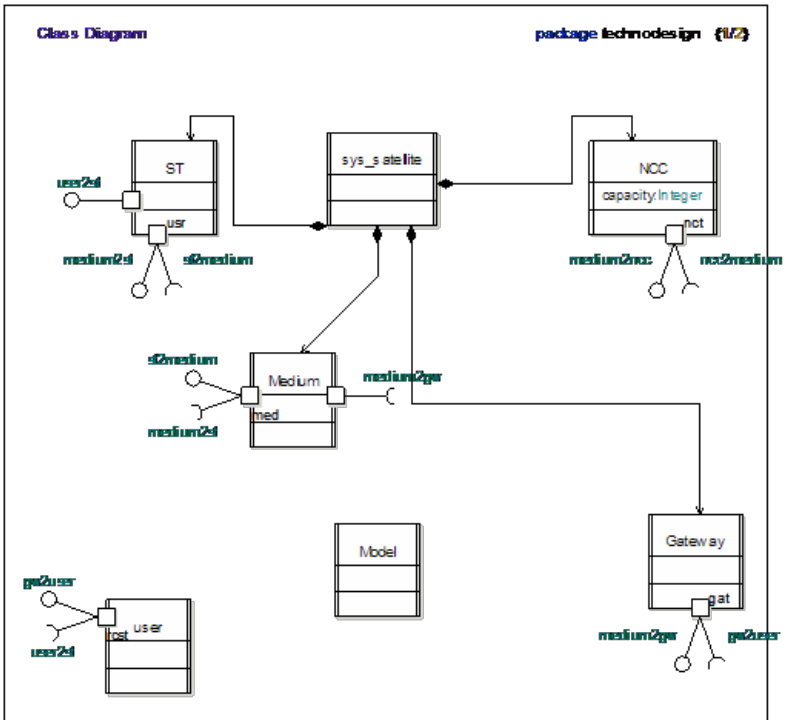


Fig. 3.3 Modèle de technologie d'une liaison satellite

Il s'agit ici d'un modèle simplifié mais cette technologie sera expliquée plus en détail dans le chapitre sur les études de cas. Ici l'objectif est juste de présenter

ce qu'est un modèle de technologie. Ainsi, sur ce modèle on fait abstraction des applications et on ne représente que la technologie du réseau considéré.

Modélisation de l'architecture d'émulation

La phase suivante consiste à définir le modèle d'architecture d'émulation qui sera implémenté au niveau du système d'émulation. Le modèle d'architecture d'émulation est obtenu après abstraction du modèle de technologie. Nous avons choisi de nous positionner au niveau IP pour mettre en oeuvre l'émulation. Ainsi, notre objectif pour la modélisation de l'architecture d'émulation est de se retrouver dans une configuration où le système d'émulation fournit un service de niveau IP c'est à dire qu'il est représenté comme étant un canal de communication de bout en bout capable d'appliquer des modifications au niveau des paramètres des paquets des flux. La méthodologie UML est également utilisée. Ainsi, comme le montre la figure 3.4 suivante, une expérience sera composée du système à tester qui peut être un protocole ou alors une application, et du système d'émulation lui même composé d'un canal de communication.

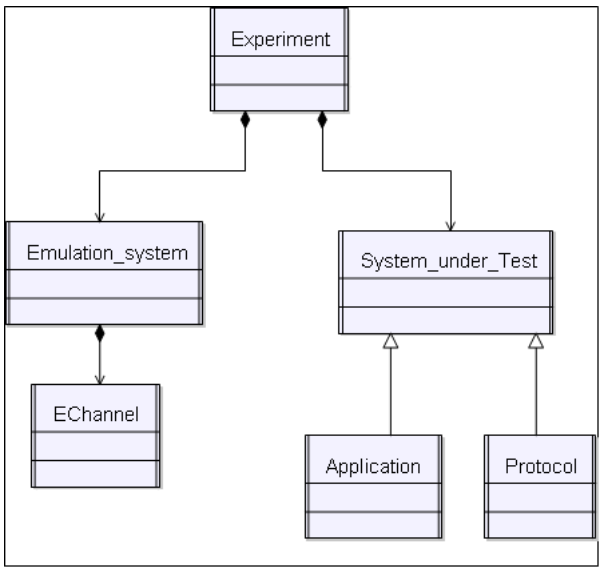


Fig. 3.4 Modèle UML d'émulation

Le diagramme de classes de la figure 3.5 présente une modélisation possible du canal d'émulation. Il est composé d'un ou plusieurs noeuds qui peuvent introduire des modifications au niveau du délai, des pertes et de la bande passante des paquets des flux, et qui sont associés à des modèles leur permettant d'accomplir ces actions.

Ainsi, comme présenté sur la figure 3.6, le canal d'émulation est modélisé comme étant une composition de noeuds d'émulation. Ces noeuds permettent de réaliser des actions (retard, perte, altération, etc.) au niveau des paquets du trafic traversant le système d'émulation. Un noeud d'émulation peut être vu comme un conditionneur de trafic élémentaire dans le sens où il est capable d'exécuter des actions élémentaires

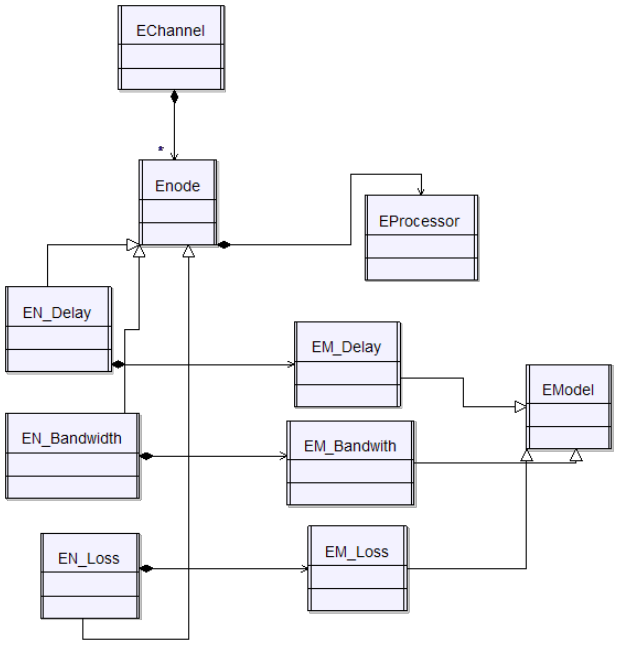


Fig. 3.5 Diagramme de classes du canal d'émulation

d'émulation. Dans l'exemple présenté, le trafic issu des applications ou du protocole à tester, transite par les noeuds d'émulation. Tout d'abord un noeud de pertes introduit des pertes suivant un modèle de pertes, ensuite un délai est appliqué par le noeud de délai et enfin le dernier noeud composant le canal d'émulation applique une bande passante au paquet. Les noeuds d'émulation sont sous le contrôle du Channelmodel.

Le canal d'émulation

Le résultat de l'émulation, c'est-à-dire ce que perçoit l'utilisateur n'est autre que le comportement du canal d'émulation. Les comportements observés pouvant être très simples (cas où le canal n'est composé que d'un seul noeud d'émulation) ou alors beaucoup plus complexes (enchaînement de plusieurs noeuds). Le canal d'émulation est une composition de noeuds d'émulation.

Le noeud d'émulation

Le noeud d'émulation est un élément composant le canal d'émulation. Il peut effectuer une tâche d'émulation en fonction d'un modèle sur les paquets du trafic qui traversera le système d'émulation. Sur la figure 3.5, on remarque qu'il est composé d'un processeur d'émulation ainsi que d'un modèle d'émulation. Le processeur d'émulation applique au paquet l'action décidée par le modèle d'émulation.

Le processeur d'émulation

Une action d'émulation est réalisée grâce à un processeur d'émulation. Cet objet

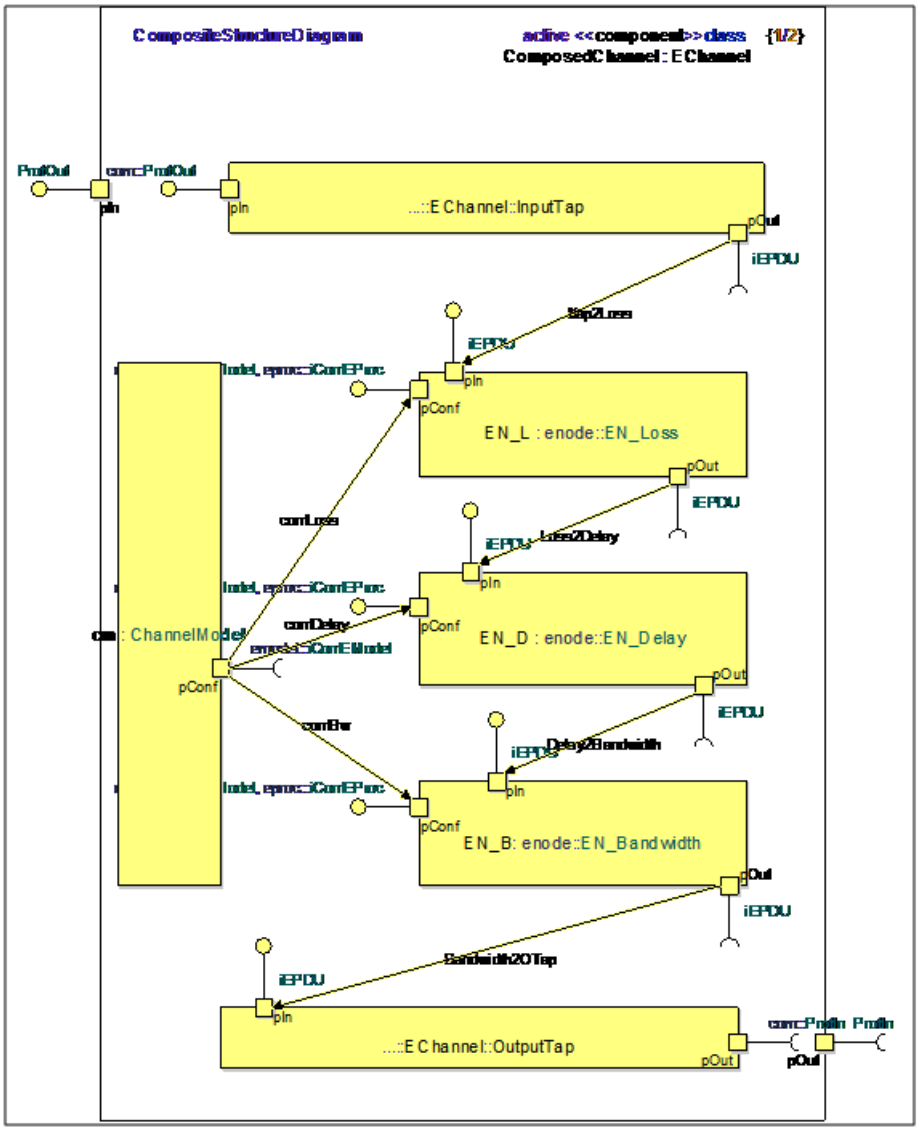


Fig. 3.6 Composite Structure Diagram du canal d'émulation

est en réalité celui qui remplit le rôle de trafic shapper dans le sens où il applique les modifications au niveau des caractéristiques des paquets.

Quelques exemples de modifications sont présentés avec les diagrammes d'état de quelques actions possibles d'émulation.

Le modèle d'émulation

Le modèle d'émulation spécifie le comportement de l'émulation. Il transmet au processeur d'émulation, aux instants adéquats, les ordres à exécuter afin de reproduire le comportement correspondant au système à émuler. Les actions de base sont par exemple, l'application d'un délai, d'une bande passante, l'introduction de pertes.

A partir de ces actions de base, il sera possible de construire des comportements plus complexes en les combinant, et ce, en fonction des besoins de l'utilisateur. Ce modèle permet de conduire ou de contrôler le processeur d'émulation dans le sens où il informe le processeur des actions à accomplir avec les paquets qu'il reçoit. Un exemple de modèle d'émulation est présenté sur le diagramme d'état de la figure 3.7.

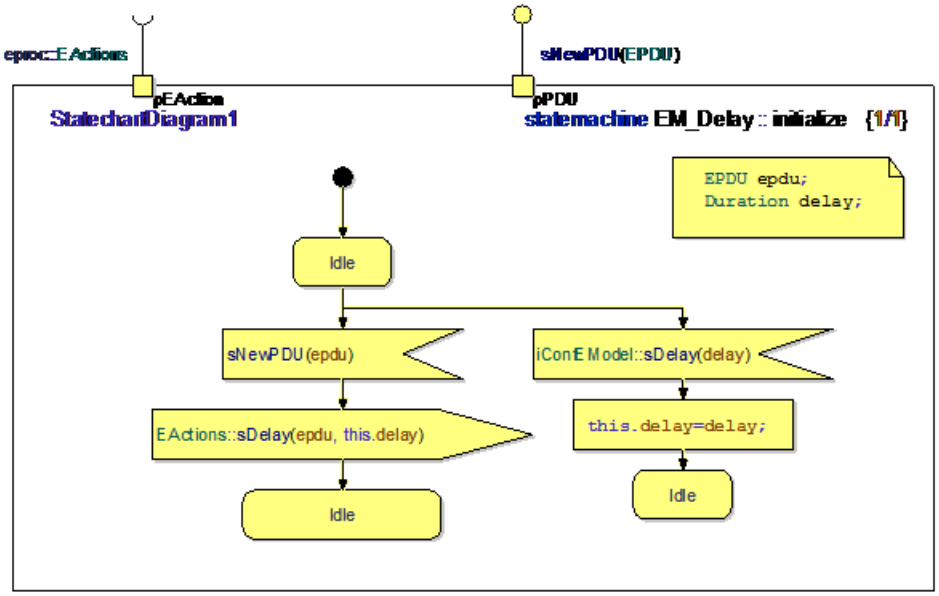


Fig. 3.7 Exemple de modèle d'émulation de délai

Lorsqu'un paquet arrive au niveau du processeur, il émet un signal au modèle correspondant. Ici par exemple, le modèle est un modèle de délai. Le modèle reçoit donc l'information qu'un paquet est disponible et il enverra sa réponse sous forme de signal au processeur une fois que le délai spécifié sera écoulé. Il est entendu que des modèles bien plus complexes peuvent être réalisés par l'utilisateur et ce, en fonction de ses besoins.

3.3.4 Phase de vérification

Après avoir modélisé notre système d'émulation, il faut le vérifier. La question qui se pose est de savoir si ce modèle permet effectivement de mettre en place, en temps réel, des modifications au niveau des paramètres des flux et plus précisément au niveau des paquets. Ces modification peuvent être des délais et des pertes. Généralement de manière intuitive, on réalise une première vérification en fonction des résultats obtenus lors de la conception des modèles. Cependant, elle ne permet pas de couvrir l'ensemble des cas possibles. Ce type de vérification repose sur une analyse des résultats de la simulation des modèles. Plusieurs vérifications peuvent être mises en place dans l'objectif de vérifier que le modèle obtenu peut être considéré comme étant une bonne abstraction de la réalité. En d'autres termes, vérifier que les

valeurs moyennes obtenues par simulation des modèles sont conformes à celle relevées lors de la phase d'analyse. Pour y parvenir, nous avons utilisé l'outil TURTLE [48] développé notamment par L.Apvrille [4] lors de sa thèse.

Nous sommes dans le cas de figure où nous souhaitons vérifier une spécification d'un modèle d'émulateur réalisant des actions en temps réel. La spécification a été réalisée en utilisant UML. Or, TURTLE est un profil UML temps réel. Etant données les phases amont du cycle de développement de système, nous allons faire de la vérification à priori.

Avec TURTLE on dispose d'une sémantique formelle pour exprimer le temps. C'est un profil outillé dans le sens ou on va pouvoir utiliser RTL qui est un outil de vérification par analyse d'accessibilité qui utilise le langage formel RT-LOTS, et Ttool qui est une interface UML vers RT-Lotos.

Notre objectif ici est uniquement un objectif de vérification temporelle des exigences que nous aurons spécifiées. Nous ne nous attachons pas à la vérification d'architectures ou alors de tautologies. Ainsi l'outil TURTLE se trouve être le plus adapté.

Le modèle d'émulation obtenu doit être vérifié afin de s'assurer de la conformité du modèle avec les attentes de l'utilisateur. Cela dit, il faut remarquer qu'ici la vérification n'intervient pas au niveau du protocole ou de l'application à tester mais plutot au niveau du service rendu par le système d'émulation. En effet, l'objectif est de s'assurer que les contraintes qui pourront être spécifiées seront respectées au niveau du modèle d'émulation mais également au niveau de l'implémentation.

Exemple de vérification

La figure 3.8 présente un exemple de vérification.

Ce qui nous intéresse ici est la vérification du comportement. Une abstraction de l'application a été réalisée. Dans cet exemple, nous allons considérer une application d'échange de paquets. Le comportement d'émulation traité ici est l'introduction d'un délai variable (de 1700 unités de temps à 2000 unités de temps) ou la perte d'un paquet. Le principe que nous avons utilisé et qui est abordé de manière plus précise dans les travaux de thèse de B.Fontan, est de mettre en place un observateur temporel.

Comme le montre la figure 3.9, le processeur d'émulation applique au paquet le conditionnement dépendant de l'ordre reçu du modèle d'émulation. Si il reçoit 0 il applique un délai de 400 unités de temps au paquet et le transmet sinon s'il reçoit 1 il perdra le paquet. Il s'agit ici d'un comportement simple. Les diagrammes suivants de la figure 3.10a et 3.10b représentent le comportement du modèle ainsi que de l'observateur.

La génération des graphes d'accessibilité montre qu'effectivement les paquets seront reçus au minimum après un délai de 1700 unités de temps et au maximum après 2000 unités de temps. Ceci permet d'affirmer que le modèle de notre système d'émulation produira un résultat d'émulation conforme aux besoins et attentes de l'utilisateur.

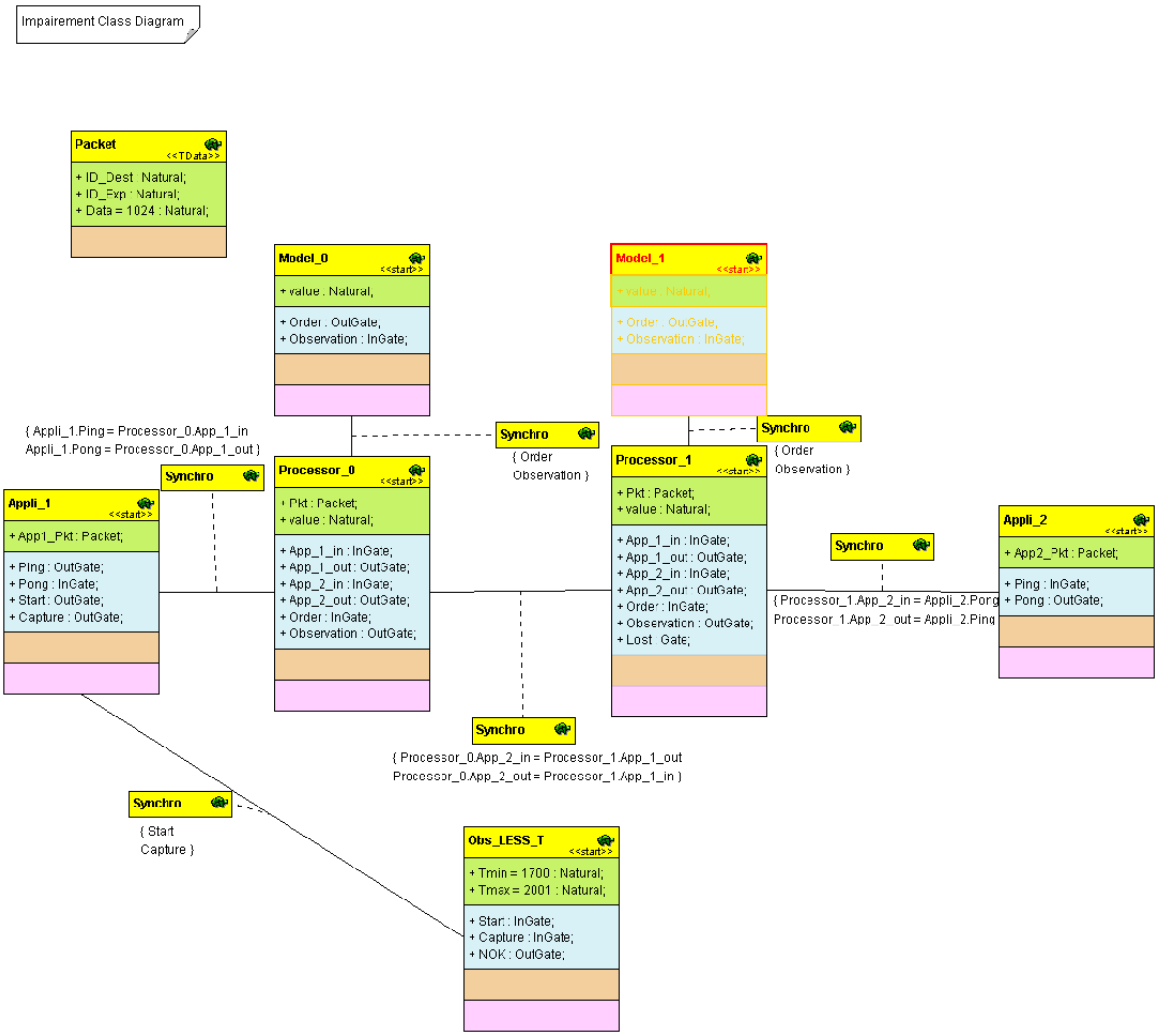


Fig. 3.8 Principe de vérification

3.3.5 Phase d'implémentation

L'implémentation du modèle d'émulation qui aura été vérifié va permettre d'obtenir un émulateur qui normalement répondra aux besoins spécifiés par l'expérimentateur. Cette implémentation a pour l'instant été réalisée sur la plate-forme NINE (NINE Is a Network Emulator) définie à l'ENSICA. Elle utilise Dummynet présenté précédemment comme conditionneur de trafic auquel nous avons apporté quelques améliorations. Cela dit, elle pourrait l'être en utilisant n'importe quel conditionneur de trafic. L'implémentation de notre système d'émulation ainsi que la plate-forme d'émulation seront présentées plus précisément dans le chapitre 5 avec les cas d'étude.

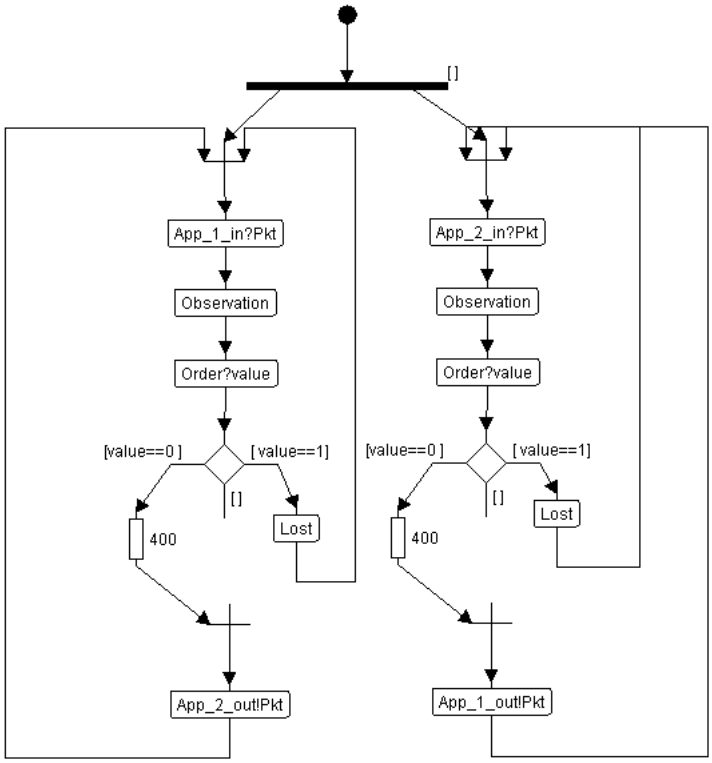
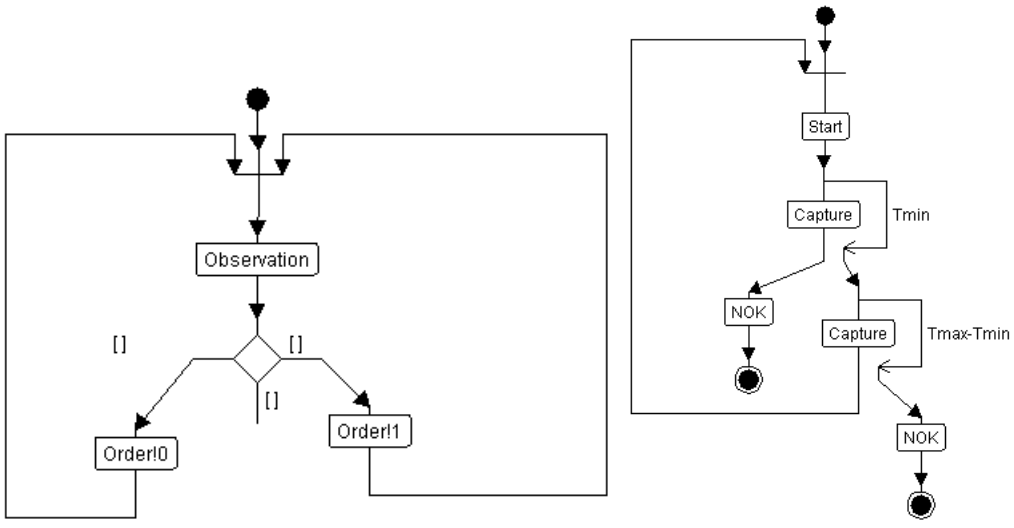


Fig. 3.9 Description du comportement du processeur



(a) Comportement du modèle (b) Comportement de l'observateur

Fig. 3.10 Comportements du modèle et de l'observateur

3.4 CONCLUSION

La méthodologie d'émulation proposée se base sur une phase d'analyse qui permet de définir le besoin de l'utilisateur de l'émulation. Ensuite une phase de modélisation

permet dans un premier temps avec le modèle de technologie d'obtenir une première modélisation. Cela permettra ensuite de déduire le modèle d'émulation qui sera finalement implémenté. Une vérification est effectuée entre la conformité des attentes de l'utilisateur et les différents modèles ainsi qu'avec l'implémentation.

Dans cette section nous avons présenté le fonctionnement des noeuds d'émulation composant le canal de communication, mais nous n'avons pas abordé les modèles qui conduisent ces noeuds. C'est ce qui nous ferons dans le chapitre suivant.

Chapitre 4

Emulation de comportements

Dans ce chapitre, j'aborderai les points suivants :

- ⋆ modélisation d'un comportement ;
- ⋆ modélisation du trafic concurrent ;
- ⋆ equation-based emulation ;
- ⋆ émulation active ;

Lorsque l'on parle d'émulation réseaux, il faut considérer deux aspects importants : tout d'abord, l'émulation de comportements dépendant de la technologie et des protocoles associés qui constitue le coeur même de l'émulation ; puis l'émulation d'événements extérieurs pouvant survenir sur la technologie réseau considérée qui a pour but d'augmenter le réalisme de l'émulation rendue.

Dans ce chapitre, deux contributions seront présentées. La première se situe au niveau de l'émulation de comportement. Après un état de l'art des méthodes de modélisation de comportement, les limites de ces approches seront présentées ainsi qu'une solution pour y répondre. Il s'agit de l'émulation active. Une seconde contribution concerne la modélisation des aspects extérieurs et pour ce faire, nous allons nous intéresser à la modélisation du trafic concurrent circulant sur le réseau d'expérimentation. Ces deux aspects constitueront le modèle complet d'émulation.

4.1 COMMENT MODÉLISER UN COMPORTEMENT ?

L'émulation de réseau permet, comme cela a été présenté précédemment, de reproduire un comportement d'un réseau cible donné sur une plate-forme d'expérimentation. Cette reproduction du réseau cible conduit le plus souvent à réaliser des concessions au niveau du réalisme de l'émulation rendue. En effet, pour mettre en oeuvre le comportement du réseau cible sur la plate-forme d'émulation, il faut le modéliser. Or, un modèle est une vue abstraite, donc simplifiée, de la réalité. Ainsi

ces simplifications entraînent inévitablement un niveau de précision et de réalisme moins élevé.

L'objectif de cette section est de présenter les différents modèles possibles permettant de reproduire un comportement réseau donné, mais également de faire ressortir leurs limites notamment dans des situations où ces modèles s'avèrent insuffisants. Ainsi, deux contributions de cette thèse seront présentées : l'approche d'émulation active, ainsi qu'une utilisation particulière de l'émulation active se basant sur des modèles d'émulation utilisant des équations permettant de définir un comportement.

4.1.1 Présentation des différents types de modèles de comportement

Une définition du comportement est la suivante : *« Le comportement d'un système quelconque est la partie de son activité qui se manifeste à un observateur »*. Un comportement réseau se résume à l'activité observable du réseau. Puisque nous avons choisi de nous positionner au niveau IP, pour caractériser un comportement réseau en terme de paramètres de QdS nous allons considérer les bandes passantes, les délais et les taux de pertes. Les modèles permettant de caractériser le comportement sont de différents types et nous allons en voir quelques uns.

4.1.2 Les modèles statiques

Afin de mettre en oeuvre des expérimentations dans des conditions particulières et pas forcément réalistes, il est possible d'utiliser des modèles statiques, c'est-à-dire des modèles où les paramètres n'évoluent pas au cours du temps. Il s'agit de la façon la plus « simple » de mettre en oeuvre l'émulation. Généralement, dans ce type d'approche, la topologie ainsi que les caractéristiques de la technologie réseau considérée n'évoluent pas. Elle consiste à définir et à fixer les valeurs des différents paramètres de QdS de l'outil d'émulation. Concrètement, il s'agit d'émuler le comportement réseau tel qu'il a été décrit à un instant donné. Cette émulation est mise en oeuvre grâce à des conditionneurs de trafic qui ont été présentés dans le chapitre 1.

Ces modèles sont utiles car ils permettent sans trop de difficultés de vérifier le fonctionnement d'un protocole ou d'une application dans des conditions précises comme par exemple les conditions optimales de fonctionnement. Ceci a pour but de valider une première version du protocole ou de l'application ou encore de reproduire des résultats. Dans le cadre de tests préliminaires les modèles statiques peuvent servir à mettre en évidence un comportement en faisant varier un paramètre donné tout en fixant les autres, par exemple, la mise en évidence du comportement du protocole ou de l'application avec des délais différents tout en fixant les autres paramètres de QdS. Il se peut également que l'expérimentateur cherche à tester des conditions particulières telles que la recherche de limites de fonctionnement.

Lors des tests préliminaires des protocoles multicast par satellite du projet DIPCAST par exemple, pour émuler une liaison satellite, une émulation statique suffisait. En effet, l'objectif était juste de s'assurer du bon fonctionnement des mécanismes. Ainsi, la liaison satellite se résumait à un canal de bout en bout de 300ms

de délai. Cette modélisation de très haut niveau du satellite suffisait car le principal besoin était de vérifier le comportement du protocole dans un contexte de délai important.

Ainsi, le principal avantage de ce type de modèle d'émulation est la relative simplicité d'utilisation qui permet d'obtenir des résultats précis ainsi que de bonnes performances. Cependant, même si ce type de modèle peut s'avérer suffisant pour de nombreux besoins, il apparaît un besoin en terme de contrôle du comportement émulé. En effet, les configurations simples proposées par les conditionneurs de trafic ne permettent pas de produire un comportement réseau précis dans le sens où il n'est pas possible de prendre en compte des algorithmes complexes contrôlant les modifications à effectuer au niveau des paquets. Ainsi le réalisme se trouve être affecté. En effet, les conditions rencontrées sur un réseau opérationnel sont rarement figées. Ce type de modèles ne permet pas de mettre en oeuvre des évolutions des paramètres. Dès lors que le comportement du réseau à émuler devient complexe c'est-à-dire qu'il induit une dynamicité au niveau des paramètres de QdS, ce type d'émulation ne suffit pas. Prenons pour exemple, l'accès à une ressource partagée. Reproduire ce comportement au niveau de l'outil d'émulation suppose être capable de faire évoluer les valeurs des différents paramètres de QdS au cours du temps. La principale limite de ce type de modèles d'émulation se situe donc ici.

4.1.3 Les modèles dynamiques

Comme cela a été dit précédemment, les conditions rencontrées sur un réseau opérationnel donné sont souvent variables. Cette variabilité se traduit par une évolution des paramètres de QdS due soit au comportement induit par les protocoles de bas niveau, ou à des événements survenant sur le réseau comme par exemple des congestions, des variations météorologiques (dans le cas de réseaux satellites, ou sans fil), perturbations électromagnétiques, etc. . Les modèles statiques ont montré leurs limites à ce niveau et il faut disposer de modèles proposant une évolution des paramètres de QdS au cours du temps. On peut distinguer plusieurs types de modèles dynamiques : l'utilisation de scénarios temporisés comme cela a été présenté par Herrscher and al[30]. et repris sur la plate-forme d'émulation NINE [37], le rejeu de traces Noble and al. [7], ou des approches hybrides utilisant des combinaisons de ces deux techniques.

Scénarios

La scénarisation est la première méthode à laquelle on pense lorsque l'on veut reproduire un comportement dynamique. Il s'agit de permettre à l'utilisateur de créer une description temporelle de l'expérience qu'il souhaite réaliser. Cette description correspond à une suite d'actions à réaliser lors d'intervalles de temps. Un exemple de scénario : dans l'intervalle de temps compris entre 10ms et 15ms, le système d'émulation devra appliquer une modification au niveau du délai et du taux de pertes des paquets du ou des flux qui transiteront au niveau de ses interfaces.

L'utilisation des scénarios permet à l'expérimentateur de contrôler les actions à réaliser au niveau du système d'émulation. Cette approche introduit une dynamicité au niveau de l'émulation, ce qui la rend plus précise que celle obtenue avec les modèles statiques. En reprenant l'exemple du projet DIPCAST, lors de la phase de test, un autre objectif était de mettre en évidence la fiabilité du protocole en fonction des différentes conditions de propagation. La figure 4.1 montre un exemple de chronogramme qui a servi à définir des scénarios pour paramétrer le système d'émulation. Ces valeurs sont issues de mesures réelles récupérées sur un environnement satellite. L'objectif était d'évaluer la fiabilité au niveau transport d'un protocole. Il fallait donc introduire des pertes correspondant à différentes conditions de propagation.

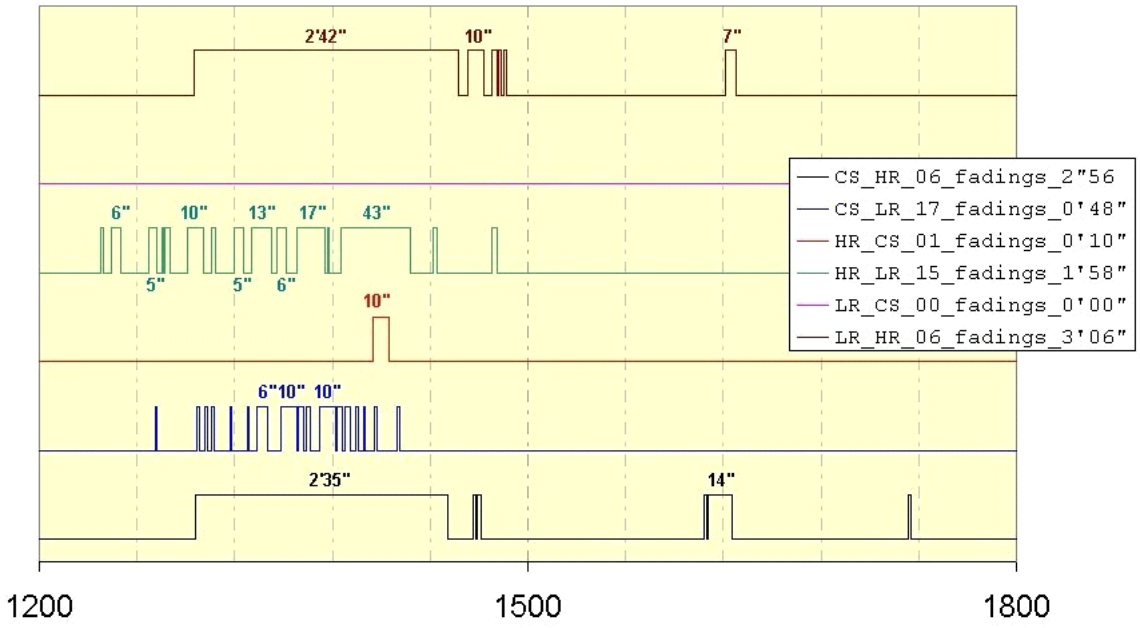


Fig. 4.1 Evolution de la fiabilité de la transmission selon différentes conditions de propagation

Les lignes de ce chronogramme correspondent à différentes conditions de propagation (HR : Heavy rain, CS : Clear sky, LR : Light rain). Ainsi, il est possible de reproduire des conditions atmosphériques variables qui se traduisent par une évolution du taux de pertes du canal de communication (passant de 0% à 100% sur des intervalles de temps).

Cependant, la granularité des intervalles de mise à jour des paramètres a un impact sur le niveau de réalisme de l'émulation rendue. Il est clair qu'un intervalle de mise à jour très grand ne propose pas un niveau de réalisme important. Néanmoins, il faut noter qu'un intervalle réduit, implique une complexité plus importante au niveau du système d'émulation car il induit un nombre de mises à jour plus important. Il s'agira donc de trouver une solution proposant à la fois un niveau de complexité maîtrisable tout en ne sacrifiant pas la précision et le réalisme.

Implémentation Pour implémenter les scénarios d'émulation, nous nous sommes tournés vers *XML (eXtended Markup Language)* le format de description défini par le consortium *W3C (World Wide Web Consortium)* qui s'est maintenant imposé comme un standard de fait. L'utilisation de XML pour écrire les scénarios d'émulation permet d'avoir des fichiers compréhensibles par un être humain et de pouvoir les valider formellement à l'aide de schémas. Cet aspect de validation est très important car un utilisateur doit être assuré que le scénario qu'il va utiliser est sémantiquement juste pour pouvoir placer une confiance minimale dans l'émulation qui va être rendue. Ainsi, lorsqu'un utilisateur souhaite émuler des conditions particulières, il peut écrire le scénario à la main et le valider grâce à l'utilisation des schémas qui lui sont fournis.

Le formalisme RELAX NG compact Le formalisme *RELAX NG compact* est défini par le consortium *OASIS (Organization for the Advancement of Structured Information Standards).* Ce formalisme est proche de celui d'une grammaire Backus-Naur, ce qui le rend beaucoup plus lisible qu'un schéma XML classique (au format XSD).

La grammaire utilisée par *RELAX NG compact* est relativement simple et se compose :

- d'éléments (notés *element*) qui vont permettre de définir les balises XML,
- d'attributs (notés *attribute*) qui permettent de définir les attributs que l'on retrouve à l'intérieur d'une balise XML.

```
- <carnet_d_adresses>
  - <contact>
      <nom>Martin</nom>
      <prenom>Pierre</prenom>
      <surnom>Pierro</surnom>
      <tel id="bureau">0561616161</tel>
      <tel id="portable">0660606060</tel>
      <mail>pierro@boulot.com</mail>
    </contact>
  - <contact>
      <nom>Dupont</nom>
      <prenom>Jaques</prenom>
      <tel id="bureau">0561616161</tel>
      <mail id="perso">jacques.dupont@yahoo.com</mail>
      <mail>jdupont@entreprise.com</mail>
    </contact>
</carnet_d_adresses>
```

(a) Exemple de carnet d'adresse en XML

```
start = carnet_d_adresses

carnet_d_adresses =
    element carnet_d_adresses {
        contact+
    }

contact = element contact {
    (
        element nom {xsd:string}&
        element prenom {xsd:string}&
        element surnom {xsd:string}?
    ),
    element tel {
        attribute id { "bureau" | "portable"},
        xsd:integer
    }+,
    element mail {
        attribute id { xsd:string }?,
        text
    }*
}
```

(b) schéma associé

Fig. 4.2 Exemple de schéma RELAX NG compact associé à un fichier XML

Supposons que nous voulions donner le schéma *RELAX NG compact* du fichier XML décrivant un carnet d'adresse qui est présenté sur la figure 4.2. Sans trop rentrer dans les détails, voici quelques éléments de syntaxe nécessaires pour la compréhension de ce schéma ainsi que ceux que nous présenterons dans la suite de ce chapitre.

- (...) : Les parenthèses permettent de grouper différents morceaux du schéma. Le nom et le prénom d'un contact sont indissociables l'un de l'autre.
- * : Ce symbole permet d'exprimer une multiplicité $0..n$. Il ne peut être utilisé que sur des éléments. Dans l'exemple qui nous intéresse cela veut dire que les adresses e-mails (*element mail*) ne sont pas obligatoires mais qu'il peut y en avoir plusieurs.
- + : Ce symbole permet d'exprimer une multiplicité $1..n$. Tout comme le symbole précédent, il ne peut être utilisé que sur des éléments. Le carnet d'adresses doit donc avoir au moins un contact et chaque contact doit avoir au moins un numéro de téléphone pour pouvoir être contacté.
- ? : Ce symbole permet d'exprimer qu'un élément ou un attribut peut être omis dans le fichier XML. Dans le carnet d'adresses, l'élément *surnom* désignant un contact n'est pas obligatoire. L'attribut *id* permettant de différencier des adresses e-mail peut également être omis.
- , : La virgule est un symbole d'association impliquant une idée d'ordre. Ainsi deux éléments ou attributs séparés par une virgule dans le schéma doivent être décrits dans le même ordre dans le fichier XML (X,Y != Y,X). Sur l'exemple nous voyons qu'il faut donc toujours commencer un contact par le bloc nom, prénom et éventuellement le surnom, puis donner un numéro de téléphone et ensuite éventuellement une adresse e-mail.
- & : Le symbole & permet d'exprimer une association sans notion d'ordre. Ainsi deux éléments ou attributs séparés par un symbole & dans le schéma devront être présent dans le fichier XML mais leur position ne sera pas importante (X&Y = Y&X). Pour un contact il est donc possible de commencer par spécifier le surnom avant de donner le nom et le prénom.
- | : Ce symbole permet d'exprimer un choix entre deux éléments. Le téléphone a ainsi un attribut obligatoire qui peut prendre soit la valeur "bureau" soit la valeur "portable".

Enfin, le formalisme *RELAX NG compact* permet d'utiliser les types XSD pour caractériser le type d'un élément ou d'un attribut. Par exemple, le numéro de téléphone sera dans l'exemple représenté par un type *xsd :integer*. Il est ainsi possible d'utiliser dans les schémas les types xsd :ID et xsd :IDREF qui permettent d'identifier un élément puis de le référencer dans la suite du document. Une présentation plus complète de ce formalisme est disponible dans le tutoriel [13].

Exemple de scénario Le scénario d'émulation va décrire le comportement du réseau à émuler au cours du temps. Il contiendra les valeurs des différents paramètres nécessaires à l'émulation. La figure suivante présente un exemple de scénario.

L'élément *scenario* sert à stocker l'ensemble des règles caractérisant les conditions de QdS sous forme de scénario temporisé. Le schéma permettant de valider le *scenario* est décrit sur la figure 4.3. Un scénario temporisé se découpe en une suite de dates et à chaque date correspond une suite de mises à jour de la QdS des liens. L'utilisation d'un type *xsd :IDREF* dans le schéma permet de s'assurer que

le lien sur lequel la mise à jour va avoir lieu a bien été défini. Les paramètres de QdS dynamiques sont spécifiés dans l'élément *link_ update* et sont valables à la date spécifiée dans l'élément englobant *date*. Ces paramètres vont ensuite rester valables jusqu'à la prochaine spécification d'un *link_ update* pour ce lien.

```
scenario =
    element scenario {
        element number_of_event { integer },
        element time_step { duration_unit, real},
        date+
    }

date =
    element date {
        attribute id { xsd:ID },
        attribute start { positiveReal },
        duration_unit,
        link_update+
    }

link_update =
    element link_update {
        attribute hidden { integer }?,
        attribute hidden_id { xsd:string }?,
        element on_link {attribute id { xsd:IDREF}},
        (
            element bandwidth { bandwidth_unit & real}&
            element delay { integer }&
            (
                element plr { positiveReal } |
                element loss_pattern { xsd:string }
            )&
            element queue_size {integer}?
        )
    }
```

(a) schéma

```
- <scenario>
    <number_of_event>360</number_of_event>
    <time_step unit="ms">500.0</time_step>
  - <date ident="t0.0" start="0" unit="ms">
    - <link_update>
        <on_link id="M1 to F1" />
        <bandwidth unit="b/s">4260000.0</bandwidth>
        <plr unit="percent">0</plr>
        <delay>0</delay>
        <queue_size>8</queue_size>
      </link_update>
    + <link_update>
    </date>
  + <date ident="t7.5" start="7500" unit="ms">
  + <date ident="t30.0" start="30000" unit="ms">
```

(b) exemple XML associé

Fig. 4.3 Extrait d'un scénario d'émulation pour la partie scénario

Ce scénario étant utilisé pour faire de l'émulation de niveau IP, les paramètres de QdS à manipuler pour faire évoluer la QdS d'un lien sont : les bandes passantes, les délais et les pertes. Comme nous pouvons le voir sur le schéma présenté sur la figure 4.3, il y a deux gestions possibles des pertes : soit spécifier un taux de pertes fixe, soit utiliser un fichier où sont spécifiées les positions des pertes.

Rejeu de traces

Si on considère que la précision de la reproduction des conditions est l'objectif principal que doit remplir un système d'émulation, une solution pour obtenir les meilleurs résultats est le rejeu de traces. Les plate-formes mettant en oeuvre l'émulation par rejeu de traces comme par exemple celle proposée par Noble et al. [7] reproduisent le comportement d'un réseau cible sur un réseau d'expérimentation en se basant sur des traces capturées préalablement. Le rejeu de traces est en fait un type particulier de scénario. Ce scénario correspondra à celui enregistré sur un réseau cible sur une période donnée. Le modèle de comportement obtenu sera exactement celui du réseau cible.

Ce modèle conduira par la suite le conditionneur de trafic. Dans ce cas, des traces (mesures) sont récupérées, analysées puis rejouées. Ce type d'émulation repose sur trois phases. Tout d'abord une phase de collecte des traces, puis une phase de distillation qui correspond en fait à l'analyse des traces. Enfin une phase de modulation où les traces seront rejouées.

Cette approche a l'avantage de proposer un comportement d'émulation très proche du comportement du réseau cible : on rejoue l'évolution des conditions d'un réseau cible donné. Mais finalement, on se rend compte que même s'il s'agit d'un modèle dynamique, ce type d'émulation est relativement figé dans le sens où les conditions reproduites par les traces correspondent à des mesures réalisées à des instants donnés sur le réseau cible. L'utilisateur n'a donc que peu de contrôle sur les paramètres : il ne peut pas les faire évoluer et de ce fait, ne peut pas tester des conditions particulières. De plus il faut noter que la phase de distillation peut s'avérer relativement délicate à mettre en oeuvre. En effet, on peut se poser la question de savoir quelles sont les informations pertinentes parmi toutes les valeurs récupérées. Et donc comment être sûr de ne pas "manquer" une donnée importante.

Ce type d'approche permet d'émuler des topologies réseau complexes tout en proposant un bon niveau de réalisme. Néanmoins, le problème majeur réside dans le fait que cette approche ne propose pas d'interactivité avec l'expérimentateur. Il ne peut pas faire varier les paramètres de QdS selon ses besoins. Par exemple, le test des conditions particulières comme la recherche des limites de fonctionnement s'avère difficile.

Utilisation de simulateur hors ligne

Une autre solution consiste, comme cela a été développé dans la thèse d'E.Conchon [15], à utiliser un simulateur à événements discrets qui sera chargé de pré-calculer les valeurs des paramètres à insérer dans un scénario qui conduira le système d'émulation. L'intérêt d'une telle méthode est de pouvoir diminuer le nombre de calculs à réaliser (dans le cadre de réseaux sans fil) et ainsi pouvoir réduire la complexité (en termes de calculs) des modèles à utiliser. De plus, elle permet d'augmenter de manière non négligeable le niveau de réalisme et la précision de l'émulation rendue dans

le sens où la simulation hors ligne permet d'utiliser des modèles bien plus complexes car n'étant pas soumis à une contrainte de temps réel.

La limitation de ce type d'approche se situe au niveau de la restitution de l'émulation proposée. En effet, il faut que le système d'émulation soit capable de restituer le même niveau de précision que celui proposé par le simulateur. Par exemple, si le simulateur fournit des résultats de l'ordre de la microseconde, il faut que le système d'émulation propose une telle granularité sinon le comportement rendu par le système d'émulation ne sera pas en adéquation avec celui obtenu par simulation. De même, cette approche implique de la part de l'expérimentateur une connaissance préalable de tous les événements pouvant survenir lors de son expérience afin de définir le scénario à simuler. De plus, cette solution ne permet pas la prise en compte d'événements extérieurs autres que ceux faisant partie du scénario simulé.

4.1.4 Conclusion

Dans cette première partie, nous avons vu que pour modéliser un comportement réseau, il existe plusieurs solutions possibles en fonction des besoins. Ainsi, les différents types d'émulation présentés permettent de satisfaire un nombre important de besoins. Ils permettent tout d'abord de mettre en place les tests fonctionnels préliminaires (modèles statiques). Puis, l'impact de paramètres peut être mis en évidence en fixant les valeurs de paramètres et en faisant évoluer les autres avec des scénarios. Cependant le résultat d'émulation obtenu n'est pas vraiment réaliste dans le sens où le modèle d'émulation est vraiment simplifié.

Ces modèles montrent leurs limites dès lors qu'il s'agit de reproduire des comportements complexes comme par exemple un accès à une ressource partagée. Dans la section qui suit, nous proposons donc une nouvelle approche qui étend l'émulation dynamique pour répondre aux limites rencontrées : l'émulation active. Dans cette nouvelle approche, l'émulation sera conduite par le trafic lui-même et elle sera capable de réagir à des événements induits par ce trafic.

4.2 L'ÉMULATION ACTIVE

Une limitation forte apparaît avec les modèles d'émulation présentés dans la section précedente. Ils ne peuvent imiter des mécanismes complexes qui utilisent une interaction avec le trafic généré par les applications ou les protocoles. Ainsi, le concept d'émulation active qui est proposé dans [44] fait allusion aux réseaux actifs. L'objectif sera la capacité de réagir à un événement survenant sur le réseau et ainsi décider de la conduite à tenir en fonction de cet événement. Ce type d'émulation permet d'expérimenter de façon plus précise les applications ou les protocoles grâce à sa capacité à réagir à des événements spécifiques. Les décisions à prendre peuvent dépendre de modèles basés sur des équations ou alors sur des machines à états ou encore des chaînes de Markov. Ce type d'émulation permet ainsi de répondre au besoin d'interactivité.

4.2.1 Présentation du concept

Certains réseaux opérationnels produisent des variations au niveau des paramètres de QdS qui nécessitent des mécanismes ainsi que des modèles complexes afin d'être émulés. La complexité de ce type d'émulation peut également provenir du comportement dynamique des protocoles utilisés sur ces réseaux. Les protocoles peuvent réagir en fonction de paramètres internes comme leurs propres mécanismes ou alors en fonction de facteurs externes tels que le trafic transitant par le système d'émulation. Prenons l'exemple d'une liaison satellite et plus particulièrement une des méthodes d'accès utilisée pour le partage de ressources sur la voie retour : le DAMA. Cette méthode d'accès sera présentée en détail dans le chapitre suivant. Elle propose quatre classes d'allocation de trafic pouvant être combinées, certaines dépendantes des caractéristiques du trafic et d'autres non. Ainsi, le comportement de cette méthode d'accès n'est pas entièrement prévisible. C'est pourquoi des configurations dynamiques du système d'émulation sont nécessaires. La seule manière d'obtenir un comportement d'émulation réaliste consiste à rendre le modèle d'émulation capable de réagir en temps réel à divers facteurs tels que le trafic expérimental qui traverse le système d'émulation. Ainsi, l'émulation doit être « active » : le trafic expérimental peut modifier la configuration du système d'émulation et, en conséquence, ces modifications auront un impact sur le trafic circulant à travers le système d'émulation.

4.2.2 Architecture d'émulation active

Afin de permettre la prise en compte des événements survenant sur le réseau, il faut disposer d'une architecture capable de réagir à des situations ou des événements se produisant au niveau des interfaces du système d'émulation. La figure 4.4 présente cette architecture composée de deux parties : une partie dont le but est de récupérer les informations au niveau des flux de données et une autre embarquant le modèle d'émulation dont le rôle est de calculer et d'appliquer les modifications au niveau des paramètres des flux transitant par le système d'émulation.

Description de l'architecture d'émulation

L'architecture d'émulation que nous proposons permet l'implémentation du concept d'émulation active. Cette architecture doit être capable de prendre en compte les informations du trafic qui transite au niveau des interfaces du système d'émulation et également de réagir en fonction des modèles d'émulation actifs. Comme cela est présenté sur la figure 4.4, l'architecture se compose de deux types de modules : les modules de bas niveau dépendants de l'implémentation ainsi que des modules de haut niveau qui eux sont indépendants de l'implémentation. L'architecture propose un niveau d'abstraction élevé pour l'émulation dans le but de simplifier la création de comportements réseau. Ainsi grâce aux modèles de haut niveau, il est possible de paramétrer la topologie du réseau considéré ainsi que de définir les règles qui permettront de reproduire le comportement souhaité. De plus, cette architecture doit permettre la prise en compte des autres modèles de comportement disponibles

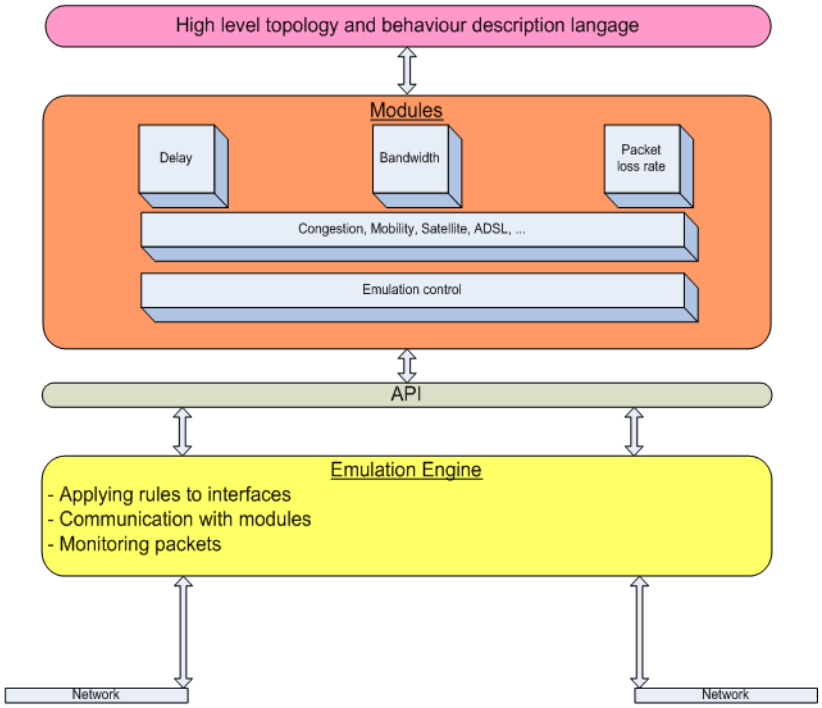


Fig. 4.4 Architecture d'émulation active

actuellement avec l'émulation comme les scénarios. La combinaison de ces différents modèles d'émulation permet d'émuler des comportements de réseaux complexes et surtout ce système est facilement déployable.

Les modules de haut niveau sont divisés en deux groupes :

- L'élément de plus haut niveau est un langage de haut niveau permettant de configurer la plate-forme d'émulation d'un haut niveau d'abstraction et de manière relativement simple. Nous avons choisi, comme c'est le cas pour l'approche à base de scénarios, d'utiliser un langage basé sur un formalisme XML qui permet de décrire la topologie du réseau ainsi que les possibles configurations des différents éléments composant cette topologie.
- Le second ensemble d'éléments correspond à des modèles d'émulation réseau pré existants mettant en oeuvre des comportements réseaux basiques tels que des délais, des pertes et bande passante, etc. De nouveaux modules peuvent être developpés en utilisant ces modules de base. Ces modules permettent d'émuler des comportements simples ou alors un peu plus complexes prenant en compte l'interaction avec le trafic traversant l'émulateur. Un exemple de modules serait ici, un lien satellite, un accès ADSL, un accès sans fil 802.11 etc. . Ces modules de haut niveau restent malgré tout indépendants de l'implémentation du système d'émulation. Le module de contrôle d'émulation fait également partie de cet ensemble d'éléments. Il gère les informations remontant des modules de bas niveau. Un exemple serait l'application d'un masque pour récupérer uniquement le trafic concerné par un module.

Les modèles de bas niveau sont en charge de l'application des modifications des paramètres de QdS ainsi que du monitoring des informations pour les modules de haut niveau. Ils agissent aussi bien sur les paquets que sur le processeur d'émulation.

Le module de récupération des informations

Ce qui fait l'originalité de cette approche, c'est qu'elle permet de réagir en temps réel aux événements pouvant survenir sur le réseau. Pour y parvenir il faut disposer d'un moyen de monitorer en temps réel les paquets circulant sur le réseau. Ainsi nous avons ajouté à l'architecture d'émulation des observateurs capables de réaliser ces actions. La granularité de ce module a été placée au niveau paquet car les événements pouvant survenir sur le réseau peuvent tout aussi bien être une augmentation de débit qu'une apparition d'un paquet contenant des informations particulières comme un paquet SYN/ACK dans le cas du protocole TCP, par exemple.

Les observateurs sont placés au niveau des interfaces du système d'émulation et ils analysent le trafic qui y transite au cours d'une expérience d'émulation. Ils sont capables de détecter en temps réel les événements qui surviennent au niveau des interfaces de l'outil d'émulation. Ces événements peuvent être une arrivée de paquets particuliers, une rafale de paquets, un certain débit etc. . Puis ils remontent ces informations au module actif de décision. Ainsi ces observateurs permettent d'obtenir à tout instant des informations sur les paquets circulant au niveau de l'interface du système d'émulation sur laquelle ils sont positionnés.

Au niveau implémentation de ces observateurs, nous avons utilisé les librairies Libpcap [38] car elles permettent d'accéder au niveau paquet et également parce qu'elles ont l'avantage d'être en libre accès. Ainsi nous avons réalisé *un sniffeur* de paquets capable de remonter les informations sur le trafic aux modules de haut niveau.

Le module actif de décision

Ce module, comme son nom l'indique, a pour mission de prendre des décisions en fonction des informations qui lui seront transmises du module de récupération d'informations sur les paquets. Il reçoit en temps réel les informations sur l'état du trafic, et des paquets qui circulent sur le réseau. En fonction de ces données, ce module détermine les valeurs des paramètres de QdS à appliquer. Les modèles d'émulation active sont capables de réagir en fonction d'événements survenant sur le réseau d'expérimentation provoqués par le protocole ou l'application à tester ou alors provenant de sources extérieures. Ces événements peuvent être dépendants du temps, d'un scénario pour le paquet, issus de modèles basés sur les données contenues dans les paquets, dépendants de signaux extérieurs tels qu'un signal de positionnement GPS, ou alors de machines à états. Par exemple, un événement pourrait être la perte d'un paquet particulier associé à un flux en fonction de la réception d'autres paquets ou alors en fonction d'une machine à états associée au flux de paquets.

4.2.3 Modèles de comportement actifs

Les modèles actifs permettent de contrôler le système d'émulation et plus précisément le moteur d'émulation, en utilisant différents formalismes comme la modélisation UML avec des machines à états, les chaînes de Markov, les réseaux de Petri. Ces modèles permettent de caractériser les modifications induites par les protocoles ou mécanismes de niveau inférieur à la couche IP. En d'autres termes, ils pourront par exemple reproduire des pertes dues à des congestions, ou alors dues à un médium, etc.

Nous allons considérer deux types de modèles : les modèles probabilistiques avec les chaines de Markov et les modèles à base d'événements avec les automates à état fini.

Modèles basés sur des chaînes de Markov

Le formalisme des chaînes de Markov permet de modéliser un comportement par un automate à états finis avec des transitions probabilisées. Il est adapté dans le cadre de comportement temporisé du réseau où le nombre d'états est fini et où la probabilité d'être dans un état ne dépend que de l'état précédent. L'exemple de la figure suivante 4.5 présente une modélisation simple d'un canal introduisant des pertes probabilistes.

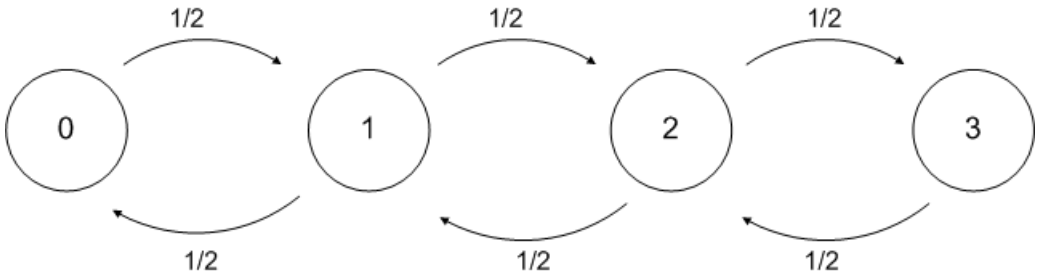


Fig. 4.5 Exemple de chaîne de Markov modélisant des pertes probabilistes

L'automate à états finis avec transitions probabilisées de la figure 4.5 permet d'obtenir les probabilités d'avoir n pertes consécutives. A l'état 0, on n'observe aucune perte, à l'état 1, une perte, etc. Pour avoir 3 pertes consécutives la probabilité sera $p(3) = (1/2)^3$.

Ainsi, les chaines de Markov permettent d'introduire une dynamicité au niveau du comportement mais elles ne proposent que des changements d'états basés uniquement sur des probabilités. Ceci peut s'avérer suffisant en fonction des besoins de l'expérimentateur. Cependant, si les passages d'un état à un autre ne sont pas liés à des probabilités mais liés, par exemple, à l'évaluation de conditions particulières, il faut considérer une solution. Ainsi, l'utilisation de modèles basés sur des machines à états sera plus adaptée car ici, il s'agit d'une modélisation basée sur des événements.

Modèles basés sur des machines à état ou Finite State Machine (FSM)

Les chaînes de Markov permettent de reproduire un comportement en se basant sur une approche probabilistique. Cependant si on se place dans un contexte où l'objectif est de modéliser un comportement, non plus basé sur des probabilités pour les changements d'état, il faut utiliser des machines à état ou automates à état finis. Le principe est de vérifier une condition de manière générale et d'appliquer une action en fonction du résultat de cette vérification. Les machines à état permettent une modélisation de comportement basée sur une approche événementielle. Nous verrons ici un exemple basé sur cette approche, pour modéliser le comportement de l'émulateur. L'intérêt se situe dans le fait qu'il est possible de travailler à une granularité fine (au niveau paquet) et ainsi avoir un meilleur contrôle sur l'émulation. Les machines à état permettent de modéliser le comportement du canal de communication.

En prenant un exemple avec le protocole TCP, on peut décider que l'on souhaite perdre un paquet d'acquittement sur deux lors d'une transmission. Le modèle d'émulation actif basé sur une machine à état est présenté sur la figure 4.6 suivante. Ainsi, la condition à vérifier est la réception du paquet d'acquittement. Un compteur se déclenche alors et ainsi, une fois qu'il atteint la valeur x, l'acquittement reçu est perdu.

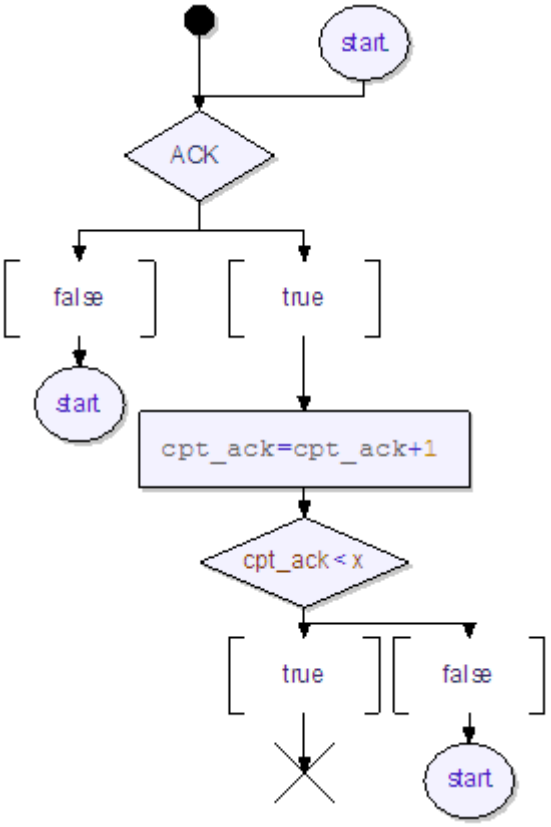


Fig. 4.6 Exemple de machine à état

Cette solution apporte un niveau de précision plus important que les approches précédentes dans le sens où il est possible de contrôler de manière plus fine le comportement du système d'émulation et plus précisément du canal de communication.

Cependant, cette méthode basée sur des machines à états demande, pour être mise en place, d'obtenir des informations sur le trafic traversant le système d'émulation. Ainsi elle doit être couplée à une solution d'émulation active.

Finalement, cette approche d'émulation de comportements permet de mettre en oeuvre de nombreux comportements réseau et ce, de manière relativement fine. Cependant, une limite de cette approche se situe au niveau de la partie cross trafic qui est difficile à mettre en oeuvre. En effet, les automates à états finis proposent une dynamique au niveau des comportements mais si on considère des effets extérieurs comme le trafic concurrent, cette approche introduit une complexité supplémentaire au niveau de la modélisation et des calculs à réaliser dans le sens où il faudrait coupler plusieurs machines pour y arriver.

4.2.4 Conclusion

Dans cette section, une contribution pour répondre au besoin d'interactivité a été présentée. Il s'agit de l'émulation active. Le principe est de placer des observateurs au niveau des interfaces du système d'émulation qui seront en charge de détecter les différents événements pouvant se produire mais également d'informer en temps réel les modules actifs de l'état du trafic. Les modèles d'émulation actifs peuvent être basés sur plusieurs méthodes selon la vision que l'utilisateur a du comportement qu'il souhaite reproduire : pour un comportement probabilistique, l'utilisation de chaînes de Markov ; pour une approche événementielle, les automates à état finis, les réseaux de Pétri ou autres approches hybrides. Ainsi, cette solution permet d'étendre la notion de dynamicité de l'émulation et permet, en considérant l'aspect comportement de l'émulation, d'apporter un meilleur niveau de réalisme. Par contre si on s'intéresse aux événements extérieurs, ces modèles risquent d'introduire une complexité au niveau des calculs à réaliser. Nous allons nous intéresser dans la suite de ce chapitre à une possibilité de modéliser de manière moins complexe le trafic concurrent sur le réseau.

4.3 LE TRAFIC CONCURRENT

Au niveau de la recherche de réalisme, une des problématiques intéressantes liés à celle de l'émulation correspond à la prise en compte des événements extérieurs tels que le trafic concurrent. En effet, lorsque l'on souhaite émuler un réseau donné, on définit une topologie réseau. L'émulation de cette topologie permet de mettre en oeuvre une émulation statique. Il est possible d'émuler un réseau ou une situation réseau dans un contexte idéal, c'est-à-dire que les limitations introduites au niveau du système d'émulation ne dépendent que des limites des différentes technologies considérées. Par exemple, l'émulation d'une topologie ADSL induira des limitations

au niveau des paramètres tels que le débit soit de l'ordre de 8 MBit/s, le délai de 20ms et le taux de perte nul.

Cependant, en considérant l'objectif de réalisme et de précision, la problématique liée au trafic concurrent apparaît clairement. En effet, le modèle d'émulation de base n'inclut généralement pas le trafic concurrent au trafic étudié qui est émis sur le réseau considéré. Ce trafic, en partie à l'origine de la dynamicité rencontrée sur les réseaux actuels, devrait être pris en compte au niveau du système d'émulation. Cette section va donc présenter les différentes manières de générer ou d'émuler ce trafic.

4.3.1 Trafic réel

La première possibilité et la plus intuitive pour cette mise en oeuvre est de générer réellement ce trafic. Ceci peut être réalisé en connectant des sources de trafic réel au système d'émulation, en utilisant des générateurs de trafic (MGEN[61], NetPerf [1], etc.) ou encore des applications. La distinction entre trafic interne ou externe est établie en fonction de la localisation des sources de trafic. Si elles se trouvent au sein de l'émulateur elles sont dites internes et sinon elles sont externes. La figure 4.7 présente un exemple de génération de trafic externe.

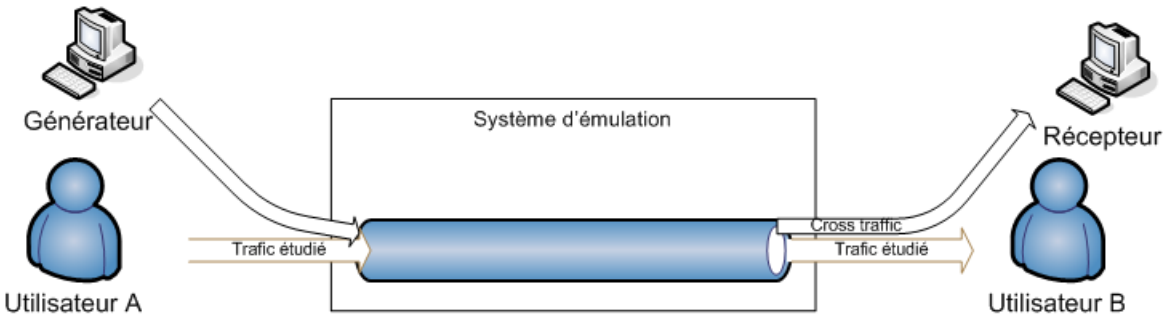


Fig. 4.7 Génération de trafic réel

L'utilisation de sources de trafic externes a l'avantage de réduire la charge (traitements à réaliser) au niveau du système d'émulation. Cependant il apparaît que la mise à l'échelle d'une telle solution s'avère difficile. En effet, si on considère une expérience d'émulation nécessitant une vingtaine de sources de trafics, on se rend compte qu'une telle solution serait très coûteuse en terme d'équipements à mettre en oeuvre. De plus, nous nous plaçons dans un objectif de simplification, il serait donc intéressant d'éviter de tout reproduire.

4.3.2 Trafic virtuel

Dans l'objectif de réduire le nombre d'équipements nécessaires à l'introduction de multiples trafics concurrents, une solution consiste à virtualiser ce trafic. La figure 4.8 montre comment cette approche est mise en place. En fait, il s'agit de générer du trafic virtuel de manière interne au sein du système d'émulation. Ce trafic entre en concurrence de la même façon que pour le cas précedent, avec le trafic réel étudié.

Pour le mettre en oeuvre, des modifications sont réalisées au niveau du noyau et des utilitaires tels que pktgen [55] peuvent être utilisés pour générer ce trafic.

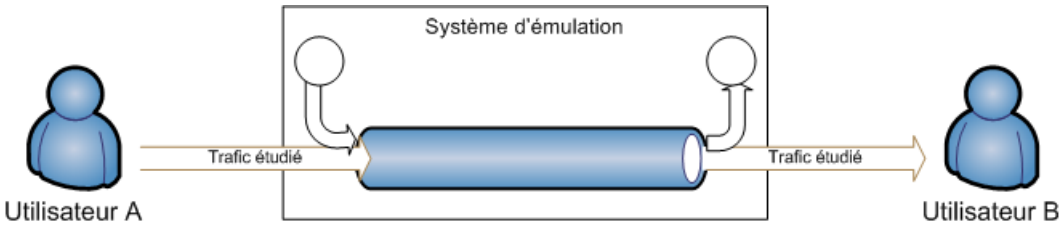


Fig. 4.8 Génération de trafic virtuel

Avec cette approche, il est possible de prendre en compte un nombre potentiellement grand de sources de trafic. Cependant, l'introduction de sources de trafic internes entraîne une augmentation des calculs à réaliser par le système d'émulation. Ceci peut induire des dégradations au niveau du résultat de l'émulation et donc des pertes au niveau du réalisme et de la précision.

4.3.3 Trafic généré à base de traces

La modélisation basée sur des traces récoltées sur un réseau opérationnel donné permet de prendre en compte le trafic concurrent, mais ici encore, il ne s'agit que du trafic enregistré pendant l'intervalle de mesure. Même si il ne s'agit que du trafic concurrent propre à une expérience, il a l'avantage d'être très précis puisqu'il s'agit du trafic circulant réellement sur le réseau. Cependant, la limite de ce type d'approche réside dans le fait qu'il n'y a aucune interaction possible avec le trafic étudié. En d'autres termes, cette solution propose la prise en compte du trafic extérieur de manière très précise mais elle se restreint aux variations de trafic intervenues durant la période de collecte des traces. Ainsi, pour un utilisateur souhaitant émuler une condition réseau très précise à un instant donné, cette solution est la meilleure. Par contre, il n'a aucun contrôle sur ce trafic extérieur.

4.3.4 Discussion

Avec les modèles présentés précédemment, il est possible de prendre en compte le trafic concurrent de diverses manières : soit en le générant réellement, soit en le générant de façon virtuelle ou alors en le rejouant en fonction de traces de trafic.

Supposons que l'on veuille émuler une congestion réseau. De quels moyens ou méthodes dispose t-on pour la mettre en oeuvre ? Une solution consiste à injecter un flux réel à notre émulateur. Ainsi ce flux rentrerait en concurrence avec le flux de l'application ou du protocole à tester. Cette solution assure un bon niveau de réalisme dans le sens où le flux injecté est un flux réel et qu'il produira une congestion réelle. Cependant, on se trouve confronté au problème de ressources nécessaires à la mise en place d'une telle solution dès lors que le nombre de flux à injecter devient important. En effet, plus le nombre de flux sera important plus il faudra de ressources

pour les mettre en place. De plus, ceci implique un nombre plus important de paquets à traiter par le système d'émulation.

Or, il n'est pas possible d'augmenter les ressources indéfiniment, donc une autre solution consiste à virtualiser le flux à injecter au niveau de l'émulateur. Ainsi le système d'émulation serait capable d'appliquer au flux considéré par l'expérimentation, des modifications correspondant à l'apparition de la congestion. Le problème de ressources en termes de machine à déployer est résolu, mais, le problème de ressources en termes de calculs n'est pas tout à fait réglé. De plus, il faut cependant disposer d'un modèle de congestion. Le problème qui se pose ici est de savoir comment obtenir ce modèle ?

4.3.5 Les modèles basés sur des équations de comportement

Nous avons vu que les modèles d'émulation existants se limitent à deux objectifs : soit favoriser la précision au détriment de la complexité de mise en oeuvre soit l'inverse. Or, tout l'intérêt de l'émulation consiste à trouver le meilleur compromis entre ces deux objectifs. Ainsi, nous avons choisi de nous intéresser à une solution basée sur la métrologie consistant à étudier les traces de trafic pouvant être obtenues soit par des mesures réelles sur un réseau réel ou alors obtenues par des simulations pour définir une équation du comportement du trafic à reproduire par le système d'émulation.

L'idée ici est la suivante : il s'agira de montrer que le concept d'émulation active peut être couplé à un mécanisme de prise de décision basé sur des modèles de comportements déduits d'équations de trafic. Ainsi ce type de modèle permettra une interactivité avec l'utilisateur mais également avec le trafic circulant à travers l'émulateur. En effet, le système d'émulation sera capable d'appliquer un comportement (évolution des paramètres du flux en termes de débit, délai et taux de pertes) en fonction d'une action de l'utilisateur. Cette évolution sera conduite par un modèle de comportement du trafic. Mais il se peut également que celui-ci doive réagir en conséquence d'un état dans lequel se trouverait le réseau.

L'exemple suivant, basé sur une congestion réseau, servira d'illustration. L'objectif est de proposer une méthode permettant d'émuler un comportement basé sur des équations. Nous avons retenu la solution d'utiliser un outil de simulation à événement discret pour simuler les comportements à reproduire. Ici nous allons considérer des congestions, et en déduire une équation et l'implémenter au niveau du système d'émulation. Pour y parvenir nous allons nous baser sur un exemple utilisant un flux TCP de type Reno. L'objectif est de pouvoir construire un modèle d'émulation représentant le comportement moyen d'un flux TCP traversant un canal de communication offrant un service de type Best Effort. Il s'agira ensuite de déterminer, grâce à une approche basée à la fois sur la simulation et le test sur une plate-forme expérimentale, l'évolution des paramètres que sont le délai de transmission, le débit et le taux de perte pour le flux TCP considéré.

Ainsi, nous avons choisi tout d'abord d'étudier l'impact sur un flux TCP d'un autre flux TCP puis de plusieurs flux TCP et enfin la mise en concurrence de flux

TCP et UDP. La taille des paquets transmis considérée est de 1472 octets (seuil théorique de fragmentation). Nous allons uniquement nous intéresser au débit et donc le faire varier. Dans un premier temps nous allons considérer un débit de 100Kb/s. Un flux TCP y est injecté et on obtient le résultat suivant :

Nous allons ensuite considérer plusieurs cas :

- 1er cas : 1 flux TCP en concurrence avec un flux non réactif de type UDP. Ceci permettra d'obtenir une idée sur le débit effectivement préservé par TCP par rapport au flux UDP. On fera varier le débit UDP.
- 2ème cas : 1 flux TCP en concurrence avec un flux réactif de type TCP. Il sera possible d'établir une comparaison entre les cas réactifs et non réactifs.
- 3ème cas : 1 flux TCP en concurrence avec un flux réactif et un flux non réactif. Ainsi des premiers modèles de l'évolution de la bande passante et du délai d'un flux TCP en concurrence avec d'autres flux.
- 4ème cas : 1 flux TCP en concurrence avec plusieurs flux UDP (2 puis 3).

Il sera possible de définir une équation du comportement du flux TCP en fonction de plusieurs cas donnés et de pouvoir implémenter cette équation au niveau du système d'émulation afin qu'il reproduise l'évolution des paramètres. La figure suivante 4.9, par exemple, présente le résultat obtenu par simulation de 2 flux TCP concurrents. Ensuite l'objectif est de pouvoir déterminer la pente de la courbe du flux TCP considéré et de pouvoir déterminer une allure générale.

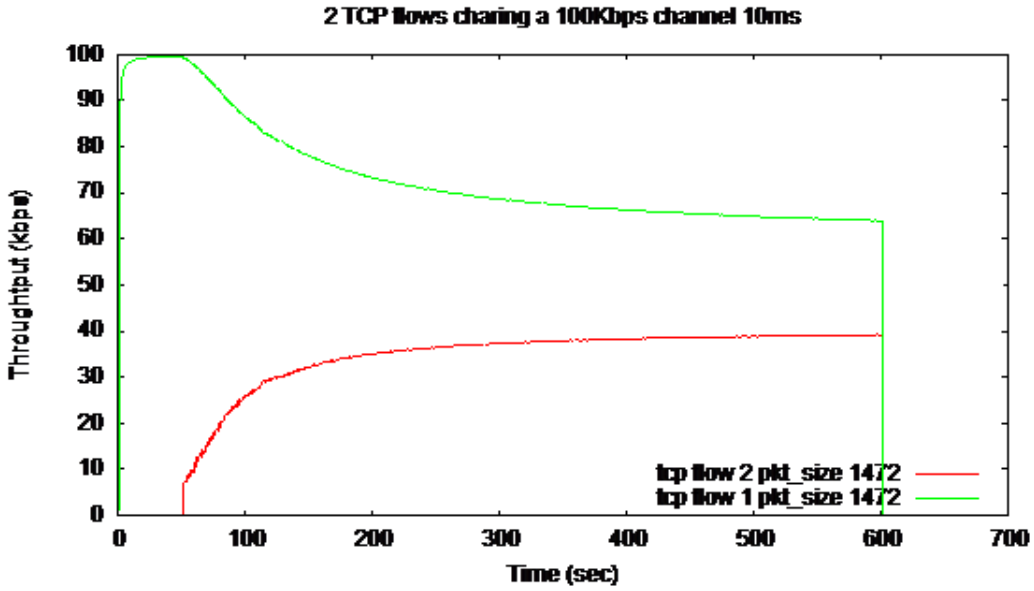


Fig. 4.9 Deux flux TCP se partageant un canal

Ainsi il est possible de déterminer l'équation du flux 1 (vert).

Le modèle mathématique est du type saturation / dé-saturation comme présenté sur la figure suivante 4.10 :

L'équation de cette courbe est de la forme :

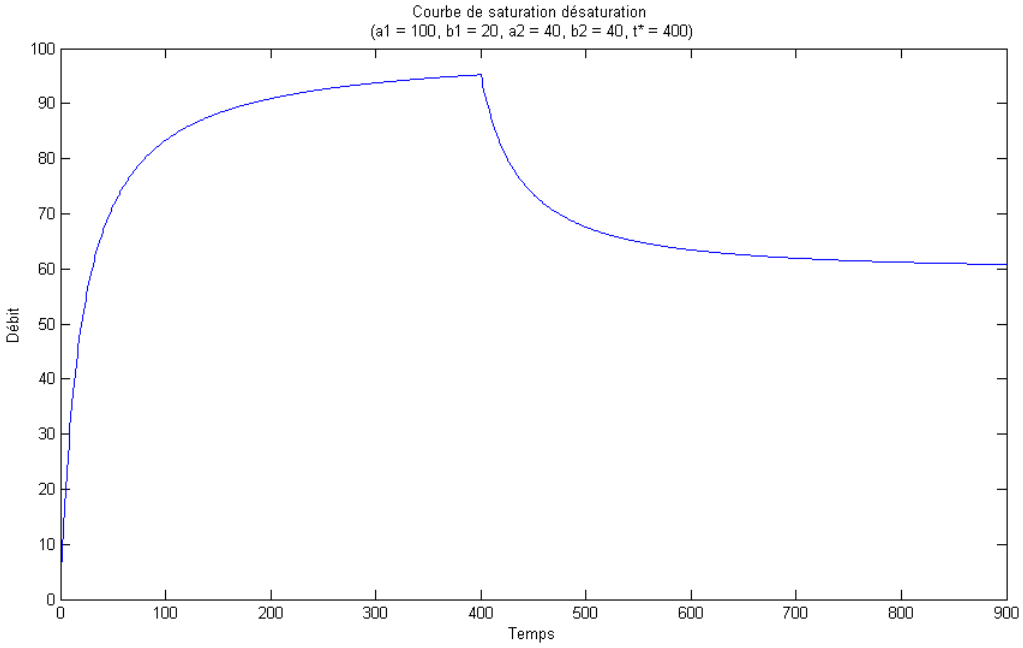


FIG. 4.10 Exemple de modèle de saturation / dé-saturation

$$z(t) = y_1(t) - y_2(t)$$

avec

$$y_1 = \frac{a_1.t}{b_1 + t} \quad y_2 = \frac{a_2.(t - t^*)}{b_2 + (t - t^*)}$$

Rappelons qu'il s'agit d'une somme et différence de modèles de Michaelis-Menten de la forme :

$$y = \frac{a.t}{b + t}$$

ou alors de la forme :

$$y = \frac{a.(t - t_0)}{b + (t + t_0)}$$

dans le cas où il y a un délai pour l'application de t_0.

Si on considère le cas général où

$$y = \frac{a.t}{b + t}$$

comme présenté sur la figure 4.11.

Si on souhaite calculer le t_{50} qui attribue la valeur $\frac{a}{2}$ à y :

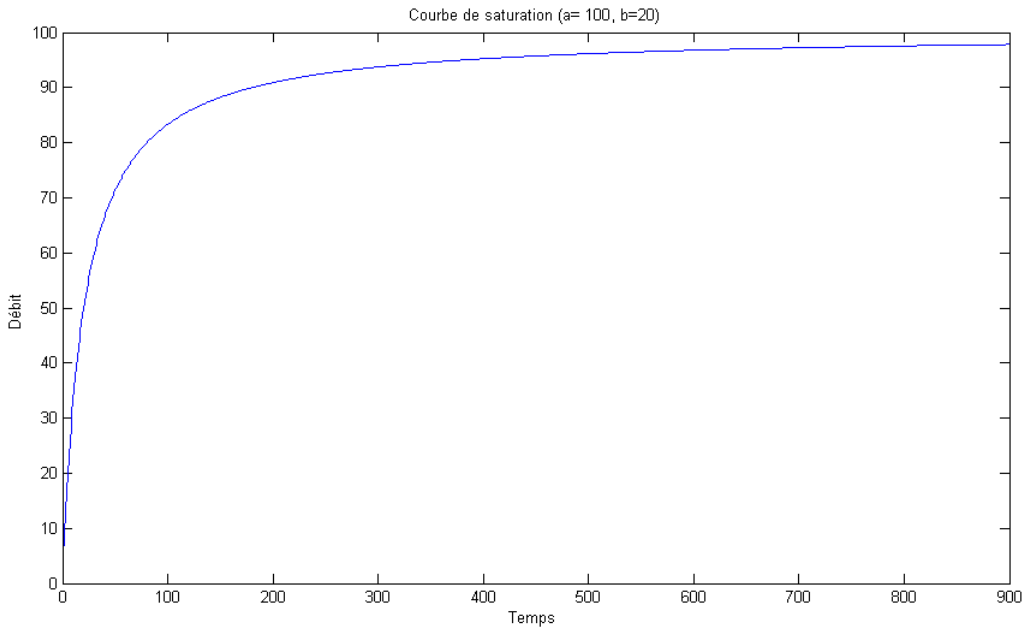


Fig. 4.11 Courbe de saturation

$\frac{a}{2} = \frac{a.t_{50}}{b+t_{50}}$ donc, $a.b + a.t_{50} = 2a.t_{50}$ ce qui implique que $a.b = a.t_{50}$ ainsi, $b = t_{50}$. Ainsi plus b est petit, plus la courbe converge rapidement vers la saturation a.

La phase suivante consiste à réaliser une estimation des paramètres inconnus (a_i), (b_i). Pour y parvenir nous allons utiliser une régression non-linéaire grâce à un logiciel de calcul mathématique. On obtient donc les valeurs des paramètres.

Il en est de même pour les autres cas. Nous avons utilisé la même méthode de calcul pour les délais. Ainsi nous avons défini une équation de comportement pour le paramètre de débit correspondant à l'ajout d'un flux.

4.4 CONCLUSION

Dans ce chapitre nous avons vu qu'il existe diverses façons de reproduire un comportement réseau en fonction des besoins des utilisateurs. Tout d'abord, la modélisation du comportement dépendant de la technologie ainsi que des protocoles utilisés. Pour y parvenir, différents types de modèles ont été présentés ainsi que leurs avantages et leurs limites. Il en est ressorti un besoin d'interactivité avec le trafic circulant à travers le système d'émulation dès lors qu'il s'agissait de modéliser des comportement un peu plus complexes comme par exemple un accès à une ressource partagée.

Pour allier un bon niveau de réalisme et de précision ainsi qu'une complexité maitrisée nous avons proposé une nouvelle approche de modélisation de comportements de réseau : l'émulation active. Elle permet de répondre au besoin d'interactivité

et également d'augmenter le réalisme de l'émulation en augmentant la précision de l'émulation rendue du fait qu'il faille travailler au niveau des paquets. L'émulation active peut être mise en oeuvre avec des modèles d'émulation basés, selon l'approche utilisée (probabilistique, événementielle), sur des formalismes tels que des chaînes de Markov, des machines à états, des réseaux de Pétri, etc. .

Pour augmenter la précision des modèles et donc d'une certaine façon du réalisme, nous nous sommes intéressés à la prise en compte des événements extérieurs tels que le trafic concurrent pouvant circuler sur le réseau considéré. Ainsi des modèles basés sur des équations de comportement ont été proposés pour améliorer l'interactivité avec l'utilisateur mais également avec le trafic circulant à travers l'émulateur.

Le chapitre suivant présentera l'illustration du concept d'émulation présenté dans ce chapitre grâce à un cas d'étude basé sur un projet européen.

Chapitre 5

Cas d'étude : émulation satellite dans le projet EuQoS

Dans ce chapitre, j'aborderai les points suivants :

- ⋆ présentation de la plate-forme d'expérimentation ;
- ⋆ présentation du projet EuQoS ;
- ⋆ présentation du contexte satellite ;
- ⋆ intégration de l'accès satellite dans le projet EuQoS.

Dans le chapitre précédent nous avons présenté les différentes solutions possibles pour la reproduction de comportements. Le cas d'étude présenté dans ce chapitre s'inscrit dans le cadre du projet Européen EuQoS. Il nous permettra de mettre en oeuvre des concepts que nous avons présentés précédemment à l'aide de deux applications. La première, introductive, concerne l'émulation du système EuQoS (dont une présentation de l'architecture sera faite) pour l'évaluation du protocole de transport ETP, et la seconde s'intéresse plus précisément à la modélisation d'une technologie associée à un réseau d'accès : un accès satellite.

5.1 PRÉSENTATION DU PROJET EUQOS

Le projet EuQoS (European Quality of ServiceEnd-to-end Quality of Service support over heterogeneous networks) est un projet européen composé de 24 partenaires (laboratoires de recherche et entreprises) ; l'ENSICA et le LAAS-CNRS participent à ce projet.

5.1.1 Objectifs

L'objectif du projet EuQoS est la conception et la réalisation d'un réseau européen hétérogène (composé de réseaux d'accès basés sur des technologies différentes : xDSL, LAN, WiFi, Satellite, etc.) offrant des garanties de qualité de service de bout en bout. On entend par garantie de qualité de service, des garanties de bande passante et temporelles en comparaison avec le service « Best Effort », proposé majoritairement dans les réseaux IP, qui n'offre aucune garantie.

Le système EuQoS est basé sur une décomposition du paradigme de QoS de bout en bout en considérant, d'une part l'axe vertical (plans de service, de contrôle et transport) et d'autre part l'axe horizontal (différentes technologies réseau, en particulier réseaux de coeur et réseaux d'accès). En considérant le plan de service, il permet à l'appelant de contacter l'appelé (c'est-à-dire d'obtenir son adresse IP) et également de s'entendre sur les codecs qui seront utilisés. De plus, ce plan propose un service de QoS à la demande pour un utilisateur (capable de gérer l'authentification, l'autorisation, et la collecte d'informations d'un utilisateur). Ce niveau est donc en charge de l'autorisation, de l'authentification et de l'enregistrement des utilisateurs mais également du filtrage des requêtes de QoS - en accord avec le profil de l'utilisateur c'est-à-dire des caractéristiques spécifiées lors de l'enregistrement.

Le plan de contrôle permet le déploiement de l'architecture EuQoS en facilitant l'adoption par les différents domaines de leurs solutions spécifiques. Ceci est réalisé grâce à la spécification du niveau Network Techology Independent (en charge de gérer le domaine au niveau IP) et au niveau Network Technology Dependent (qui est en charge de l'exécution des algorithmes spécifiques à chaque technologie réseau considérée). L'interface entre les deux permet à n'importe quel fournisseur d'accès d'implémenter sa propre solution de QoS juste en implémentant son propre Ressource Allocator (RA), spécifique à sa technologie.

5.1.2 Architecture EuQoS de bout en bout

L'architecture de EuQoS est une architecture de bout en bout avec deux visions : tout d'abord une vue déploiement réseau sur un certain nombre de systèmes autonomes (AS) et une vision application sans considérer les AS. La vision déploiement réseau est présentée sur la figure suivante 5.1.

Cette architecture est composée de réseaux d'accès et de réseaux de coeur. Ces différents réseaux d'accès sont basés sur des technologies diverses (wifi, satellite, xDSL etc.).

Comme le montre la figure 5.1 l'architecture EuQoS est basée sur l'approche standard « diviser et conquérir » qui consiste à réduire la taille d'un problème considéré en le découpant en plus petites parties. Ainsi pour atteindre cet objectif, une des principales forces du système EuQoS réside dans le fait que les interactions entre d'une part les clients et les serveurs, et d'autre part entre les différents plans de l'architecture EuQoS sur les différents AS, sont clairement spécifiés.

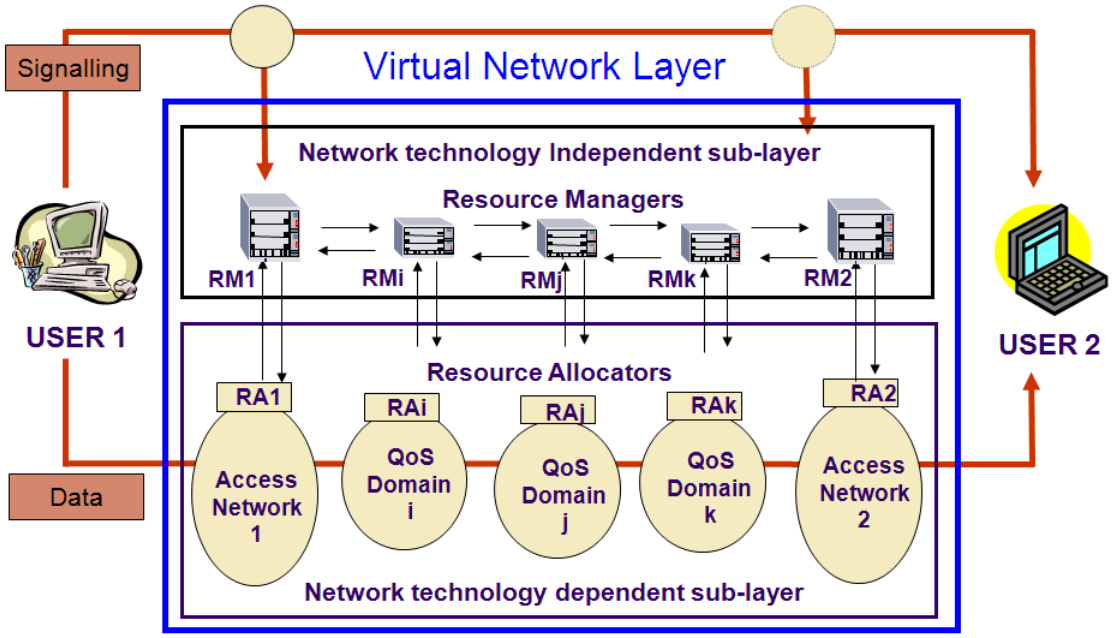


Fig. 5.1 Architecture EuQoS de bout en bout

5.1.3 La Qualité de Service de bout en bout dans EuQoS

Les différentes classes de service EuQoS

L'architecture EuQoS offre différentes classes de service permettant de satisfaire les besoins divers des utilisateurs et des applications.

Les différentes Classes de Service (CdS) fournies par l'architecture EuQoS sont les suivantes :

- Service RT : il s'agit d'une classe de service qui nécessite un délai, une gigue et taux de perte faibles pour la délivrance des paquets ; elle nécessite également d'être servie à un débit pré-configuré.
- Service NRT : cette classe offre une grande probabilité de délivrance des paquets « dans le profil » et la possibilité de transmettre plus de paquets si les ressources sont disponibles.
- Service Standard : cette classe n'offre aucune garantie de délivrance des paquets.

5.1.4 Test du protocole Enhanced Transport Protocol (ETP)

Dans le cadre du projet EuQoS, le protocole de transport Enhanced Transport Protocol (ETP) développé par E. Exposito lors de sa thèse [21] est utilisé de manière à introduire la notion de qualité de service dans la couche transport. L'objectif

est de disposer d'un support d'expérimentation pour valider le fonctionnement et l'intégration de ce protocole dans le cadre du projet. Il nous servira à introduire le cas d'étude EuQoS.

Présentation du protocole ETP

ETP est un framework permettant d'utiliser et de composer des mécanismes entre eux dans le but de répondre à des besoins applicatifs. ETP est en fait un protocole de la couche Transport orienté Qualité de Service (QoS) en mode connexion et orienté messages. La figure 5.2 présente une vision globale des différents mécanismes du protocole ETP.

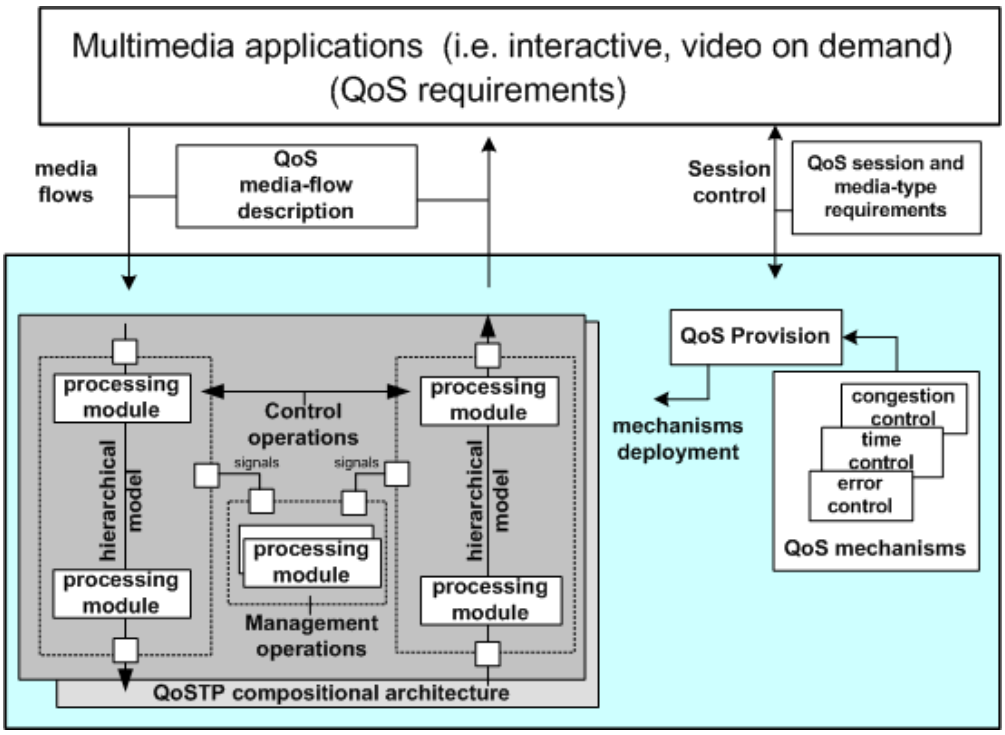


Fig. 5.2 Vision globale des mécanismes du protocole ETP

ETP offre des mécanismes prenant en compte l'ordre partiel, la fiabilité, le contrôle de congestion et des services de communication de bout en bout pouvant être configurés de manière statique ou dynamique au cours du temps, et ce, en fonction des besoins applicatifs. Les services ETP sont implémentés en composant des mécanismes configurables afin d'assurer un contrôle sur la QoS.

Définition du besoin d'émulation

Le besoin, dans ce contexte, se traduit par la nécessité de disposer d'un support permettant de mettre en oeuvre l'expérimentation du protocole ETP. Or, pour avoir des conditions reproductibles et facilement contrôlables, le choix de l'émula-

tion s'imposait. De plus, avant de réaliser l'intégration effective du protocole dans l'architecture EuQoS, il était nécessaire de disposer d'un support permettant de réaliser des tests pour valider cette intégration. L'émulation permet de reproduire le comportement global des différents réseaux d'accès et d'offrir un service de bout en bout équivalent à celui que l'on doit obtenir sur le réseau EuQoS réel.

La mise en place de l'émulation du réseau EuQoS doit donc offrir différents niveaux de QdS qui correspondent aux configurations possibles du réseau EuQoS : la QdS offerte par les divers réseaux d'accès et de coeur.

5.1.5 La plate-forme d'expérimentation générique

Pour mener nos expérimentations, nous avons utilisé la plate-forme NINE. Son architecture est présentée dans la section suivante.

Architecture générique

La figure 5.3 présente l'architecture générique la plate-forme d'émulation NINE (Nine Is a Network Emulator). Elle est composée d'un routeur émulateur basé sur Dummynet ainsi que d'éléments terminaux sur lesquels sont déployés les protocoles ou applications à expérimenter et d'un gestionnaire d'émulation en charge de la configuration de la plate-forme, incluant les terminaux et le conditionneur de trafic.

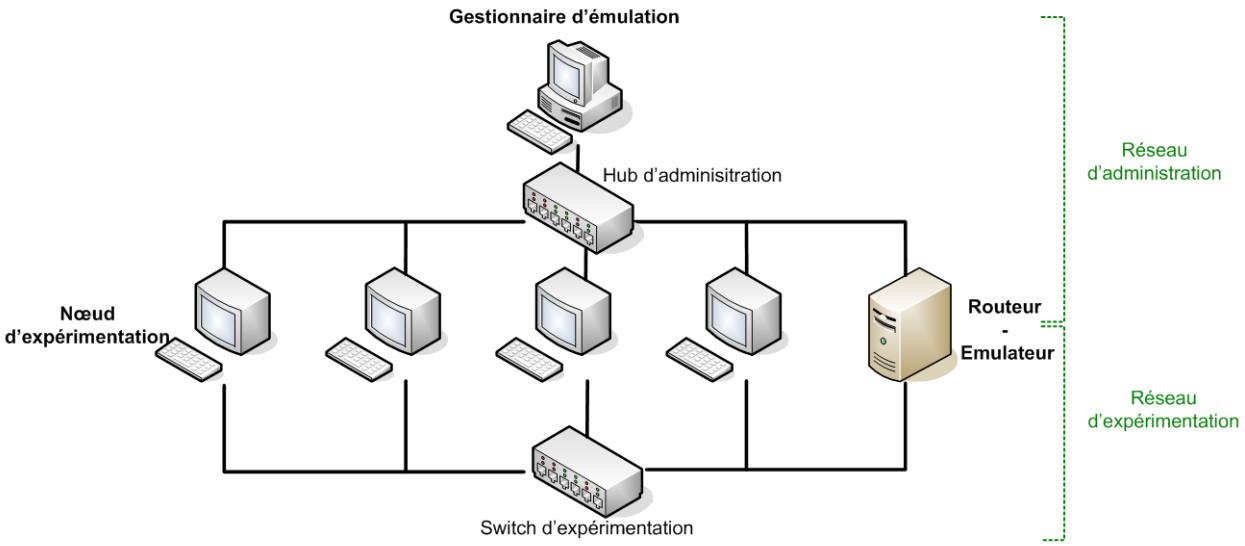


Fig. 5.3 Architecture de la plate-forme d'émulation NINE

Cette plate-forme se compose de deux réseaux : un réservé à l'administration et l'autre pour les expérimentations. Le fait d'utiliser deux réseaux permet de s'assurer que le trafic d'expérimentation n'entrera pas en concurrence avec le trafic d'administration (trafic permettant le paramètrage ainsi que les mises à jour des équipements de la plate-forme).

Nous avons choisi d'utiliser Dummynet comme conditionneur de trafic du fait de ses caractéristiques et de sa relative simplicité d'utilisation, comme cela a été présenté dans le chapitre 2 (2.3.4) : il fonctionne en effet au niveau IP et permet d'utiliser des canaux de communication, appelés *pipes,* pour contraindre le trafic dans le but de reproduire des conditions réseau prédéfinies en termes de délais, bande passante et taux de pertes. Il est également possible de gérer des files d'attente associées à ces pipes. Dummynet offre l'avantage de pouvoir enchaîner des pipes pour faire subir à un paquet plusieurs contraintes successives. Ceci est utile pour émuler à haut niveau une topologie réseau composée de plusieurs technologies différentes (par exemple, dans [17], modélisation d'une topologie composée d'un accès satellite et d'un réseau ADSL).

Architecture de la plate-forme utilisée

L'architecture proposée sur la figure 5.4 permet de mettre en oeuvre l'émulation du réseau EuQoS. L'architecture utilise la plate-forme d'émulation NINE, et deux machines terminales sont utilisées. Au niveau du routeur / émulateur (machine sat), le comportement d'un réseau équivalent à EuQoS est reproduit. La machine *sat-ctrl* fait office de gestionnaire d'émulation.

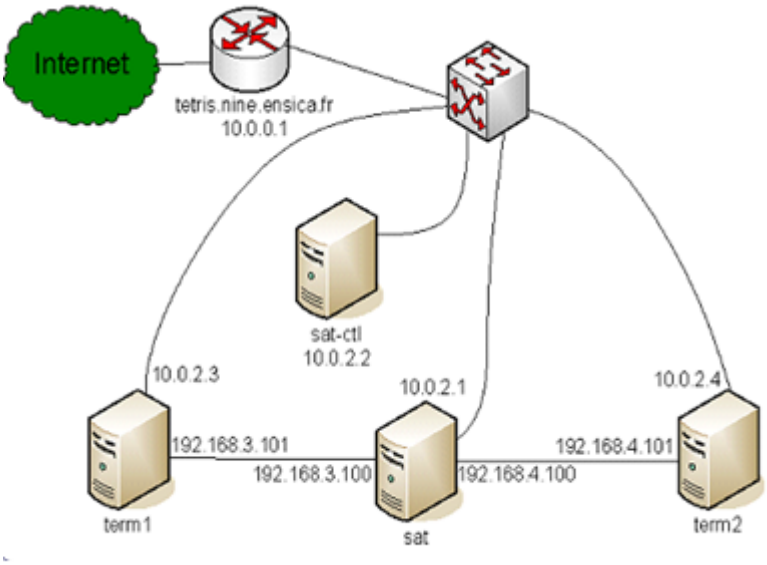


Fig. 5.4 Architecture de la plate-forme d'émulation

Implémentation

La figure suivante 5.5 représente une vue schématique de l'abstraction que nous avons réalisée. L'objectif est de proposer une émulation de bout en bout basée sur un canal de communication. Ce canal reproduit en temps réel l'évolution des paramètres de QdS.

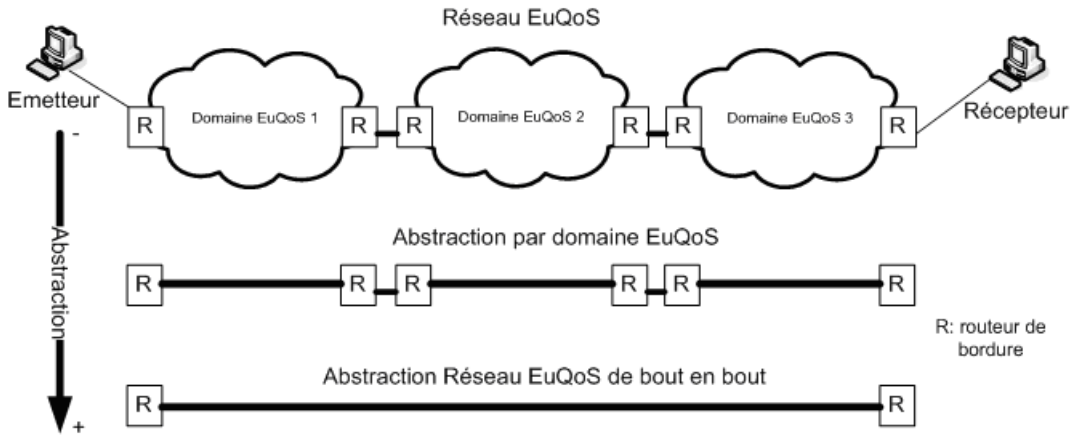


Fig. 5.5 Abstraction réseau EuQoS

Dans cette étude de cas, nous utilisons l'émulation pour reproduire le comportement général du réseau EuQoS. Les différents mécanismes à reproduire étant assez simples, nous avons choisi d'utiliser une approche d'émulation dynamique conduite par des scénarios d'émulation. Nous allons ré-utiliser l'ordonnanceur défini dans [15], et définir des fichiers XML de description de haut niveau que nous traiterons afin de paramétrer la plate-forme et également afin de générer un scénario d'émulation.

Description générale des modèles XML utilisés

La première phase consite à définir la topologie du réseau sur lequel les expérimentations vont être menées. Pour cela, deux descriptions doivent être réalisées : une caractérisation des différents domaines existants puis une description de la topologie du réseau à émuler. La figure 5.6 suivante présente un extrait de description des différents domaines.

Ce fichier correspond donc à une description de haut niveau des caractéristiques des différents domaines existants. Pour chaque domaine considéré, trois classes de services sont utilisées (RT, NRT et BE). Pour ces classes, les paramètres de QoS sont définis en termes de délais, taux de pertes, et de bande passante maximale. Avec le délai de propagation du médium par unité de distance et la distance de traversée du domaine, le gestionnaire d'émulation calcule le délai moyen de propagation.

Ensuite il s'agit pour l'utilisateur de définir la topologie du réseau à émuler. Cette topologie est donc composée des différents domaines que devra emprunter le trafic. La figure suivante 5.7 propose un extrait de description des chemins de QoS.

Dans ce fichier, l'utilisateur doit paramétrer :

- le type de SLA souhaité (contrat) ;
- la bande passante désirée dans l'unité spécifiée ;
- le type d'emulation (par exemple onePipe correspondant à un canal de bout en bout qui représente en fait l'agrégation des différents chemins définis.) ;

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Parametrage des differents domaines de QoS a traverser-->
<domain>
        <network type = "geoSatellite">
                <RT>
                        <propagDelayDefault unit="ms">300</propagDelayDefault>
                        <propagDelay unit="ms">
                                <distance unit="km">300</distance>
                                <value>1</value>
                        </propagDelay>
                        <plr>0.01</plr>
                        <bpMax unit="bps">2048000</bpMax>
                </RT>
                <NRT>
                        <propagDelayDefault unit="ms">450</propagDelayDefault>
                        <propagDelay unit="ms">
                                <distance unit="km">200</distance>
                                <value>1</value>
                        </propagDelay>
                        <plr>0.05</plr>
                        <bpMax unit="bps">2048000</bpMax>
                </NRT>
                <BE>
                        <propagDelayDefault unit="ms">1200</propagDelayDefault>
                        <propagDelay unit="ms">
                                <distance unit="km">75</distance>
                                <value>1</value>
                        </propagDelay>
                        <plr>0.2</plr>
                        <bpMax unit="bps">2048000</bpMax>
                </BE>
                </network>
...

</domain>
```

Fig. 5.6 Exemple de description du domaine

- la topologie (l'utilisateur paramètre les différents domaines qu'il souhaite traverser) ;
- le scénario (durée de l'expérience et charge du réseau).

A l'aide de ces deux fichiers de description et d'un troisième fichier correspondant à une description logique de la plate-forme (adresse IP et nom des machines terminales participant à l'expérience et de la machine d'émulation), un fichier correspondant à la description de l'expérience est créé. Pour y parvenir, nous avons développé un programme Java qui fait office de simulateur simplifié et qui se charge de la génération du scénario d'émulation correspondant. Un exemple de description de l'expérience correspondant à un scénario d'émulation est proposé sur la figure 5.8 suivante.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- End to end Path file example -->
<e2ePath>
        <requiredSLA>
                <CoS>RT</CoS>
                <bwe2e unit="bps">100000</bwe2e>
        </requiredSLA>
        <emulationType>onelink</emulationType>
        <topology>
                <direction>
                        <origin>term1</origin>
                        <endpoint>term2</endpoint>
                </direction>
                <network>
                        <id>sat1</id>
                        <type>geoSatellite</type>
                        <link>return</link>
                        <position>1</position>
                </network>
                <network>
                        <id>sprint</id>
                        <type>sprintCoreNetwork</type>
                        <position>2</position>
                </network>
                <network>
                        <id>wifi1</id>
                        <type>wifi</type>
                        <link>forward</link>
                        <position>3</position>
                </network>
        </topology>
        <scenario>
                <duration unit="ms">100000</duration>
                <load>
                        <time>0</time>
                        <rate>0.05</rate>
                </load>
                <load>
                        <time>10</time>
                        <rate>0.10</rate>
                </load>
        </scenario>
</e2ePath>
```

Fig. 5.7 Extrait de description d'un chemin de QoS

La première partie de ce fichier est utile au gestionnaire ou contrôleur d'émulation. Elle lui permet de paramètrer les équipements de la plate-forme avec les adresses correspondantes. La seconde partie de ce fichier entre les balises *scenario* correspond au scénario d'émulation. Il s'agit d'un scénario temporel décrivant l'évolution des conditions de QoS au cours du temps. Ainsi aux dates correspondantes, les mises à

```
<?xml version="1.0" encoding="UTF-8"?>
<emulation_experiment>
        <emulated_platform>
                <emulated_host id="term1">
                        <ip_address>192.168.141.1</ip_address>
                        <ip_mask>255.255.255.0</ip_mask>
                        <gateway>192.168.141.100</gateway>
                        <member_of>Emulated EuQoS Network</member_of>
                        <mapped_on>term1</mapped_on>
                </emulated_host>
                <emulated_host id="term2">
                        <ip_address>192.168.142.1</ip_address>
                        <ip_mask>255.255.255.0</ip_mask>
                        <gateway>192.168.142.100</gateway>
                        <member_of>Emulated EuQoS Network</member_of>
                        <mapped_on>term2</mapped_on>
                </emulated_host>
        </emulated_platform>
        <connectivity>
                <link id="term1 to term2">
                        <sender>term1</sender>
                        <receiver>term2</receiver>
                </link>
        </connectivity>
        <scenario>
                <number_of_event>1</number_of_event>
                <time_step unit="ms">10</time_step>
                <date id="t0.0" start="0.0" unit="ms">
                        <link_update>
                                <on_link id="term1 to term2" />
                                <bandwidth unit="b/s">1000000</bandwidth>
                                <delay unit="ms">650</delay>
                                <plr unit="percent">0.0248005</plr>
                                <queue_size>5</queue_size>
                        </link_update>
                </date>
        </scenario>
</emulation_experiment>
```

Fig. 5.8 Exemple de scénario d'émulation

jour des liens seront effectuées. Dans l'exemple présenté, nous allons considérer une transmission entre un émetteur *term1* et un récepteur *term2.* A l'instant *t0=0.0ms* le lien entre term1 et term2 sera paramétré de telle sorte que la bande passante soit de 1Mb/s, le délai de 650ms, le taux de pertes de 0.0248005 et la taille de la queue du système d'émulation de 5 paquets.

Une fois le fichier de description de l'expérience généré, la phase suivante consiste à jouer ce scénario sur le système d'émulation. Pour ce faire, nous utilisons un ordonnanceur qui envoie en temps réel les commandes de mise à jour à la plateforme en fonction des événements du scénario.

5.1.6 Résultats d'expériences

Ces expériences ont été réalisées avec N.Van Wambeke [62]. Leur but est d'évaluer le comportement du protocole ETP dans l'architecture EuQoS. Etant donné la nature des CdS EuQoS, plusieurs solutions de composition ont été définies afin de produire le service le plus adéquat en fonction des besoins temporels des applications streamées et non streamées. Ces compositions sont présentées sur le tableau 5.1 suivant.

	Appli streamée (VoD)	Appli non streamée (FTP)
Standard	ETP[TFRC]	ETP[TFRC+EC]
Téléphonie & Appli RT interactive	ETP[RC]	ETP[RC+EC]

Tab. 5.1 Compositions ETP pour les différentes CdS EuQoS

avec EC : Error Control –TFRC : Congestion Control
Plusieurs cas vont être considérés en fonction des classes de service RT et Standard. Les mesures seront réalisées au niveau applicatif pour les protocoles UDP et TCP ainsi que pour ETP en considérant les deux services (Standard et RT Interactive / Telephony). Pour les scénarios utilisant une application non streamée, nous utilisons une application de type FTP. Cette application envoie des données à la couche transport aussi vite qu'elle le peut. Pour chaque scénario nous présenterons le débit applicatif en réception et les pertes au niveau applicatif. Sur les figures de débit, l'abscisse représente le temps en secondes et l'ordonnée le débit en Kbit/s ; pour celles représentant les pertes, l'abscisse représente le temps en secondes et l'ordonnée les pertes par unité de temps (secondes). Dans le cas d'application streamée nous utiliserons une application de type vidéo à la demande.

Cas d'application non streamée dans le cas d'une réservation dans la classe RT

Dans ce scénario, un débit crête de 400Kbit/s est réservé pour la classe RT.

1. Avec UDP

   Avec le protocole UDP, on constate que la limitation de bande passante (400Kbit/s) est effective sur les figures 5.9 et 5.10. Tout le trafic au-delà de cette limite est perdu, d'où les pertes visibles sur la figure 5.10 car le protocole UDP ne dispose pas de mécanisme capable de récupérer les erreurs. Etant donné que l'on utilise une application non streamée, le trafic n'est pas conditionné au niveau de l'application ce qui provoque les pertes dues à UDP.

2. Avec TCP
   En utilisant TCP il n'y a pas de perte comme le montre la figure 5.12. Ceci est normal étant donné que TCP est un protocole offrant des garanties de fiabilité. On constate également que le débit (figure 5.11) est légèrement inférieur parce que TCP n'utilise pas toute la capacité qui lui est allouée. Même si il y des

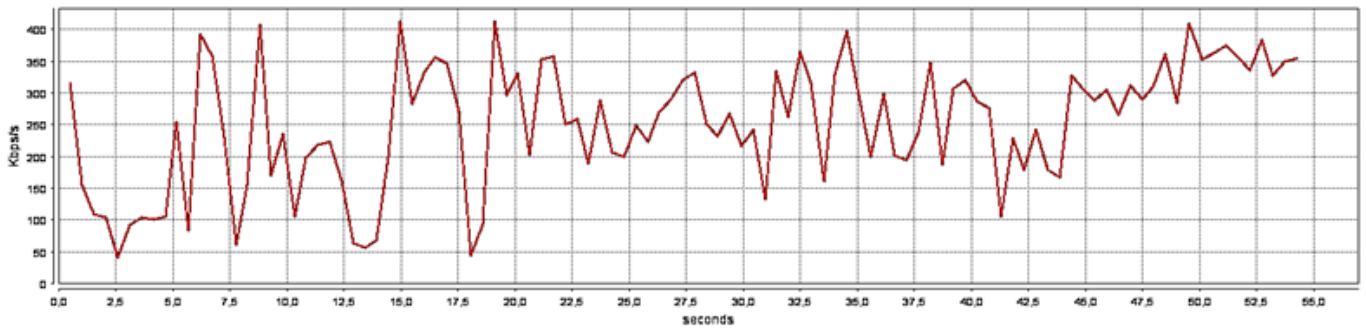


Fig. 5.9 Débit au niveau applicatif pour le cas non streamé en utilisant UDP

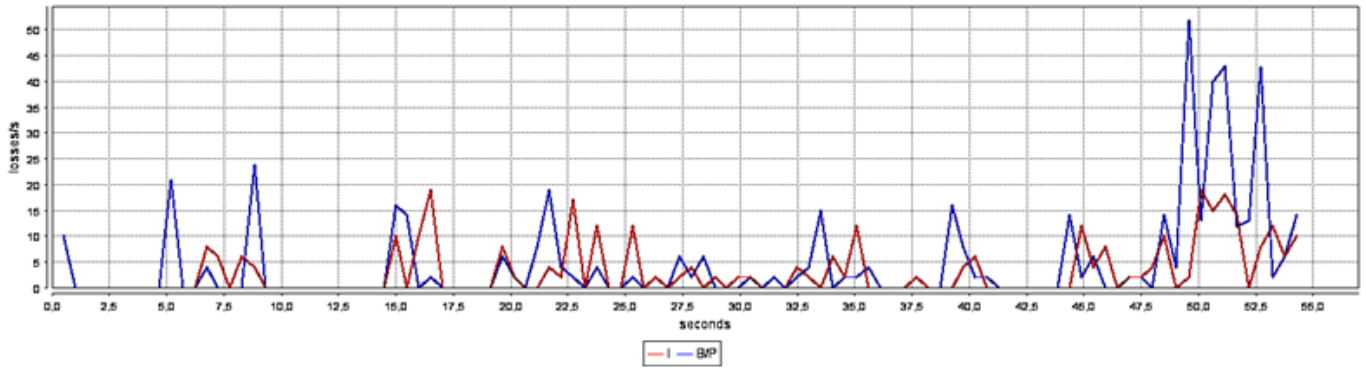


Fig. 5.10 Taux de pertes pour une application non streamée utilisant UDP

pertes lors de la transmission, grâce aux mécanismes de récupération et de fiabilisation de TCP, elles ne sont pas visibles au niveau applicatif. Par contre, ces mécanismes qui cherchent à utiliser toute la capacité disponible sur le lien vont souvent générer des pertes au moment où ils atteignent la capacité

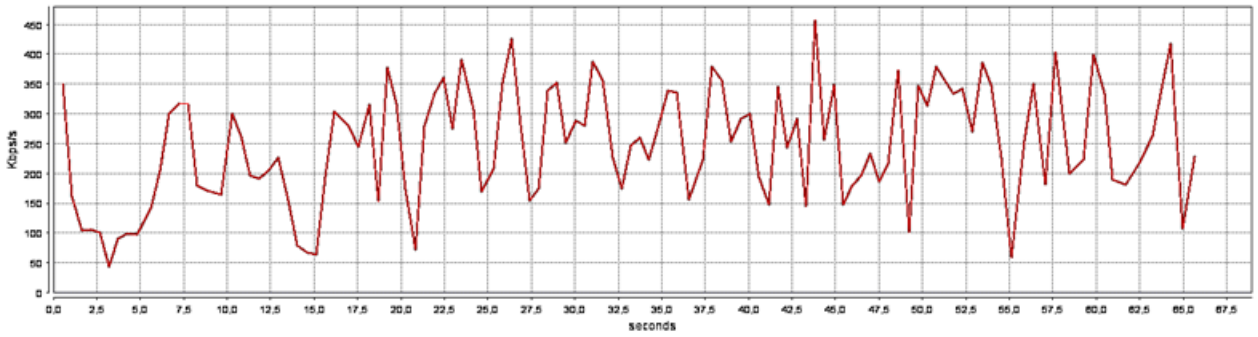


Fig. 5.11 Débit au niveau applicatif pour le cas non streamé en utilisant TCP

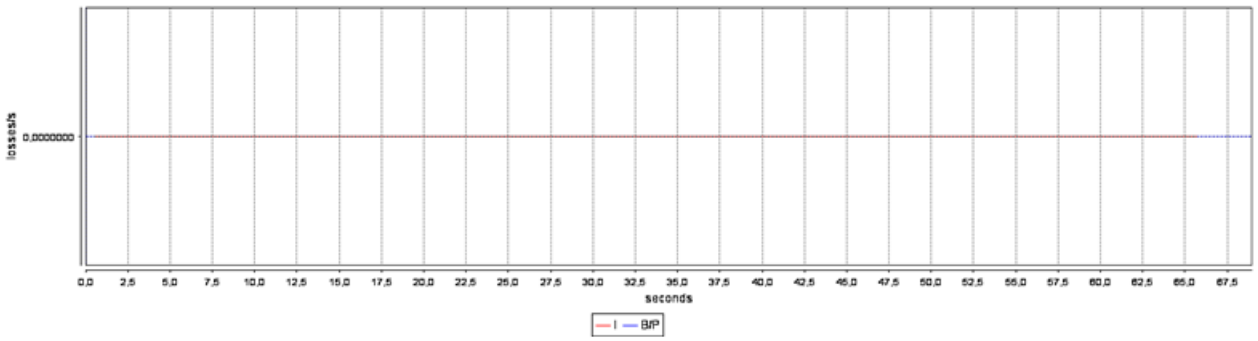


Fig. 5.12 Taux de pertes pour une application non streamée utilisant TCP

maximale de ce lien, ce qui se traduit par une augmentation globale du délai de transmission du fichier.

3. Avec ETP pour la CdS application RT interactive

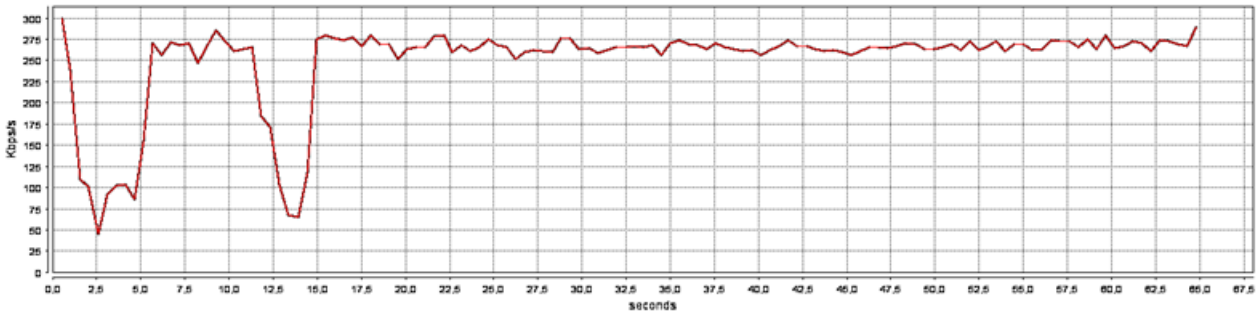


Fig. 5.13 Débit au niveau applicatif pour le cas non streamé en utilisant ETP (CdS RT Interactive)

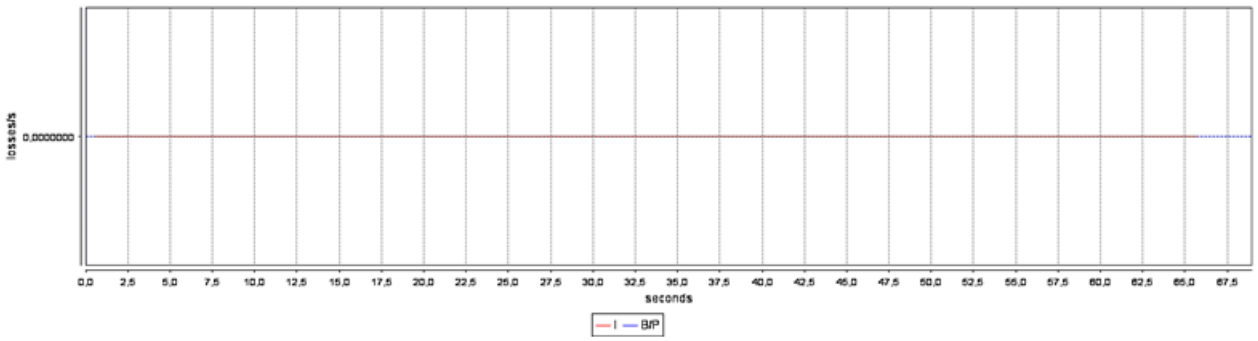


Fig. 5.14 Taux de pertes pour un profil d'application non streamée utilisant ETP (CdS RT Interactive)

Pour les applications non streamées dans le cas RT, ETP met en oeuvre des mécanismes de conditionnement de trafic et de contrôle d'erreurs. En comparant les résultats des figures 5.13 et 5.14 avec ceux obtenus avec le protocole UDP, on remarque qu'ETP offre de meilleurs résultats au niveau de la fiabilité. Par rapport à TCP, le temps de transfert est meilleur avec ETP (environ 2 secondes de mieux, environ 65s pour ETP et 67s pour TCP). Grâce aux mécanismes de conditionnement, ETP est donc capable de mieux utiliser la bande passante disponible car il se base sur la connaissance de la bande passante qui lui est attribuée, pour contrôler son débit d'émission. On constate en effet sur la courbe de la figure 5.13 que l'émission est réalisée de manière lissée au niveau de la capacité réservée de la classe.

Cas d'application non streamée utilisant la classe de service Standard
Dans le cas de la classe de service Standard ou Best Effort, la bande passante moyenne allouée à ETP est d'environ 400Kbits/s. Il s'agit de la bande passante résultant de la mise en concurrence de flux avec le flux étudié.

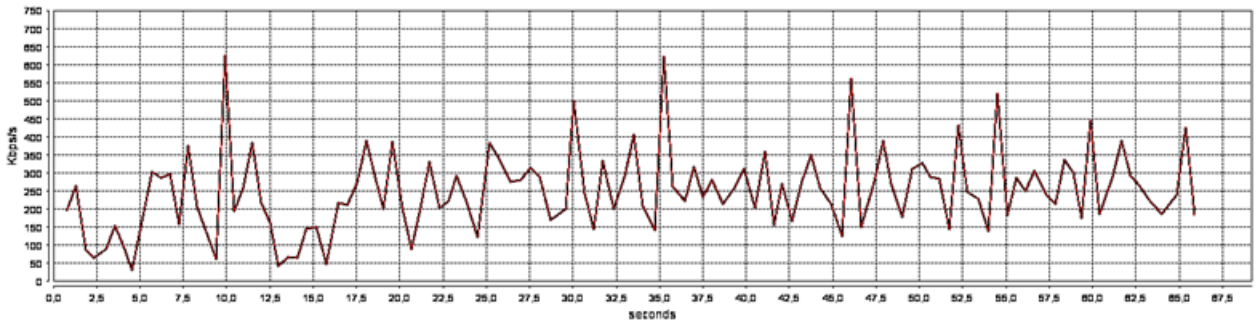


Fig. 5.15 Débit au niveau applicatif pour le cas non streamé en utilisant ETP (CdS Standard)

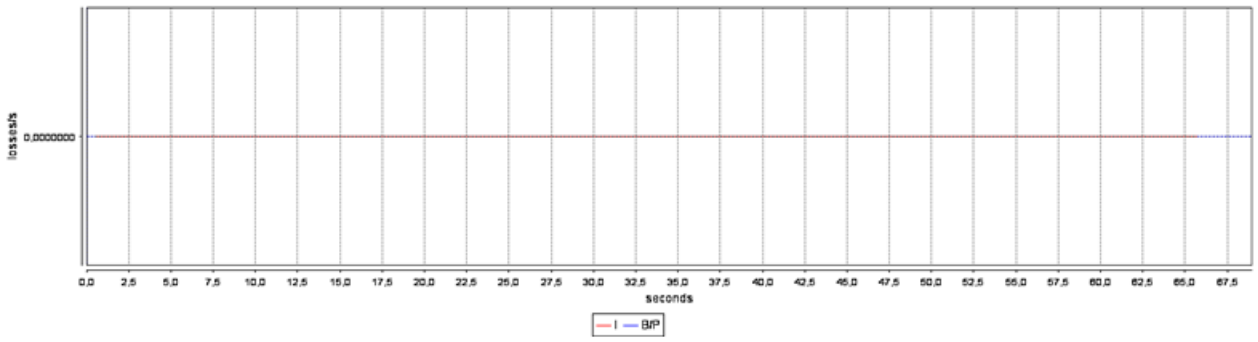


Fig. 5.16 Taux de pertes pour un profil d'application non streamée utilisant ETP (Cds Standard)

Dans un contexte Best effort, l'algorithme TFRC de contrôle de congestion est associé à ETP avec un contrôle d'erreurs basé sur SACK (selective acknowledgment).

On remarque dans ce cas (figures 5.15 et 5.16) que le comportement d'ETP est similaire à celui obtenu avec TCP. Grâce à l'algorithme SACK, ETP est capable de récupérer toutes les erreurs. De plus, le temps de transmission est d'environ 67s (valeur proche de celle obtenue avec TCP dans le cadre d'une réservation de 400Kbit/s).

Ainsi, cette première évaluation du protocole ETP dans le cas d'applications non streamées, montre à la fois l'efficacité de l'émulation et du protocole dans le cas RT et Standard. Dans le cas RT, ETP dispose d'une connaissance du débit maximal alloué sur le canal de communication et de ce fait, les performances se trouvent être meilleures. En effet, il émet directement à ce débit et il n'y a pas de pertes comme c'est le cas avec le protocole UDP qui n'assure pas une transmission fiable. La différence avec TCP se situe au niveau de l'utilisation de la bande passante disponible. ETP offre de meilleurs résultats car TCP n'émet pas à la capacité maximale du lien d'où un gain de temps lors de la transmission. Dans le cas Standard, ETP permet grâce aux mécanismes de contrôle de congestion (TFRC) et contrôle d'erreurs (SACK) de fiabiliser la transmission tout en gardant une efficacité des ressources disponibles.

Cas d'application streamée dans le cas d'une réservation dans la classe RT

Nous allons considérer le profil de trafic présenté sur la figure 5.17 suivante qui correspond à un stream vidéo.

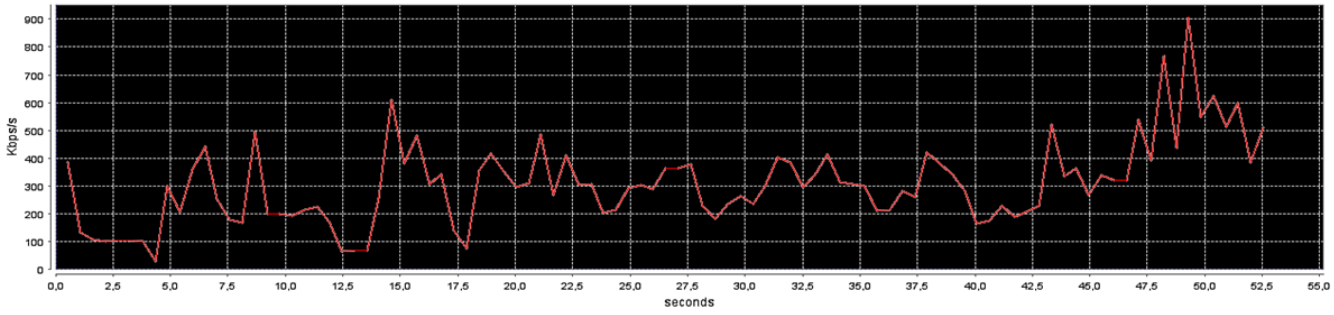


Fig. 5.17 Profil de trafic considéré

Pour les trois expériences suivantes, une réservation au débit crête de cette video (1200Kbit/s) est effectuée dans la classe de service RT.

1. Avec UDP
   D'après les figures 5.18 et 5.19 on constate que la bande passante du profil n'est pas affectée car la réservation est effectuée au niveau du débit crête de la vidéo. Il n'y a donc pas de pertes de données et l'application reçoit les données sans délai.
2. Avec TCP

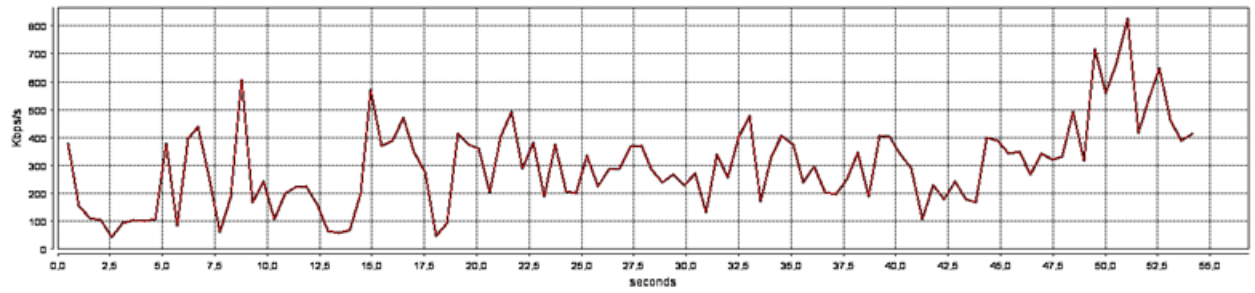


Fig. 5.18 Débit au niveau applicatif pour le cas streamé en utilisant UDP

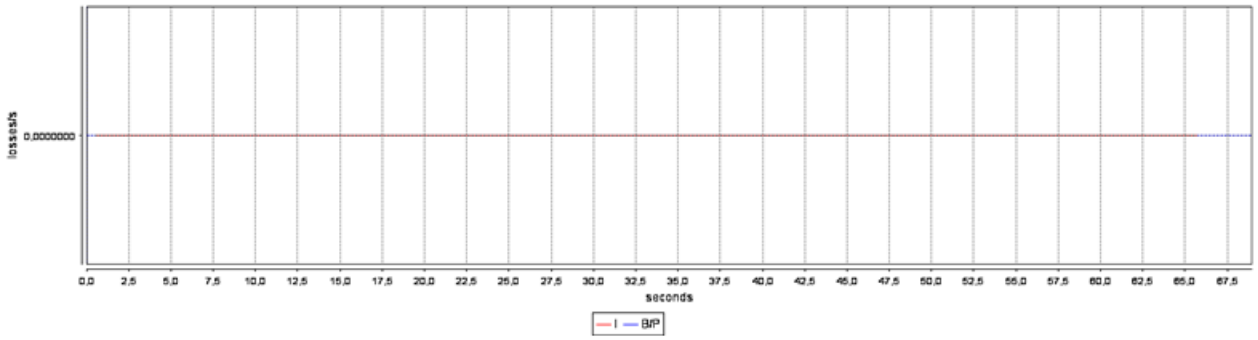


Fig. 5.19 Taux de pertes pour une application streamée utilisant UDP

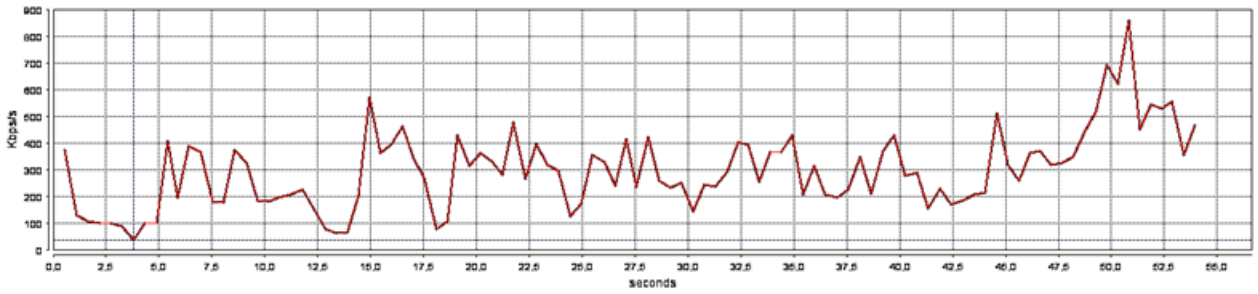


Fig. 5.20 Débit au niveau applicatif pour le cas streamé en utilisant TCP

La figure 5.20, montre que le profil de débit de TCP est légèrement altéré par rapport à celui obtenu avec UDP du fait des mécanismes de contrôle de congestion mis en place par le protocole. Cependant, ceci n'affecte pas la qualité de la présentation de la vidéo.

3. Avec ETP pour la CdS application RT interactive
   Dans le cas de QdS RT, un conditionnement ou shaping du trafic est réalisé au niveau de la couche transport par le protocole ETP. Ceci permet de s'assurer que l'application respecte le profil considéré. Le conditionnement de trafic au

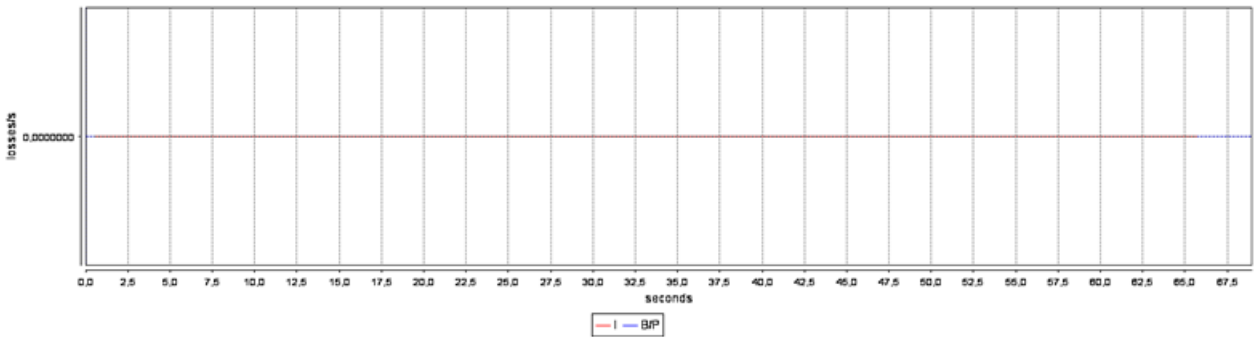


Fig. 5.21 Taux de pertes pour une application streamée utilisant TCP

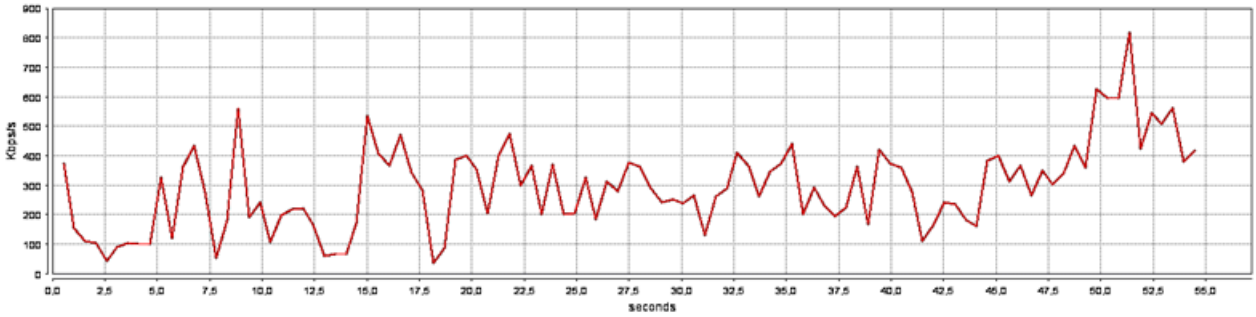


Fig. 5.22 Débit au niveau applicatif pour le cas streamé en utilisant ETP (CdS RT Interactive)

niveau d'ETP est moins agressif que le contrôle de congestion de TCP puisque le profil n'est pas altéré (en comparaison avec le profil UDP) jusqu'à environ 52s où un burst est lissé par l'algorithme de conditionnement.

On constate sur les figures 5.18, 5.20, et 5.22 que les résultats obtenus pour les protocoles UDP, TCP et ETP (RT) sont relativement proches. Ce comportement

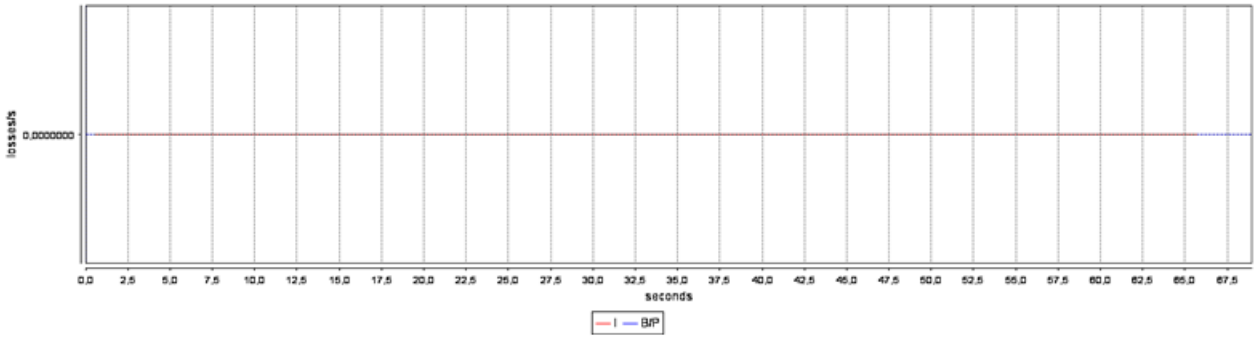


Fig. 5.23 Taux de pertes pour un profil d'application streamée utilisant ETP (CdS RT Interactive)

s'explique par le fait que l'on utilise une application streamée. Le conditionnement est réalisé au niveau de l'application. Ainsi dans le cas d'UDP, de TCP et d'ETP une réservation est réalisée au niveau du débit crête du flux.

Conclusion

Dans cette première partie nous avons présenté le réseau EuQoS avec ses caractéristiques. Nous avons montré une première utilisation possible de l'émulation dans ce contexte avec l'expérimentation d'un protocole de transport ETP. Etant donné que les mécanismes à émuler pour la mise en place de l'évaluation du protocole ETP n'étaient pas très complexes, une émulation basée sur des scénarios temporels suffisait. Les résultats ont permis de montrer que dans le cas d'un flux streamé, le protocole offre de meilleures performances que les protocoles UDP ou TCP grâce à sa connaissance préalable du débit du canal de communication.

Dans la suite de cette étude de cas, nous allons nous intéresser à l'étude puis à la modélisation d'un accès satellitaire.

5.2 PRÉSENTATION DU CONTEXTE SATELLITE

C'est en 1965 qu'apparaît pour la première fois un réseau commercial de télécommunications par satellite géostationnaire avec Intelsat. Depuis, cette technologie n'a cessé d'évoluer pour être aujourd'hui l'un des moyens les plus utilisés pour les transmissions de données. Il existe quatre types de satellites différenciés par leur orbite :

- Low Earth Orbit : LEO (orbites basses entre 200 et 800 km). Elles sont à défilement et le satellite ne reste visible au-dessus d'un point que quelques minutes. Pour sa capture, il nécessite des antennes suiveuses de dimensions convenables. Ces orbites sont utilisées par les navettes, les laboratoires spatiaux, l'observation et la photographie de la Terre, la météorologie, ainsi que les satellites militaires.
- MEO (orbites moyennes entre 10 000 et 15 000 km). Elles sont à défilement, cependant le satellite reste visible au-dessus d'un point pendant quelques heures. Leurs utilisations sont du même ordre que pour les satellites en orbite basse (LEO)
- HEO (orbite héliosynchrone, fortement elliptique dans un plan quasi polaire), présente le satellite à la verticale d'un point, tous les jours à la même heure et sur la face éclairée de la Terre. Mêmes applications, altitudes vers 800 km.
- Geostationnary Earth Orbit : GEO (orbite de 35900 km au dessus de l'équateur). Le satellite suit la même période de révolution que la Terre (environ 23h56min). Cette altitude lui permet de paraître immobile dans un référentiel géocentrique c'est à dire par rapport à la Terre. Ce type d'orbite couplé à des réflecteurs offre une couverture relativement importante : avec un seul satellite près de 42% de la surface terrestre sont couverts. Cependant, ces satellites, par l'altitude de leur orbite, ont des délais de transmission moyens élevés de l'ordre

de 280 ms.

La liaison satellite considérée dans cette étude est un lien satellite géostationnaire (GEO).

Utiliser un satellite comme moyen de communication présente quelques avantages. Tout d'abord, ce type de réseau, à la différence des réseaux filaires, ne nécessite pas la mise en place d'infrastructures complexes et offre naturellement un support de diffusion à grande échelle. De plus, ils ne sont pas contraints par les frontières nationales. L'installation des terminaux sous la zone de couverture est simple.

Cependant ce type de médium de communication a quelques inconvénients comme par exemple la bande passante qui se trouve être limitée car elle doit être partagée entre les différents utilisateurs. Ceci implique le prix de revient d'une telle liaison. De plus, il faut noter qu'une liaison satellite est sensible aux conditions météorologiques. En effet, la qualité des communications est différente par temps clair ou sous un orage.

Le mode de transport de données utilisé est le mode paquet. Deux modes de transport sont généralement utilisés dans les systèmes satellite : AAL5/ATM et MPE/MPEG.

La technologie ATM a été utilisée dans les systèmes satellite dans un but d'interopérabilité avec les systèmes terrestres dans lesquels elle était déployée.

Le mode de transport MPEG provient du standard de diffusion de télévision numérique pour le transport de contenus audio et vidéo et utilise des paquets MPEG2-TS. L'encapsulation est réalisée par la couche MPE (Multi Protocol Encapsulation) selon la norme ISO/IEC 13818-6.

5.2.1 Architecture considérée

L'architecture dans cette étude est présentée sur la figure 5.24. Elle est composée d'un segment spatial - le satellite en orbite géostationnaire - et d'un segment terrestre sans lequel les satellites ne pourraient pas être administrés d'une part et ne pourraient pas délivrer d'informations d'autre part. Ce segment terrestre comprend des terminaux Satellite Terminal (ST) qui ont accès au satellite, une station passerelle (Gateway) qui permet de connecter le satellite à l'Internet, et un Network Control Center (NCC) dont le rôle est de gérer l'accès à la ressource satellite.

5.2.2 La norme Digital Video Broadcasting (DVB)

Le Digital Video Broadcasting (DVB) [18] est une norme européenne de diffusion numérique créée pour la télévision. Elle est associée au format de compression MPEG-2 [35]. Cette norme définit comment transmettre des signaux MPEG-2 en utilisant différents supports tels que le satellite, le câble, et la diffusion terrestre ainsi que la façon de protéger le signal.

Différents standards DVB existent et peuvent être regroupés dans les catégories suivantes :

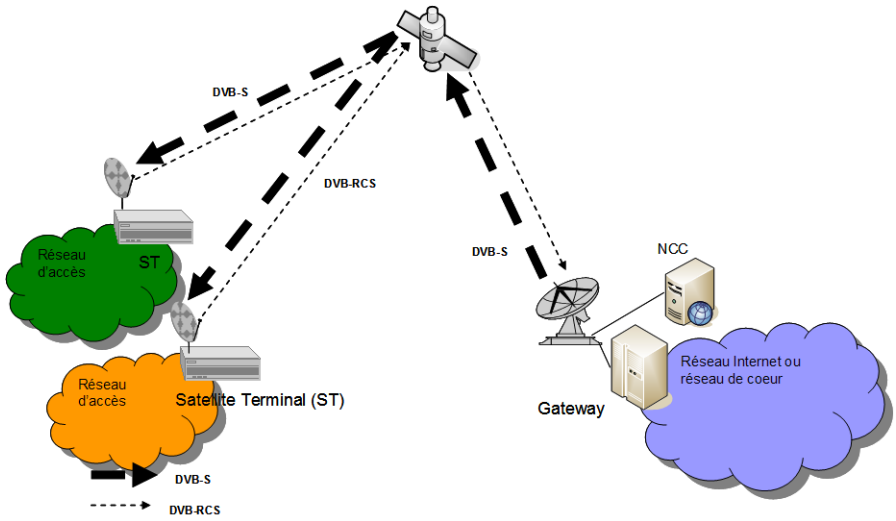


FIG. 5.24 Architecture satellite étudiée

- Les standards de transmission qui permettent le transport des Elementary Streams (ES) MPEG-2 sur différents supports. Les trois grands standards qui sont le DVB-S (Satellite), DVB-T (Hertzien / Terrestre) et le DVB-C (Câble). Il existe d'autres standards plus récents comme le DVB-H (Handheld) qui est couplé au DVB-T et qui est destiné à la réception sur terminaux mobiles.
- Les standards d'interaction qui traitent des voies retour. On y trouve les standards DVB-RC* (DVB-Return Channel for *). Parmi ceux-ci nous pouvons citer le DVB-RCS (Return Channel for Satellite) pour la voie retour par Satellite.
- Les standards d'interfaçage et de codage avec notamment le DVB-CI (DVB Common Interface)

A l'exception des États Unis, du Mexique, du Canada et de la Corée du Sud, le DVB a été adopté par tous les pays du monde pour la télévision et la radio numérique.

Le MPEG-2

Le standard MPEG-2 est à la base de la norme DVB. Il regroupe des méthodes de multiplexage particulières qui sont basées sur le découpage des flux élémentaires (Elementary Stream) en PES (Packetised Elementary Stream) [35]. Ces paquets contiennent un flux vidéo dont le codage et les en-têtes contiennent les informations nécessaires à son décodage. Ce paquet est appelé flux élémentaire ou Elementary Stream (ES).

Le DVB-S

Le Digital Video Broadcasting par Satellite (DVB-S)[9, 12] est un standard ouvert de transmission de la télévision numérique par satellite géostationnaire initié en 1993. Il regroupe des informations sur les protocoles utilisés pour le multiplexage

des données en se basant sur le standard MPEG-2, des méthodes d'accès et des informations sur le codage canal ainsi que sur la modulation du signal.

Le DVB-S définit le mode de transmission sur un lien unidirectionnel satellite de données audio et vidéo vers des terminaux équipés de cartes DVB-S. L'architecture classique d'un système DVB-S (le plus souvent dédié aux services télévisuels) s'appuie sur un satellite transparent mono faisceau et une Gateway. Les terminaux ne sont uniquement capables de recevoir les données.

Méthode d'accès

Dans le cas du DVB-S, les stations qui ont accès au système satellite sont équipées de transpondeurs et sont généralement appelées Gateway. Les méthodes d'accès utilisées sont pour la majeure partie des méthodes propriétaires. Elles reposent sur une division fréquentielle de la bande passante et parfois temporelle pour les systèmes qui embarquent une intelligence à bord. La Gateway va émettre à un débit constant et permanent. Le bourrage est utilisé relativement fréquemment.

Le DVB-RCS

Avec le DVB-S, on dispose d'avantages tels que : une grande capacité, un débit constant et une bonne adéquation à la télévision mais tout cela uniquement de manière unidirectionnelle : de la Gateway vers les terminaux. L'avènement de nouveaux services IP comme la Voix sur IP, Vidéo à la demande, etc., a fait apparaître de nouveaux besoins et plus particulièrement en terme d'interactivité. Or cette interactivité n'est pas prévue par le DVB-S. Les solutions possibles [19] étaient basées, dans un premier temps, sur l'idée d'utiliser un lien terrestre pour les requêtes provenant des utilisateurs. Cependant ce lien de retour n'étant pas toujours disponible, l'idée était de pouvoir disposer d'un canal interactif permettant au terminaux d'instaurer un dialogue avec les serveurs ou les différents utilisateurs de l'Internet, et ce, en utilisant le satellite.

Le DVB-RCS [11, 10](Digital Video Broadcast - Return Channel System) est une norme soutenue et développée par l'European Telecommunications Institute (ETSI [20]) en 1999. Elle décrit comment le trafic bidirectionnel des données devrait être transmis via satellite. Cette norme d'accès Internet par satellite autorise des débits allant jusqu'à 8 Mbits en flux descendant et jusqu'à 2 Mbps en flux montant. Suivant le DVB-RCS, le terminal satellite du client peut recevoir une transmission standard de DVB envoyée par la station satellite maîtresse. La transmission depuis l'installation client vers cette même station peut également être envoyée via la même antenne.

La norme permet de transporter le protocole IP mais prend en compte également de nombreux protocoles de routage (RIP, IGMP) et de transport (RTP, UDP, TCP). Dans la norme, les terminaux sont appelés Return Channel Satellite Terminal (RCST).

Le système BBI (Broadband Interactive Service) de Satlynx [58] est un exemple de mise en oeuvre de la norme DVB-RCS, mais cette dernière comporte un grand

nombre d'options qui rend l'interopérabilité des terminaux problématique. Le consortium Satlabs Group [26], créé à l'initiative de l'agence spatiale européenne, cherche à promouvoir le déploiement de la norme à grande échelle. Il dispose d'un groupe de travail sur l'interopérabilité des terminaux où l'on retrouve les principaux fabriquants de terminaux pour l'Internet par satellite (EMS Technology [59], Newtec Cy [16], Nera ASA [46]).

Méthode d'accès

Dans le cadre du DVB-RCS, la méthode d'accès utilisée est plus complexe que celle du DVB-S puisque le DVB-S n'a pas vocation à partager sa bande passante entre beaucoup de Gateway. Le DVB-RCS va permettre de mettre en oeuvre une certaine inter activité en facilitant l'accès à la ressource (accès au système satellite) à un grand nombre de terminaux. On se rend compte que si on disposait d'un partage de ressources fixe et constant, cela ne fonctionnerait pas. Chaque terminal envoie des messages au NCC afin de fixer et de garantir les paramètres de qualité de service (débit, délai, gigue, etc.).

La norme DVB-RCS définit une voie de retour au format MF-TDMA (Multi Frequency Time Division Multiplexing Access) qui permet de partager la capacité montante. La norme spécifie que les liaisons montantes et descendantes doivent utiliser des fréquences différentes mais est indépendante des bandes de fréquences utilisées (Ku, Ka, L, S...). Ainsi, le lien retour est décomposé en plusieurs bandes de fréquence, elles-mêmes partagées en slots temporels durant lesquels les RCST pourront transmettre des bursts de données.

Les différents bursts

Il existe quatre grands types de bursts DVB-RCS :

- Bursts de trafic TRF (Traffic) basés soit sur les cellules ATM (ATM TRF burst) soit sur des paquets MPEG2-TS (MPEG2-TS TRF burst)
- le burst CSC (Common Signalling Channel) pour l'identification du RCST auprès du NCC lors de la phase de logon.
- le burst ACQ (Acquisition) optionnel, utilisé lors de la synchronisation.
- le burst SYNC (Synchronisation) utilisé pour maintenir la synchronisation ou pour envoyer des informations de contrôle vers le NCC.

Un burst de cellules ATM est constitué :

- d'une concaténation de cellules ATM (le nombre varie en fonction de la taille de l'unité temporelle d'émission)
- d'un champ Satellite Access Control (SAC) optionnel
- d'un entête obligatoire.

Le partage de la ressource

Le mode d'accès TDMA (Time Division Multiple Access), utilisé parfois pour la voie aller, pose problème sur la voie retour. En effet, pour la voie aller, il y a le plus souvent une seule gateway qui se contente de découper sa bande allouée

entre différents groupes d'utilisateurs, tandis que dans le cadre de la voie retour, un grand nombre de terminaux veulent se partager le même canal. Aussi le TDMA ne suffit pas à éviter les collisions entre les différents bursts des terminaux actifs. Le mode d'accès pour la voie retour : le mode MF-TDMA utilise un découpage de la bande allouée en supertrames. A chaque groupe de terminaux une supertrame est allouée et celle-ci est elle-même découpée en trames. Chaque trame est constituée d'un ensemble de time-slots, chacun dédié à la transmission d'un burst précis. La figure 5.25 suivante présente ce découpage.

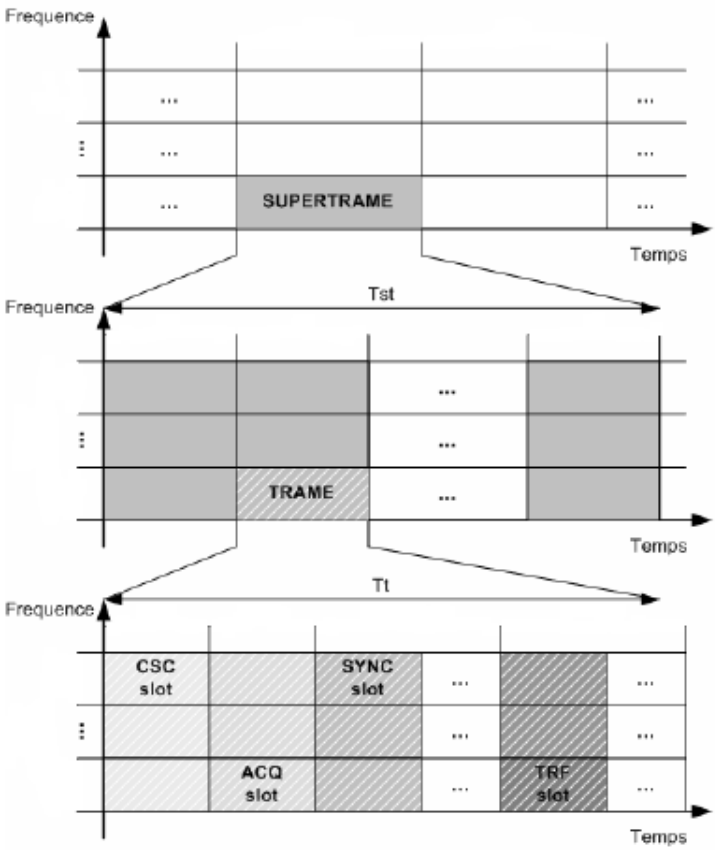


Fig. 5.25 Format d'une supertrame DVB-RCS

Il existe deux types de partage MF-TDMA de la bande passante : un mode statique et un mode dynamique. La méthode statique utilise des time-slots de même durée et de codage identique. Cette technique réduit grandement les temps d'attente d'émission sur le média et les messages à envoyer. Plus complexe, la méthode dynamique se construit sur des time-slots de taille variable, permettant une meilleure flexibilité, et donnant au NCC la possibilité d'attribuer de plus ou moins grands time-slots, en fonction de l'allure du trafic. Toutefois cette méthode demande plus de calculs de la part du NCC (elle doit transmettre les informations de chaque time-slot), ce qui risque de dégrader le temps d'accès au support.

La méthode d'allocation de ressources peut être de plusieurs types. L'accès au support peut être fixé pour toute la durée d'une communication, ce qui permet d'allouer à un terminal ou à un groupe d'utilisateurs une partie fixe de la bande passante. Les méthodes de ce type sont souvent gérées par la fonction CAC (Connection Admission Control). La bande passante pour une communication peut aussi être attribuée dynamiquement, ce qui permet de s'adapter au trafic, et ainsi d'optimiser l'utilisation de la capacité satellite. Ces méthodes, appelées DAMA (Demand Assignment Multiple Access), sont toutefois plus complexes à mettre en oeuvre et introduisent un temps d'accès au support variable et souvent complexe à modéliser.

Nous allons détailler le fonctionnement et la complexité de modélisation de cette technique d'accès à la ressource (DAMA) dans la section suivante.

5.2.3 Principe de fonctionnement de l'allocation en DAMA

Le DAMA est un mécanisme dynamique d'allocation de ressources à la demande, de niveau 2 (MAC) permettant de partager au mieux les ressources disponibles sur la voie retour, et ce, entre les différents terminaux. Il combine des techniques d'allocation fixe et d'allocation dynamique. Il diffère des méthodes statiques qui se contentaient d'allouer des ressources par terminal à la connexion de celui-ci. Cette allocation restait inchangée durant toute la durée de la connexion ce qui entraînait un gaspillage de la ressource du fait de l'impossibilité de réutiliser les slots inutilisés par un terminal. Le DAMA utilise un mécanisme de signalisation pour la partie dynamique : chaque Terminal Satellite calcule la capacité dont il a besoin pour écouler son trafic (en se basant sur le remplissage des buffers MAC), puis il envoie une requête de capacité à un NCC (centre de contrôle du réseau satellitaire). Le NCC traite alors les demandes dans l'ordre et envoie périodiquement à l'ensemble des RCSTs un plan d'allocation des ressources (TBTP). Le RCST distribue ensuite les ressources allouées au trafic se trouvant dans les buffers.

Avec le DVB-RCS, quatres classes d'allocation de capacité sont définies :

- Constant Rate Assignment (CRA) : Allocation du trafic à débit constant, et ce, pour toute la durée de la connexion entre le RCST et le système satellitaire, sans signalisation. Le délai de transmission est constant et se réduit au délai de propagation de la liaison satellite. Dans cette classe d'allocation, la capacité est garantie. Si la connexion est acceptée, le terminal bénéficie de la bande passante pour laquelle il a effectué une requête pour toute la durée de la connexion. Ainsi, le trafic n'est pas sujet à des délais d'ordonnancement et de mise en file d'attente. Le NCC doit s'assurer, à l'initialisation de la connexion que la capacité totale disponible en CRA pour tous les terminaux satellite n'excède pas la capacité totale disponible sur la liaison satellite. Le terminal satellite ayant du CRA dans son contrat (SLA : Service Level Agreement), n'établit pas de requête de capacité. Ainsi, le délai associé à cette capacité est d'environ 300ms (rtt pour un satellite géostationnaire).
- Rate-Based Dynamic Allocation (RBDC) : Allocation du trafic à débit variable. Des requêtes sont émises en fonction du débit moyen entrant sur le RCST.

L'allocation d'un débit constant est valable uniquement sur une période de temps donnée. Ceci convient à un trafic ayant un débit soutenu. Il existe deux composantes : une garantie (RBDCmax) définie lors de l'établissement de la connexion et une non garantie, les deux étant obtenues au moyen de requêtes de capacité.

- Volume Based Dynamic Capacity (VBDC) : Allocation du trafic en fonction du volume. Des requêtes reflétant le volume de données reçu dans les buffers à écouler sont émises. Cette capacité est purement dynamique et n'assure aucune garantie d'allocation.
- Guaranteed Volume Based Dynamic Capacity (G_VBDC) : Allocation du même type que le VBDC mais avec un minimun garanti.
- Free Capacity Assignment (FCA) : Allocation des ressources restantes. Cette capacité est allouée sans requête en fonction des disponibilités des ressources.

Le tableau 5.2 suivant récapitule les caractéristiques de ces différentes classes d'allocation.

Type de classe d'allocation	Garantie	A la demande
CRA	Oui	Non
RBDC	Oui, jusqu'à un maximum	Oui, basé sur un taux et un timer
VBDC	Non	Oui, basé sur le volume
G_VBDC	Oui	Oui, basé sur le volume
FCA	Non	Non (distribué automatiquement)

Tab. 5.2 Résumé des caractéristiques des classes d'allocation

Nous allons maintenant nous intéresser un peu plus en détail au fonctionnement de ce mécanisme :

Lorsqu'un paquet arrive, en fonction des capacités disponibles (CRA, RBDC, VBDC), plusieurs stratégies peuvent se mettre en place. Le cycle de Requête / Allocation, présenté dans [22, 3] est illustré sur la figure 5.26. En supposant que les trames durent 53ms et que les supertrames sont composées de 10 trames, on obtient le fonctionnement suivant.

Les requêtes d'allocation de capacité (CR) sont émises dans les slots réservés à la signalisation en début de supertrame. Les réponses (TBTP) sont émises toutes les 10 trames.

Plusieurs illustrations du fonctionnement de ce protocole d'accès à la ressource vont être présentées.

Cas 1

On suppose dans un premier cas qu'il y a du CRA disponible et que le débit du flux associé au paquet est inférieur à MaxCRA qui correspond au seuil maximum disponible dans cette capacité d'allocation. Le paquet est alors transféré du ST vers la Gateway en passant par le lien satellite. Le délai de transfert est alors de l'ordre

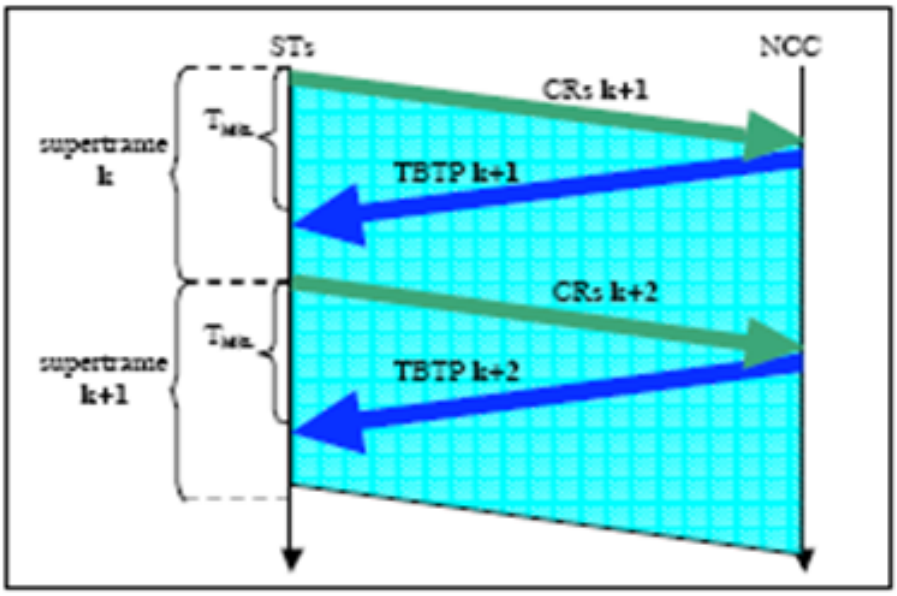


FIG. 5.26 Principe de requête / allocation du DAMA

de 270 à 300ms ce qui correspond au temps nécessaire pour atteindre le satellite en orbite géostationnaire. La Gateway le transmettra ensuite au destinataire situé dans le réseau terrestre.

Cas 2

Maintenant, supposons que l'utilisateur dispose dans son contrat (SLA) de CRA et de RBDC et que dans un premier temps le débit du flux dans lequel se trouve le paquet soit inférieur à MaxCRA. Le paquet sera alors transféré comme dans le cas précédent. Si maintenant, on se trouve au delà de MaxCRA, l'excédent sera alors mis en queue et une requête RBDC sera alors envoyée au NCC. Si la demande est inférieure à MaxRBDC, la requête sera accordée en RBDC. Le RBDC, permet de réaliser une allocation de bande passante à la demande. Un timer est alors déclenché au niveau du ST. Ce temps d'attente correspond en fait au temps inter requête. Pendant cette période, le débit est évalué. Au niveau du NCC un second timer est enclenché. Celui-ci permettra de dire pendant quelle durée sera affecté le débit demandé.

Cas 3

Supposons maintenant qu'il n'y ait pas de capacité disponible en CRA et en RBDC mais uniquement en VBDC. Pour réaliser une allocation en VBDC, le ST va évaluer le volume reçu entre deux requêtes. Lors de l'évaluation du volume de données reçues, s'il dépasse la taille maximum du buffer MAC, les paquets reçus en plus sont alors perdus. Maintenant, si le buffer MAC n'est pas plein, les paquets sont

bufférisés et une requête est alors émise au NCC afin d'écouler le volume reçu. Le NCC répondra en fonction de la capacité dont il dispose et enverra au ST un nombre de slots dans les trames composant la supertrame. Une fois que le ST reçoit cette information (message TBTP), il met alors en forme le trafic pour occuper au mieux l'espace accordé. Si le nombre de slots est suffisant alors tout le trafic peut être émis sinon le reliquat est conservé dans la file d'attente et pourra potentiellement être émis dans les futurs plans d'allocation.

Cas 4

Il se peut également que le ST dispose dans son contrat de capacité en CRA et en VBDC. Dans ce cas, le débit CRA est toujours disponible et tout ce qui excède ce débit est demandé au moyen de requêtes au NCC.

Enfin, si l'ensemble des requêtes a été satisfait, le NCC peut accorder plus de ressources au ST. L'allocation ainsi réalisée est du FCA (Free Capacity Assignment). Il s'agit d'une capacité qui peut être percue comme un bonus pour le ST.

5.3 INTÉGRATION DE L'ACCÈS SATELLITE À EUQOS

De toutes les technologies présentes pour les réseaux d'accès dans le projet EuQoS, l'accès satellite est le seul qui ne soit pas physiquement déployé. La technologie satellite est émulée car le contrôle complet d'un système satellite réel est impossible ou trop onéreux. L'objectif est donc de proposer une émulation d'un système satellitaire offrant le meilleur niveau de réalisme possible avec un minimum de ressources.

Plusieurs solutions étaient possibles pour mettre en oeuvre cet accès satellite :

- Dans le contexte du développement de protocoles de bas niveau, l'émulation satellite peut être basée sur des implémentations réelles de protocoles (par exemple DAMA, protocoles de routage adaptatifs, etc.) utilisant des modèles d'émulation filaire pour éviter d'utiliser l'interface réelle sans fil. Le comportement complet des protocoles et la signalisation sont donc réellement implémentés, alors que seulement le lien physique est émulé, ceci dans le but d'introduire le délai de bout en bout dû à une orbite géostationnaire. Ce type de solution a l'avantage de rendre un service d'émulation très précis mais a l'inconvénient d'être complexe et coûteux à développer et à maintenir dès lors que les protocoles évoluent.
- L'approche d'émulation active se veut être une alternative. En effet, au lieu de reproduire l'accès à la ressource satellite basée sur une implémentation complexe du comportement des protocoles de bas niveau, seul l'effet sur le transfert de donnée sera émulé en fonction d'un modèle d'émulation actif.

5.3.1 Objectif

L'objectif de cette intégration est de proposer un support d'expérimentation basé sur un système satellite émulé et également de pouvoir évaluer son impact au sein

de EuQoS sur les applications et protocoles de communication de bout en bout. Une approche d'émulation est utilisée pour implémenter cette technologie. Ainsi, l'émulateur est en mesure de produire en temps réel un comportement au plus proche du système satellite considéré. L'objectif ici n'est pas d'évaluer les performances d'une solution satellite mais plutôt de produire un comportement qui corresponde à celui d'un réseau satellite. Avec l'émulateur satellite que nous proposons, il est possible de contrôler l'évolution de la QdS grâce à des modèles que nous avons définis. L'objectif est d'introduire cet émulateur au sein de l'architecture du réseau EuQoS puis d'évaluer l'impact d'un lien satellite sur un réseau multi domaine proposant de la QdS.

5.3.2 Intégration du système satellitaire émulé dans l'architecture EuQoS

La figure 5.27 présente l'intégration d'un système satellite au sein de l'architecture EuQoS.

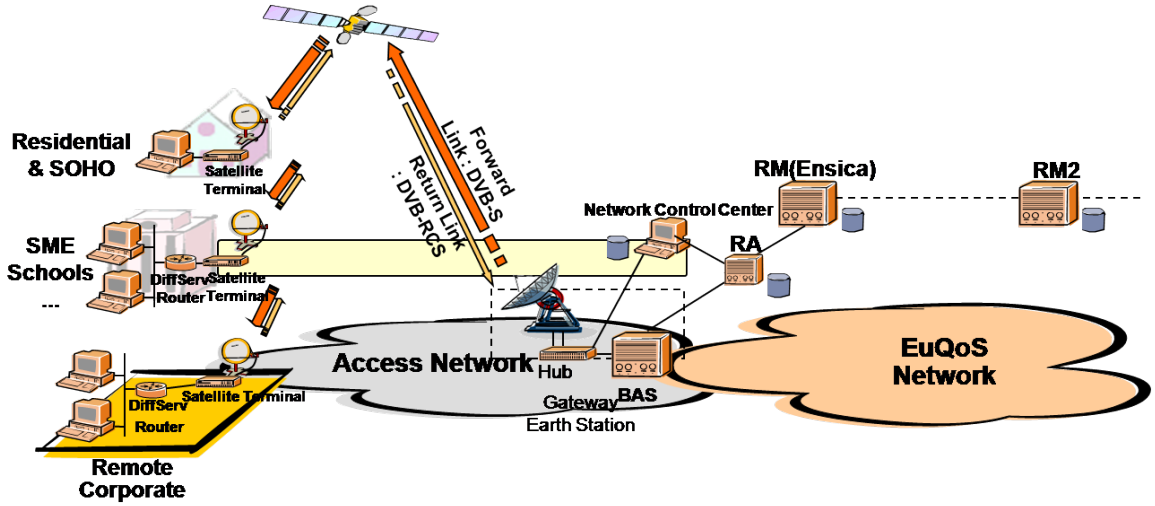


Fig. 5.27 Liaison satellite dans le réseau EuQoS

Sur la figure 5.27 les éléments importants à prendre en compte sont le Ressource Allocator (RA) et le Ressource Manager (RM).

Le rôle principal du RA est d'allouer les ressources. Il implémente des algorithmes de contrôle d'admission propres à chaque technologie réseau considérée. Il implémente également la configuration du système sous-jacent pour réserver la ressource (différents routeurs et noeuds du domaine).

Le RM est en charge de l'admission et de la signalisation de bout en bout. Il fait des requêtes auprès du RA afin que celui-ci vérifie la disponibilité des ressources.

Comme le montre la figure 5.27, le RA et le RM se trouvent du même côté (au niveau du système satellite) de la Gateway et du NCC. Ceci a été décidé afin de limiter la signalisation (délais) entre les deux et également pour permettre une gestion centralisée de l'allocation des ressources satellite.

Le RA est en liaison avec le NCC (centre de contrôle réseau) de manière à avoir des informations en temps réel sur le niveau d'utilisation des ressources. Il est également en liaison avec le routeur de bordure (BAS) du domaine satellite pour le configurer en temps réel en fonction des admissions de trafic. Il va également configurer à distance l'ensemble des terminaux satellite concernés par les admissions de trafic. La figure 5.28 présente une vision simplifiée de l'architecture du RA satellite et de l'émulateur intégré dans le réseau EuQoS. Comme nous pouvons le constater, deux entités distinctes ont accès au contrôle de l'émulation par l'intermédiaire de la classe *AccessClass*. D'une part, le RA satellite et en particulier le module de configuration du réseau sous-jacent (*SatelliteUNModule*). Ce module va en effet réserver des ressources sur le système d'émulation à chaque nouveau flux admis par l'intermédiaire de *AccessClass*. Une configuration en temps réel permet à ce flux d'utiliser la classe de service, dans lequel il a été admis, sur le système satellitaire. D'autre part, une entité de contrôle basée sur des scénarios accède également au contrôle de l'émulation. Ces scénarios sont définis dans un fichier XML et spécifient de manière temporelle les aspects du trafic concurrent à émuler au cours d'une expérience.

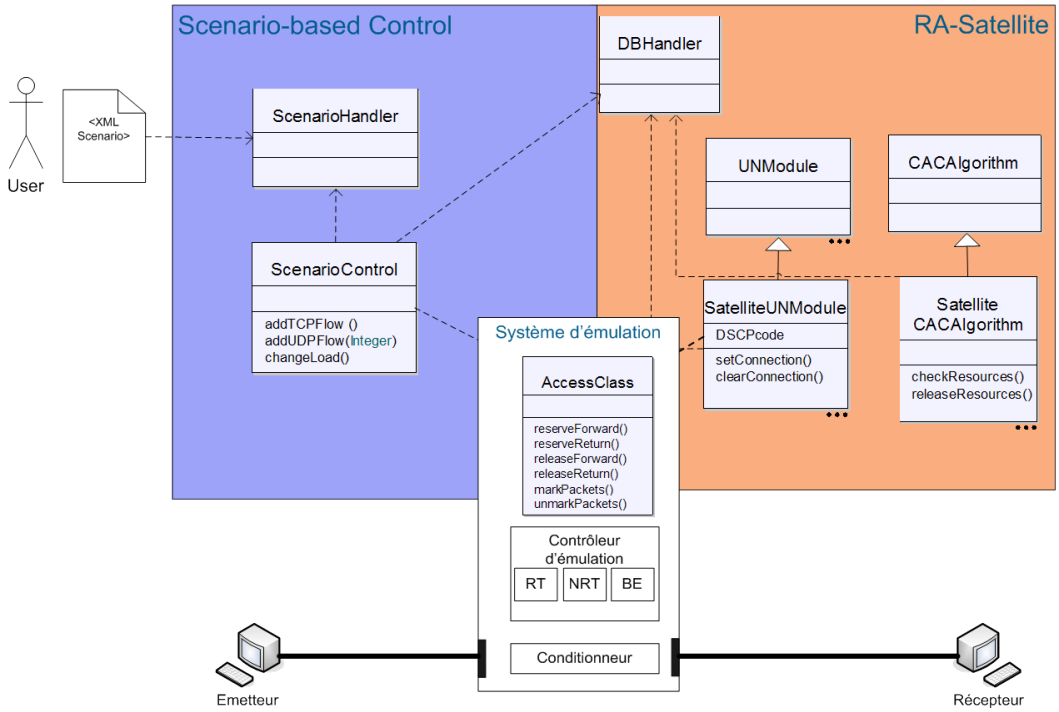


Fig. 5.28 Intégration de l'architecture satellite à EuQoS

Le coeur de l'émulation, c'est-à-dire le contrôleur d'émulation (qui implémente les diverses classes de service) et le conditionneur, vont être décrits en détail dans la section suivante.

5.3.3 Modélisation et implémentation de l'émulation satellite

La modélisation de l'accès satellite, et plus particulièrement du mécanisme de DAMA sur la voie retour, ne peut être réalisée avec une approche d'émulation classique. Il faut être capable de réagir en temps réel en fonction du trafic entrant dans l'émulateur. L'émulation active met en oeuvre cet aspect d'interactivité avec le trafic.

La modélisation de l'accès satellite a été réalisée. Il s'agit de modéliser les différents composants de la technologie considérée. La figure 5.29 représente une modélisation du modèle de technologie de la liaison satellite. Le système satellite est composé du ST, du NCC de la Gateway et du médium qui correspond au satellite.

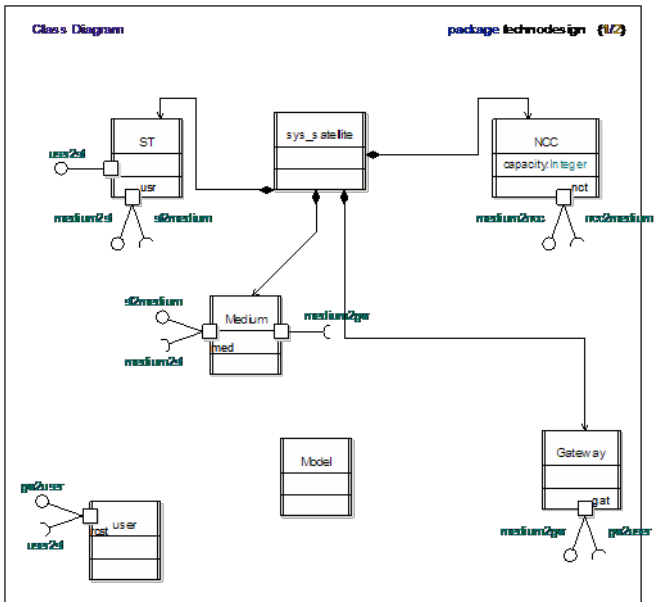


Fig. 5.29 Modèle de technologie d'un accès satellite

La figure 5.30 suivante représente le diagramme de cas d'utilisation de notre système. Il nous permet de donner une vision globale du comportement fonctionnel du système satellite. Trois phases sont considérées pour une connexion : l'établissement, le transfert de données et la fermeture de la connexion. L'établissement de connexion comporte une phase de synchronisation et une phase de loggin, mais elle intègre également une phase de négociation du contrat (SLA). Le contrat peut inclure des garanties pour les classes d'accès CRA, RBDC, VBDC.

La modélisation des classes d'allocation CRA est mise en place en utilisant un modèle d'émulation simple étant donné qu'il s'agit d'une capacité garantie et constamment allouée. Ainsi un modèle statique est utilisé pour représenter la propagation et un modèle dynamique pour représenter les pertes du médium.

Nous allons maintenant considérer la modélisation du VBDC. En effet, le fonctionnement du VBDC dépend du trafic et de ce fait, il nous permet d'illustrer l'utilisation de l'émulation active. La figure 5.31 représente une modélisation du VBDC.

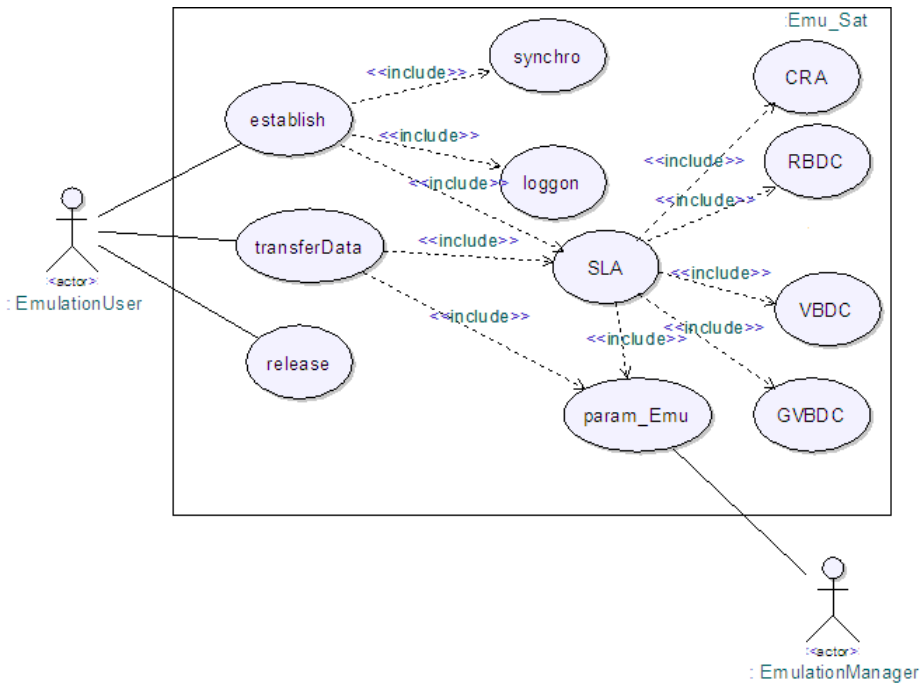


Fig. 5.30 Diagramme de cas d'utilisation de l'accès satellite

Afin de reproduire le comportement global de ce mécanisme, nous l'avons décomposé en comportements de base que nous avons implémentés au niveau des noeuds d'émulation. Ces noeuds vont être enchaînés et ainsi ils produiront le comportement souhaité.

L'objectif du modèle d'émulation est de reproduire les effets sur le trafic du comportement du mécanisme considéré. Ainsi chacun des noeuds représente un aspect du comportement du VBDC.

Le premier enode EN_NCCRequest reproduit l'intervalle dans lequel deux requêtes sont envoyées du ST vers le NCC. Ce comportement est mis en place au niveau du noeud en utilisant un délai circulaire. Ce noeud est également en relation avec le noeud actif qui, grâce aux ports d'écoute, récupère les informations sur le trafic (volume de données entrant sur le premier enode : correspondant à la somme de la taille des paquets).

Le second enode, EN_RequestPropag est un noeud de délai. Il se charge de reproduire le temps de propagation des requêtes du ST vers le NCC, leur traitement, et le retour de la réponse au ST. Ce délai inclut donc un paramètre variable (autour de 100ms) qui correspond au temps de traitement des requêtes au niveau du NCC.

Le troisième enode a la charge de l'émulation de la bande passante. Il reproduit le débit auquel le ST émet les données sur le lien satellite. La limitation de ce débit dépend du trafic entrant et des calculs réalisés par l'enode actif EN_ActiveBWCtrl en fonction du contrat qui existe et du niveau de bande passante utilisée. Ces informations lui auront été transmises par le EN1 (EN_NCCRequest).

Le quatrième enode correspond au délai de propagation du système satellite.

L'enode actif EN_ActiveBWCtrl est basé sur une machine à état et est capable de calculer la bande passante à appliquer aux flux.

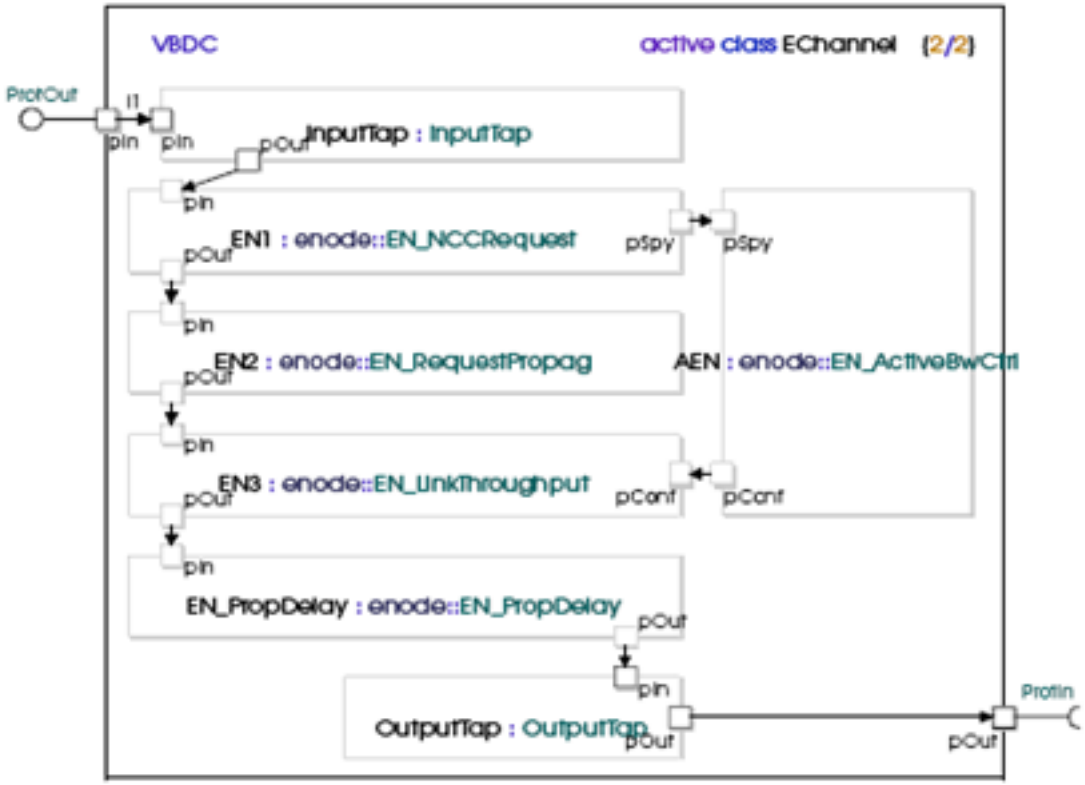


Fig. 5.31 Modélisation du VBDC

Au niveau de l'implémentation, chaque Enode passif correspond à un pipe Dummynet dont les paramètres de QoS sont déterminées par les caractéristiques de l'enode considéré, et qui peuvent évoluer dans le temps. Le enode actif correspond à la partie du contrôleur d'émulation qui gère une classe de service spécifique tel que représenté sur la figure 5.28. Nous utiliserons l'approche d'émulation active basée sur une machine à état pour reproduire le comportement du DAMA.

5.3.4 Résultats

Pour réaliser nos expériences la plate-forme de la figure 5.32 est utilisée. Elle est également basée sur NINE.

Les mesures ont été réalisées sur la plate-forme 5.32. Ces résultats vont permettre d'évaluer le réalisme de l'émulation obtenue en établissant des comparaisons. Pour y parvenir nous allons comparer les résultats obtenus dans le cadre du projet TRANSAT [40] sur une plate-forme de niveau 2. Sur cette plate-forme, uniquement

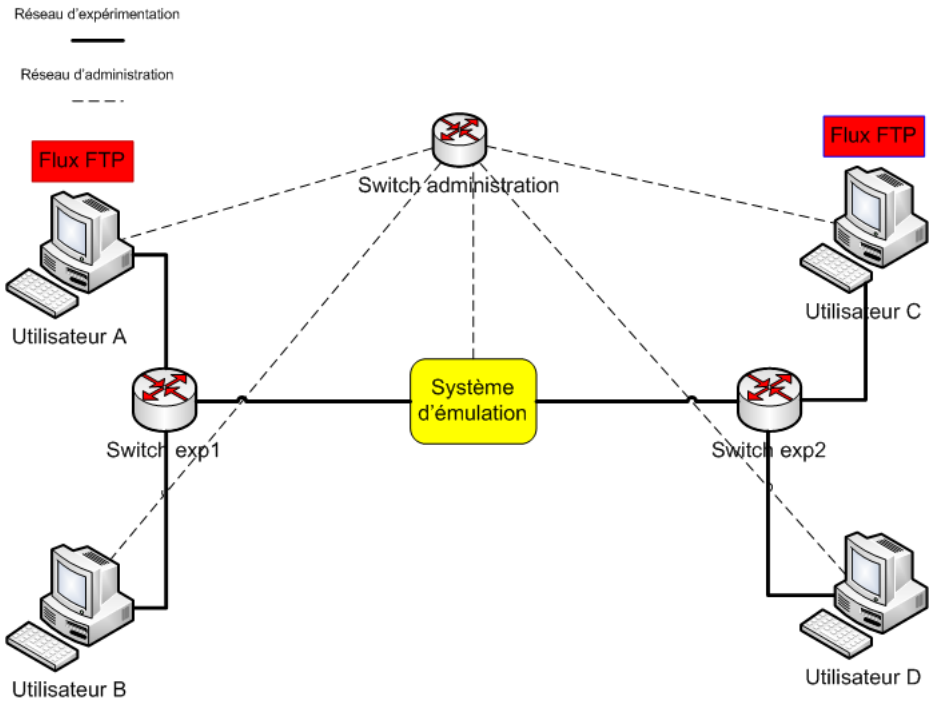


Fig. 5.32 Plate-forme d'expérimentation

la transmission est émulée et tous les mécanismes de contrôle et d'admission sont implémentés. Les résultats obtenus sur cette plate-forme sont comparés avec ceux obtenus avec notre système d'émulation de niveau 3.

Pour les expérimentations, une application émettant à débit constant de 128Kbit/s est utilisée. Ce flux est transmis sur la voie retour du satellite et une réponse est envoyée pour chaque paquet sur la voie aller pour mesurer le RTT. La voie aller est dimensionnée de telle sorte que les réponses n'aient pas à faire face à des congestions.

Nous allons considérer deux cas, un premier où le système d'émulation sera configuré pour proposer uniquement du VBDC et un second où il proposera du CRA et du VBDC. Pour ces deux cas, le RTT sera mesuré pour les paquets. Les buffers sont surdimensionnés de telle sorte qu'il n'y ait pas de pertes dues à des dépassements de capacité.

La courbe de la figure 5.33 illustre les variations induites par la classe d'accès VBDC dans le cas où la capacité de cette classe est de 512Kbit/s (soit bien supérieure au débit d'émission de l'application). La variation cyclique du délai reproduit le comportement correspondant à l'algorithme d'allocation de capacité VBDC. Les requêtes de capacité correspondant au trafic présent dans les buffers sont émises à intervalles réguliers (530ms dans notre implémentation).

Lorsqu'une requête de capacité est accordée, tout le trafic présent dans les queues du ST est émis rapidement (au taux du VBDC garantie dans le contrat). Le délai des paquets diminue de manière cyclique comme le montre la figure 5.33. Ceci est dû au fait que le trafic arrivant juste avant que la requête ne soit envoyée attend moins

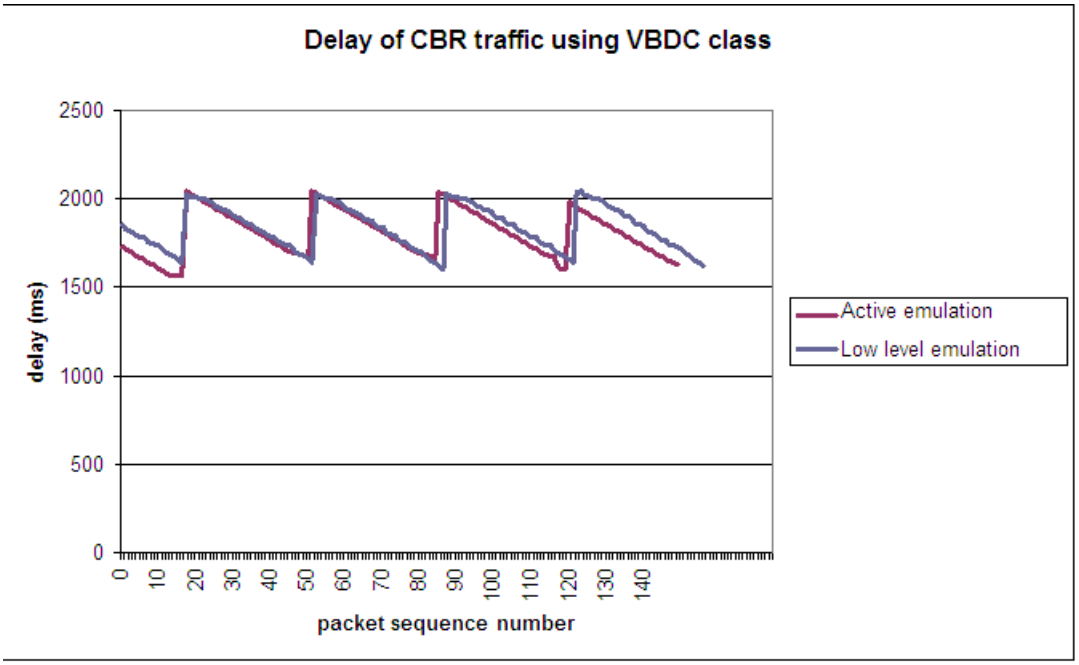


Fig. 5.33 Délai des paquets en VBDC

longtemps dans le buffer que celui arrivant juste après que la requête soit envoyée. Le trafic arrivant juste après la requête doit attendre 530ms pour que la prochaine requête soit émise. De plus, le débit garantit est plus important que le débit du flux. Ainsi pour ces paquets, le RTT augmente jusqu'à environ un délai de 2 secondes.

La courbe de la figure 5.34, présente la variation du délai dans le cadre d'une classe d'accès VBDC où la capacité disponible est de 128Kbit/s ce qui correspond au débit d'émission de l'application. On observe également une variation cyclique du délai, mais celle-ci est moins importante que pour le cas de la figure 5.33. Ceci est dû au fait que la capacité disponible en VBDC est égale au débit de l'application. Ainsi, le temps d'attente des paquets dans le buffer est constant et on peut remarquer que le délai du trafic CBR dans ces conditions tend à se lisser.

La comparaison des résultats obtenus grâce à notre système d'émulation et ceux mesurés sur la plate-forme de niveau 2 sont quasiment identiques. Ainsi nous pouvons valider le comportement du VBDC que nous avons émulé.

La courbe de la figure 5.35 présente la composition des capacités VBDC et CRA qui peut s'avérer utile pour l'implémentation d'applications nécessitant de fortes garanties de qualité de service.

Nous considérons toujours que l'application émet à un débit constant de 128Kbit/s. La bande passante disponible en CRA est de 64Kbit/s et en VBDC également de 64Kbits/s. Le débit d'émission de l'application est donc de deux fois la capacité disponible en CRA. Ainsi, pendant la période initiale du transfert de données, des paquets sont alors stockés dans les buffers. En conséquence, des requêtes de capa-

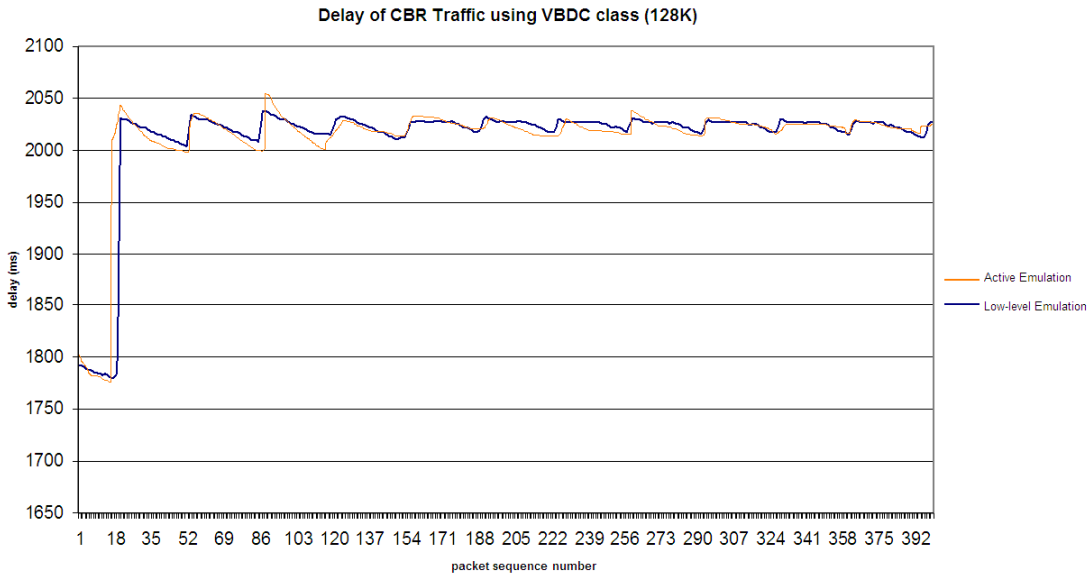


Fig. 5.34 Evolution du délai des paquets dans le cas d'une classe VBDC à 128K

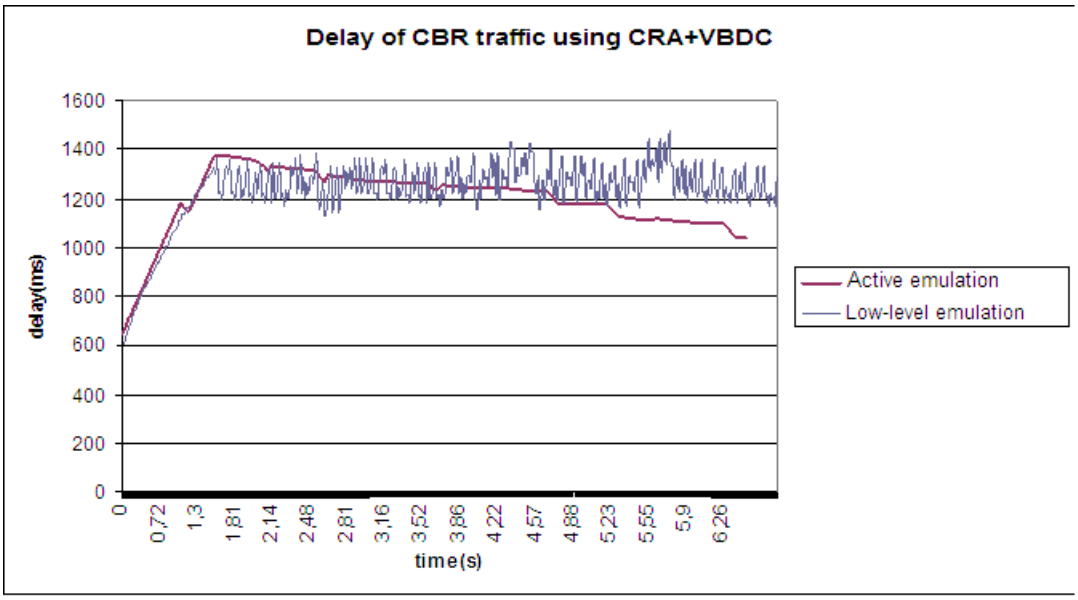


Fig. 5.35 Délai des paquets en CRA et VBDC

cité pour la classe VBDC sont émises. L'accumulation initiale des paquets dans ces buffers entraîne une augmentation du délai de bout en bout, comme le montre la figure 5.35. Le délai se stabilise ensuite du fait de l'allocation de la capacité qui a été requise précédemment. Ainsi la bande passante allouée devient égale au débit d'émission de l'application.

Ces résultats montrent quelques comportements intéressants du système d'émulation. La comparaison des résultats obtenus sur la plate-forme de niveau 2 et ceux obtenus avec le système d'émulation de niveau 3 permet de dire que notre approche d'émulation offre une bonne reproduction des mécanismes.

Description de l'expérience

Nous allons maintenant nous intéresser au cas présenté sur la figure 5.36 où plusieurs flux (un flux HTTP (démarrant à t=60s et s'arretant à t=240s) ainsi qu'un flux de VoIP (démarrant à t=120s et se terminant à t=300s)) vont entrer en concurrence sur la voie retour avec un flux FTP (transfert de fichier de 3,5Mo). Ces flux seront générés réellement.

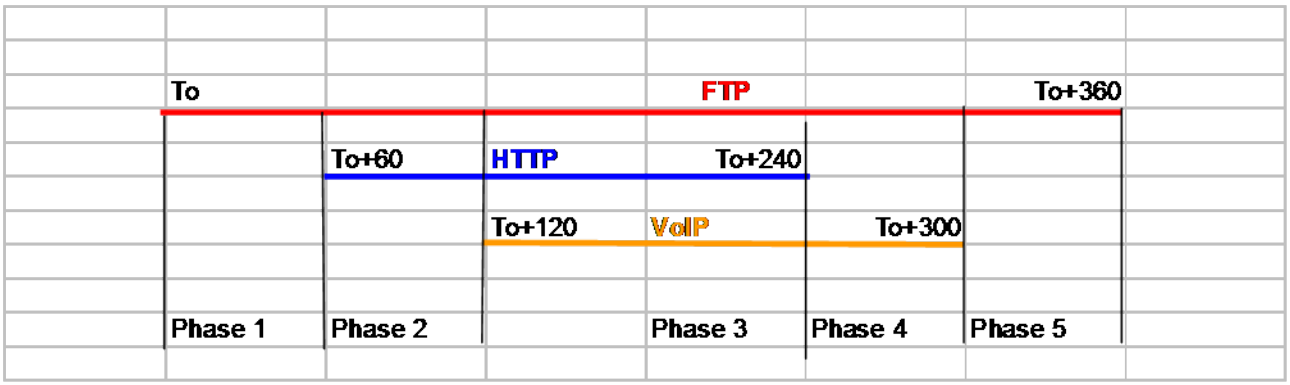


FIG. 5.36 Scénario de flux concurrents

Résultats

Il faut préciser que notre objectif est de valider le comportement que nous avons reproduit dans un cas de référence qui correspond au cas best effort. La figure 5.37 présente une comparaison entre les résultats obtenus grâce à notre émulateur et ceux nous servant de référence (résultats de la plate-forme de niveau 2) au niveau de l'évolution du débit TCP (flux FTP) dans le cas de flux concurrents.

Au niveau du scénario, on constate que l'évolution de la bande passante de TCP se fait aux mêmes instants et suivant les mêmes proportions.

La courbe *serie 2* représente les mesures réalisées sur l'émulateur de bas niveau et la courbe *serie 3* celles réalisées sur notre système d'émulation. On constate que l'allure générale des 2 courbes est la même.

Dans le cadre des délais, nous pouvons voir sur la courbe 5.38 que les résultats obtenus avec l'émulation active et ceux obtenus sur la plate-forme de bas niveau sont très proches de ceux obtenus avec le système d'émuation de bas niveau. On constate également que l'allure générale des deux courbes est la même. Pour expliquer cette

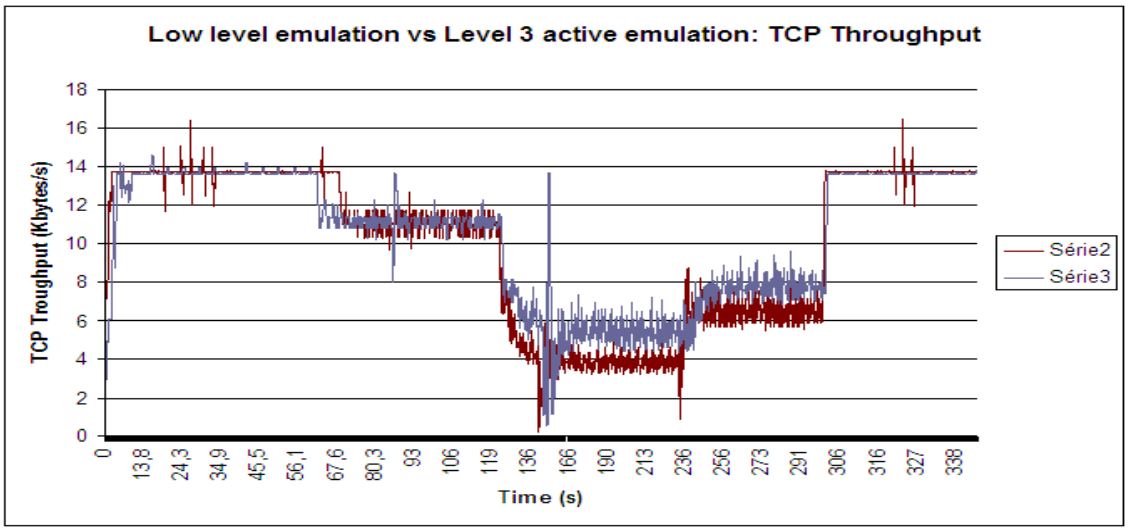


Fig. 5.37 Comparaison des résultats obtenus par émulation de niveau 2 et 3 pour le cas du débit du flux TCP

évolution du délai, il faut se référer au scénario 5.36 et à la figure de l'évolution du débit de TCP 5.37.

On se rend compte que le buffer du routeur atteint par deux fois sa limite de capacité (le délai de bout en bout des paquets n'augmente plus : le délai d'attente dans le buffer est alors maximal) ce qui se traduit sur sur la courbe de débit de TCP par des pertes de paquets. TCP diminue donc son débit d'émission grâce à ses mécanismes de contrôle de congestion. Ensuite TCP va continuer à augmenter progressivement son débit d'émission, ce qui va provoquer une nouvelle augmentation du délai d'attente des paquets de VoIP. La seconde fois où la capacité maximale du buffer est atteinte, on observe sur le scénario qu'elle est liée à l'arrêt du flux HTTP. Ceci explique la diminution plus importante du délai de bout en bout pour les paquets de VoIP. Même si nous nous sommes efforcés de travailler dans des conditions identiques, l'évolution des piles protocolaires sur les systèmes d'exploitation et notamment du protocole TCP, ainsi que la taille du buffer sur le routeur, peuvent expliquer ces légères variations.

La comparaison des résultats obtenus sur notre système d'émulation actif nous montre des résultats très proches de ceux mesurés sur la plate-forme de bas niveau. Ceci nous permet de valider le comportement de notre système d'émulation.

5.3.5 Conclusion

Dans ce chapitre, le projet européen EuQoS a été présenté. Dans le cadre de ce projet, nous avons proposé l'émulation de niveau IP du réseau satellite basé sur différentes technologies d'accès sous-jacentes. La mission de l'ENSICA était de s'occuper de l'accès satellite et nous avons donc proposé de l'émuler. Cet accès satellite est intégré à l'architecture EuQoS. Pour la modélisation du comportement

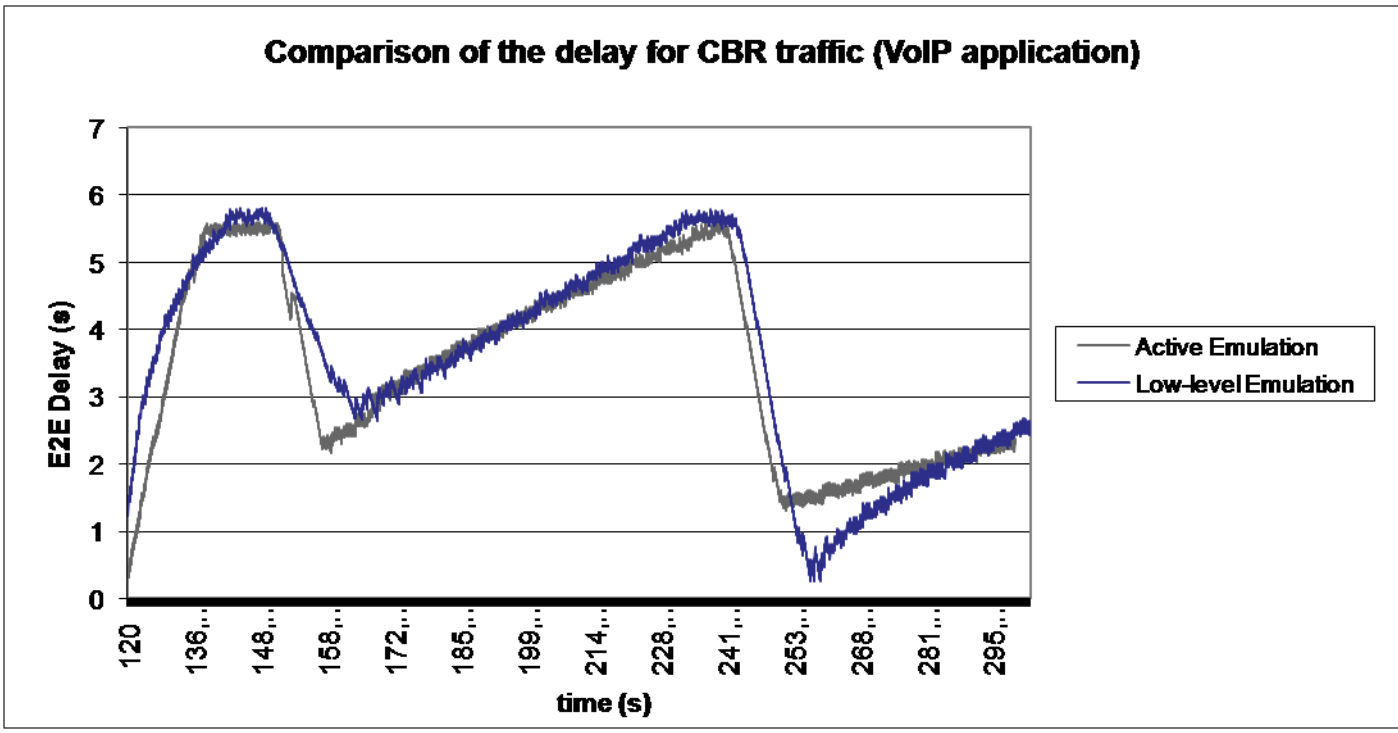


Fig. 5.38 Evolution du délai de bout en bout du trafic CBR dans le cadre d'application de VoIP

du satellite, nous avons utilisé l'approche d'émulation active et des mesures ont été réalisées. Elles ont été comparées avec des mesures réalisées sur une plate-forme de bas niveau. Les résultats ont permis de montrer que le service rendu par notre système d'émulation est statistiquement équivalent à celui rendu par un système satellite réel.

Chapitre 6

Conclusion

Dans cette thèse nous avons présenté le concept d'émulation active qui permet d'augmenter le niveau de réalisme des solutions d'émulation existantes.

Dans le premier chapitre, un état de l'art des différents moyens disponibles dans le contexte de l'expérimentation de réseau ou de protocoles a été présenté. Les limites et les avantages des solutions d'expérimentation ont été abordées. Il en est ressorti que l'expérimentation en environnement émulé s'avère un bon compromis entre le test en environnement réel et la simulation. Nous nous sommes intéressés plus particulièrement à l'émulation de niveau IP. Etant donné que le service est rendu en temps réel, ce type d'approche offre l'avantage de pouvoir utiliser des implémentations de protocoles ou d'applications réelles. Cependant, pour garantir l'exécution en temps réel, des concessions ont dû être réalisées par les différents outils mettant en oeuvre l'émulation de réseau. De ce fait, les principales limites se situent au niveau de la complexité et du réalisme de l'émulation rendue. Vouloir augmenter l'un peut être préjudiciable à l'autre et inversement. Ainsi, ce chapitre a permis d'établir un état de l'art des solutions d'émulation vis-à-vis des besoins des utilisateurs et a servi à définir un besoin global d'émulation.

Dans le deuxième chapitre, deux besoins importants pour l'émulation ont été présentés et une proposition de méthode pour émuler un réseau ou une condition donnée a été proposée. Il s'agit tout d'abord de réaliser une modélisation de l'existant puis d'abstraire le comportement pour en déduire un modèle d'émulation. Ce modèle peut être vérifié aussi bien au niveau du comportement que des aspects temporels. Ainsi il est possible de l'implémenter pour obtenir un émulateur qui fournit un service en conformité avec les spécifications de l'utilisateur.

Le troisième chapitre s'intéresse plus précisément à la reproduction d'un comportement réseau. Pour y parvenir, deux aspects importants ont été considérés. L'ému-

lation de comportements dépendant de la technologie et des protocoles associés que l'on souhaite reproduire, ce qui constitue le coeur même de l'émulation ; puis l'émulation d'événements extérieurs pouvant survenir sur la technologie réseau considérée et ce, dans le but d'augmenter le réalisme de l'émulation rendue. Ce chapitre présente la contribution majeure de ce travail qui se situe au niveau de l'émulation de comportement : l'émulation active. Cette approche d'émulation introduit la notion d'interactivité entre le système d'émulation et le trafic circulant au niveau des interfaces de l'émulateur. Elle permet la prise en compte d'évènements pour la prise de décisions d'émulation (en fonction de modèles). Ceci peut s'avérer très utile dans le cas de la reproduction de comportements complexes. L'architecture d'émulation active introduit donc des observateurs en charge de remonter les informations sur le trafic aux modèles d'émulation. Ces modèles d'émulation actifs peuvent être basés sur différents formalismes selon la vision qu'a l'utilisateur du comportement réseau qu'il souhaite reproduire : pour un comportement probabilistique, l'utilisation de chaînes de Markov ; pour une approche événementielle, les automates à états finis, les réseaux de Pétri ou autres approches hybrides. Ainsi, avec l'émulation active, il est possible d'étendre la notion de dynamicité de l'émulation et en considérant l'aspect comportement de l'émulation, d'apporter un meilleur niveau de réalisme.

La modélisation des aspects extérieurs est également présentée. Elle permet d'augmenter le réalisme de l'émulation rendue. Le cas du trafic concurrent circulant sur le réseau d'expérimentation est abordé. Le résultat de cette modélisation peut rentrer dans le cadre d'une architecture d'émulation active.

Ainsi, ces deux aspects (comportement et aspects extérieurs) constituent le modèle complet d'émulation.

Le quatrième chapitre présente deux cas d'étude basés sur le projet EuQoS. Le premier cas d'étude a permis d'utiliser une approche d'émulation basée sur des scénarios pour l'évaluation du protocole de transport ETP, l'objectif étant d'apporter un support d'expérimentation pour ce protocole représenté par un canal de communication introduisant des variations au niveau des paramètres de QoS.

La seconde étude de cas est basée sur la modélisation d'un accès satellite et plus particulièrement d'un mécanisme d'allocation de ressources sur la voie retour : le DAMA. L'utilisation de l'émulation active s'est imposée du fait de la complexité du mécanisme (requêtes en fonction de volume de données par exemple). Elle a permis de reproduire les effets du DAMA sur le trafic. Une comparaison des résultats obtenus a été réalisée avec des mesures réalisées sur une plate-forme de niveau 2 (où uniquement la propagation est émulée mais où tous les mécanismes de niveau 2 sont implémentés). Les résultats obtenus sur le système d'émulation que nous avons proposé se sont avérés très proches de ceux observés sur la plate-forme de niveau 2.

Cette évaluation a permis de valider l'adéquation du comportement de notre système d'émulation vis-à-vis de la modélisation mais également d'une implémentation de niveau inférieur. Il s'agit donc d'une solution originale permettant à moindre coût

d'obtenir une émulation avec un bon niveau de réalisme.

Comme perspectives de recherche, dans un premier temps, nous pouvons nous attacher à l'amélioration des modèles d'émulation actifs. Il serait intéressant de permettre la prise en compte de nouvelles technologies réseau ou encore de nouveaux mécanismes.

Pour automatiser le passage du modèle de la technologie au modèle d'émulation, un langage ou algèbre pourrait être proposée. Elle permettrait de réduire automatiquement la topologie à un canal de communication qui serait ensuite reproduit au sein du système d'émulation. Ceci aurait l'avantage de simplifier au maximum l'émulation.

Il serait intéressant de pouvoir intégrer les profils d'émulation à Turtle. En effet, Turtle couvre également la phase de déploiement et intègre la génération de code Java. Or actuellement, l'implémentation du système d'émulation est réalisée manuellement à partir de la modélisation que nous avons réalisée. Il serait donc intéressant de disposer d'une génération automatique de code. On pourrait ainsi disposer d'un système d'émulation totalement en accord avec les spécifications et ainsi pouvoir réduire les erreurs d'implémentation.

Concernant l'approche que nous avons proposée, basée sur les équations de modèles de comportement, elle mérite d'être étendue à d'autres conditions pouvant survenir sur un réseau. En effet, nous nous sommes intéressés au cas d'une congestion mais il existe de nombreuses autres conditions qui pourraient être modélisées.

Pour avoir un outil vraiment performant, il serait intéressant d'évaluer la faisabilité d'un émulateur sur un système temps réel. En effet, même si nous avons introduit une phase de vérification du comportement de l'émulateur dans la méthodologie, avec un système temps réel (RTAI [56], LinuxRT [23] etc.) on disposerait d'un moyen permettant de garantir l'exécution des actions d'émulation.

Nous pourrions également nous intéresser à l'utilisation de l'émulation active dans le cadre des réseaux sans fil. Il serait en effet intéressant d'utiliser l'approche d'émulation active pour la modélisation d'un accès 802.11. Il s'agirait d'utiliser l'émulation active conduite par des machines à états pour reproduire les mécanismes d'accès au canal. De plus, il serait intéressant d'établir une comparaison entre cette approche d'émulation active et la solution proposée dans [15] pour l'émulation de réseaux sans fil.

Liste des publications

Conférences internationales avec comite de lecture

1. L. Dairaine, E. Exposito, H. Thalmensy "*Towards an Unified Experimentation Framework for Protocol Engineering*", dans les actes de la conférence 2006 on Service Oriented Architectures in Converging Networked Environments (SOCNE06) à Vienne en 2006.
2. M.Gineste, H.Thalmensy, L.Dairaine, P.Senac, M.Diaz, "*Active Emulation of a DVB-RCS Satellite Link in an End-to-end QoS-oriented Heterogeneous Network*", dans les actes de la conférence 23rd AIAA International Communications Satellite Systems Conference (ICSSC 2005), à Rome en Italie, en Septembre 2005.
3. L. Lancerica, L. Dairaine, F. de Belleville, H. Thalmensy, C. Fraboul, "*MITV, A solution for Interactive TV Based on IP Multicast over Satellite*", dans les actes de la conférence IEEE International Conference on Multimedia and Expo, (ICME) en Juin 2004.

Rapport de recherche

1. Hervé Thalmensy, Laurent Dairaine, Florestan de Belleville, *Mesures DIPCAST sur la plateforme avancée : adéquation MITV et DIPCAST*, Rapport de contrat DIPCAST, Août 2003
2. Hervé Thalmensy, Laurent Dairaine, Florestan de Belleville, *Mesures DIPCAST sur la plateforme de base : adéquation MITV et DIPCAST*, Rapport de contrat DIPCAST, Juillet 2003

Bibliographie

[1] Netperf. NetPerf . http ://www.netperf.org/netperf/netperfpage.html.

[2] *Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols*, volume 3, 2005.

[3] Olivier Alphand. *"Architecture de qualité de service pour systèmes satellites DVB- S/RCS dans un contexte NGN"*. PhD thesis, These de doctorat de INPT, 2005.

[4] Ludovic Apvrille. *"Contribution à la reconfiguration dynamique de logiciels embarques temps-reel : application à un environnement de télécommunication par satellite"*. PhD thesis, These de doctorat de Institut National Polytechnique de Toulouse, 2002.

[5] K.Walsh P.Mahadevan D.Kosic J.Chase A.Vahdat, K.Yocum and D.Becker. Scalability and accuracy in a large-scale network emulator. In *Proceedings of the 5th Symposum on Operating Systems Design and Implementation (OSDI)*, December 2002.

[6] Andy Bavier, Nick Feamster, Mark Huang, Larry Peterson, and Jennifer Rexford. In VINI veritas : Realistic and controlled network experimentation. In *Proceedings of ACM SIGCOMM 2006*, Pisa, Italy, September 2006.

[7] G.T Nguyen B.D Noble, M.Satyanarayanan and R.H Katz. Trace-Based Mobile Network Emulation. In *Proceedings of ACM SIGCOMM'97*, September 1997.

[8] Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhan, Ya Xu, and Haobo Yu. Advances in Network Simulation. *IEEE Computer*, 33(5) :59–67, May 2000.

[9] Digital Video Broadcasting. Digital video broadcasting ; implementation guidelines for data broadcasting. *ETSI technical report*, 01/2003.

[10] Digital Video Broadcasting. Digital video broadcasting ; interaction channel for satellite distribution systems ; guidelines for the use of en 301 790. *ETSI technical report*, 01/2003.

[11] Digital Video Broadcasting. Digital video broadcasting ; interaction channel for satellite distribution systems. *ETSI EN 301 790*, 03/2003.

[12] Digital Video Broadcasting. Digital video broadcasting ; dvb specification for data broadcasting. *ETSI Norm*, 11/2004.

[13] J. Clark, J. Cowan, and M. Murata. RELAX NG Compact Syntax Tutorial, 2003.

[14] CNRS. Grid 5000. French ACI, http ://www.grid5000.org, 2003.

[15] Emmanuel Conchon. *"Définition et mise en œuvre d'une solution d'émulation de réseaux sans fil"*. PhD thesis, These de doctorat de INPT, 2006.

[16] Newtec cy. Newtec cy : http ://www.newtec.eu/.

[17] DIPCAST. Dipcast homepage : http ://dipcast.netvizion.fr/fr/bas.phtml.

[18] DVB. DVB Homepage : http ://www.dvb.org.

[19] E.Duros, W.Dabbous, H.Izumiyama, Y.Zhang. A link layer tunneling mechanism for unidirectionnal links. *RFC 3077*, 2001.

[20] ETSI. ETSI Homepage : http ://www.etsi.org.

[21] Ernesto Exposito. *"Spécification et mise en oeuvre d'un protocole de transport orienté Qualite de Service pour les applications multimedias"*. PhD thesis, These de doctorat de Institut National Polytechnique de Toulouse, 2003.

[22] Slim Abdellatif Thierry Gayraud Frédéric Nivor, Pascal Berthou. Amélioration de l'Allocation Dynamique de Ressource dans un Système Satellite DVBS/RCS. *Colloque Francophone sur l'Ingénierie des Protocoles - CFIP 2006*, Tunisie 2006.

[23] FSMLabs. Linux rt : http ://www.fsmlabs.com/.

[24] Sachin Ganu, Haris Kremo, Richard Howard, and Ivan Seskar. Addressing Repeatability in Wireless Experiments using ORBIT Testbed. In *TRIDENTCOM 05 : Proceedings of the First International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities (TRIDENTCOM'05)*, pages 153–160. IEEE Computer Society, 2005.

[25] Emmanuel Jammeh Martin Fleury Martin J. Reed Gilberto Flores Lucio, Marcos Paredes-Farrera. Opnet modeler and ns-2 : Comparing the accuracy of network simulators for packet-level analysis using a network testbed. *ICOSMO*, 2 :700–707, 2003.

[26] Satlabs Group. Satlabs group : http ://satlabs.org/.

[27] S. Hemminger. `http://developer.osdl.org/shemminger/netem/`.

[28] D. Herrscher, A. Leonardi, and K. Rothermel. Modeling Computer Networks for Emulation. In *Proceedings of PDPTA '02*, pages 1725–1731, June 2002.

[29] D. Herrscher and K. Rothermel. A Dynamic Network Scenario Emulation Tool. In *Proceedings of ICCCN 2002*, pages 262–267, October 2002.

[30] Daniel Herrscher and Kurt Rothermel. A Dynamic Network Scenario Emulation Tool. In *Proceedings of the 11th International Conference on Computer Communications and Networks (ICCCN 2002)*, pages 262–267, Miami, October 2002.

[31] Herve Thalmensy, Laurent Dairaine, Florestan de Belleville. Mesures DIPCAST sur la plateforme avancee : adéquation MITV et DIPCAST. Technical report, Rapport de contrat DIPCAST, Aout 2003.

[32] Herve Thalmensy, Laurent Dairaine, Florestan de Belleville. Mesures DIPCAST sur la plateforme de base : adéquation MITV et DIPCAST. Technical report, Rapport de contrat DIPCAST, Juillet 2003.

[33] David B. Ingham and Graham D. Parrington. Delayline : A wide-area network emulation tool. *Computing Systems*, 7(3) :313–332, 1994.

[34] *Basic Reference Model for Open System Interconnection.* Int. Standards Org., 1984. ISO 7498.

[35] ISO/IEC. Mpeg-2 Generic coding of moving pictures and associated audio information. *ISO/IEC 13818*, 1996.

[36] G. Jourjon P. Casenove F. Tan E. Exposito E. Lochin L. Dairaine, E. Exposito. IREEL : Remote Experimentation with Real Protocols and Applications over Emulated Network. In *Proceedings of ACM ITICSE 06 poster session*, 2006.

[37] Laurent Lancerica, Laurent Dairaine, Florestan de Belleville, Herve Thalmensy, and Christian Fraboul. Mitv - a solution for an interactive tv based on ip multicast over satellite. In *ICME*, pages 2159–2162. IEEE, 2004.

[38] LIBPCAP. Tcpdump/libpcap. 2002. The Tcpdump Group.

[39] L.Peterson and al. A blueprint for introducing disruptive technology into the internet. In *ACM HotNets-I Workshop*, 2002.

[40] M. Gineste, P. Sénac. Quality of service solutions for satellite communication. *ICN*, April 2005.

[41] D. Mahrenholz and S. Ivanov. Real-Time Network Emulation with ns-2. In *In proceedings of The 8-th IEEE International Symposium on Distributed Simulation and Real Time Applications*, 2004.

[42] M.Allman and S.Ostermann. One : The ohio network emulator. Technical report tr-19972, computer science, Ohio University, 1997.

[43] M.Carson and D.Santay. Nist net : A linux-based network emulation tool. *ACM SIGCOMM Computer Communications Revieuw*, 33 :111–126, 2003.

[44] L.Dairaine P.Senac M.Diaz M.Gineste, H.Thalmensy. Active emulation of a dvb-rcs satellite link in an end-to-end qos-oriented heterogeneous network. In *ICSSC*, 2005.

[45] N.Djurak M.Mikuc, Z.Puljiz and M.Zec. Imunes : An integrated multiprotocol network emulator/simulator. http ://www.tel.fer.hr/imunes, 2005.

[46] Nera. Nera asa : www.nera.no/.

[47] OPNET. Opnet technologies homepage : http ://www.opnet.com.

[48] C.Lohr J.P Courtiat P. de Saqui-Sannes, L.Apvrille. Turtle : Timed UML and RT-Lotos Environnement.

[49] K. Pawlikowski, H.-D.J. Jeong, and J.-S. Ruth Lee. On Credibility of Simulation Studies of Telecommunication Networks. *IEEE Communications Magazine*, pages 132–139, January 2002.

[50] Vern Paxson and Sally Floyd. Why we don't know how to simulate the internet. pages 1037–1044, 1997.

[51] Qualnet. Qualnet. scalable network technologies. http ://www.scalable-networks.com/, 2001.

[52] Kishore Ramachandran, Sanjit Kaul, Suhas Mathur, Marco Gruteser, and Ivan Seskar. Towards Large-Scale mobile network emulation through spatial switching on a wireless grid. In *Proceedings of the ACM SIGCOMM 2005 Workshop on experimental approaches to wireless network design and analysis (E-WIND-05)*, Philadelphia, PA, 2005.

[53] RENATER. Renater homepage : http ://www.renater.fr.

[54] L. Rizzo. Dummynet : A Simple Approach to the Evaluation of Network Protocols. *ACM Computer Communication Review*, 27(1) :31–41, January 1997.

[55] R.Olsson. Pktgen the Linux Packet Generator. In *Proceedings of the Linux Symposium*, July 2005.

[56] RTAI. Rtai : https ://www.rtai.org/.

[57] S. Sanghani, T. Brown, S. Bhandare, and S. Doshi. Ewant : Emulated wireless ad hoc network test-bed, 2003.

[58] Satlynx. Satlynx : http ://www.satlynx.com/.

[59] EMS TECHNOLOGIES. Ems : http ://www.ems-t.com/.

[60] R.Van Mook M.Van Oosterhout P.Schroeder J.Spaans T.Graf, G.Maxwell and P.Larroy. Lnux advanced routing & traffic control. http ://lartc.org, 2005.

[61] L. Usnr. Mgen. http ://cs.itd.nrl.navy.mil/work/mgen/index.php.

[62] Nicolas Van Wambeke, François Armando, Christophe Chassot, and Ernesto Exposito. A model based approach to communication protocol's self-adaptation. In *MUE*, pages 1024–1029, 2007.

[63] S. Y. Wang, C. L. Chou, C. H. Huang, C. C. Hwang, Z. M. Yang, C. C. Chiou, and C. C. Lin. The design and implementation of the NCTUns 1.0 network simulator. *Computer Networks*, 42(2) :175–197, 2003.

[64] S. Y. Wang and Y. B. Lin. NCTUns Network Simulation and Emulation for Wireless Resource Management. *Wireless Communications and Mobile Computing*, 5(8) :899–916, December 2005.

[65] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. An Integrated Experimental Environment for Distributed Systems and Networks. In *Proceedings of OSDI02*, 2002.

[66] R.Sharma X.W.Huang and S.Keshav. The entrapid protocol development environment. In *IEEE Infocom*, 1999.

[67] Xiang Zeng, Rajive Bagrodia, and Mario Gerla. Glomosim : A library for parallel simulation of large-scale wireless networks. In *Workshop on Parallel and Distributed Simulation*, pages 154–161, 1998.

[68] P. Zheng and L.M. Ni. EMPOWER : A Network Emulator for Wireless and Wireline Networks. In *Proceedings of IEEE INFOCOM 2003*, 2003.

Emulation de réseaux au niveau IP pour l'expérimentation de services et de protocoles de communication. Application aux réseaux satellites

Dans le cadre de l'évaluation de protocoles, plusieurs techniques sont utilisées : l'expérimentation en environnement réel, la simulation et l'émulation.

Dans cette thèse, l'approche d'émulation est étudiée car elle permet la mise en œuvre d'applications ou de protocoles réels dans un environnement contrôlable dont les caractéristiques (bande passante, délais, pertes) reproduisent en temps réel le comportement d'un réseau cible pour étudier et comparer différents protocoles et applications.

Cette thèse proposera des techniques basées sur le couplage d'approches telles que la métrologie ou la simulation pour reproduire efficacement et de manière flexible le comportement de différents types de réseaux.

Au cours de ce travail, nous avons participé à différents projets (DIPCAST et EuQoS) pour l'émulation de contexte satellite, et proposés une architecture d'émulation pour un lien satellite DVB-RCS ainsi qu'une méthodologie d'émulation pour l'émulation de réseaux à qualité de service.

Mots clés : émulation, DAMA, réseau satellite, émulation active

An IP level network emulation solution for services and protocols evaluation applied on satellite networks

In the context of protocol evaluation, several solutions are used: live tests, simulation and emulation.

We focus on network emulation which allows the use of real applications or protocol implementations in a controllable environment. For this purpose, several characteristics (bandwidth, delay, losses) allow the real time reproduction of a target network behavior. In this thesis, we propose solutions based on metrology and simulation to reproduce in a flexible but efficient way the behavior of different network types such as the satellite one. We were involved in several projects (DIPCAST and EuQoS) for emulation of a satellite environment and we proposed an architecture for emulating the DVB-RCS satellite link. A methodology for quality of service network is also proposed.

Keywords: emulation, DAMA, satellite network, active emulation