Formalisation et test d'IPv6

Abdelghani Benharref, Zineb Berbich, Mohamed Salah Bouassida, Rachida Dssouli, Isabelle Chrisment

Formal Specification and Testing IPv6 Protocols

Formalisation et test d'IPv6

A. Benharrefsfn] , Z. Berbich , M.S. Bouassida , I. Chrisment, R. Dssouli

Octobre 2003

Formalisation et test d'IPv6

A. Benharref[*] [†]sfn] , Z. Berbich[‡], M.S. Bouassida[§], I. Chrisment[¶], R. Dssouli[‖]

Projets MADYNES et Concordia University (Canada)

Rapport de recherche
Octobre 2003
20 pages

Résumé : Ce document présente les activités de recherche qui ont été menées conjointement entre le projet MADYNES du LORIA et Concordia University de Montréal dans le cadre d'une coopération financée par le FFCR (Fonds France-Canada pour la Recherche). Ces activités de recherche portent sur la formalisation et le test des protocoles IPv6.

Mots-clé : méthodes formelles, SDL, modèlisation, conformité IPv6 et tests d'interopérabilité, MLD, Multicast, Réseaux actifs)

(Abstract: pto)

[*] Concordia University,Canada
[†] [
[‡] Concordia University,Canada
[§] LORIA
[¶] LORIA
[‖] Concordia University, Canada

LORIA
Campus scientifique
BP 23, 54506 VANDŒUVRE LÈS NANCY (France)
Téléphone : 03 83 59 30 30 - International : +33 3 3 83 59 30 30
Télécopie : 03 83 27 83 19 - International : +33 3 83 27 83 19

Formal Specification and Testing IPv6 Protocols

Abstract: This document presents research activities achieved between MADYNES project from LORIA and Concordia University from Montreal, within the framework of a cooperation funded by FFCR (Fonds France-Canada for Research). These research activities are related to formalizing and testing IPv6 protocols.

Key-words: Formal methods, SDL, modeling, IPv6 conformance and interoperability testing, MLD, Multicast, Active Networks

1 Introduction

Dans ce document, nous présentons le projet de coopération entre MADYNES et Concordia University. L'objectif principal du projet est la génération automatique des suites de tests de conformité et d'interopérabilité des protocoles d'Internet nouvelle génération plus connue sous le vocable IPv6 [?]. Le projet se compose de plusieurs étapes:

1. le développement de spécifications formelles en SDL [?] et leur validation ainsi que la définition de l'architecture de test ;
2. la génération des suites de test pour toutes les composantes d'interface (réseau et application) pour l'hôte et le routeur et pour une sélection limitée de configuration d'environnements ;
3. l'expérimentation sur les réseaux français et canadiens.

Développement et validation de spécifications formelles La modélisation des diverses composantes de la suite de protocoles IPv6 a suivi la décomposition fonctionnelle. On a modélisé les protocoles suivants : IPv6, ICMPv6, NDP, MLD et le Routage [?] [?] [?] [?]. Le découpage fonctionnel donne toutes les composantes d'interfaçage direct et indirect qu'il est important de spécifier et de valider. L'avantage de ce découpage est qu'il permet la réutilisation individuelle des composantes en vue de leur intégration dans des environnements différents. Toutes les composantes sont spécifiées en SDL [?] et validées dans le cadre de ce projet de collaboration. L'étape de modélisation est la plus exigeante en terme de travail. C'est une étape manuelle. Ensuite les spécifications formelles sont validées avec les outils existants. La méthodologie suivie se résume comme suit : extraction des informations des RFCs et des normes existantes, modélisation sous forme d'automates à états finis étendus par les variables (qui est le modèle de SDL), description de l'architecture du système à modéliser et du comportement des protocoles en langage formel SDL pour obtenir des spécifications. Les spécifications sont alors validées via la simulation et l'application de scénarios de test. Enfin on génère des suites de test automatiquement en utilisant les outils commerciaux.

La génération automatique des suites de test La génération automatique de test consiste à développer et à utiliser des algorithmes de dérivation de test à partir de spécifications formelles exécutables. Comme nous avons modélisé certains protocoles en SDL qui est un langage de description formel exécutable, nous avons tout simplement utilisé les outils de Telelogic (Suite SDL de TAU et ObjectGeode [?]). Les cas de test sont générés par objectif et selon la couverture automatique.

L'expérimentation La première phase de l'expérimentation consistait en la comparaison entre les cas de test obtenus manuellement et les cas de test obtenus via la génération automatique à partir des spécifications développées par l'équipe de Montréal. L'équipe de Nancy avait fait un travail de développement d'environnement appelé FLAME [?] pour lequel certains cas de test sont générés manuellement. Cette étape est également la validation

des spécifications formelles en comparaison avec les cas de test qui ont déjà servi sur une plate-forme existante. La deuxième serait de compléter la génération automatique des tests d'interopérabilité et de les appliquer à l'environnement FLAME.

Les deux équipes de recherche ont fait un travail que l'on peut qualifier de complémentaire et ont développé une expertise dans le test distribué, le test fondé sur les modèles tels les automates à états finis, les automates étendus par les variables et le temps.

L'équipe de Montréal composée de Rachida Dssouli (professeur), Abdelghani Benharref (étudiant Ph.D.), Zineb Berbich et Huang Zheing (étudiantes MSc) a travaillé sur la partie en amont en optant pour l'utilisation d'une méthodologie de génération automatique des suites de test fondée sur les techniques formelles. Trois étudiants ont participé au projet, ils ont modélisé plusieurs protocoles de la pile IPv6, développé les spécifications correspondantes en SDL et généré automatiquement les cas de tests.

L 'équipe de Nancy composée d'André Schaff (professeur), Isabelle Chrisment (maître de Conférence), Olivier Festor (chargé de recherche INRIA), Ghassan Cheddoud (doctorant), Isabelle Astic (ingénieur expert), et de Mohamed Salah Bouassida (étudiant MSc) a concentré ses efforts sur la partie avale par la mise en œuvre des test dans un environnement actif, qui est un environnement d'exécution d'IPv6 appelé FLAME. L'équipe a ainsi développé manuellement des cas de tests pour certains protocole de la pile IPv6.

Dans la suite du document, nous reprenons plus en détail chacune des trois étapes réalisées dans le cadre du projet.

2 Développement et validation de spécifications formelles

2.1 Modélisation avec SDL, vérification et validation de spécifications formelles

Dans cette section, nous expliquons la méthodologie et les techniques utilisées pour la formalisation, la vérification et la validation de modèles et la génération de cas de tests. Les phases importantes sont données dans les sections suivantes.

2.2 Extraction des informations techniques

Tous les protocoles Internet (IPv4 et IPv6) sont décrits en détail dans des documents RFC (Request For Comment) émanant d'organismes de normalisation. Les implantations doivent refléter au moins les parties obligatoires qui sont qualifiées par "must" ou "should" de leurs RFCs; ces informations doivent être extraites correctement et vérifiées avec précaution. Toute mauvaise interprétation à ce stade peut conduire à une implantation erronée. Pendant cette première étape de modélisation plusieurs ambiguïtés et/ou incomplétude d'informations vont apparaître. En général ce sont des parties de la norme (spécifications) laissées aux soins des développeurs. Ces parties ou problèmes doivent être résolus avant de progresser dans la

formalisation. À la fin de cette phase, le comportement extrait est représenté sous une forme semi-formelle tels que des tables, des systèmes à transitions, FSM, Petri Net.

2.3 Construction d'automates à états finis étendus par les variables (EFSMs)

Dans cette phase, les spécifications semi-formelles obtenues lors de la première phase sont traduites en un ensemble d'automates à états finis (Extended Finite State Machine) [**?**]. Les EFSMs ainsi obtenues sont enrichies ou augmentées par les définitions de contraintes des protocoles. Cet enrichissement peut inclure l'ajout des variables d'états à chaque automate et la définition des messages échangés entre les EFSMs tels que donnés dans les RFCs.

2.4 Description en SDL

SDL est un langage de description formelle fondé sur le modèle des EFSMs. Le passage des EFSMs vers SDL est en effet très simple et le format graphique de SDL facilite la traduction. Cependant, certains aspects doivent être définis pour la première fois à ce stade de traduction. La décomposition du système en modules afin de déterminer son architecture, les structures de données, les temporisateurs, les signaux et leurs paramètres doivent être sélectionnés et les processus (les composantes finales contenant le comportement) doivent être codés. Le niveau d'abstraction est déterminé pour chaque module en fonction des contraintes du langage et du niveau de détail désiré.

Dans ce projet nous avons utilisé l'environnement de composition de spécifications SDL connu sous le nom de SDL editor de l'outil ObjectGeode commercialisé par Telelogic. Nous avons également utilisé le simulateur de cet environnement pour la vérification de la syntaxe et les tests unitaires des spécifications développées.

2.5 Vérification et validation des modèles SDL

Afin de s'assurer que le modèle SDL a bien été créé correctement, le modèle doit être vérifié de telle sorte à capturer et corriger toutes les erreurs du type étreintes fatales "deadlock", boucles sans progression "Livelock" et autres erreurs. Ensuite le système modélisé doit subir le test de régression afin de s'assurer que des erreurs n'ont pas été introduites lors des corrections et que les fonctionnalités spécifiées n'ont pas été altérées. L'usage du simulateur SDL est important dans cette phase. La validation tente de répondre à la question suivante : a-t-on construit le bon modèle? Est-ce que le modèle reflète bien ce qui est décrit dans les RFCs? Dans cette phase l'usage du simulateur combiné avec les scénarios a été important. La simulation permet de comparer les comportements spécifiés via les scénarios et de les comparer avec les réponses du système. Toute ereur détectée est indiquée par le simulateur. Les scénarios sont extraits des tests générés manuellement par l'équipe du LORIA.

3 La génération automatique des suites de test

3.1 Génération des suites de tests TTCN

Après la validation des modèls SDL, nous avons utilisé le générateur "TestComposer", une partie de l'outil ObjectGeode, pour générer la suite de tests TTCN [?]. C'est un outil de génération de test pour la conformité conçu pour générer, automatiquement ou interactivement, des suites de tests pour une Implémentation Sous Test (IUT ou Implementation Under Test) et l'architecture de tests [?]. TestComposer prend en entrée un modèle SDL, une spécification de l'environnement de test et un ensemble éventuellement vide des objectifs de tests définis par l'utilisateur. Ces objectifs de test peuvent être construits interactivement en utilisant le simulateur SDL, ou écrits en utilisant les MSCs ou les obervateurs. Cet outil complète l'ensemble des objectifs de tests par des objectifs de tests additionnels sur la base de la couverture. Ces suites de tests générées sont stockées dans une base de données.

3.2 Exemple de spécification et suite de test MLD

Dans cette section, nous présentons rapidement la spécification SDL pour MLD, donnons un exemple de suite de tests TTCN générées par ce protocole et les étapes nécessaires pour fournir les suits de tests exécutables en langage C dans une plateforme active [?].

3.2.1 Spécification

Le protocole choisi pour les tests est le protocole MLD : *Multicast Listeners Discovery* [?], c'est un protocole de gestion de groupes multicast, de niveau lien local qui permet à un routeur de détecter la présence de nœuds sur l'ensemble de ses liens désirant recevoir des paquets relatifs à un groupe multicast. Le protocole MLD permet ainsi de tenir à jour, continuellement, la liste des adresses multicast disposant de *listeners*. Cette information est utilisée par les protocoles de routage multicast pour pouvoir acheminer les paquets vers les nœuds intéressés.

MLD est utilisé par les routeurs pour trouver les membres sur leurs liens locaux. Il doit être implémenté à la fois dans les routeurs et dans les nœuds qui veulent recevoir des paquets multicast. Les nœuds ne répondent pas aux requêtes multicast des autres nœuds, les routeurs le font. Si un routeur écoute sur des adresses multicast, il doit répondre à ces requêtes. Le système SDL de MLD consiste en deux blocs, de définition de structure de données, de signaux et d'un ensemble de procédure. Le block NodeBlock contient le processus définissant le comportement d'un simple nœud et le bloc RouterBlock contient le processus spécifiant le comportement d'un routeur.

3.2.2 Exemple de suite de test TTCN

Une des propriétés fondamentales qui devrait être satisfaite par n'importe quelle implémentations MLD est que seul un routeur par lien local, à un moment donné, devrait être sélectionné pour répondre aux requêtes multicast. Le routeur sélectionné (appelé "Querier")

Comportement	Description	Commentaires
?Query		Envoi d'un message "Query"
!Report		Réception d'un message "Report"
!Query		Réception d'un message "Query" d'un routeur avec une adresse IP
		Le routeur doit arrêter d'envoyer des messages "Query"
?Query	F	Erreur

FIG. 1 – *Code TTCN*

est celui avec l'adresse IP la plus petite. Si un routeur, dans l'état "Querier", reçoit une message Query d'un autre routeur avec une autre IP plus petite, il devra passer dans l'état "Non Querier". La tableau de la Figure 1 présente un résumé de la partie dynamique du code TTCN correspondant à ce scénario.

3.2.3 Exemple de suites de tests exécutables

Les suites de tests sont générées manuellement de TTCN vers C et exécutées dans un environnement actif appelés FLAME [**?**]. L'approche active permet le dynamisme et la décentralisation des tests nécessaires pour les fonctionnalités IPv6 (multicast, mobilité) où la stimulation et les événements peuvent se produire n'importe où. Dans cet environnement actif, les suites de test sont chargées dans un "Tester Node" qui exécutera la suite de test et rapportera le résultat observé. Le même test décrit dans le paragraphe précédent est reproduit ici. La procédure test consiste dans les étapes suivantes:

- telnet sur le "Tester Router" (TR).
- Le TR charge le code de la procédure test.
- Le TR envoie un message "General Query" au routeur sous test (RUT) avec une adresse IP inférieure à celle utilisée par le RUT.
- Si le TR reçoit encore un message "General Query" du RUT, le test échoue.
- Affichage d'un message d'erreur

4 Expérimentation

Parallélement aux spécifications et à la génération automatique de tests, nous avons réalisé manuellement les mêmes scénarios relatifs au protocole MLD dans un environnement d'exécution de tests

4.1 Introduction

La mise au point de tests sur le protocole MLD pouvait être élaborée suivant deux approches : approche non active et approche active.

Approche non active

L'approche non active est basée sur les réseaux traditionnels qui comportent un nombre restreint et fixe de services implantés dans les équipements et qui n'offrent aucun moyen d'en ajouter de nouveaux. En conséquence, l'approche non active ne permet pas de modifier dynamiquement le comportement global du réseau. Ainsi, des tests MLD réalisés suivant l'approche non active seront statiques et ne présenteront au testeur aucun aspect dynamique. Chaque scénario de test doit être manuellement installé sur les machines choisies et le testeur ne pourrait en aucun cas superviser tous les tests via une seule machine console.

Approche active

L'approche active se base sur le fait que tout ou une partie des composants d'un réseau actif dans les différents plans (signalisation, supervision et données) sont programmables dynamiquement par des entités tierces (opérateur, fournisseur de services, applications, usagers). Ainsi, la contribution d'une infrastructure active pour la réalisation des tests MLD présente des avantages :

- Le déploiement dynamique du code. En effet, le code des différents tests peut être téléchargé d'une machine console et exécuté sans aucune installation manuelle.
- La capacité de télécharger les scénarios de tests à n'importe quel endroit du réseau.
- La souplesse et facilité de réalisation des tests du protocole pour l'administrateur testeur, en effet, à partir d'une machine console, il peut superviser tous les scénarios de tests qu'il a mis en place à n'importe quel endroit du réseau.
- Un test MLD aura la forme d'une sonde programmable qui exécute le scénario de test chargé d'une machine de contrôle, obtient les résultats du test, les affiche sur l'écran du même hôte de contrôle et enfin s'achève.

Les avantages de l'approche active par rapport à l'approche traditionnelle pour la réalisation des tests de MLD nous ont conduit à choisir l'approche active. Plusieurs travaux de recherche ont été menés pour concevoir des architectures de réseau actif, parmi ces architectures on peut citer : SmartPackets développée chez BBN, Switchware développée chez SMI et ALE, Active IP développée chez MIT, ANTS suite de Active IP. Toutes ces architectures sont présentées dans [**?**]

4.2 L'environnement Flame

L'environnement d'exécution qui a été utilisé pour mettre en place les scénarios de tests MLD est l'environnement FLAME qui est un environnement d'exécution actif au sein de l'équipe MADYNES.