Vérification et analyse de performance d'un protocole de gestion de session

Dario Vieira, Ana Cavalli

Vérification et analyse de performance d'un protocole de gestion de session

Dario Vieira et Ana Cavalli

Laboratoire Samovar (CNRS) and GET/INT
9, rue Charles Fourier
91011 Evry Cedex - France
{*ana.cavalli, dario.vieira*}*@int-evry.fr*

RÉSUMÉ. *Le Managed Session Protocol (MSP) est un nouveau protocole de gestion de session qui nous avons développé en collaboration avec INTEL-Cambridge [CAM]. MSP est exécuté au dessus de la couche d'un protocole de transport et il offre des services telles que la diffusion (broadcasting), le multi-streaming, les communications multi-cibles (multi-homing) et le multi-session. Ce protocole peut être utilisé par différentes applications, par exemple le protocole BGP peut bénéficier des fonctionnalités offertes par MSP. Cet article présente MSP, ainsi que les techniques de vérification formelle et d'analyse de performance, mises en place pour, d'une part, valider les propriétés de correction de MSP, et d'autre part, valider la mise en oeuvre de MSP. Pour la vérification, nous avons utilisé la technique de model checking et l'outil Simple Promela Interpreter (SPIN). En outre, nous avons employé la méthode de la logique temporelle pour la conception des invariants et des propriétés de MSP. Pour l'analyse de performance, nous avons utilisé la mise en oeuvre de MSP dans l'eXtensible Open Router Platform (XORP) [HAN 05] en utilisant BGP comme application client.*

ABSTRACT. *Managed Session Protocol (MSP) is a new session maintenance protocol that we have proposed with collaboration of INTEL-Cambridg. MSP operates on top of the Transmission Control Protocol (TCP). Furthermore, Multiple Managed Session Protocol (MMSP) is sub-protocol that enhances MSP functionalities. MSP/MMSP provides services such as broadcasting, multi-streaming, multi-homing and multi-session. In this paper, it is proposed and carried out formal verification techniques to validate the correctness of MSP. It is applied model checking techniques by using Simple Promela Interpreter (SPIN) model checker in order to verify different correctness claims. Besides, it is made use of temporal logic so as to design and establish invariance and liveness properties of MSP.*

MOTS-CLÉS : *Vérification, SPIN, Logique Temporelle, Analyse de Performance*

KEYWORDS: *Verification, SPIN, Temporal Logic, Performance Analysis*

L'objet – 8/CFIP 2006, pages 1 à 12

1. Introduction

Le Managed Session Protocol est un nouveau protocole de gestion de session. Il fournit des services tels que la détection de l'échec de connexion et la remise en marche automatique de la session de transport. Visant à fournir certains services tels que le multi-session, le multi-homing, le multi-streaming et le broadcasting, nous proposons également un sous protocole, qui est exécuté au dessus du MSP, que nous appelons Multiple Managed Session Protocol (MMSP). Le Border Gateway Protocol (BGP) est une application, parmi les différentes applications potentielles, qui peut utiliser les propriétés du MSP. Par exemple, les hôtes BGP peuvent utiliser MSP afin de créer automatiquement des sessions multiples entre leurs pairs.

Malgré l'importance pratique de MSP, il n'a jamais été formellement vérifié. Dans cet article, nous avons mis en route les premiers essais afin d'accomplir la vérification formelle du MSP. Le model checker de SPIN [HOL 03] a été utilisé pour vérifier MSP. D'ailleurs, le protocole et l'heuristique d'inférence sont spécifiés en utilisant le langage PROcess MEta LAnguage (PROMELA), où un simulateur de protocole peut effectuer des simulations aléatoires ou guidées. SPIN exécute une recherche approfondie pour vérifier qu'une propriété donnée est valide sous toutes les simulations possibles du système. Dans le modèle qui est présenté dans cet article, toutes les propriétés ont été codées en utilisant la Logique Temporelle Linéaire (LTL). Après, elles sont traduites en PROMELA. L'utilisation de LTL a comme avantage la possibilité d'employer n'importe quel model checker pendant la phase de vérification.

Par ailleurs, la proposition d'un nouvel protocole de session pose la question de sa performance. Pour répondre à cette question, nous avons fait la mise en oeuvre de MSP en XORP, et puis nous avons exécute des expériences en utilisant XORP et l'*Integrated Multiprotocol Network Emulator/Simulator* (IMUNES) [ZEC 04].

Cet article est organisé comme suit. Dans la section 2, les principales définitions et caractéristiques du MSP sont introduites. La section 3 introduit les principales définitions du model checking et du SPIN-PROMELA. Cette section présente aussi un bref aperçu des travaux menés dans le domaine de la vérification et l'analyse de performances des protocoles. Puis, il est décrit comment les différents composants du système sont mis en application et effectuent la vérification en utilisant SPIN. Les propriétés qui ont été conçues dans LTL sont également présentées. Dans la section 4, la mise oeuvre de MSP en XORP est introduite. Ensuite, dans la section 5, il est décrit l'expérience qu'a été fait avec MSP et BGP en utilisant XORP et IMUNES. Enfin, la section 7 présente les conclusions et les futures directions de ce travail.

2. Préliminaires

Dans cette section sont présentés les principaux concepts et propriétés de MSP. Tout d'abord, nous introduisons la motivation de la définition de MSP.

2.1. *Contexte*

Le besoin d'un protocole tel que MSP, est apparu suite à une analyse détaillé du Border Gateway Protocol (BGP). En fait, le RFC de BGP définit un sous-protocole qui met en application un type de gestion de session réseau. Ce sous-protocole est employé pour maintenir une session réseau bidirectionnel sans erreur sur laquelle des messages sont échangés. En effet, ce sous-protocole résout un problème très général, et il contient très peu de traits relatifs à BGP. Il pourrait être traité comme un protocole indépendant. Ainsi, ce fait nous a motivé pour concevoir un nouveau protocole appelé Managed Session Protocol (MSP).

Cependant, nous avons conçu MSP en ayant comme objectif de proposer un protocole de gestion réseau plus général, comme il est illustré par la Figure 1, de telle manière que d'autres applications (par exemple, des protocoles de signalisation, ou des applications de synchronisation de bases de données) puissent se servir des fonctionnalités de MSP.

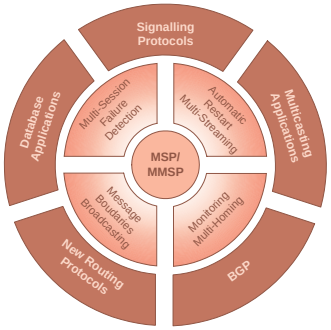

Figure 1. *MSP : Un Protocole de Gestion Réseau plus Général*

Afin de répondre aux exigences d'extensibilité, l'architecture de MSP est composée de deux couches, comme représenté par la Figure 2. Une couche est composé de MSP lui-même, qui fournit des services tels que la détection d'échec de connexion et la reprise automatique de la session de transport, et l'outre couche est composé du Multiple Managed Session Protocol (MMSP) qui fournit des services tels que la multi-session, le multi-homing et la diffusion.

2.2. *Managed Session Protocol*

Le *Managed Session Protocol* (MSP) est un nouveau protocole de gestion de session qui fournit de connexions de réseau transparentes aux applications ordinaires en utilisant seulement les mécanismes des espaces utilisateurs. Au demeurant, MSP est exécuté au dessus de la couche transport. La Figure 3 illustre l'architecture de MSP.

MSP peut offrir les services suivants :

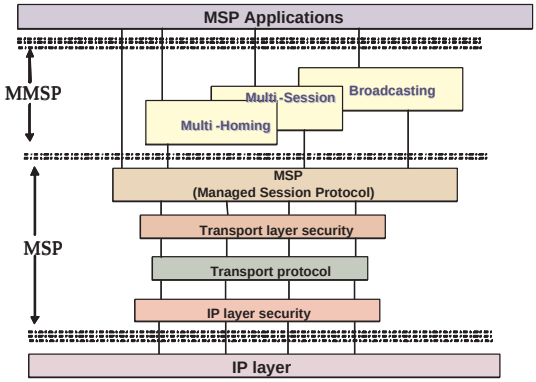

Figure 2. *MSP Logical Components*

√ La remise en marche automatique . Quand un échec de connexion a lieu, MSP peut automatiquement remettre en marche la session de transport avec l'hôte. Cette propriété de MSP peut réduire le délai lié à un hôte qui a échoué.

√ Détection d'échec. Le mécanisme de détection d'échec du MSP est basé sur *Bidirectional Forwarding Detection* (BFD) [KAT 05]. De ce fait, MSP peut vérifier non seulement l'accessibilité physique mais également l'état opérationnel du plan de contrôle.

√ Surveillance de l'état. Les applications sont informées sur tous les changements de la session de transport comme quand un hôte vient de se connecter ou d'échouer, ou quand une connexion est demandée par d'autres hôtes.

√ Conservation des frontières des messages d'application. Cela est utile quand les données d'applications ne sont pas en flot continu mais arrivent en flots de données logiques que le récepteur manipule séparément.

Le *IP security protocol suite* (IPsec) [KEN 98] fournit les services de sécurité requis par les hôtes MSP pour vérifier l'intégrité des messages, l'identité des émetteurs, et le fait que le récepteur MSP est réellement le destinataire prévu de chaque message. En outre, des fonctionnalités telles que le contrôle de congestion et de flux, la découverte de chemin et la fragmentation de données des applications, sont manipulées par la couche d'un protocole de transport au lieu du MSP.

2.3. *Le protocole Multiple Managed Session Protocol (MMSP)*

MMSP est exécuté au-dessus du MSP comme l'illustre la Figure 3. Ses principales fonctionnalités peuvent être décrites comme suit :

√ La propriété *Multi-streaming*. MMSP permet la livraison de plusieurs flots de données indépendants sur une connexion. Ceci permet un schéma de livraison où

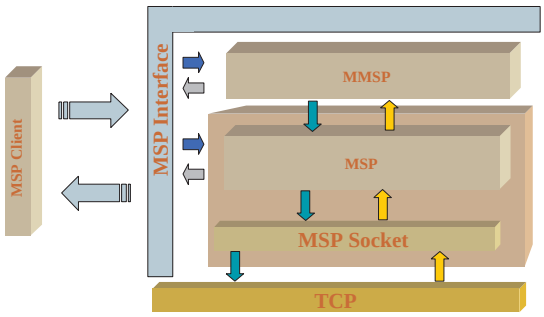

Figure 3. *Managed Session Protocol*

l'ordre des messages est seulement maintenu par flots pour réduire Head-Of-Line (HOL).

√ La propriété *Multi-homing*. MMSP fournit un support transparent aux communications entre deux hôtes dont un ou tous les deux sont *multi-homed*. Ces possibilités peuvent être employées pour construire des chemins redondants entre deux hôtes.

√ La propriété *Broadcasting*. MMSP fournit le support transparent à la diffusion. En conséquence, les applications peuvent par exemple, envoyer des messages spécifiques pour un groupe particulier d'hôtes comme l'illustre la Figure 4.

√ La propriété *Multi-session*. MMSP fournit le support à *multi-session*. En conséquence, MMSP permet à des sessions multiples d'être automatiquement créées sans overhead de configuration additionnel ou utilisation d'adresses additionnelles.

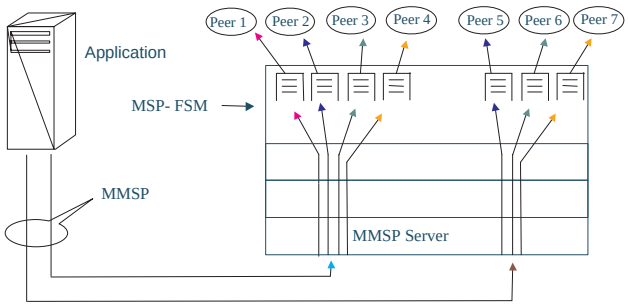

Figure 4. *Multiple Managed Session Protocol (MMSP)*

3. Model Checking

La mise en oeuvre de la vérification des propriétés des protocoles de communication réseau est une tâche laborieuse. De nombreuses méthodes sur la vérification formelle pour les protocoles de communications réseau ont déjà été publiées. Les

principales techniques utilisées pour l'analyse des systèmes complexes sont la simulation, le test, la vérification déductive et le model checking. La simulation et le test comprennent la mise en oeuvre des expérimentations avant le déploiement des systèmes, tandis que la vérification déductive se rapporte habituellement à l'utilisation des axiomes et des règles pour valider la correction des systèmes. Le model checking est une technique automatique très utilisée pour la vérification des systèmes distribués à états finis.

Le model checking a été employé avec succès dans la pratique pour vérifier des protocoles de communication réseau. Il y a beaucoup d'outils qui peuvent être utilisés afin d'effectuer la vérification telle que *C Model Checker* (CMC) [MUS 04] et (SPIN) model checker [HOL 03]. Par exemple, CMC est employé dans [MUS 04] afin de vérifier la mise en oeuvre du protocole de routage *Ad hoc On Demand Distance Vector* (AODV). [DON 03] se servent de SPIN pour vérifier certaines propriétés des protocoles. Il y a aussi, en outre, beaucoup de méthodes dans la littérature qui emploient une combinaison de model checking et de raisonnement formel pour accomplir la vérification formelle des protocoles. [SMI 02] emploient une méthode formelle de raisonnement afin de prouver des propriétés des protocoles transaction-TCP et TCP-SACK, tandis que [BHA 02] emploie une combinaison de SPIN et de raisonnement formel pour vérifier des propriétés des protocoles de routage du type vecteur distance.

En conséquence, la description du comportement et des corrections du MSP ont été codés en utilisant PROMELA [HOL 03]. Au demeurant, la vérification formelle des propriétés de la sûreté et de vivacité a été mise en route en employant le model checker SPIN [HOL 03] qui examine exhaustivement tous les comportements permis par MSP.

3.1. *Exigences de correction*

Une formule de logique temporelle se compose de prédicats, d'attributs, d'opérateurs booléens ($\vee, \wedge, \neg, \Rightarrow, \Longleftrightarrow$), d'opérateurs de quantification ($\forall, \exists$) et d'opérateurs temporels tels que $\Box$ (toujours), $\diamondsuit$ (un jour), $\diamond$ (un jour passé), qui sont employés pour raisonner sur le passé et le futur.

Des propriétés des systèmes répartis exprimables en utilisant la langage de la logique temporelle peuvent être classifiées comme suit :

$\surd$ Des propriétés d'invariance, également appelées les propriétés de sûreté, peuvent être exprimées comme : "rien de mauvais ne se produit pendant l'exécution du système", c'est-à-dire, le système n'atteint jamais un état indésirable. En général, elles sont exprimées par des formules telles que : $\Box Q$ ou $\Box(p \hookrightarrow \Box q)$, qui peuvent être lues respectivement comme : "toute exécution du système vérifie Q" et "si P est vraie à l'état initial, alors Q est immédiatement vraie et elle restera vrai pendant le reste de l'exécution du système".

$\surd$ Des propriétés de vivacité. Ces propriétés, qui peuvent être formulées comme : "quelque chose de bon aura lieu pendant l'exécution du système", expriment des re-

lations causales et les propriétés dynamiques du système. En général, elles sont exprimées par des formules telles que : $(p \hookrightarrow \diamond q)$ ou $\Box(p \hookrightarrow \diamond q)$, qui peuvent être lues respectivement comme : "si P est vrai à l'état initial, alors Q sera vrai dans le futur" et "toute les fois où P est vrai, alors Q sera vraie dans le futur".

La logique temporelle a été employée pour indiquer et établir quelques propriétés remarquables d'invariance et de vivacité du protocole MSP. Les propriétés de correction doivent être formalisées afin de capturer toutes les caractéristiques critiques du MSP. On s'est particulièrement concentré sur les propriétés de propriétés de maintenance du MSP. Par exemple, la preuve de la correction du MSP implique d'établir les propriétés de sûreté et de vivacité décrites ci-dessous :

$\surd$ Propriété de Sûreté : si une session est ouverte entre deux hôtes, alors elle reste ouverte si des ressources sont disponibles et si les utilisateurs n'ont pas demandé de clore la session. Plus formellement : $\Box(P \hookrightarrow \Diamond(P \wedge Q \wedge \neg U))$

où P et Q représentent la propriété : "les hôtes P et Q ouvrent une session" et U la propriété : "les utilisateurs ont demandé de clore la session".

$\surd$ Propriété de vivacité : si certains utilisateurs demandent de clore une session ou quelques paquets ne remplissent pas les conditions de contrôle, alors la session sera close. Plus formellement : $(U \vee \neg CR \hookrightarrow \Diamond C)$ où U représente la propriété : les "utilisateurs désirent clore une session" et CR représente "les conditions de contrôle" et C "la fermeture de la session".

Les propriétés de sûreté et de vivacité garantissent que MSP satisfait également la propriété d'arrêt qui déclare qu'éventuellement aucune action dans MSP n'est disponible.

D'autres propriétés de vivacité qui ont été prouvées sont les suivantes :

$\surd$ Si la session est close parce que quelques paquets ne remplissent pas les conditions de contrôle, plus tard le mécanisme de remise en marche (restart mechanism) sera activé. Plus formellement : $(C \wedge \neg CR \hookrightarrow RS)$ où C et CR représentent la fermeture et les propriétés de contrôle et le RS représente la propriété de remise en marche. Cette propriété montre l'exactitude du mécanisme de remise en marche.

$\surd$ Il est toujours vrai que, si des hôtes A et B ne sont pas connectés, alors ils établiront une connexion dans le futur. Cette propriété peut être exprimée en LTL comme $(\Box((\Diamond\Box(!p1 \text{ and } !q1)) \hookrightarrow (\Diamond\Box(p2 \text{ and } q2))))$.

4. La mise en ouvre du protocole MSP

Pour la mise en oeuvre de MSP, nous avons divisé les tâches de fond en sous-processus, auxquels on associe des intervalles de temps limités. Par la suite, on exécute ces sous-processus de manière séquentielle, tout en gérant d'éventuelles priorités.

Afin de lancer une association MSP (ou une connexion MSP), les clients MSP indiquent une des adresses de transport qui définit l'hôte auquel ils veulent se conn-

necter. Un identificateur d'association, qui est traité localement par MSP, sera retourné si la connexion est etablie. Si MSP ne peut pas ouvrir une connexion avec l'hôte désiré, une erreur est retournée aux clients MSP.

Cet identificateur doit être fourni comme paramètre à chaque fois que des clients MSP demandent des services. Par exemple, si les clients MSP désirent envoyer des données vers d'autres hôtes, ils doivent fournir, en plus d'autres paramètres, l'identificateur de l'association qui sera employé par MSP pour envoyer les données. Prenons comme autre exemple le cas où des clients MSP désirent clore la connexion avec un hôte spécifique. Dans ce cas, ils doivent fournir à MSP l'identificateur de l'association qu'ils désirent clôturer.

Les clients MSP doivent pouvoir demander des services en passant des primitives à MSP et ils devraient pouvoir recevoir des notifications de MSP pour plusieurs événements. De ce fait, MSP fournit deux types d'interfaces à ses clients : des interfaces normales et des interfaces abstraites. Une interface normale est employée par des clients MSP afin de demander des services à MSP (par exemple, pour envoyer des données ou pour obtenir des informations sur l'état d'une association). Les interfaces abstraites quant à elles, sont employées par des clients MSP afin de recevoir des notifications sur des événements tels que des données qui arrivent, l'établissement ou la clôture d'association.

MSP fournit des interfaces afin que ses clients puissent s'enregistrer sur des événements qu'ils souhaitent être notifiés. En conséquence, MSP fournit également des interfaces afin que ses clients puissent supprimer les enregistrements des événements qu'ils ne souhaitent plus être notifiés.

5. Analyse de Performance

Dans cette section, nous présentons les résultats préliminaires de l'analyse de performance de MSP. Pour cela, nous avons analysé la mise en oeuvre de MSP dans XORP en employant BGP comme client de MSP. De plus, nous nous sommes servis de la simulation en employant *l'Integrated Multiprotocol Network Emulator/Simulator* (IMUNES) [ZEC 04].

5.1. *Simulation*

Le but des simulations est d'évaluer l'exécution de MSP. D'ailleurs, ces simulations valident l'architecture de MSP. En outre, une des métriques qui nous avons employée pendant les expériences s'appelle *Routing Message Overhead* (RMO). RMO est défini comme *le nombre total de messages UPDATE* échangés dans le réseau par unité de temps.

Nous avons effectué deux types d'expériences lors des simulations :

$\surd$ *UP* : Un noeud BGP annonce une seule destination. Ensuite, nous attendons jusqu'à ce que le système converge vers une état stable.

$\surd$ *Down* : Dans l'état stable produit par l'expérience UP, le noeud BGP retire la destination qui a été annoncée. Ensuite, nous attendons à nouveau jusqu'au à ce que le système converge vers un nouveau état stable.

Ces simulations ont été effectuées en utilisant différentes configurations du réseau pour chaque type d'expérience.

5.1.1. *Environnement d'expérimentation*

Afin de faciliter l'analyse de performance, lors des simulations, nous avons fait quelques simplifications. Dans les topologies réseau, chaque AS est composé de un seul routeur. En plus, juste un seul AS est choisi comme *AS source*, qui annonce et plus tard retire un préfixe réseau d. Lors des simulations, d est le seul préfixe réseau qui est pris en compte.

Nous avons employé la famille de topologies appelée *CLIQUE* comme topologie réseau. Une famille de topologie CLIQUE est définie comme suit :

Definition 1 *Une configuration réseau de la taille n dans la famille CLIQUE se compose de n systèmes autonomes avec des routeur R_1, R_2,..., R_n, tels qu'il y a un lien entre R_i et R_j pour $i, j = 1, 2, ..., n, i \neq j$. En conséquence, une CLIQUE se compose de systèmes autonomes dans une topologie complètement maillée.*

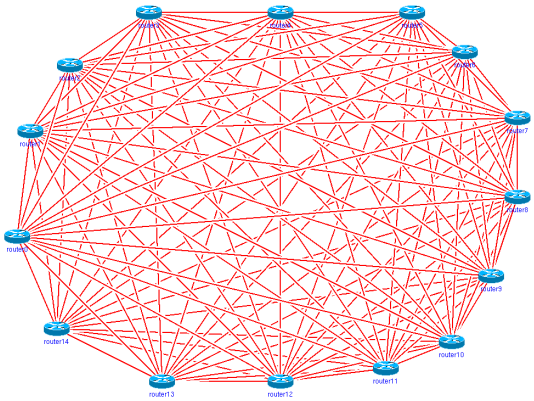

Figure 5. *Clique of size 15*

Dans la phase *Down*, il est possible d'avoir pour chaque routeur un grand nombre de chemins alternatifs vers le routeur source. Cette topologie a été signalée par Labovitz [LAB 00] comme le pire cas possible dans le modèle réseau. La Figure 5 montre une topologie réseau CLIQUE de taille quinze.

Pour chaque expérience que nous avons effectué, nous définissons une fonction $\mathcal{F}$ prenant les paramètres suivants : n (nombre de systèmes autonomes) ; di (le délai induit, en secondes, sur tous les messages UPDATEs avant de les envoyer) ; ld (link delay - en secondes). Afin de simplifier l'analyse des expériences, tous les liens ont la même valeur de ld. Un routeur (r_i) converge quand il traite tous les messages UPDATE qu'il reçoit, même si ces messages ne le font pas choisir à nouveau une meilleure route. Son temps de convergence, c'est-à-dire, le temps écoulé à partir du moment où le routeur source envoie son premier message UPDATE jusqu'au moment où le routeur (r_i) traite son dernier message est dénoté $time_{r_i}$. En conséquence, nous définissons la période de convergence d'une expérience donnée comme le temps maximum de convergence parmi tous les routeurs n dans le système :

$$time(f, \vec{p}) = \max_{0 \leq i \leq n} (time_{r_i}) \tag{1}$$

Dans l'équation 1, f représente l'expérience UP ou $Down$. Le nombre de messages UPDATE dans une expérience donnée est défini comme le nombre total d'annonces plus le nombre total de retraits qui ont été envoyés par les routeurs :

$$updates(f, \vec{p}) = \sum_{0 \leq i \leq n} (nra_i + nrw_i) \tag{2}$$

Tableau 1. *Résultats d'expérience sans MSP.*

	No. of UPDATEs	RMO	Convergence Time
Clique 5	190	3,28	58
Clique 8	721	6,62	109
Clique 10	5207	8,06	646
Clique 15	6166	7,5	823

Tableau 2. *Résultats d'expérience avec MSP.*

	No. of UPDATEs	RMO	Convergence Time
Clique 5	73	2,4	30
Clique 8	622	4,07	153
Clique 10	6799	8,86	767
Clique 15	7272	8,16	891

Dans les tableaux 1 et 2, nous montrons les résultats que nous avons obtenu avec des expériences en utilisant des CLIQUEs de tailles cinq, huit, dix et quinze. De plus, nous avons réalisé des expériences en utilisant la mise en oeuvre de BGP dans XORP avec et sans MSP. Au demeurant, dans ces tableaux, *No. of UPDATEs* représente $updates(f, \vec{p})$ dans l'équation 2. En conséquence, ces expériences ont prouvé que MSP améliore les performances de BGP.

6. Travaux Existants

Dans la littérature, il y a des travaux qui ont été développés pour traiter du problème des connexions fiables. Dans ce cadre, quelques projets ont exploré l'idée d'ajouter des fonctionnalités au protocole TCP afin de surmonter le problème d'échec de connexion. Par exemple, [SNO 01] propose un méthode qui ajoute des nouvelles options à TCP pour permettre la migration des connexions d'un hôte vers l'autre, tandis que [ZAG 03] propose une architecture qui permet aux systèmes de supporter des "crashes" sans clore la connexion TCP. [ZAN 02] proposent deux techniques, appelées *rocks* et *racks*, qui fournissent une mobilité transparente de la connexion réseau, y compris des mécanismes pour le rétablissement et pour la détection d'échec de connexion. Cependant, ces systèmes utilisent une technique problématique, celle du standard KEEPALIVE de TCP, afin de détecter des échecs de connexion. En outre, ils ne fournissent pas les services tels que le multi-homing, ou le multi-session qu'on trouve dans MSP.

[STE 00] propose un protocole de gestion de session appelé *Stream Control protocol* (SCTP). Comme MSP, SCTP fournit des services additionnels comme la conservation des frontières des messages, le multi-homing et le multi-streaming. En comparaison avec SCTP, MSP offre plusieurs avantages par des services additionnels tels que le multi-session, la diffusion et la reprise automatique de la session transport. En outre, MSP fournit une architecture plus modulaire que celle fournie par SCTP. Les lecteurs intéressés pour un analyse plus détaillé de la comparaison de MSP et SCTP peut se rapporter au [CAV 05].

7. Conclusion

Dans cet article, nous avons présenté un nouveau protocole de gestion de session appelé Managed Session Protocol (MSP). De plus, nous avons présenté la vérification des propriétés relevantes pour la correction du protocole, nous avons décrit la mise en oeuvre de MSP en XORP, et nous avons présenté aussi quelques résultats concernant l'analyse de performance. Ces analyses ont montré, tout d'abord, la viabilité de MSP. Puis, elles ont montré qu'on peut réduire la complexité liée à l'architecture de BGP en séparant les fonctionnalités liées à la gestion de session d'autres fonctionnalités.

La prochaine étape de ce travail sera d'utiliser les connaissances obtenues par la vérification formelle de MSP afin de généraliser cette approche et, en conséquence, pouvoir vérifier autres protocoles de communications réseau. En effet, une nouvelle approche de test basée sur des invariants a déjà été conçue. Cette approche peut être appliquée au test de protocoles de communication tel que MSP mais également aux protocoles de routage comme BGP. Concernant à l'analyse de performance, la suite de ce travail sera d'utiliser un simulateur comme PlanetLab [PAG], qui permet d'obtenir des résultats plus réels, afin d'analyser plus précisément l'overhead du MSP sur les applications.

8. Bibliographie

[BHA 02] BHARGAVAN K., OBRADOVIC D., GUNTER C. A., « Formal Verification os Standards for Distance vector Routing Protocols », *J. ACM*, , 2002, p. 538-576.

[CAM] CAMBRIDGE I. R., http ://www.intel-research.net/cambridge/.

[CAV 05] CAVALLI A., GRIFFIN T. G., VIEIRA D., « A Comparison Between Two Session Maintenance Protocols », *IEEE AICT 2005, Lisbonne, Portugal*, , 2005.

[DON 03] DONG Y., HOLZMANN G., SMOLKA S., « Fighting livelock in the GNU i-Protocol : a case study in explicit-state model checking », *STTT*, , 2003, p. 505-528.

[HAN 05] HANDLEY M., KOHLER E., GHOSH A., HODSON O., RADOSLAVOV P., « Designing Extensible IP Router Software », *In the proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI) 2005*, 2005.

[HOL 03] HOLZMANN G. J., *The Spin Model Checker*, Addison Wesley, September 2003.

[KAT 05] KATZ D., WARD D., « Bidirectional Forwarding Detection », Internet Draft - RFC, March 2005.

[KEN 98] KENT S., ATKINSON R., « Security Architecture for the Internet Protocol », Internet Draft - RFC 2401, November 1998.

[LAB 00] LABOVITZ C., AHUJA A., ROSE A., JAHANIAN F., « Delayed Internet Routing Convergence », *In Proc. ACM SIGCOMM*, August/September 2000.

[MUS 04] MUSUVATHI M., ENGLER D., « Model Checking Large Network Protocol Implementions », *In NSDI*, , 2004, p. 155-168.

[PAG] HOME PAGE P., http ://www.planetlab.org.

[SMI 02] SMITH M. A., RAMAKRISHNAN K. K., « Forma Specification and Verification of Safety and Performance of TCP Selective Acknowledgment », *IEEE/ACM Trans. Netw.*, , 2002, p. 193-207.

[SNO 01] SNOEREN A., ANDERSEN D., BALAKRISHNAN H., « Fine-grained failover using connection migration », *In Proc. 3rd USENIX symp. on Internet Technologies and Systems (USITS)*, , 2001, p. 97-108.

[STE 00] STEWART R. R., XIE Q., MORNEAULT K., SHARP C., SCHWARZBAUER H. J., TAYLOR T., RYTINA I., KALLA M., ZHANG L., PAXSON V., « Stream Control Transmission Protocol », http ://www.ietf.org/rfc/rfc2960.txt, October 2000.

[ZAG 03] ZAGORODNOV D., MARZULLO K., ALVISI L., BRESSOUD T. C., « Engineering Fault-Tolerant TCP/IP Servers Using FT-TCP », *In Proc. IEEE Intl. Conf. on Dependable Systems and Networks (DSN)*, , 2003.

[ZAN 02] ZANDY V., MILLER B., « Reliable network connections », *In Proc. ACM MobiCom*, , 2002.

[ZEC 04] ZEC M., MIKUC M., « Operating System Support for Integrated Network Emulation in IMUNES », *in Proceedings of the 1st Workshop on Operating System and Architectural Support for the on demand IT InfraStructure / ASPLOS-XI*, , 2004.