UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À

L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE

EN

MATHÉMATIQUES ET INFORMATIQUE APPLIQUÉES

PAR

MARTIN ST-AMAND

NOUVELLE MÉTHODE DE DISTRIBUTION DES CLÉS DE CRYPTAGE DANS LES COMMUNICATIONS MULTICAST

NOVEMBRE 2003

À Nathalie et à Amélie

REMERCIEMENTS

Je remercie sincèrement tout d'abord M. François Meunier, de l'Université du Québec à Trois-Rivières, d'avoir pris le temps d'évaluer mon mémoire et je remercie Mme Dominique Seret, de l'Université René Descartes, Paris V, malgré la distance, de prendre aussi le temps et l'énergie nécessaire à cette évaluation.

Je remercie mon professeur, M. Boucif Amar Bensaber, de m'avoir proposé de faire la maîtrise et de son aide tout au long de mes longs travaux. Son sens de la minutie et du travail bien fait m'a permis d'offrir des articles de qualité et de finir mon mémoire. Son esprit critique m'a permis de remettre en question ma méthode et moi-même bien des fois, ce qui nous a été bénéfique pour mes recherches et pour moi.

Je remercie ma femme et ma fille pour leurs encouragements, pour leur soutient et pour les bons moments qui ont fait oublier les mauvaises passes. Avec un travail de longue haleine, vous m'avez fait reprendre mon souffle et vous m'avez redonné courage bien des fois.

Je remercie mes parents de m'avoir encadré alors que je développais mon potentiel dès le plus jeune âge.

Je remercie André Boumso pour ces conseils de recherche et pour les bonnes discussions autour de la machine à café. Merci pour l'ouverture d'esprit contagieuse, ton esprit ouvert et pour des échanges vrais et profonds.

RÉSUMÉ

Lors d'une communication de groupe appelée communication Multicast, un émetteur transmet des paquets d'information vers des récepteurs, appelés abonnés ou participants. Lors d'une transmission encryptée, l'entité responsable de la création des clés de cryptage et de les acheminer est appelée Key Distribution Center (KDC).

Une optimisation des méthodes point à point doit être effectuée pour réduire la quantité de paquets envoyés lors de l'ajout et du départ d'un ou de plusieurs participants.

La méthode ROKEN-SKI est présentée dans cet ouvrage. Elle combine deux méthodes déjà proposées par la communauté de recherche : les variables complémentaires, appelées identifiants dans cet ouvrage et les clés de hachage, appelées clés glissées.

Aussi, nous utiliserons une méthode d'attribution du nom des clés qui permet d'augmenter la clarté des arbres de hiérarchie de clés et de faciliter le traitement lors du rafraîchissement des clés.

Je désignerai ma méthode par l'appellation ROKEN-SKI (Refresh-Oriented Key Naming and Slided Keys and Identifiers). Pour bien démontrer l'efficacité de la technique, elle sera comparée dans les tests avec une technique existante.

ABSTRACT

In group communications (known as Multicast communication), an entity called transmitter sends packets of information to several receivers (called subscribers or participants). In a secure communication, the entity responsible of the creation and the transmission of the encryption keys is called a Key Distribution Center (KDC)

An optimisation of the Unicast methods must be carried out to reduce the amount of packets transferred on a join and a leave of one or a group of participants.

The ROKEN-SKI method is presented in this document. It combines two methods that were proposed by the scientific community: the complementary variables, called identifiers in this work and the hashed keys, called slided keys. I also used a method for naming the keys in the tree, which allows improved clarity of the hierarchical key trees and facilitate the treatment when refreshing the keys.

I will call my method ROKEN-SKI (Refresh-Oriented Key Naming and Slided Keys and Identifiers). For showing how my method improves the communications, it will be compared to an existing method.

TABLE DES MATIÈRES

Page

LISTE DES FIGURES

LISTE DES TABLEAUX

CHAPITRE 1

LA COMMUNICATION DE GROUPE : LE MULTICAST

1.1 Introduction

Depuis la nuit des temps, les communications occupent une place importante dans la vie des hommes. La diffusion d'information pour informer les masses s'est faite sur les places publiques, puis avec les manuscrits. Vint ensuite les livres, les journaux et les Télex.

Maintenant qu'Internet est implanté dans une grande partie du monde, des moyens existent pour distribuer de l'information rapidement et gratuitement. Il manquait toutefois un mode de communication à grande échelle, c'est à dire à un très grand nombre de destinataires à la fois. C'est dans cette veine que la diffusion (broadcast) a été mise sur pied. Ce type de transmission diffuse des informations à toutes les adresses d'un réseau.

Mais un réel problème se posait lorsque les destinataires se situaient hors du réseau dans lequel l'émetteur se trouvait : le message de diffusion n'est visible que dans son réseau. C'est dans cette optique que les travaux de recherches sur la communication de groupe, le *Multicast*, ont commencé, vers 1994.

1.2 Définitions

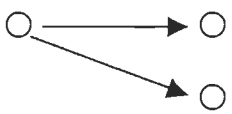
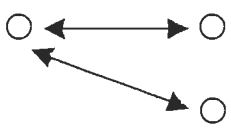
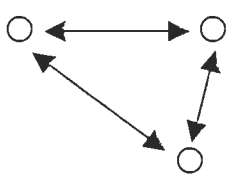
Un protocole est dit **échelonnable** s'il conserve les mêmes propriétés observées avec un petit groupe, que lorsqu'il est employé dans un grand groupe.

Lorsque les données sont trop grandes pour être envoyées toutes d'un coup, on les divise en **ADU** (Application Data Unit) et on met ces ADU dans les paquets à envoyer. Les ADU se définissent clairement comme: unité de données logiquement séparables utiles à l'application (ex : 1 frame vidéo, quantité slides, etc). [8]

1.2.1 Conversations

Une conversation est un transfert d'information entre deux participants d'un groupe actif.

Il existe 4 types de conversations entre des participants, que ce soit les composants de base d'une association de groupe ou dans une sous-association du groupe: [9]

- unidirectionnelle 
- bidirectionnelle 
- multidirectionnelle 

- point à point unidirectionnelle 

1.2.2 Description des entités

Les sessions Multicast sont créées par un Initiateur, une entité responsable de :

- débuter la session
- déterminer les paramètres suivants :
    - le nombre d'abonnés possible
    - les détails sur les connexions
    - la qualité de Service
    - la sécurité

Souvent, l'initiateur est l'émetteur. L'émetteur est l'entité qui émet les paquets vers les destinataires.

Les abonnés sont les entités (ordinateurs, applications) qui se sont joints au groupe et qui vont recevoir les données. Lorsque les communications Multicast sont encryptées, les abonnés reçoivent le nom de participants.

Les superviseurs (réparateurs, serveurs) sont des intermédiaires entre le maître et les abonnés qui sont responsables de certaines opérations lors du ré-acheminement des informations erronées (voir figure 1).

1.3 Multicast fiable

Dans plusieurs applications comme la transmission vidéo ou audio, même si les abonnés reçoivent mal une certaine quantité d'informations, cela ne serait pas catastrophique. Dans certains cas, comme dans le transfert de fichier, il est impératif pour l'émetteur de transférer toutes les informations aux abonnés et de s'assurer qu'ils ont bien reçu toutes les parties (ADU) des informations à transmettre. Le Multicast fiable (Reliable Multicast) est donc un mode de transfert Multicast sans erreur.

1.3.1 Processus d'acquiescement

Il existe deux grandes familles de protocoles de Multicast fiables, basés sur leurs différences au niveau de leur façon de savoir si leurs abonnés ont bien reçu les paquets. Une famille est basée sur des paquets d'acceptation, Positive acknowlegment (ACK), et l'autre sur des paquets de non-réception, Negative acknowlegment (NACK).

1.3.1.1 Mécanismes basés sur ACK

Avec les méthodes basées sur ACK, une demande de confirmation est faite pour chaque envoi. Lorsque l'abonné reçoit correctement les fragments de données, les ADU, il envoie à l'émetteur un paquet ACK pour lui signaler qu'il a bien reçu l'information.

Si l'émetteur ne recevait pas de ACK de la part d'un abonné, il devrait faire réacheminer les informations.

Il est important de connaître les avantages et inconvénients des différentes façons d'utiliser les paquets ACK.

Avantages :

- l'émetteur détient un maximum d'informations sur les conditions de réception;
- l'émetteur peut utiliser les informations pour faire un bon contrôle de congestion et un bon débit de transmission;
- l'émetteur est assuré que tous les abonnés ont reçu l'information.

Inconvénients :

- les ACK normaux causent des défaillances à l'émetteur (par la taille de la mémoire utilisée pour le contrôle et par l'implosion des messages lors de la réception);
- les ACK ne conviennent pas à de grands groupes;
- l'émetteur doit attendre d'avoir les ACK de tous les participants avant de pouvoir continuer, dans le cas où il y a erreur de transmission, un temps précieux est gaspillé;
- les ACK peuvent ne pas se rendre à l'émetteur, et donc il peut attendre une réponse qui ne viendra jamais, même si l'abonné a bien reçu l'information.

1.3.1.2 Mécanismes basés sur NACK

Avec les méthodes basées sur NACK, une demande est faite pour chaque envoi erroné. Lorsque l'abonné ne reçoit pas correctement les fragments de données, les ADU, il envoie à l'émetteur un paquet NACK pour lui signaler qu'il y a eu des erreurs durant la transmission.

Dès que l'émetteur reçoit un NACK de la part d'un abonné, il devrait faire réacheminer les informations.

Avantages :

- l'émetteur n'a plus besoin de savoir exactement combien de récepteurs il y a;
- la gestion de la correction d'erreur est plus simple, elle est assurée par les récepteurs;
- l'état de l'émetteur peut être réduit car il n'a pas besoin de garder une trace de l'état des récepteurs;
- seulement un NACK est nécessaire pour avertir qu'il y a des erreurs ;
- plus besoin d'attendre après toutes les réponses avant de passer à la transmission suivante.

Inconvénients :

- plus difficile pour l'émetteur de savoir s'il peut vider ses tampons d'envoi;
- des mécanismes supplémentaires au niveau session sont requis si l'émetteur veut savoir si un récepteur quelconque a reçu les données.

1.3.1.3 Agrégation de messages d'acquiescement

Une des méthodes employées pour la retransmission des informations erronées est la suivante: des superviseurs sont désignés parmi les abonnés pour assurer la re-diffusion à des sous-groupes [29]. Chaque superviseur s'occupe d'un sous-groupe et fait le relais à partir du superviseur-maître.

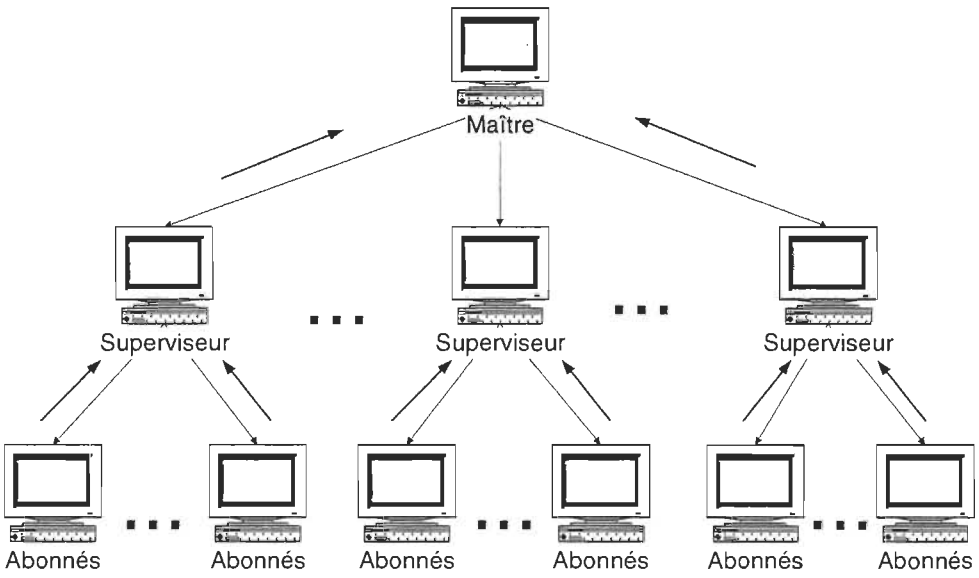


Figure 1: Agrégation de ACK/NACK

Les arbres d'acquiescement:

- plus robustes et plus faciles à configurer que les autres méthodes;
- pour être facilement échelonnable, l'arbre doit être généré par les récepteurs et sa gestion doit se faire sans que l'émetteur intervienne, sinon la capacité d'échelonnage sera diminuée;

- la retransmission peut se faire par l'entremise des nœuds intermédiaires sans que le principal émetteur ait besoin d'intervenir;
- les nœuds de l'arbre pourront :
    1. combiner les ACK en envoyant un ACK quand tous leurs enfants ont envoyé des ACK;
    2. combiner les ACK en listant tous les enfants qui ont envoyé les ACK;
    3. envoyer un ACK combiné avec une liste des exceptions NACK.

1.4 Transmissions entre les entités

Lors des sessions Multicast, plusieurs échanges se font entre le groupe et l'émetteur. L'émetteur doit effectuer des transmissions pour :

- L'acheminement de données : l'émetteur doit effectuer l'envoi des données vers le groupe.
- Gestion des acquittements : l'émetteur doit compiler les acquittements qu'il reçoit.
- Gestion des retransmissions : l'émetteur doit s'assurer que tous les abonnés ont reçu les paquets.
- Contrôle de la congestion : l'émetteur doit s'assurer que ces paquets n'engendrent pas une trop grande utilisation des ressources réseaux qui le relient avec le groupe, en modifiant la taille et la vitesse de transmission des paquets sortants.
- Gestion de la qualité de service : l'émetteur doit s'assurer que les abonnés reçoivent correctement les paquets.

- Gestion de la sécurité : si une communication cryptée est désirée, l'émetteur et le centre de distribution de clés (KDC) doivent s'assurer que les paquets sont encryptés et que tous les abonnés du groupe aient accès aux clés de décryptage.

Le schéma ci-dessous montre les différentes fonctionnalités utilisées pour faire l'échange entre l'émetteur et le participant.

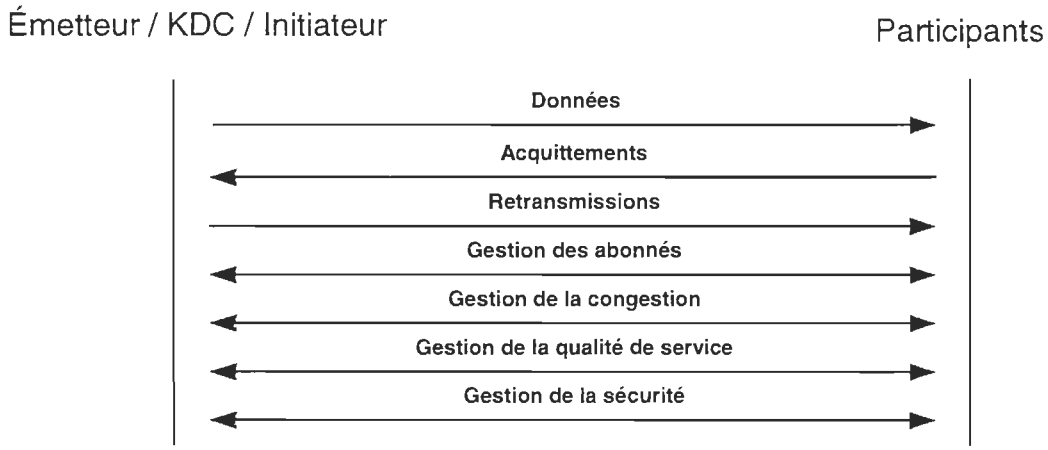


Figure 2: Échanges entre émetteur et participants

Comme ces échanges sont nombreux, pour qu'un protocole Multicast soit échelonnable, il faut qu'il minimise ces échanges. Plus le nombre d'abonnés est grand et plus le nombre de transmissions sera élevé.

1.5 Correction d'erreurs

La correction d'erreur peut être effectuée de trois façons : la retransmission, la réplication et les méthodes Forward Error Correction.

Le premier et le plus utilisé, est la retransmission des informations erronées. C'est une méthode qui assure une correction efficace, mais consomme énormément de bande passante pour l'envoi des informations.

La méthode de la réplication est utilisée lorsque les informations sont répétées. Un paquet erroné serait plus long à retransmettre que d'en émettre un nouveau. Par exemple, la position d'un objet dans un jeu. Pas de retransmissions, donc pas de surcharge réseau. Elle est utilisée dans des applications interactives.

Les techniques FEC (forward error correction) consistent à prendre un groupe de paquets qui doivent être envoyés et d'effectuer un OU logique (XOR) avec eux et envoyer le résultat du calcul avec chacun des paquets. Si le récepteur reçoit le XOR et 2 paquets sur 3, il peut régénérer le paquet perdu sans retransmission. Les techniques FEC sont efficaces pour réduire les transmissions de réparation lorsque les erreurs surviennent peu fréquemment et sont utilisées souvent dans des transferts de fichier.

1.6 Applications qui utilisent le Multicast

Dans les communications, dès que le nombre de récepteurs est supérieur à un, le Multicast peut être utilisé. Des tests ont été effectués pour des jeux à plusieurs joueurs

[3], des serveurs vidéos [4][5] et des serveurs web [6] et l'utilisation du Multicast a été bénéfique dans tous les cas. Le tableau 1 dresse une liste des applications qui utilisent le Multicast.

	Temps réel	Non Temps réel
Multimédia	Serveur vidéo Vidéo-conférence Internet audio Évènements multimédia	Réplication Livraison sur demande
Données seulement	Cotes de la bourse Nouvelles « Tableau blanc » Jeux en-ligne Cache web	Acheminement de données Réplication de base de données

Tableau 1: Applications utilisant le Multicast

1.7 Différences entre Unicast et Multicast

Les requêtes Internet dites point à point ou Unicast sont effectuées lors d'un échange entre une seule entité requérante et une seule entité servante. Ce type de requête représente 99.99% des requêtes Internet, c'est-à-dire utilisation du Web, échanges de fichiers ou toute connexion à un service.

Le terme Multicast est employé pour désigner une transmission d'information entre un ou des émetteurs vers plusieurs récepteurs. Plusieurs types de Multicast peuvent être effectués, dépendant du nombre d'émetteurs permis dans le groupe. Une connexion 1 à N schématisée dans la figure 3, représente une transmission d'information en une seule fois d'un émetteur vers N récepteurs et une connexion de N à M, schématisée dans la figure 4,

représente une transmission d'information en une seule fois entre N émetteurs et M récepteurs. Les communications N à M ne seront pas traités dans cet ouvrage.

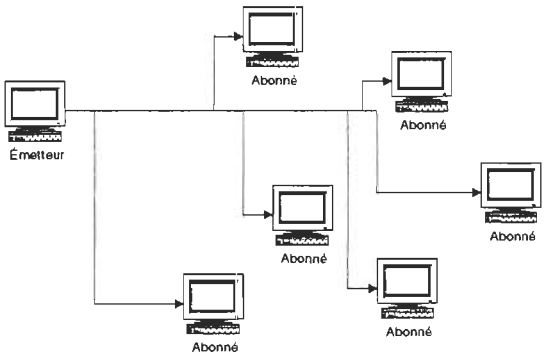


Figure 3: Multicast 1 à N

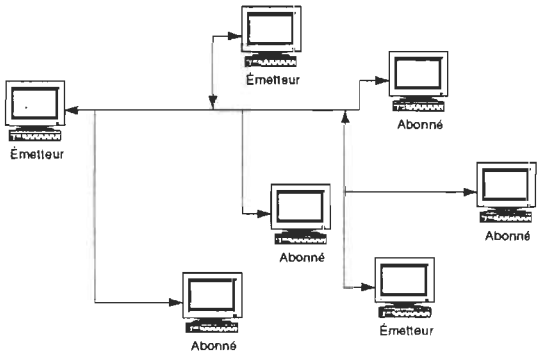


Figure 4: Multicast N à M

Lors d'une communication Multicast, l'ensemble des destinataires est appelé **Groupe** et l'entité responsable de l'émission des paquets d'informations est appelée **Émetteur**. Les destinataires sont nommés **Abonnés** lors d'une session Multicast non-cryptée et **Participants** lors d'une communication cryptée.

Lorsque l'émetteur veut envoyer une information à tous les destinataires, il établit une connexion avec l'adresse de groupe et transmet l'information une seule fois. Les méthodes diffèrent selon les diverses méthodes de duplication d'information sur le réseau. Une étude plus approfondie des méthodes de transmission est présentée à la section1.3.

Les recherches sur les communications Multicast ont permis d'assurer un certain nombre d'avantages par rapport aux communications Unicast. Les principaux restent la diminution de la gestion des connexions chez l'émetteur et la diminution de l'utilisation

de la bande passante sur le lien Internet de l'émetteur et sur le réseau jusqu'aux destinataires.

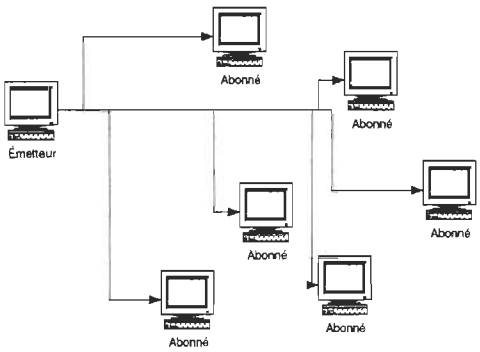


Figure 5: Des transmissions Unicast

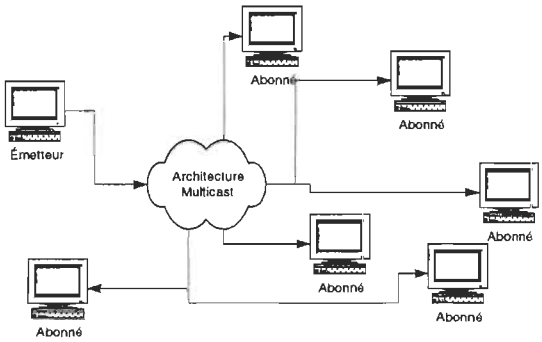


Figure 6: Une transmission Multicast

Comme dans la figure 5, les transmissions Unicast utilisent une connexion par abonné tandis que le Multicast (figure 6) utilise une connexion pour le groupe et envoie les paquets ayant une adresse de groupe comme destination. Le nombre de connexions est donc considérablement diminué si le nombre d'abonnés est grand. En général, le terme grand groupe désigne un groupe de 1000 et plus.

1.8 Architectures et protocoles

1.8.1 IP Multicast

Le protocole IP Multicast [1] gère la transmission des paquets Multicast sur le réseau Mbone (voir section 1.3.3). Le protocole IP Multicast utilise l'architecture du Mbone pour dupliquer les paquets d'information que l'émetteur envoie au groupe.

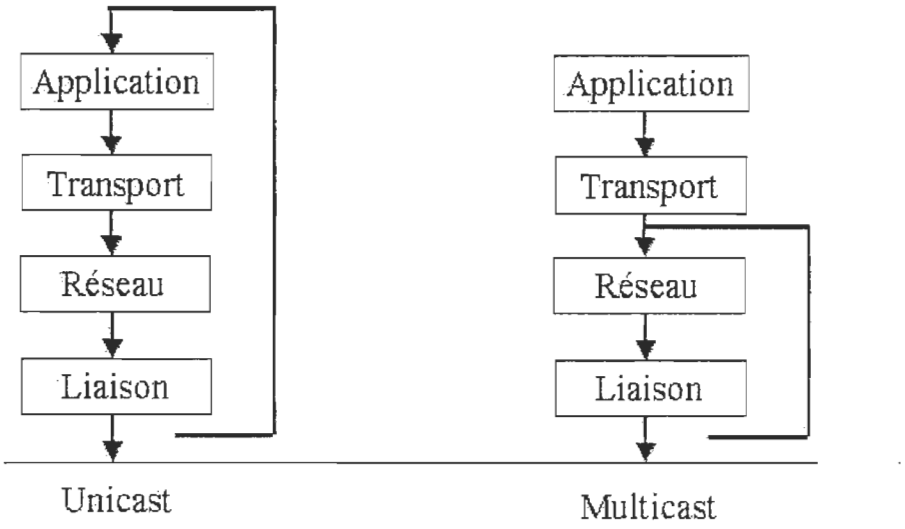


Figure 7: Utilisation de la pile TCP/IP

Lorsqu'un émetteur effectue un envoi en mode Unicast, il utilise une connexion à partir de la couche application. S'il utilise plusieurs connexions, les ressources utilisées seront de beaucoup supérieures à celles utilisées par le protocole IP Multicast, qui n'utilise que les parties inférieures pour la connexion au groupe (figure 7).

1.8.2 IGMP (Internet Group Management Protocol)

Les instructions de commandes pour la gestion du groupe au niveau réseau (la couche 3 de la pile TCP/IP) sont gérées par le protocole IGMP [2]. Lorsqu'un abonné veut se joindre au groupe, il envoi un paquet *JOIN* avec l'adresse du groupe. L'architecture Multicast va ensuite envoyer la requête à son routeur Multicast qui acceptera ou refusera la demande d'adhésion.

Pour s'assurer qu'il y ait encore des abonnés qui vont recevoir ses transmissions, le routeur peut demander un écho au groupe. S'il reçoit une confirmation, il sait qu'il y a au

moins un participant encore disponible pour recevoir. Une confirmation générée par le client IGMP qui reçoit la demande en premier est suffisante pour que le routeur continue la diffusion sur cette partie du réseau. Les autres n'ont pas besoin de répondre.

1.8.3 Le MBONE

Depuis le commencement vers 1994, les recherches en Multicast nécessitaient de grands besoins d'architecture pour effectuer des tests. Les chercheurs ont créé une architecture dédiée à la recherche en Multicast : le MulticastBone, ou Mbone, est composé de routeurs spéciaux, contenant des instructions qui permettent de dupliquer des paquets d'information pour les acheminer à tous les membres d'un groupe. Ces routeurs Multicast sont présents dans toutes les régions du globe, tel qu'indiqué à la figure 8.

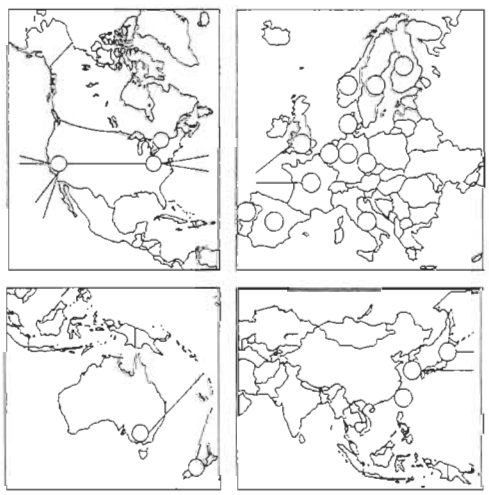


Figure 8: Le Mbone

L'architecture actuelle du Mbone utilise des routeurs Multicast et des routeurs Unicast. Comme les routeurs Internet ne supportent pas tous le protocole IP Multicast, les paquets

sont encapsulés dans des paquets Unicast et transférés de routeur en routeur, jusqu'à ce qu'un routeur Multicast se retrouve sur le chemin du paquet. Il doit donc décapsuler le paquet Unicast et retirera le paquet Multicast. Le routeur va alors produire des copies du paquet et les acheminer dans tous les chemins qui relient le routeurs aux abonnés. Cette méthode est démontrée dans la figure 9.

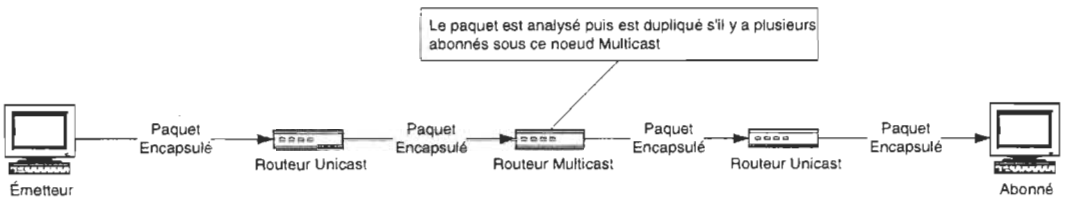


Figure 9: Encapsulation des paquets IP Multicast

Les routeurs Multicast communiquent entre eux par des protocoles de routage Multicast.

Dans le chapitre 2, je fais un tour d'horizon des différentes techniques existantes pour échanger les clés de cryptage lors des différentes opérations effectuées dans le groupe.

Dans le chapitre 3, je décris la méthode que je propose pour échanger les clés de cryptage lors qu'un participant se joint au groupe ou le quitte.

Dans le chapitre 4, je décris comment ma méthode a été testée et dans le chapitre 5, je présente les différents résultats obtenus lors des tests.

CHAPITRE 2

MÉTHODES DE DISTRIBUTION DES CLÉS LORS D'UNE COMMUNICATION MULTICAST

2.1 Introduction

Lors d'une communication de groupe (dite communication Multicast), une entité appelée émetteur transmet des paquets d'informations vers plusieurs récepteurs (appelés abonnés). Lors de la transmission, les paquets acheminant l'information se retrouve sur un réseau public qu'est Internet. Si les données sont « sensibles », des méthodes de cryptage doivent être utilisées.

Dans la recherche en sécurité Multicast, on perçoit deux axes de recherche : l'authentification et la transmission des clés. L'authentification est la vérification de l'identité d'une entité dans le groupe et ne sera pas traitée dans ce mémoire.

Des méthodes de communications sécures comme ISAKMP [10] ou des méthodes d'envoie de clés comme Diffie-Hellman [11] sont bien adaptées pour le point à point. Par contre, elles ne sont pas optimales lorsqu'elles sont utilisées en Multicast. Des méthodes d'optimisation doivent être utilisées pour réduire le temps d'envoi des informations et le temps de calcul d'encryption. Des méthodes doivent être créées pour les besoins des communications Multicast. Étant donnée que plusieurs gestions se font en même temps

sur la même bande, le but à atteindre est d'obtenir une méthode de cryptage des communications de groupe qui utilise le moins possible la bande passante.

2.1.1 Considérations

On doit prendre en compte certaines considérations avant d'analyser les différentes techniques de distribution de clés dans une communication Multicast. Le RFC 2627 [12] définit les conditions de base pour effectuer une session cryptée en Multicast.

2.1.1.1 Entités en jeu

Initiateur : entité qui détermine les besoins pour une session Multicast, qui définit les attributs de sécurité pour la session et qui fabrique la liste des participants de groupe. La liste sera gérée par l'émetteur ou par une entité différente.

Gestionnaire des clés (Key Distribution Center, KDC) : entité responsable d'acheminer les clés à tous les participants. On peut parler aussi de racine dans certains cas.

Participant : abonné faisant partie de l'association de groupe (group association). C'est à dire dans une communication Multicast dont les messages sont encryptés.

2.1.1.2 Clés

Une clé est un nombre ou une structure nécessaire pour décoder un texte crypté. Le groupe utilise plusieurs types de clés, dépendant des besoins.

Des clés personnelles sont remisent aux participants lorsqu'ils se joignent au groupe. Une clé personnelle servira à décoder les messages destinés à un seul participant (dis messages point à point ou Unicast). Ces messages proviennent du KDC ou de l'émetteur.

Des clés de groupe sont utilisées pour décoder des messages destinés à plusieurs participants du groupe (dit messages de groupe ou Multicast).

2.1.1.3 Confidentialité

On peut qualifier trois sortes de confidentialité (secrecy) [13]:

Confidentialité de clé de groupe (group key secrecy) : garantie qu'il est impossible pour un adversaire de découvrir aucune des clés de groupe.

Confidentialité du passé (Backward secrecy) : garantie qu'il est impossible qu'une entité passive, connaissant un lot de clés, de découvrir des clés anciennes

Confidentialité du futur (Forward secrecy) : garantie qu'il est impossible qu'une entité passive, connaissant un lot de clés anciennes, de découvrir des clés nouvelles.

2.1.2 Opérations sur le groupe

- Ajout : lorsqu'un participant se joint au groupe.
- Départ : lorsqu'un participant quitte le groupe. Il peut aussi s'agir d'une expulsion du groupe. Certains protocoles définissent une différenciation entre les deux mais nous traiterons les deux dans la même opération.

- Simple : lorsqu'il y a un ajout/départ d'un seul participant.
- Multiple : lorsqu'il y un ajout/départ de plusieurs participants à la fois.
- Séparation : lorsqu'un groupe se scinde en deux ou plusieurs groupes.
- Fusion : lorsque deux ou plusieurs groupes en deviennent un seul.
- Pertes de clés : lorsqu'un participant ne reçoit pas les messages contenant les clés lui permettant de décoder les messages suivants. Seulement quelques protocoles en font allusion.

2.1.2.1 Critères de comparaison

Les critères de comparaison des différentes approches sont:

- le temps requis pour établir la communication de groupe;
- le nombre de transmissions lors des opérations;
- l'efficacité des opérations sur le groupe;
- l'utilisation de la mémoire et le stockage.

2.2 Approches centralisées à un niveau

2.2.1 Distribution manuelle

Ces approches [14][12][15] sont basées sur les méthodes d'encryption des communications point à point. Elles utilisent des clés privées et publiques pour encrypter et décrypter les messages envoyés de l'émetteur vers les récepteurs. Un message crypté avec la clé publique de B ne peut être décrypté qu'avec la clé privée de B. La figure 10 permet de mieux comprendre les étapes de l'envoi d'un message encrypté.

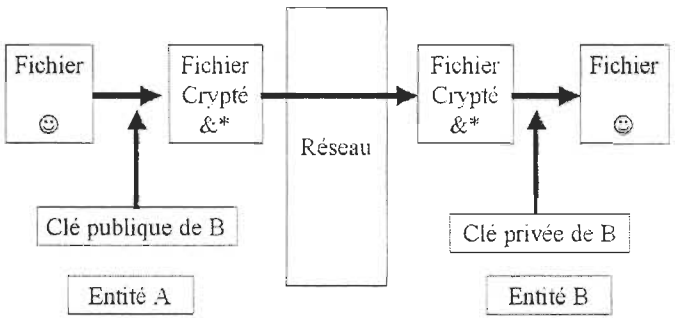


Figure 10: Utilisation de clés privées/publiques en point à point

Le diagramme du groupe est organisé comme une hiérarchie (figure 11). Une entité centrale, appelée racine ou gestionnaire des clés (Key Distribution Center ou KDC) est souvent l'initiateur du groupe. C'est elle qui est responsable de distribuer les clés aux participants du groupe.

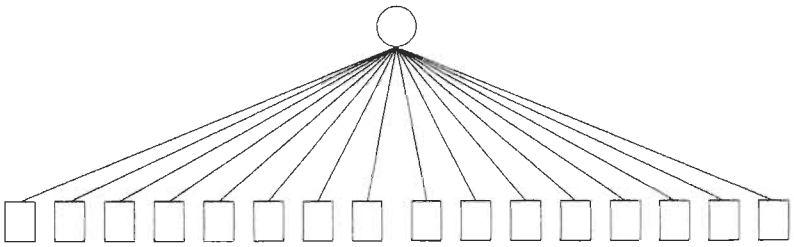


Figure 11: Organisation d'un groupe selon la distribution manuelle

Pour chaque envoi d'information de l'émetteur vers un destinataire, une génération de paquet crypté est effectuée. Donc, si le groupe est composé de n participants, il y aura donc n paquets encryptés individuellement et envoyés à chacun des participants, lorsqu'une information doit être envoyée à tout le monde sans utiliser la clé de groupe.

La clé de groupe est utilisée pour envoyer un message à tous, comme un message de données ou d'informations générales. Pour lire ces messages « généraux », le participant doit avoir en sa possession la clé de groupe. La clé de groupe est envoyée à chacun des participants lors de la création du groupe ou à un participant qui s'ajoute au groupe par la suite. Cette clé est encryptée avec la clé personnelle du participant.

L'entité KDC fabrique des clés de rechange à être utilisées lorsque la sécurité est compromise. Ces clés de rechange peuvent aussi être transmises après une certaine période de temps [12], déterminée d'avance par l'initiateur. Les nouvelles clés de groupes sont envoyées à chacun, encryptées avec la clé individuelle de chacun des participants.

2.2.1.1 Création du groupe

L'initiateur fixe les conditions de sécurité (grosseur des clés de cryptage, règles de rejet, etc.), génère la liste des participants choisis. L'initiateur envoi une demande à tous les participants qui veulent faire partie de la session. À chaque réponse positive, l'initiateur et le participant échangent leurs clés publiques pour effectuer une communication cryptée.

2.2.1.2 Ajout simple

Lors de l'ajout d'un participant, on lui envoie sa clé personnelle et ensuite, une nouvelle clé de groupe est créée et elle est envoyée à chacun des participants dans un paquet encrypté avec la clé publique respective à chacun des participants.

2.2.1.3 Ajout multiple

Lors de l'ajout de participants, on envoie pour chacun des « nouveaux » leur clé personnelle et une nouvelle clé de groupe est créée et elle est envoyée à chacun des participants dans un paquet encrypté avec la clé publique respective à chacun des participants.

2.2.1.4 Départ simple

Lors du départ/rejet d'un participant, une nouvelle clé de groupe est créée et est envoyée à chacun des participants restants en utilisant un paquet crypté avec leur clé individuelle.

2.2.1.5 Départ multiple

Lors du départ/rejet de participants, une nouvelle clé de groupe est créée et elle est envoyée à chacun des participants restants en utilisant un paquet crypté avec la clé publique respective à chacun des participants.

2.2.1.6 Séparation

Une clé de groupe est créée pour chacun des deux groupes. La clé de groupe du premier groupe est envoyée à chacun des participants de ce groupe en utilisant sa clé personnelle et la clé du second groupe est envoyée à chacun les participants de ce groupe en utilisant aussi sa clé personnelle.

2.2.1.7 Fusion

Une nouvelle clé de groupe est créée et elle est envoyée à chacun des participants dans un paquet encrypté avec sa clé publique.

2.2.1.8 Perte de clés

Lorsqu'un participant ne reçoit pas la clé de groupe, il doit attendre que le temps de la clé soit écoulé et reçoit enfin la clé de groupe lui permettant de décoder les messages « généraux ».

2.2.1.9 Avantages

- déjà bien utilisé dans les applications actuelles;
- ne requiert pas de méthodes sophistiquées.

2.2.1.10 Désavantages

- Dans les sessions en temps réel, cette méthode est trop lente;
- Non-échelonnable (à utiliser qu'avec de petits groupes).

2.3 Approche avec variables complémentaires

Cette méthode [12] est basée sur la génération à distance de la clé de groupe. Elle utilise la même structure que la méthode manuelle, comme la figure 12 l'illustre, mais cette méthode pourrait réduire de beaucoup les envois entre l'entité KDC et les participants. La méthode utilisée pour générer la clé de groupe est basée sur le principe suivant : chaque participant connaît une variable associée à chacun des autres participants, mais ne connaît pas la variable associée à lui-même.

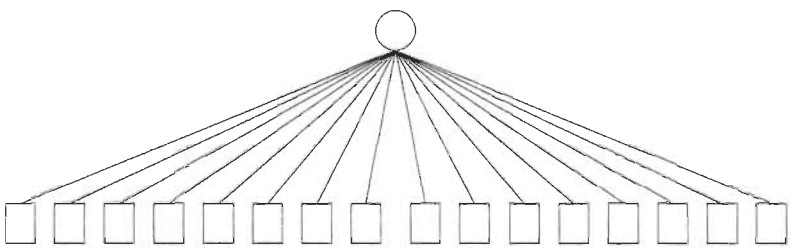


Figure 12: Organisation d'un groupe selon la distribution manuelle

En supposant que lorsqu'on passe une variable dans une fonction le résultat étant la valeur d'une clé, alors on peut faire générer à tous ceux qui connaissent la valeur de la

variable, une nouvelle clé. On aura juste à indiquer la variable à utiliser pour générer la clé.

Les participants ont donc les variables de tous les autres participants en stockage. Par exemple, si le groupe est composé de N participants, le participant aura N-1 variables en mémoire. Pour indiquer la variable à utiliser, l'entité KDC envoie un message avec le nom ou l'indice du tableau à utiliser.

Si on utilise la méthode pour générer une nouvelle clé de groupe lorsqu'un participant quitte, la racine envoie un message du type « u1 a quitté le groupe ». Ceci devient une indication pour trouver quelle variable utiliser pour fabriquer la nouvelle clé.

L'initiateur de la session envoie alors la variable complémentaire à tous les participants sauf celui qui est associé à la variable. Il envoi une clé de groupe en même temps.

2.3.1 Création du groupe

Lors de la mise sur pied du groupe, on génère une variable complémentaire pour chacun des participants du groupe. Cette variable est déterminée par une opération mathématique, une fonction pseudo aléatoire. L'entité KDC envoie chacune des variables à chacun des participants, sauf leurs variables associées.

L'envoi suit l'algorithme suivant :

Boucler pour i de 1 à N
    Boucler pour j de 1 à N

```
            Si i ≠ j
                Envoyer( participant[i], variable[j] )
        Fin boucle
    Fin boucle
```

2.3.2 Ajout simple

Lorsqu'un participant se joint au groupe, l'entité KDC (ou l'initiateur) génère une variable complémentaire et l'envoi à tous les autres participants. L'entité KDC envoie au participant arrivant les variables de tous les autres participants.

2.3.3 Ajout multiple

Lorsque plusieurs participants se joignent au groupe, l'entité KDC (ou l'initiateur) génère une variable complémentaire pour chacun d'entre eux et l'envoi à tous les autres participants. L'entité KDC envoie aux participants arrivants les variables de tous les autres participants.

2.3.4 Départ simple

Lorsqu'un participant quitte le groupe ou il est banni, l'entité KDC envoie un message du type « le participant xxx a quitté le groupe ». Tous les participants peuvent alors régénérer la clé de groupe en utilisant la variable complémentaire associée au participant qui a quitté.

2.3.5 Départ multiple

Lorsque plusieurs participants quittent le groupe ou si ils sont bannis, l'entité KDC choisit le dernier « quittant » et envoie un message du type « le participant xxx a quitté le groupe, ainsi que xxx, xxx … ». Tous les participants peuvent alors régénérer la clé de groupe en utilisant la variable complémentaire associée au participant qui a quitté et peuvent supprimer de leur listes les variables complémentaires associés aux autres « quittants ».

2.3.6 Séparation

Lorsque le groupe se scinde en plusieurs groupes distincts, on doit recalculer et redistribuer de nouvelles variables complémentaires, pour chacun des groupes.

2.3.7 Fusion

Lorsque plusieurs groupes se réunissent en un seul, les variables complémentaires d'un groupe doivent être redistribuées dans le groupe comme si chacun des membres du groupe était des nouveaux participants.

2.3.8 Perte de clés

Si un participant ne recevait pas un message d'un départ, par exemple, alors il ne serait pas capable de décoder les messages suivants. Si tel est le cas, deux opérations différentes peuvent être effectuées.

Une fonction pourrait être ajoutée pour essayer de retrouver parmi les variables celle qui a été utilisée pour faire la nouvelle clé. Cette méthode prendrait beaucoup d'opérations de cryptage surtout si le nombre de participants est grand.

Une demande pourrait être envoyée au serveur pour savoir quel était le dernier quittant ou pour avoir une mise à jour du tableau de variables.

2.3.9 Avantages

- À chaque départ : aucune distribution de clés;
- Un message unique de petit format.

2.3.10 Désavantage

- Quand on supprime plusieurs utilisateurs en même temps, ils peuvent faire collusion. Cette méthode s'explique de la façon suivante : un participant ne connaît pas la variable complémentaire qui lui permettrais de rester dans le groupe, sauf si un autre participant du groupe lui donne. Ainsi, des participants partageant leur variable complémentaire annule les possibilités d'exclusion d'un participant de groupe.
- La quantité de stockage est énorme quand la taille augmente. Ce qui rend la méthode non-échelonnable.

2.4 Approches centralisées en arbres

La méthode en arbre hiérarchique [16][12] découle des travaux d'optimisation lors des départs de participant, notamment les travaux de Fiat et Naor [17] sur des modèles mathématiques appliqués aux transferts de clés.

Plutôt que d'utiliser les clés personnelles de chacun pour envoyer la nouvelle clé de groupe, l'entité KDC peut utiliser des clés de sous-groupes pour encrypter les clés de groupe. Wong et al. [16] définissent les ensembles suivants : U : ensemble de participants, K : ensembles des clés et R : ensemble des relations entre U et K.

La technique diminue le nombre d'envoie de clés à envoyer en point à point entre l'entité KDC et les participants.

- Note importante : l'arbre des clés est différent de l'arbre de transmission des données Multicast.

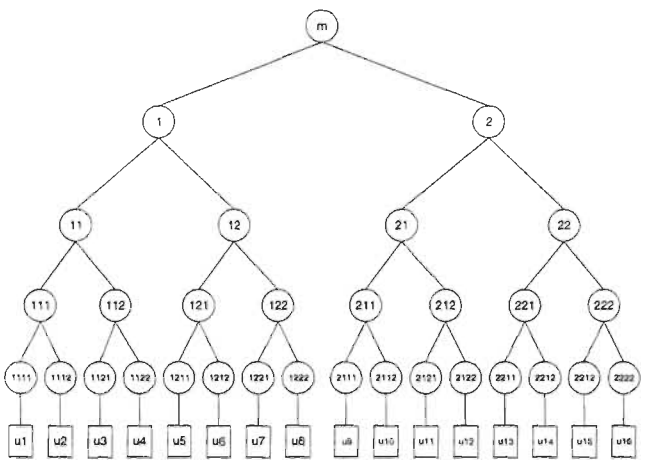


Figure 13 : Arbre hiérarchique

Dans l'arbre, comme la figure 13 le démontre, il y a une clé pour chaque feuille et une clé pour chaque nœud, racine comprise. Chaque participant doit posséder toutes les clés correspondantes aux nœuds qui le relie au KDC. Les feuilles ne connaissent pas les clés dont elles n'ont pas besoin.

Les participants sont représentés par des carrés dans la figure 13 et les clés elles sont représentées par des cercles.

Lorsque des messages doivent être envoyés à tous les participants, on utilise la clé de groupe. Cette clé est connue de tous et elle représente la racine de l'arbre en figure 13.

Lorsque des messages s'adressent à un sous-groupe, l'émetteur utilise les clés particulières et peut envoyer des messages à un sous-groupe directement en Multicast.

Si le message s'adresse à un participant seulement, sa clé privée peut être utilisée. La communication se fait alors en point à point.

2.4.1 Création du groupe

L'initiateur fabrique un arbre binaire ou non, balancé ou non, avec d niveaux, avec la liste des participants. L'initiateur prend souvent la charge du KDC. Chaque participant est alors géré comme un ajout de participant.

2.4.2 Ajout simple

Dans la figure 14, si le participant 1 s'ajoute au groupe, les clés en gras indiquent quelles sont les clés qui doivent être changées. Elles devront être envoyées à tous leurs descendants.

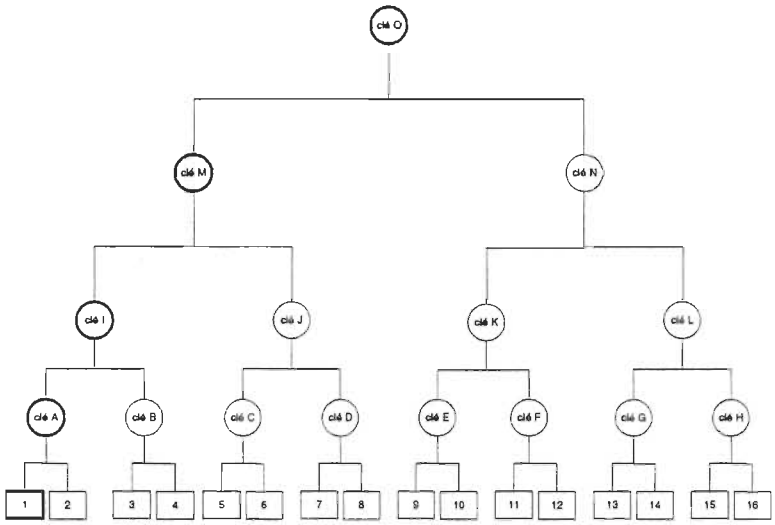


Figure 14: Les clés devant être changées

L'entité KDC envoie au participant 1 sa clé personnelle.

Toutes les clés que le participant 1 possède doivent être renouvelées. De nouvelles clés de sous-groupe sont nécessaires pour préserver le niveau privé des anciens messages. Ces clés sont générées par une fonction mathématique.

La clé de sous-groupe (clé A dans l'exemple) est encryptée avec la clé personnelle du participant 1. Notée : $\{K_A\}_{K1}$

L'entité KDC envoie au participant 1 sa clé de groupe A.

La clé de sous-groupe (clé A dans l'exemple) est encryptée avec la clé privée du participant 2. Notée : $\{K_A\}_{K2}$

La clé de sous-groupe A est envoyée à tous les participants nécessitant cette clé. Dans l'exemple, les participants 1 et 2 doivent la recevoir.

La clé de sous-groupe I est encryptée en utilisant la clé de sous-groupe A. $\{KI\}_{KA}$

La clé de sous-groupe I est encryptée en utilisant la clé de sous-groupe B. $\{KI\}_{KB}$

La clé de sous-groupe I est envoyée à tous les participants nécessitant cette clé. Dans l'exemple, les participants 1 à 4 doivent la recevoir.

La clé de sous-groupe M est encryptée en utilisant la clé de sous-groupe I. $\{KM\}_{KI}$

La clé de sous-groupe M est encryptée en utilisant la clé de sous-groupe J. $\{KM\}_{KJ}$

La clé de sous-groupe M est envoyée à tous les participants nécessitant cette clé. Dans l'exemple, les participants 1 à 8 doivent la recevoir.

La clé de groupe O (appelée aussi K_m) est encryptée en utilisant la clé de sous-groupe M. $\{KO\}_{KM.}$

La clé de groupe O est encryptée en utilisant la clé de sous-groupe N. $\{KO\}_{KN}$

La clé de groupe est envoyée à tous les participants.

2.4.3 Ajout multiple

Chaque nouveau participant est ajouté au groupe avec la même méthode qu'avec un ajout simple.

2.4.4 Départ simple

Lors du départ d'un participant, il faut recalculer et réacheminer toutes les clés qu'il possédait (sous-sous-groupe, sous-groupe, groupe), pour préserver la sécurité des nouveaux messages.

Toutes les clés qu'avait le participant 1 doivent être renouvelées. De nouvelles clés de sous-groupe sont nécessaires pour préserver le niveau privé des anciens messages. Ces clés sont générées par une fonction mathématique.

Dans la figure 15, si le participant 1 quitte le groupe, les clés en gras indiquent celles qui doivent être changées. Elles devront être envoyées à tous leurs descendants.

La clé de sous-groupe A est encryptée avec la clé privée du participant 2. Notée : $\{K_A\}_{K2}$

L'entité KDC envoie au participant 2 sa nouvelle clé de groupe A.

La clé de sous-groupe I est encryptée en utilisant la clé de sous-groupe A. $\{KI\}_{KA}$

La clé de sous-groupe I est encryptée en utilisant la clé de sous-groupe B. $\{KI\}_{KB}$

La clé de sous-groupe I est envoyée à tous les participants nécessitant cette clé. Dans l'exemple, les participants 2 à 4 doivent la recevoir.

La clé de sous-groupe M est encryptée en utilisant la clé de sous-groupe I. $\{KM\}_{KI}$

La clé de sous-groupe M est encryptée en utilisant la clé de sous-groupe J. $\{KM\}_{KJ}$

La clé de sous-groupe M est envoyée à tous les participants nécessitant cette clé. Dans l'exemple, les participants 2 à 8 doivent la recevoir.

La clé de groupe O (appelée aussi K_m) est encryptée en utilisant la clé de sous-groupe M. $\{KO\}_{KM.}$

La clé de groupe O est encryptée en utilisant la clé de sous-groupe N. $\{KO\}_{KN}$

La clé de groupe est envoyée à tous les participants restants.

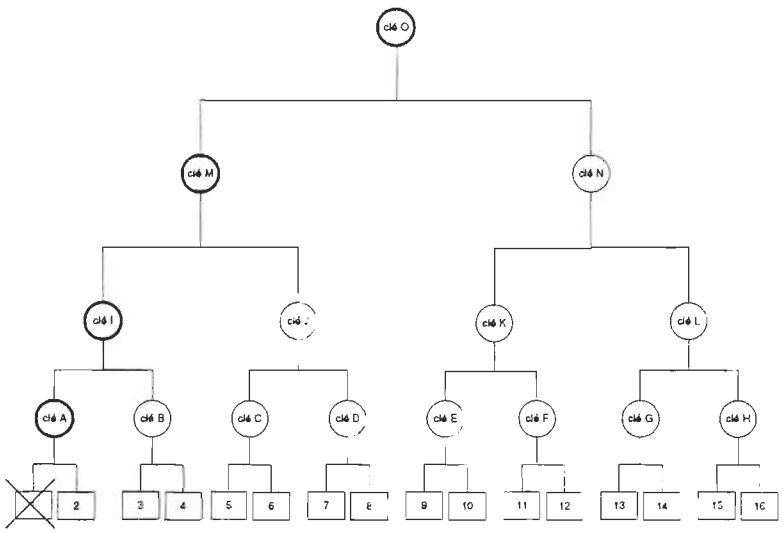


Figure 15 : Retrait d'un participant

2.4.5 Départ multiple

Chaque participant quittant est retiré du groupe avec la même méthode qu'avec un départ simple.

2.4.6 Séparation

Lors de la séparation du groupe en plusieurs groupes, les clés de sous groupes peuvent être utilisées comme clé de groupe dans chacun des nouveaux groupes. Dans l'exemple, si le groupe se décompose en deux groupes, les clés M et N seraient utilisées comme nouvelles clés de groupe.

2.4.7 Fusion

Lors de la fusion entre plusieurs groupes, chacune des clés de groupe est utilisée comme clé de sous groupe et une nouvelle clé de groupe est générée pour le nouveau groupe. La nouvelle clé est ensuite envoyée, encryptée avec les clés de sous-groupes.

2.4.8 Perte de clés

Si un participant ne recevait pas un message d'un départ, il ne serait pas capable de décoder les messages suivants. Le participant devrait redemander les clés de sous groupe et la clé de groupe au KDC, comme s'il s'agissait d'un nouveau participant, sauf que les clés ne seraient pas mises à jour.

2.4.9 Avantages

- Coûts de stockage et de transmission des clés balancés et échelonnables;
- Les clés des nœuds peuvent être utilisées pour encrypter les messages et les clés;
- Parties de l'arbre selon des groupes homogènes (receive-only user);
- Suppression sécuritaire d'un membre;
- Transmission efficace;
- Stockage efficace;
- Approche résistante aux collusions.

2.4.10 Inconvénients

- Lors du départ d'un participant, le temps de recalcul de l'arbre prend plus d'espace mémoire et occupe plus de temps processeur;
- Lors des ajouts de participants, la quantité de clés transférées ente l'entité KDC et les participants augmente le temps de connexion d'un participant.

2.5 Comparaisons entre les méthodes

Le tableau no 2 illustre bien les différences du nombre de paquets échangés lors des différentes opérations sur le groupe.

Méthode	Envoi	Ajout	Retrait
1 à 1	N	1	1
Distribution manuelle	1	N	N
Variables complémentaires	1	N	1
Hiérarchique	1	$(Log_2(N))/2$	$(Log_2(N))/2$

Tableau 2: Nombre de paquets échangés selon la méthode

N : Nombre de participants dans le groupe

2.6 Variantes de la technique hiérarchique

Les variantes de la technique en arbre tentent de réduire la taille des paquets contenants les clés, la taille des clés ou le nombre d'envois par opération. Un tableau comparatif de ces méthodes est présenté en annexe.

2.6.1 Sans index

2.6.1.1 Efficient Large-Group Key Distribution (ELK)

Le but visé par la méthode ELK [13] de Perrig était de réduire la quantité de bande passante utilisée lors de la transmission des clés, en réduisant la taille des messages. La méthode ELK utilise des méthodes mathématiques pour réduire la taille des messages lors des envois de clés. À la place d'envoyer des clés dans les messages, l'entité KDC pourrait envoyer des « indices » (hints). Lorsque le participant reçoit le message et les

indices en même temps, il pourrait fabriquer les clés avec l'indice. L'indice est donc un générateur de clés, à distance.

La technique utilise aussi des agrégats de messages lors de l'opération de départ multiple. Cette méthode consiste à réduire le coût de transmission. Si plusieurs membres quittent en même temps ou dans un intervalle de temps prédéterminé à l'avance, l'entité KDC envoie les nouvelles clés aux participants restants en changeant les clés partagées par tous les « quittants ».

Perrig [13] semble être le seul à se préoccuper des pertes de clés. Il propose que les indices puissent être envoyés dans les mêmes paquets que les données. Cette proposition est excellente étant donnée que les données sont réacheminées par les méthodes de retransmission Multicast fiable. Dans ce cas, les opérations lors de perte de clés seraient nulles et les cas où un participant ne serait pas capable de décrypter un message sont réduits à zéro.

2.6.2 Avec Index

Les index sont générés par des nombres aléatoires et peuvent être utilisés pour générer des clés. Ils sont stockés chez l'entité KDC et chez les participants. Ils peuvent aussi s'appeler Key Encryption Key (KEK).

2.6.2.1 One-way Function Trees (OFT)

La méthode OFT [18] utilise les techniques des variables complémentaires expliquées précédemment, appliquée aux arbres hiérarchiques. Chaque clé a sa clé « cachée » qui est utilisée pour régénérer la clé (les cases en gris clair sur la figure 16, les clés que le participant u1 connaît et utilise sont en noir).

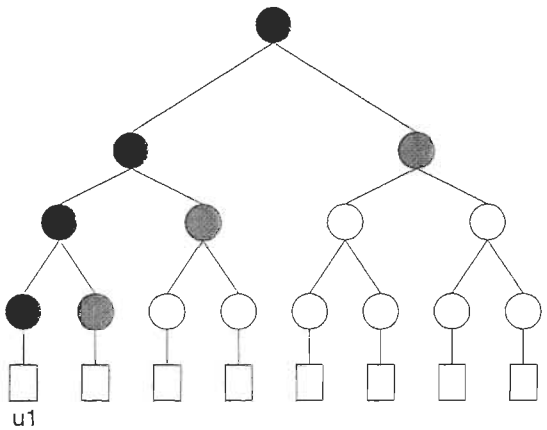


Figure 16: La méthode OFT

Lors de l'ajout d'un participant, l'entité KDC envoie les clés cachées et les clés que le participant aura besoin dans un message Unicast, ce qui peut prendre beaucoup de bande passante.

Cette technique utilise les clés des fils de gauche et de droite pour régénérer les clés. Lorsque les clés doivent être rafraîchies, les clés peuvent être rafraîchies par l'utilisation des index ou par la dérivation d'une clé de ses fils et passées dans une fonction dite « un sens » (one-way function).

2.6.2.2 Enhanced Binary trees (EHBT)

La technique EHBT [19] utilise aussi une clé cachée pour chaque clé et stockée chez le frère du possesseur de la clé.

Si un participant utilise cette clé et que son frère quitte, il utilise la clé cachée comme nouvelle clé partagée entre son frère et lui, sauf que son frère ne peut plus décrypter les messages de cette branche. L'entité KDC envoie les clés supérieures cryptées avec cette nouvelle clé, de la même façon qu'avec les arbres hiérarchiques. Les clés anciennes sont stockées dans un ensemble de clés ancêtres.

Dans les messages d'envoi des clés, la technique utilise des commandes et des index, pour donner les informations aux nœuds pour connaître les clés à utiliser pour créer les nouvelles clés.

2.6.2.3 Méthode de Canetti

Cette méthode est dérivée de la méthode OFT. Cannetti [20] propose d'utiliser la partie basse ou haute d'une clé pour en générer une autre, ce qui est plus difficile pour une entité extérieure de trouver la valeur de la clé. Il utilise la partie haute ou basse de la clé du fils de gauche ou de droite pour régénérer une clé.

2.7 Approches Distribuées

Des approches distribuées sont les méthodes qui distribuent une certaine partie du traitement ou du contrôle aux niveaux de serveurs secondaires ou intermédiaires, ou encore au niveau des participants. Ces méthodes améliorent beaucoup l'achalandage au

niveau du KDC et permettent de séparer les groupes en sous-groupes plus facilement. Des méthodes, comme la méthode Cliques [21], utilisent des contrôleurs de groupes pour la gestion des clés dans les sous-groupes.

Par contre, lorsque que l'on permet à plusieurs entités d'assurer un certain contrôle ou traitement, il faut s'assurer que les entités responsables sont bel et bien celles qu'elles disent être. Des travaux d'authentification en Multicast sont en cours pour permettent l'utilisation de ces méthodes distribuées [56].

2.7.1 Iolus

Cette approche [22] propose une subdivision du groupe Multicast en sous-groupes pour mieux distribuer le traitement et l'envoi des clés, comme illustré à la figure 17. Des superviseurs de sous-groupes (GSI) sont responsables de la transmission des clés et des données cryptées aux autres participants. Ils connaissent la clé de groupe et les clés de sous-groupes desquels ils font partie.

Les ajouts et départs se font au niveau du sous-groupe sans que l'entité centrale intervienne. Chaque sous-groupe utilise une clé de groupe différente et les rafraîchissements de cette clé sont différents à chaque groupe. L'entité KDC (GSC) n'est en relation avec les participants que lorsqu'ils s'abonnent au groupe. Ils sont alors redirigés vers les GSI. L'émetteur est en relation avec les GSI qui lui sont connexe et leur transmet les paquets cryptés.

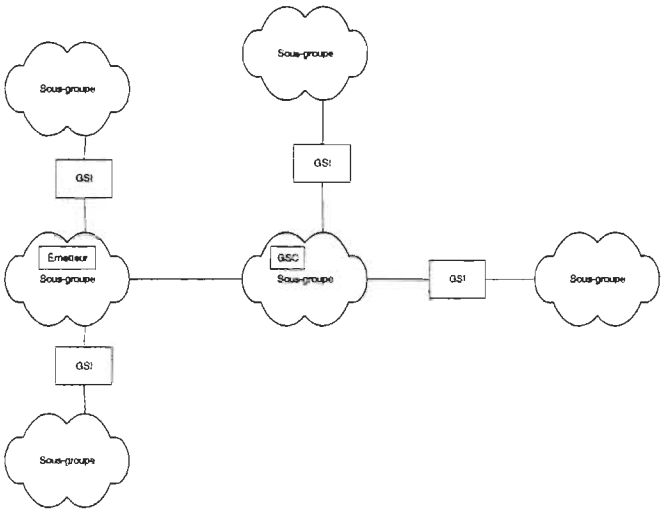


Figure 17 : Un groupe avec la méthode Iolus

Lors de l'acheminement des paquets cryptés au travers d'un sous-groupe vers un autre ou du GSI vers les participants, trois méthodes se présentent. La première méthode est de décrypter le message avec la clé de groupe et de recrypter le message avec la clé de sous-groupe et de le transmettre aux participants. Ce qui peut être très coûteux en temps d'encryption. La deuxième méthode est d'encapsuler le paquet crypté avec la clé de groupe permettant de le consulter dans un paquet crypté avec la clé de sous-groupe et de le transmettre aux participants. La troisième méthode est d'utiliser un tunnel entre l'émetteur et le GSI à la manière de IPSEC [57].

2.7.2 Hydra

Le but de la méthode Hydra [23] est de séparer le groupe Multicast en sous-groupes et de faire la gestion de la sécurité dans les différents sous-groupe, même s'ils utilisent des protocoles différents. Hydra n'utilise pas de contrôleur principal, ce qui est une méthode

pleinement distribuée. Chaque sous-groupe Hydra est supervisé par un serveur Hydra. La figure 18 présente une illustration représentant l'architecture de la méthode Hydra.

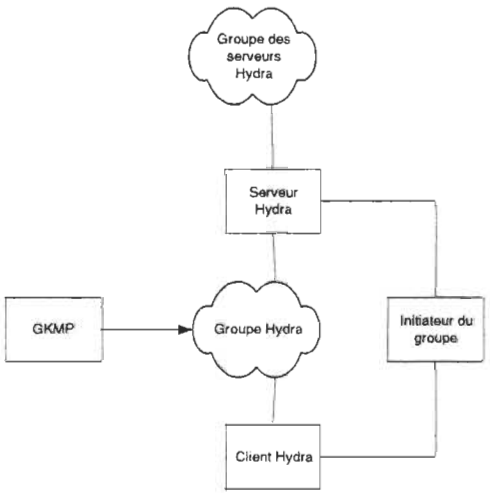


Figure 18 : La méthode Hydra

2.8 Conclusion

Les approches centralisées à un niveau sont les techniques de communications sécurisées appliquées aux communications Multicast. Le nombre de paquets pour la distribution de clés lors du départ/arrivée d'un participant est de n-1 clés. Étant donné que ces clés sont diffusées en utilisant les clés publiques de chacun des participants, cela demande énormément de traitement au niveau du KDC ou de l'émetteur.

Les approches centralisées en arbres sont issues des travaux d'optimisations et réduisent énormément la quantité de clés à envoyer et limite les envois à ceux qui en ont besoin dans l'arbre, soit environ 2 log(n) clés.

Les techniques distribuées séparent le groupe Multicast en plusieurs sous-groupes et acheminent le traitement vers des serveurs distants. Les traitements lors de départs/arrivés sont restreints au niveau des sous-groupes, ce qui est une extension de la technique en arbre.

Toutes les techniques revues dans ce chapitre sont utilisables lors d'une communication de groupe en mode crypté. Un paramétrage lors de la création du groupe peut être effectué par l'initiateur. Il pourrait prévoir la quantité de participants, d'après l'historique du groupe et la quantité de départs/arrivées après la création du groupe, ce qui aiderait à choisir la bonne méthode.

Si le groupe est petit, on aura peut-être avantage à utiliser une technique traditionnelle, déjà connue, facile à comprendre et facile à développer.

Si, par contre, la quantité de départs/arrivés est grande ou si les participants sont nombreux, l'initiateur devrait utiliser les techniques hiérarchiques. Les méthodes hiérarchiques étant plus échelonnables que les autres, permettent de maximiser les envois de clés dans les différents scénarios et permettent de réduire les transmissions de l'émetteur vers le récepteur.

Les méthodes distribuées sont très intéressantes mais elles doivent être combinées avec des techniques d'authentification, de la part de l'émetteur auprès des participants et de la part des superviseurs de sous-groupes. Lorsque les protocoles d'authentification seront échelonnables, donc applicables à de grands groupes Multicast, les approches distribuées

prendront une place de plus en plus importante dans la recherche en gestion des clés dans les communications Multicast.

Dans le prochain chapitre, je décris la méthode que je propose pour échanger les clés de cryptage lorsqu'un ou plusieurs participants se joignent au groupe ou quittent le groupe.

CHAPITRE 3

MÉTHODE PROPOSÉE : ROKEN-SKI

3.1 Introduction

Dans le RFC2627 [12], une méthode utilisant des variables complémentaires a été décrite mais a tout de suite été écartée par ses auteurs. Bien que cette approche réduise le nombre de transmissions de clés lors du départ d'un participant, elle permet trop de collusion entre les participants. Cet inconvénient peut être réduit si les variables complémentaires sont utilisées avec un arbre : le partage d'information entre les différents participants est limité. Donc si elles sont utilisées avec des techniques en arbres, ces techniques deviennent moins propices à la collusion entre les participants.

Dans certaines approches, comme les méthodes OFT [18] et EHBT [19], des clés cachées sont utilisées. Cette technique est très intéressante par sa capacité à reformer des clés avec des informations se retrouvant chez les participants. Nous pensons donc qu'avec des informations précises, les participants peuvent générer certaines clés à distance, c'est-à-dire sans qu'il n'y ait de transmissions entre l'entité KDC et les participants.

Pour faciliter l'appellation des clés dans l'arbre de transmission, nous avons songé à une nouvelle méthode de noms pour attribuer les noms aux clés. Cette technique améliore aussi les traitements lors du rafraîchissement des clés.

Dans ce chapitre, je présente une méthode qui utilise des clés cachées et des identifiants. Cette méthode est appelée Refresh-Oriented Key Naming with Slided Keys and Identifiers method, ou ROKEN-SKI.

3.2 But de notre démarche

Le but de notre démarche est de trouver le nombre de niveaux k idéal en utilisant la technique des variables complémentaires en association avec la technique des clés glissées. Ce niveau k idéal, diminuerait le plus possible les transmissions entre l'entité KDC et les participants.

3.2.1 Définitions

3.2.1.1 Identifiant

Notre approche utilise des identifiants pour régénérer les clés à distance, chez le participant. Nous désignons par **identifiant**, une information utilisable pour générer une nouvelle valeur pour une clé. L'identifiant associé à un participant est inconnue de celui-ci et il est stocké chez les participants proches de lui dans l'arborescence. Les identifiants seront détaillés davantage dans la section 3.2.5. Le mot identifiant est un synonyme de « variable complémentaire ».

3.2.1.2 Clés glissées

Nous désignons, par clé glissée, une clé qui a été obtenue à partir d'une clé d'un de ces fils. La clé du fils de gauche ou celui de droite peut être utilisé pour mettre à jour une clé. Ces clés seront détaillées davantage dans la section 3.2.6.

3.2.1.3 Rafraîchissement d'une clé

Une clé est dite rafraîchie lorsqu'elle est passée dans une fonction mathématique, le résultat du traitement génère une clé avec une valeur différente. Une fonction à sens unique ou une fonction pseudo-aléatoire peut être utilisée pour rafraîchir les clés. Nous utiliserons les fonctions à sens unique dans notre approche.

3.2.1.4 Mise à jour d'une clé

Une clé est mise à jour lorsqu'on lui attribue une nouvelle valeur, sans lien avec sa valeur actuelle.

3.2.1.5 Noms des clés

Les clés de l'arbre sont nommées selon leur position dans la hiérarchie. Les fils de la racine portent le nom 0 et 1. Les petits-fils de la racine portent les noms 00, 01,10 et 11. Plus la position hiérarchique d'une clé est profonde et plus le nom de cette clé est long. **La longueur maximale du nom d'une clé est la hauteur de l'arbre**.

3.2.1.6 Nom d'un participant

Le nom d'un participant est donné par le nom de sa clé personnelle dans l'arbre des clés. Plus le participant est situé profondément dans l'arbre et plus son nom est long.

3.2.1.7 Fonctions à sens unique

Une fonction à sens unique est une fonction mathématique qui prend une valeur en entrée x et qui donne une valeur en sortie y sans qu'il n'y ait de possibilité de retrouver x avec y. Notre approche utilisera deux fonctions à sens unique donnant des valeurs différentes en sorties, les fonctions f(x) et g(x).

3.2.2 Notations

Symboles	**Descriptions**
{ x }y	L'information x est encryptée avec la clé y.
Km ou m	Clé de groupe (clé Multicast).
M{...}	Paquet encrypté envoyé en mode Multicast
U{...}	Paquet encrypté envoyé en mode Unicast
Uk[]	Tableau de clés
x'	Nouvelle valeur d'une clé qui vient d'être changée

3.2.3 Types de paquets

3.2.3.1 Introduction

Nous décrivons ci-dessous les différents paquets utilisés dans l'échange des clés entre l'entité KDC et les participants. Tous ces paquets sont des informations cryptées contenues dans des paquets plus généraux. Le champ DESTINATION indique le destinataire du paquet général. Les participants savent à qui le paquet s'adresse par le nom de la clé contenue dans le champ DESTINATION. Les participants savent donc les destinataires du paquet avec le champs DESTINATION. Les destinataires du paquet sont tous les participants qui connaissent cette clé. Si l'émetteur veut envoyer un paquet à tous les participants, il rempliera le nom de la clé avec l'appellation m, sinon le nom de la clé sera inséré dans le champ (ex. : 0, 1 ou 010100).

Paquet : GENERAL

DESTINATION
PAQUET

Fonction : Désigne à qui s'adresse le paquet encrypté contenu dans le paquet général.

Utilisé: À chaque envoi de paquet.

Type : Unicast ou Multicast.

Champs :

DESTINATION : Entier représentant un nom d'une clé.

PAQUET : Contient un paquet crypté. Longueur variable, selon la longueur du paquet à inclure.

Paquet : KEY

NAME	KEY_VALUE

Fonction: Donne à un ou plusieurs participant(s) une clé.

Utilisé: Lors de la mise à jour d'une clé.

Type : Unicast ou Multicast.

Champs:

NAME : Entier représentant le nom de la clé à mettre à jour.

KEY_VALUE : Entier donnant la valeur d'une clé.

Paquet : KEYS

NAME	KEY_VALUE
NAME	KEY_VALUE
…	

Fonction: Donne à un ou plusieurs participant(s) une série de clés.

Utilisé: Lors de la mise à jour de plusieurs clés.

Type : Unicast ou Multicast

Champs:

NAME : Entier représentant le nom de la clé à mettre à jour.

KEY_VALUE : Entier donnant la valeur d'une clé.

Paquet : REFRESH

NAME

Fonction: Demande aux participants d'effectuer un rafraîchissement d'une clé.

Utilisé: Lors de l'ajout simple d'un participant.

Type : Multicast

Champs:

NAME: Entier représentant le nom de la clé à rafraîchir.

Paquet : REFRESHG

NAME
NAME
…

Fonction: Demande aux participants d'effectuer un rafraîchissement d'une série de clés.

Utilisé: Lors de l'ajout multiple de participants.

Type : Multicast

Champs:

NAME: Entier représentant le nom de la clé à rafraîchir.

Paquet : QUIT

NAME

Fonction: Signale aux participants du groupe qu'un utilisateur a quitté. Avec ses informations, les participants pourront effectuer la mise à jour des clés possédées par le partant, s'ils le connaissent.

Utilisé: Lors du départ simple d'un participant.

Type : Multicast

Champs: NAME: Entier représentant le nom d'un participant.

Paquet : QUITG

NAME
NAME
...

Fonction: Signale aux participants du groupe que plusieurs utilisateurs ont quitté le groupe. Ils pourront effectuer la mise à jour des clés possédées par le partant, s'ils connaissent ces clés. Ces paquets sont utilisés pour un nombre de partants prédéterminés. Si le nombre de partants dépasse un certain seuil, le nom ALL sera utilisé pour mettre à jour toutes les clés de l'arbre.

Utilisé: Lors du départ multiple de participants.

Type : Multicast

Champs: NAME: Entier représentant le nom d'un participant.

Paquet : ADDCOUSIN

NAME	ID

Fonction: Envoie un identifiant d'un nouveau participant à ses cousins dans l'arbre.

Utilisé: Lors de l'ajout simple d'un participant.

Type : Multicast

Champs:

NAME: Entier représentant le nom d'un participant.

ID : Entier représentant un identifiant.

Paquet : ADDCOUSING

NAME	ID
NAME	ID
…	…

Fonction: Envoie à un nouveau participant les identifiants de ses nouveaux voisins dans l'arbre ou envoie aux participants les identifiants des participants qui viennent de s'ajouter.

Utilisé: Lors de l'ajout simple ou lors d'un ajout multiple d'un (de) participant(s).

Type : Multicast

Champs:

NAME: Entier représentant le nom d'un participant.

ID : Entier représentant son identifiant.

3.2.4 Méthode de rafraîchissement des clés

Lors de l'ajout d'un participant, on doit rafraîchir les clés que le nouveau participant possèdera pour préserver la confidentialité du passé de la communication Multicast. Le nom du participant, donné par sa place dans l'arbre, détermine quelles seront les clés qu'il connaîtra. Donc à chaque ajout d'un participant, tous les participants du groupe et l'entité KDC doivent rafraîchir toutes les clés reliant la position du nouveau participant à la racine de l'arbre.

Pour indiquer quelles sont les clés à rafraîchir, l'entité KDC envoie un paquet Multicast REFRESH. Dans le paquet REFRESH, l'information du champ NAME est le nom de la clé qui a créé le besoin de rafraîchissement. Les informations contenues dans le nom de la clé sont très utiles : elles contiennent les noms de toutes les clés nécessitant un rafraîchissement.

Le numéro d'une clé correspond au nom du père auquel a été ajouté un chiffre. Si le fils est situé à gauche de son père, sa clé reçoit un 0 de plus, sinon elle reçoit le chiffre 1 en plus. À chaque niveau, le numéro (absolu) d'une clé contient le nom de toutes les clés reliant celle-ci à la racine de l'arbre. Avec le numéro d'une clé particulière, on peut retrouver chacun de ses parents.

De cette façon, un participant :

- connaîtra toutes les clés à rafraîchir, sans avoir besoin d'une structure de commandes dans les paquets

- n'a pas besoin de rafraîchir toutes ses clés, uniquement celles qu'il a en commun avec le nouveau participant.

Quand le participant reçoit le paquet REFRESH, il prend le nom de la clé et le passe en paramètre dans la fonction `RefreshKeys()`, dont un exemple d'implémentation serait le suivant :

```
void refreshKeys( char[] x )
{
     refresh( Km );
     i = 0;
     while (( i <= lenght(x)) ou ( x[i] == myName[i] ))
     {
          keyX << x[i];  //ajouter x[i] à la fin de keyX
          refresh( keyX )
          i++;
     }
}
```

La fonction `refresh()` prend le nom d'une clé en paramètre et la rafraîchie avec la fonction à sens unique.

```
void refresh( char[] keyX )
{
     no = lenght(keyX) ;
     tableauDesCles[no] = functionX( tableauDesCles[no] ) ;
}
```

La fonction `refreshGroupOfKeys()` prend un paquet avec plusieurs clés et lance la fonction de rafraîchissement avec chacune d'entre elles.

```
void refreshGroupOfKeys( REFRESHG paquet )
{
     nbrKeys = paquet.nbrKeys;
     current = paquet.first ;
     for( i=0; i<nbrKeys; i++)
     {
          refreshKeys( current.KEY_VALUE );
          current = current.next;
     }
}
```

La fonction `refreshALL()` rafraîchit toutes les clés avec la fonction à sens unique.

```
void refreshALL()
{
     for(i = 0; i < length( tableauDesClés ) ; i++)
     tableauDesClés[i] = functionX( tableauDesClés[i] ) ;
}
```

3.2.4.1 Exemple de rafraîchissement

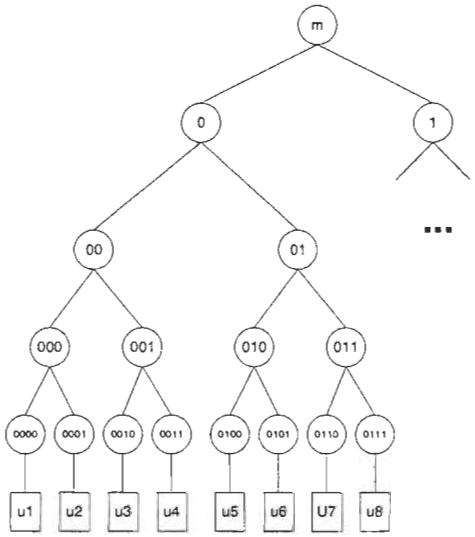


Figure 19: Rafraîchissement de l'arbre des clés

Dans l'exemple schématisé dans la figure 19, lorsqu'un participant, u1 par exemple, reçoit le paquet REFRESH, avec « 1011010 » dans le champ NAME :

Km' = f (Km)	Il rafraîchit la clé de groupe.
1 != 0	Comme la clé à vérifier n'est pas connue par lui, il termine son rafraîchissement de clé.

Lorsque le participant u1 reçoit le paquet REFRESH, avec « 0101 » dans le champ NAME:

Km' = f (Km)	Il rafraîchit la clé de groupe.
0' = f (0)	La clé numéro « 0 », est connue par lui.
01 != 00	Comme la clé à vérifier n'est pas connue par lui, il termine son rafraîchissement de clé.

Lorsque le participant u1 reçoit le paquet REFRESH, avec « 0001 » dans le champ NAME:

Km' = f (Km)	Il rafraîchit la clé de groupe.
0' = f (0)	La clé numéro « 0 », est connue par lui.
00' = f (00)	La clé numéro « 00 », est connue par lui.
000' = f (000)	La clé numéro « 000 », est connue par lui.
0001' != 0000	Comme la clé à vérifier n'est pas connue par lui, il termine son rafraîchissement de clé.

Lorsque le participant u1 reçoit le paquet REFRESH, avec « ALL » dans le champ NAME:

Il rafraîchit toutes ses clés.

3.2.5 Identifiants

Les identifiants sont utilisés pour générer une clé commune à un certain nombre d'individus, sans qu'il n'y ait de transmissions de clés lors d'un départ d'un de ces individus. Si tous les individus d'un groupe connaissent une clé qu'un individu ne connaît pas, lorsqu'il quitte le groupe, la valeur de la nouvelle clé est facile à générer. Les participants prennent l'identifiant associé au participant quittant pour générer la nouvelle valeur de la clé connue par tous, à l'exception de celui qui est associé à cet identifiant.

Notre méthode propose un choix variable de niveaux d'identifiants. Les niveaux commencent en bas de l'arbre et montent jusqu'à la racine. Plus le nombre de niveaux est

élevé et plus le nombre de participants connaissant un identifiant associé à un individu sera grand.

L'entité KDC choisit un niveau d'identifiants à utiliser. Plus ce niveau est haut et plus le nombre de paquets à transférer lors du départ d'un participant sera petit et plus le nombre d'informations à transférer lors de l'ajout d'un participant sera grand. Plus ce niveau est bas, plus le nombre d'informations à transférer lors de l'ajout d'un participant sera petit et plus le nombre de paquets à transférer lors du départ d'un participant se rapprochera des méthodes classiques de distribution en arbre. Au cours de la communication Multicast, le niveau d'identifiants pourra être changé, en créant une négociation entre l'émetteur et les participants.

La technique qui consiste à envoyer un identifiant avant l'ajout d'un participant diminue de moitié la quantité d'information à émettre si on combine l'ajout et le retrait d'un participant. Lorsque l'identifiant est envoyé, il est crypté avec la clé connue de tous ceux qui connaîtront cet identifiant. Avec la méthode classique HBT [12,16], il faudrait envoyer la nouvelle clé encryptée avec la clé de niveau inférieure droite que chacun des participants connaisse. Il faudrait aussi refaire un envoi avec la clé inférieure gauche, pour que les participants connaissant cette clé inférieure gauche puissent avoir accès à la nouvelle valeur de la clé. L'envoi d'un identifiant lors de l'ajout d'un participant permet d'envoyer simultanément à gauche et à droite la clé à mettre à jour. Donc en utilisant un identifiant, le nombre d'encryptions et de transmissions de clés est réduit de moitié.

Une comparaison des simulations utilisant différents niveaux permettront de savoir le niveau maximum pour l'utilisation des identifiants. La combinaison de l'approche avec identifiants et les méthodes de clés glissées permettrait une baisse des échanges entre l'entité KDC et les participants lors des ajouts et des départs de participants.

Une étude statistique du groupe peut être effectuée si un historique du groupe est disponible. Un groupe avec beaucoup de changements dans la liste de ses membres sera difficile à gérer.

3.2.5.1 Exemple de cas avec identifiants

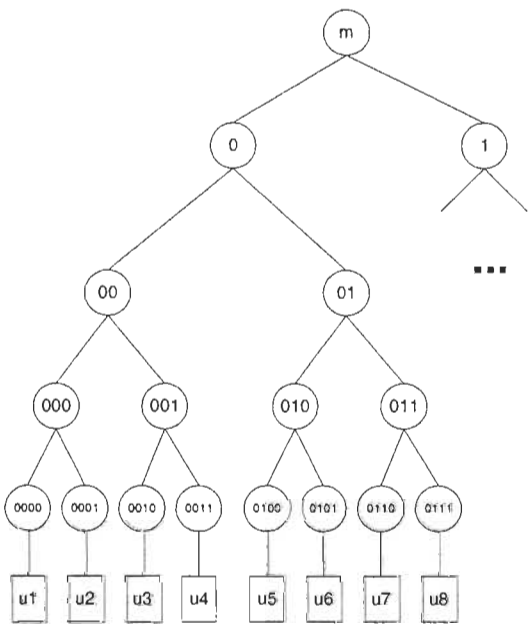


Figure 20: Identifiants

Dans l'exemple présenté à la figure 20, si des identifiants pour 3 niveaux étaient utilisés, chaque participant aura 7 cousins. Chacun de ses cousins possèdera son identifiant. Si on ne veut plus de cousin u2, on prend l'identifiant qui lui était associé et il devient la clé que tous ses autres cousins connaissent. Dans l'exemple ci-dessus, c'est la clé 0.

Les cousins proches de u2 prennent l'identifiant de niveau inférieur, que seul u1 à u4 connaissent, et ils forment la clé qu'ils connaissent. Dans l'exemple, c'est la clé 00.

Pour générer la clé partagée par le quittant et son frère, elle deviendra la valeur de la clé glissée de son frère. Dans ce cas-ci, la clé 000 devient la clé 0001', i.e. 000 = g(001).

Les clés supérieures pourront être générées avec les clés glissées.

3.2.6 Clés glissées

Les clés glissées sont des nouvelles clés obtenues par la transformation d'une clé dans l'arbre. Ces clés peuvent être les clés du fils de gauche ou de droite ou encore la clé du frère. Lorsqu'une clé doit être mise à jour, les transmissions peuvent être réduites si un certain nombre de participants peut connaître la nouvelle valeur de la clé.

En utilisant une clé connue de certains individus et en la passant dans une fonction à sens unique différente de celle utilisée dans le rafraîchissement des clés, on arrive à ce que la moitié des participants qui ont à changer la valeur d'une clé connaissent sa nouvelle valeur, sans transmissions. Le reste des participants qui ont à changer la valeur de cette

clé doivent la recevoir par transmission. Donc, le nombre de transmissions nécessaires pour faire connaître cette clé à tous les participants partageant cette clé est réduit de moitié.

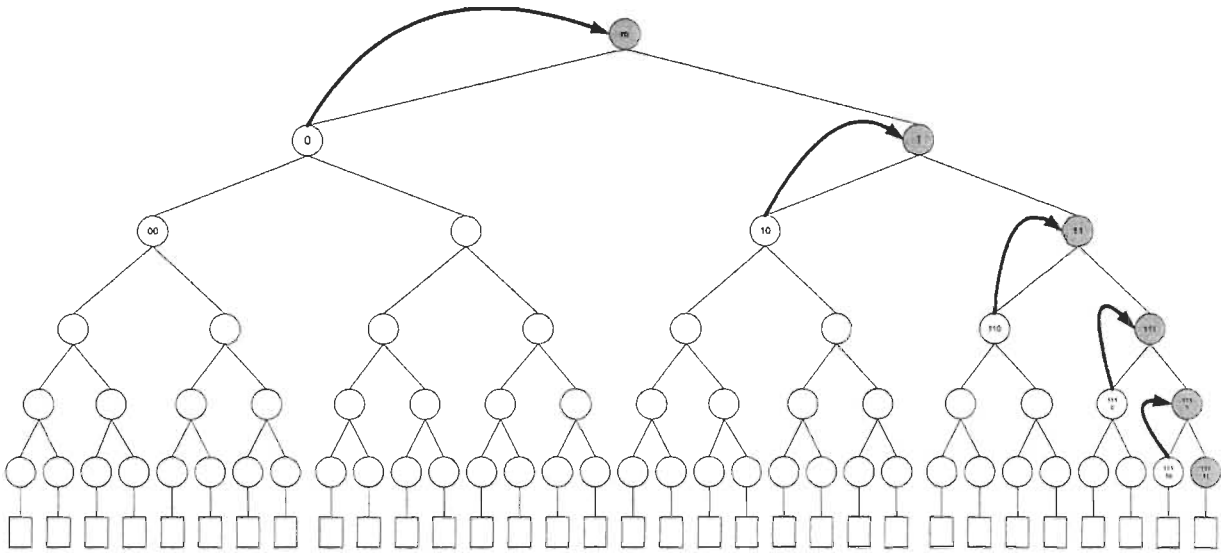


Figure 21: Clés glissées

Dans l'exemple de la figure 21, les clés en gris doivent être mise à jour. La clé utilisée pour obtenir la nouvelle valeur est choisie parmi les fils de gauche ou de droite de cette clé ; celle qui n'a pas été mise à jour est sélectionnée. Si par contre, les clés des deux fils étaient mise à jour, celle qui a été obtenue par une clé glissée sera choisie pour changer la clé (voir départs multiples). Pour la clé de groupe, la clé 0' est utilisée, pour la clé 1, la clé 10' est utilisée, et ainsi de suite. Les nouvelles clés devront être transmises aux participants restants, selon la méthode traditionnelle de transmissions des clés en arbre.

Km = g(0) 1 = g(10) 11 = g(110) 111 = g(1110) 1111 = g(11110)

3.2.7 Stockage

3.2.7.1 Key Distribution Center (KDC)

Chez l'entité KDC, toutes les clés ne doivent pas être reliées à tous les participants qui les connaissent. Avec le nom d'un participant, on peut connaître les clés qu'il connaît, sans avoir à garder cette information dans le tableau. Dans la table no 3, un tableau des clés contenant les valeurs est présenté.

Clé	Valeur
Km	...
0	...
1	...
...	...

Tableau 3: Tableau des clés

Chez l'entité KDC, les identifiants sont stockés dans le tableau ID, comme dans le tableau no 4, avec le nom des participants dans un champ et la valeur des identifiants qui leurs sont associés. La longueur du tableau correspond avec le niveau d'identifiants choisit.

Participant	Identifiant
0000	...
0001	...
00010	...
...	...

Tableau 4: Tableau des identifiants

3.2.7.2 Participants

Chez le participant, les clés sont stockées de la façon suivante : un tableau de K entrées contenant les clés (K étant le nombre de niveaux choisit). Ce tableau contient les clés que le participant connaît. Ces clés sont ordonnées selon leur rang dans l'arbre des clés. Si l'implémentation utilise les tableaux standards, l'indice déterminera la position de la clé et si l'implémentation utilise des tables de hachage, on pourra utiliser directement le nom de la clé pour trouver sa valeur.

Avec un groupe comme dans la figure 22, la clé Multicast (la racine) est stockée dans l'entrée 0 du tableau. Le fils de la racine occupe le rang 1. Le petit-fils le rang 2 et ainsi de suite. De cette façon le niveau sur lequel se situe la clé donne sa position. On peut retrouver facilement une clé avec sa **longueur**. Pour les exemples de ce document, ce tableau portera le nom de: `tableauDesCles`. Un tableau exemple est présenté avec le tableau no 5.

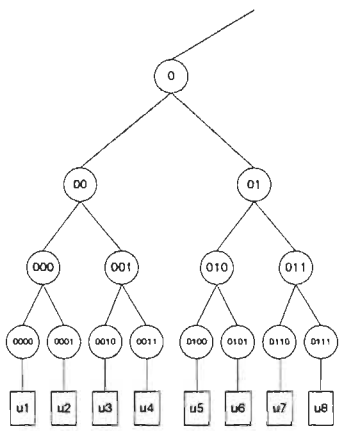


Figure 22: Une sous partie de l'arbre des clés

Indice	Nom de la clé	Valeur
0	m	…
1	0	…
2	01	…
3	011	…
4	0110	

Tableau 5 : Tableau des clés de u3

Chez tous les participants, les identifiants sont stockés dans le tableau ID, avec le nom des participants qui sont ses cousins dans un champ et la valeur des identifiants qui leurs sont associés, tel qu'illustré au tableau no 6. La longueur du tableau correspond avec le niveau d'identifiants choisit.

Participant	**Identifiant**
0000	…
0001	…
00010	…
…	…

Tableau 6: Tableau des identifiants

3.2.8 Opérations sur le groupe

3.2.8.1 Création du groupe

3.2.8.1.1 Opérations chez l'entité KDC

- Prépare un arbre qui servira à représenter les clés connues de chacun des participants
- Ajoute chacun des participants déjà inscrits au groupe, selon la méthode d'un ajout multiple
- Envoie aussi les identifiants pour les espaces vides, s'il y a lieu. Ces espaces seront comblés par les participants qui viendront se joindre au groupe par la suite.

3.2.8.2 Ajouts simples

3.2.8.2.1 Opérations chez l'entité KDC

- Ajoute le nouveau participant dans l'arbre
- Fabrique les nouveaux identifiants associés au nouveau participant
- Effectue un rafraîchissement des clés ayant besoin d'être changées
- Envoie un paquet Unicast KEY au nouveau participant, contenant les clés rafraîchies dans le champ KEY_VALUE
- Envoie un paquet Multicast ADDCOUSIN aux cousins du nouveau participant avec l'identifiant de celui-ci dans le champ ID, encrypté avec la clé commune à tous ses cousins

- Envoie un paquet Unicast ADDCOUSING au nouveau participant avec les identifiants des ses cousins dans le champ ID, et les noms de ceux-ci dans le champ NAME
- Envoie un paquet REFRESH de façon à ce qu'il n'y ait qu'un seul rafraîchissement des clés, avec les noms des nouveaux participants dans le champ NAME

3.2.8.2.2 Opérations chez les participants du groupe

- Avec un paquet Multicast REFRESH, il lit le nom du participant qui se joint au groupe dans le champ NAME et il rafraîchit les clés ayant besoin d'un rafraîchissement
- Avec un paquet Multicast ADDCOUSIN, il stocke le contenu du champ ID dans son tableau ID avec le nom du cousin en index

3.2.8.2.3 Opérations chez le nouveau participant

- Stocke les clés du champ KEY_VALUE du paquet Unicast KEY qu'il reçoit et les met dans son tableau de clés.
- Avec le champ ID du paquet Unicast ADDCOUSING qu'il reçoit, il stocke les identifiants de ses cousins dans un tableau d'identifiants et avec le champ NAME, et il copie le nom de ses cousins comme index du tableau ID.

3.2.8.3 Exemple d'ajout simple

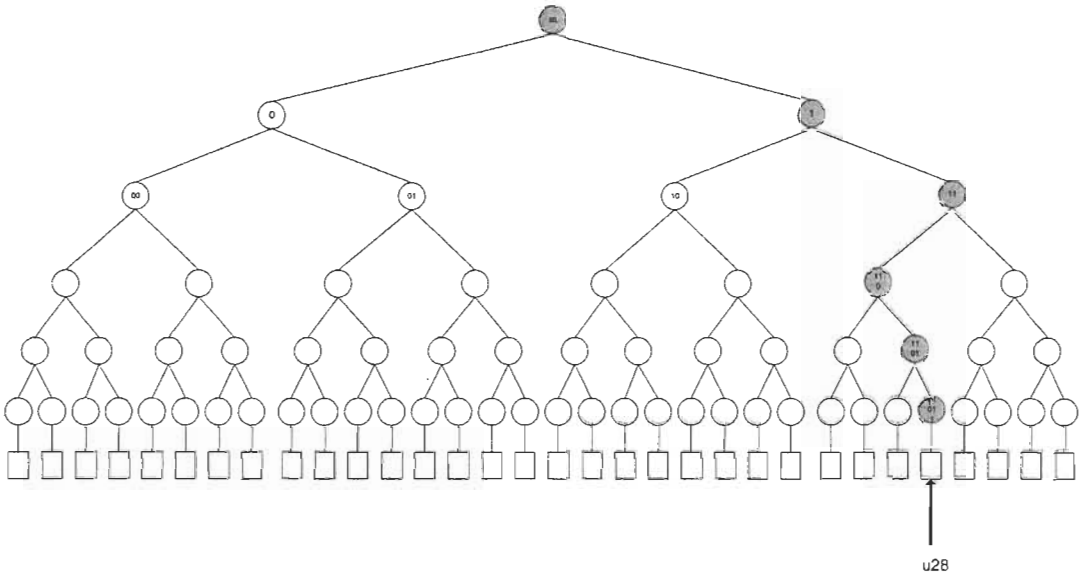


Figure 23: Ajout simple

Dans l'exemple exposé à la figure 23, le participant u28 a été ajouté au groupe. Les clés qu'il va connaître (en gris) vont être rafraîchies. L'entité KDC envoie un paquet Multicast REFRESH ayant dans le champ NAME le nom du nouveau participant, c'est-à-dire « 11011 ». En plus de la clé de groupe, les clés à rafraîchir portent donc le nom de : 1, 11, 110, et 1101. Chaque participant recevant le paquet REFRESH rafraîchit ses clés si elles sont comprises dans celles qui doivent être rafraîchies (en gris dans la figure 23).

Tout dépendant de la valeur de la quantité de voisins que u28 devrait avoir, il reçoit x identifiants pour ceux-ci avec leurs noms dans un paquet ADDCOUSING. Si l'entité KDC choisit des identifiants pour 2 niveaux, le participant u28 aura 3 voisins, donc 3 identifiants à stocker.

3.2.8.4 Ajouts multiples

3.2.8.4.1 Opérations chez l'entité KDC

- Pour chaque nouveau participant i faire
    - Ajouter les nouveaux participants dans l'arbre
    - Fabriquer des nouveaux identifiants associés au participant i
    - Effectuer un rafraîchissement des clés ayant besoin d'être changées
    - Envoyer un paquet Unicast KEY à i, contenant les clés rafraîchies dans le champ KEY_VALUE
    - Envoyer un paquet Multicast ADDCOUSIN aux cousins de i avec l'identifiant de i dans le champ ID, encrypté avec la clé commune à tous ses cousins
    - Envoyer un paquet Unicast ADDCOUSING à i avec les identifiants des cousins de i dans le champ ID, et les noms de ceux-ci dans le champ NAME
- participant i suivant
- Fin Pour
- Si le nombre de participants à ajouter est supérieur à un seuil
    - Envoyer un paquet REFRESH avec ALL dans le champ NAME de façon à ce qu'il n'y ait qu'un seul rafraîchissement des clés.
- Sinon

- Envoyer un paquet REFRESHG de façon à ce qu'il n'y ait qu'un seul rafraîchissement des clés, avec les noms des nouveaux participants dans le champ NAME

- Fin si

3.2.8.4.2 Opérations chez les participants du groupe

- Avec un paquet REFRESHG, il prend les noms des participants qui se joignent au groupe dans le champ NAME et il rafraîchit les clés ayant besoin d'un rafraîchissement.
- Avec un paquet Multicast REFRESH avec ALL dans le champ NAME, toutes les clés du participant sont rafraîchies.
- Avec un paquet Multicast ADDCOUSIN, il stocke le contenu du champ ID dans son tableau ID avec le nom du cousin en index.

3.2.8.4.3 Opérations chez les nouveaux participants

- Stocker les clés du champ KEY_VALUE du paquet Unicast KEY qu'il reçoit et les met dans son tableau de clés.
- Avec le champ ID du paquet Unicast ADDCOUSING qu'il reçoit, il stocke les identifiants de ses cousins dans un tableau d'identifiants et avec le champ NAME, il copie le nom de ses cousins comme index du tableau.

3.2.8.5 Exemple d'ajout multiple

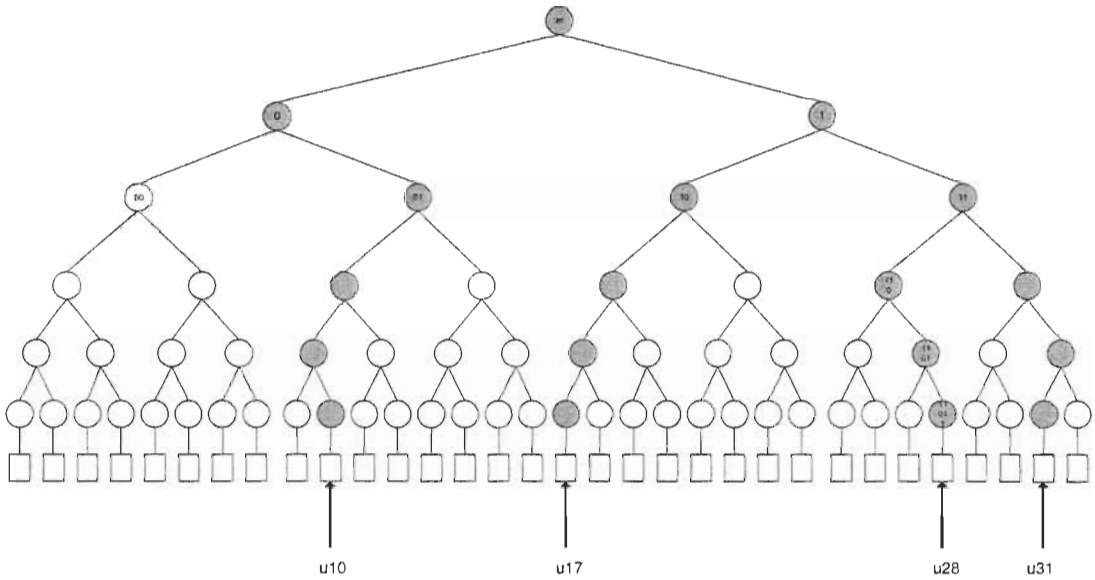


Figure 24: Ajout multiple

Dans l'exemple schématisé à la figure 24, les participants u10, u17, u28 et u31 ont été ajoutés au groupe. Les clés qu'ils vont connaître (en gris) vont être rafraîchies. L'entité KDC envoie un paquet Multicast REFRESHG avec dans le champ NAME les noms des nouveaux participants : 01001, 10000, 11011 et 11110 . En plus de la clé de groupe, les clés à rafraîchir portent donc le nom de : 0, 1, 01, 10, 11, 010, 101, 110, 0100, 1000, 1101, 1111 pour les participants déjà dans le groupe. Chaque participant recevant le paquet REFRESH rafraîchit ses clés si elles sont comprises dans celles à rafraîchir (en gris dans la figure 24).

Tout dépendant de la valeur de la quantité de voisins que chaque participant devrait avoir. Chaque participant reçoit x identifiants pour ses cousins avec leurs noms dans un paquet

ADDCOUSING. Si l'entité KDC choisit des identifiants pour 2 niveaux, chaque nouveau participant aura 3 voisins, donc 3 identifiants à stocker.

3.2.8.6 Départs simples

3.2.8.6.1 Opérations chez l'entité KDC

- Enlève le participant partant de l'arbre
- Envoie un paquet Multicast QUIT avec le nom du partant dans le champ NAME
- Effectue une mise à jour des clés ayant besoin d'être changées avec la méthode des clés glissées
- Pour chaque clé qui a été changée
    - Pour chaque participant ne connaissant pas cette clé
        - Envoie la clé encryptée avec la clé inférieure à la clé inconnue du participant
    - Prochain participant
    - Fin Pour
- Prochaine clé
- Fin Pour

3.2.8.6.2 Opérations chez les participants du groupe

- Avec un paquet QUIT, il lit le nom du participant qui quitte le groupe (le champ NAME) et il fabrique une liste (Uk) de toutes les clés qu'il doit mettre à jour, triée par la longueur du nom de la clé : de la plus grande à la plus petite.

/* Pour la technique des identifiants. */

- Pour chaque niveau d'identifiant
    - Pour chaque clé de Uk[]
        - Si le nom de la clé est identique à celui de son cousin
            - Affecter à la clé commune entre ce participant et à ce cousin, l'identifiant du partant
            - Supprimer la clé de l'ensemble Uk[]
        - Fin si
    - Prochaine clé
- Prochain niveau d'identifiant
- /* Uk[] : toutes les clés qui n'ont pas été rafraîchies en utilisant la technique des identifiants. */

/* Pour la technique des clés glissées. */

- Pour chaque clé de Uk[]
    - Si la clé fait partie de l'ensemble des clés du participant
        - Si la clé inférieure à cette clé ne fait pas partie de Uk[]

          /* Si la clé inférieure n'a pas été rafraîchie */

            /* on passe la clé dans une fonction à sens unique. */
            - la clé = g(clé inférieure de celle-ci)
            - Supprimer la clé de l'ensemble Uk[]
        - Sinon

            /* Si la clé inférieure a été rafraîchie. */

            - /* On doit recevoir la clé du KDC. */

- la clé = KEY (NAME)
  - Fin si
  - Fin si
- Prochaine clé

3.2.8.7 Exemple de départ simple

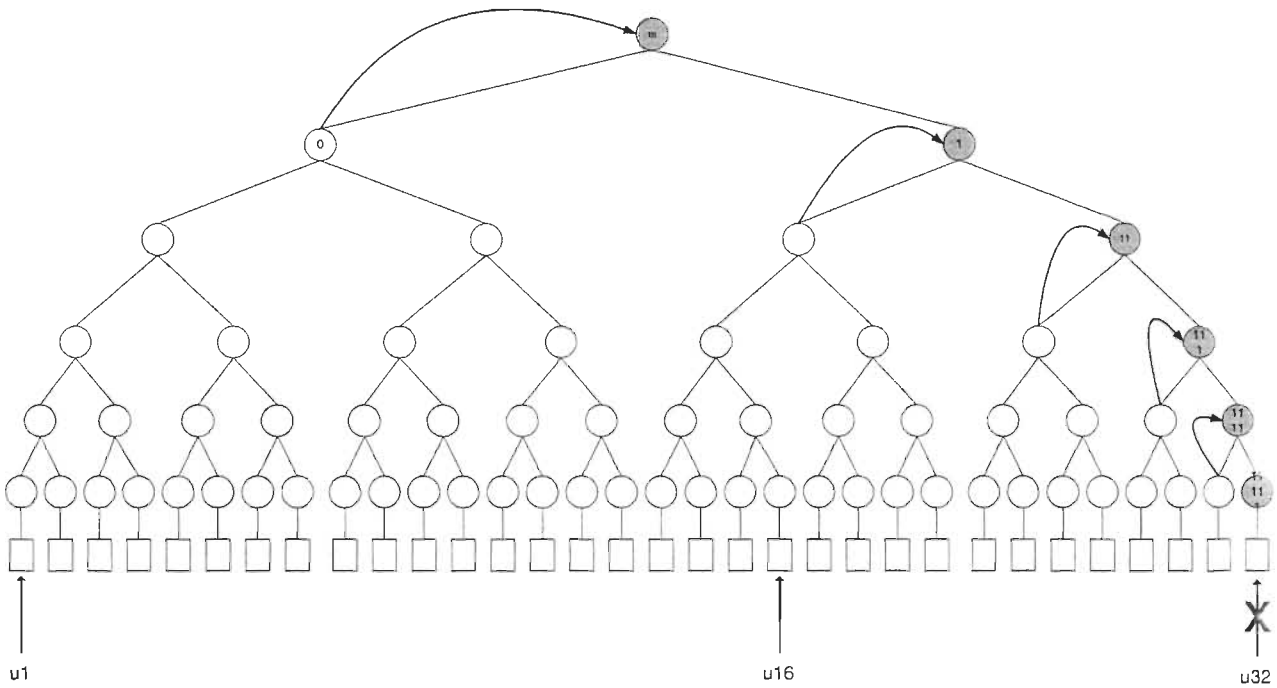


Figure 25: Départ simple

Avec un groupe comme illustré à la figure 25, dépendant du niveau d'identifiants, les clés 1111, 111, 11, et 1 pourraient être mise à jour avec les identifiants des différents niveaux associés à u32. Pour plus de détails, voir la section *3.2.5*.

La nouvelle clé de groupe est donnée par la technique de clé glissée : la clé 0, passée dans la fonction à sens unique, donne la nouvelle clé de groupe. Les participants u1 à u16 connaissent la nouvelle valeur et ne reçoivent pas la nouvelle clé. Ils sont exclus des opérations suivantes.

Peu importe le niveau d'identifiants utilisé, la clé supérieure immédiate de la clé personnelle du quittant est remplacée par la valeur transformée de celle de son frère immédiat. Dans l'exemple ci haut, 1111 = g(11110).

Si le niveau d'identifiants est à zéro, il y aura 4 niveaux utilisés avec les clés glissées. La clé 111 est mise à jour avec la clé inférieure à celle-ci qui ne nécessite pas de mise à jour, i.e. 111 = g(1110). La clé 11 est mise à jour avec la clé inférieure à celle-ci qui ne nécessite pas de mise à jour, i.e. 11 = g(110). La clé 1 est mise à jour avec la clé inférieure à celle-ci qui ne nécessite pas de mise à jour, i.e. 1 = g(10).

Tous les participants connaissant les clés inférieures qui ne nécessitaient pas de mises à jour (en blanc sur la figure 26) n'ont pas besoin de recevoir ces clés. Les participants qui ne les connaissaient pas, reçoivent une copie de la clé, encryptée avec la clé inférieure qu'ils connaissent. Par exemple, le voisin immédiat de u32, u31, reçoit {1}11'

Si le niveau d'identifiants est à un, il y aura 3 niveaux utilisés avec les clés glissées. La clé 111 est mise à jour avec l'identifiant du quittant, i.e. 111 = identifiant(11111). La clé 11 est mise à jour avec la clé inférieure à celle-ci qui ne nécessite pas de mise à jour, i.e.

11 = g(110). La clé 1 est mise à jour avec la clé inférieure à celle-ci qui ne nécessite pas de mise à jour, i.e. 1 = g(10).

Tous les participants connaissant les clés inférieures qui ne nécessitaient pas de mises à jour n'ont pas besoin de recevoir ces clés. Les participants qui ne les connaissaient pas, reçoivent une copie de la clé, encryptée avec la clé inférieure qu'ils connaissent. Par exemple, u31 reçoit {1}11'.

3.2.8.8 Départs multiples

3.2.8.8.1 Opérations chez l'entité KDC

- Pour chaque participant partant i faire
    - Enlève le participant partant de l'arbre
    - Copie le nom du participant dans une structure pour préparer le paquet QUITG.
    - Prochain participant
- Fin pour
- Envoie un paquet Multicast QUITG avec les noms des partants dans le champ NAME
- Effectue une mise à jour des clés ayant besoin d'être changées avec la méthode des clés glissées
- Pour chaque clé x qui a été changée

- Pour chaque participant ne connaissant pas la nouvelle valeur de cette clé x
  - Envoie la clé, encryptée avec la clé inférieure à la clé x que le participant connaît
- Prochain participant

- Prochaine clé

Opérations chez les participants du groupe

- Avec un paquet QUITG, il prend les noms des participants qui partent du groupe dans le champ NAME et il fabrique une liste (Uk) de toutes les clés qu'il doit mettre à jour et triée par longueur du nom de la clé de la plus grande à la plus petite.

/* Pour la technique des identifiants. */

- Pour chaque niveau d'identifiant
  - Pour chaque clé de Uk[]
    - Si le nom de la clé est identique à celui de son cousin
      - Affecter à la clé commune de ce participant et de ce cousin, l'identifiant du partant
      - Supprimer la clé de l'ensemble Uk[]
    - Fin si
  - Prochaine clé
- Prochain niveau d'identifiant

/* Uk[] : toutes les clés qui n'ont pas été rafraîchies en utilisant la technique des identifiants. */

/* Pour la technique des clés glissées. */

- Pour chaque clé de Uk[]
    - Si la clé fait partie de l'ensemble des clés du participant
        - Si la clé inférieure à cette clé ne fait pas partie de Uk[]

            /* Si la clé inférieure n'a pas été rafraîchie */

            /* on passe la clé dans une fonction à sens unique. */

            - la clé = g(clé inférieure de celle-ci)
            - Supprimer la clé de l'ensemble Uk[]
        - Sinon

            /* Si la clé inférieure a été rafraîchie. */

            /* On doit recevoir la clé du KDC. */

            - la clé = KEY (NAME)
        - Fin si
    - Fin si

Prochaine clé

3.2.8.9 Exemple de départ multiple

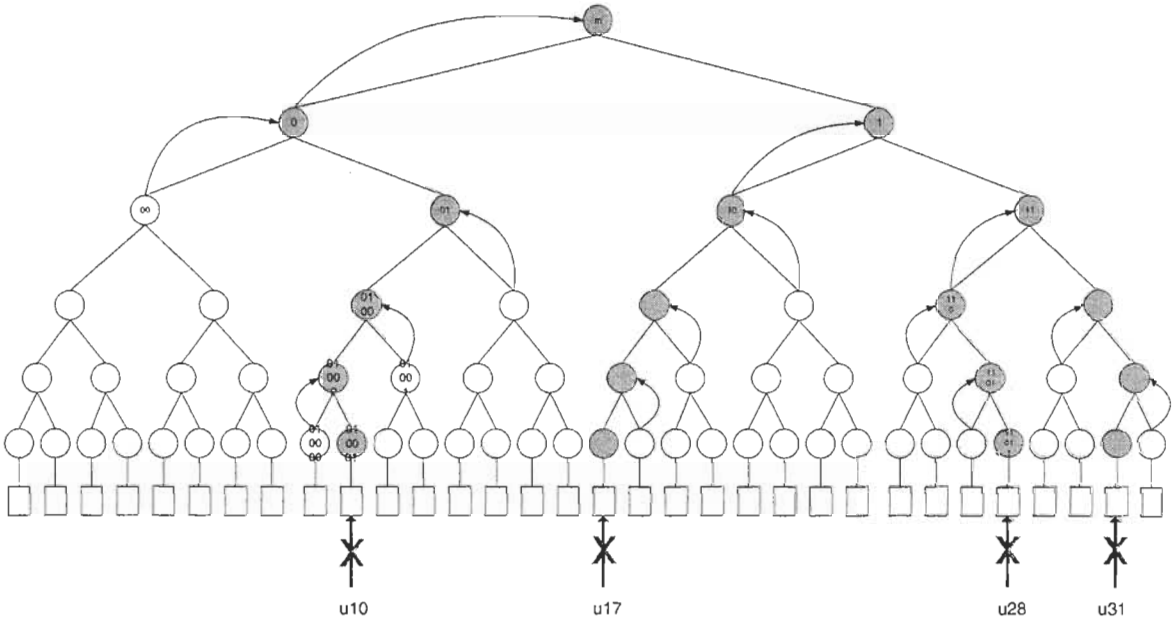


Figure 26: Départ multiple

Dans l'exemple de la figure 26, chaque clé en gris désigne les clés qui doivent être mise à jour. Dépendant du niveau d'identifiants choisi, les clés seront mises à jour selon les techniques des identifiants ou des clés glissées.

Dans l'exemple ci haut, la clé 0100 recevra la valeur de la clé 01000, peu importe le nombre de niveaux d'identifiants choisi.

La clé 010 :

Avec un niveau d'identifiant à 0, elle recevra la valeur transformée de la clé 0101, i.e. 010 = g(0101).

Avec un niveau d'identifiant de plus de 1, elle recevra l'identifiant du participant u10.

La clé 01 :

Avec un niveau d'identifiant à 1, elle recevra la valeur transformée de la clé 010', i.e. 01 = g(010').

Avec un niveau d'identifiant de plus de 1, elle recevra l'identifiant du participant u10.

La clé 0 :

Avec un niveau d'identifiant à 2, elle recevra la valeur transformée de la clé 00 (0 = g(00)).

Avec un niveau d'identifiant de plus de 2, elle recevra l'identifiant du participant u10.

Toutes les autres clés inférieures à la clé de groupe sont mises à jour de la même manière.

La nouvelle clé de groupe est donnée par la technique de clé glissée : la clé 0, passée dans la fonction à sens unique, donne la nouvelle clé de groupe. La clé de groupe sera acheminée aux participants u18 à u32 dans un paquet du type {m}1. Les participants u1 à u16 connaissent la nouvelle valeur et ne reçoivent pas la nouvelle clé.

3.2.8.10 Séparations

Lors de la séparation du groupe en plusieurs groupes, les clés des sous-groupes peuvent être utilisées comme clé de groupe dans chacun de ces groupes.

Dans l'exemple, si le groupe se décompose en deux groupes, les clés 0 et 1 seraient utilisées comme nouvelles clés de groupe.

3.2.8.11 Fusions

Lors de la fusion entre plusieurs groupes, chacune des clés de groupe est utilisée comme clés de sous-groupe et une clé de groupe est générée pour le nouveau groupe. La clé est ensuite envoyée, cryptée avec les clés de sous-groupes.

3.2.8.12 Perte de clés

Pour les clés générées à distance, les risques de perte de clés sont presque nuls, sauf en cas d'erreur de calcul dans la fonction à sens unique.

Pour les clés (de groupe et de sous-groupe) envoyées dans les paquets cryptés, les risques de perdre une clé sont égaux aux méthodes décrites dans la section « Travaux connexes ».

3.3 Conclusions

La technique que nous avons proposée vise à réduire le nombre de transmissions entre l'entité KDC et les participants lors de l'ajout et lors du départ d'un ou de plusieurs participants.

En utilisant la technique des identifiants, la quantité d'informations transmises entre l'entité KDC et les participants est réduite. Cette technique utilise des informations

partagées entre plusieurs participants sur d'autres participants sans que ceux-ci ne le sachent. En utilisant cette technique dans les arbres de clés, le nombre de participants connaissant les informations associées à un participant réduit le partage d'information et de ce fait, réduit la probabilité de collusion parmi les participants du groupe.

En utilisant la technique des « Clés glissées », la quantité de clés transmises entre l'entité KDC et les participants est réduite lors du départ d'un ou de plusieurs participants. Cette technique utilise des informations, dont des clés, qui sont connues des participants. Étant déjà connues de ces participants, ces informations n'ont pas besoin d'être transmises et peuvent donc servir à générer à distance des clés, et donc réduire le nombre de transmissions de clés.

J'ai proposé une nouvelle méthode pour nommer les clés qui facilite la compréhension des arbres de clés et leur gestion. La connaissance de toute la structure de la hiérarchie de clés n'étant plus nécessaire et ainsi le stockage est diminué.

Le chapitre suivant décrit la méthodologie utilisée pour réaliser les différentes simulations permettant d'évaluer les performances du protocole ROKEN-SKI.

CHAPITRE 4

MÉTHODOLOGIE DE SIMULATION

Pour implanter ma méthode, j'ai utilisé une technique de développement dans les recherches en réseautique : les simulateurs réseaux. Ces simulateurs permettent de diminuer les infrastructures nécessaires à la recherche en réseautique.

4.1 Les simulations réseaux

Lors de la création d'un nouveau protocole, il est intéressant d'évaluer la performance de ce protocole sur des réseaux différents et sur des architectures différentes, à moindre coût. Les simulateurs réseaux ont été créés à cet effet.

On peut définir une **simulation** comme l'exploration d'un modèle abstrait construit à partir d'observations du monde réel [24]. Une simulation est donc une expérimentation avec un comportement ou avec une architecture sous éléments contrôlés.

4.1.1 Buts

Les simulations réseaux nous permettent :

- d'observer un comportement dans divers scénarios s'il a été étudié à petite échelle et qui ne pourrait pas être observé dans un environnement à grande échelle [24].
- d'explorer des scénarios qui ne peuvent être observés de façon traditionnelle [24].

- de minimiser le coût de développement et de tests en cas d'erreur.

4.1.2 Déroulement

Le déroulement d'une simulation, comme illustré à la figure 27, ressemble énormément à la résolution de problèmes. Tout d'abord, il faut définir le problème : bien comprendre et avoir une vue d'ensemble du problème. Si les concepteurs d'un protocole sont perplexes vis-à-vis d'une fonctionnalité, d'un composant ou du comportement du protocole dans certains cas, des efforts doivent être déployés précisément dans ces domaines.

Une fois que le problème est bien cerné, il faut construire les modèles. Il faut transcrire les idées en diagrammes d'états, en algorithmes et en code pour ensuite l'implanter dans le simulateur réseau. Cette étape est très importante car il faut s'assurer que l'implémentation du protocole respecte la conception.

Lorsque l'implémentation est terminée, des simulations peuvent être effectuées. Des architectures doivent être définies et la solution (protocole, architecture, etc.) doit être appliquée à une ou plusieurs entités de l'architecture de simulation.

Si la solution est un nouveau protocole, des entités doivent être dotées de ce protocole pour l'utiliser. Des paramètres d'utilisation peuvent être réglés dans ce cas.

Si la solution est une nouvelle architecture, elle doit être ajoutée à l'architecture déjà en place. Des liaisons entre les différentes architectures existantes et la nouvelle doivent être créées.

Une fois les simulations effectuées, il faut analyser les résultats en vue de savoir si la solution répond aux attentes de ses concepteurs. Si les résultats ne sont pas corrects, l'équipe de conception doit revoir soit les paramètres de simulation, soit revoir la conception.

Après s'être assuré que la simulation est correcte, il s'agit de prendre les décisions par rapport aux interrogations de départ.

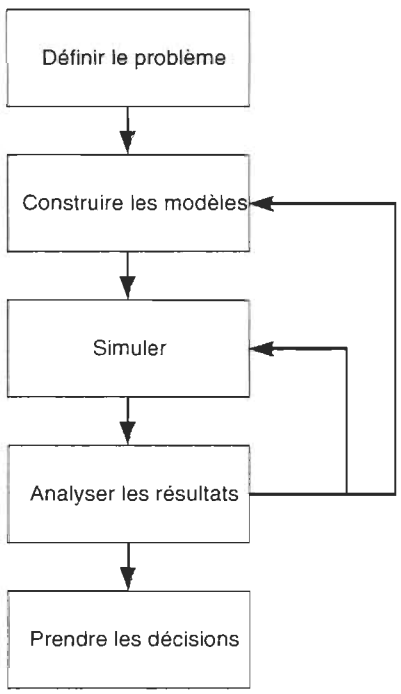


Figure 27: Étapes de la réalisation d'une simulation

4.1.3 Avantages des simulations réseaux

Lors du développement logiciel, les développeurs se retrouvent devant un certain nombre d'inconnues. Ces inconnues peuvent être rencontrées durant la période de conception, de développement et/ou d'intégration. Les simulateurs réseaux peuvent réduire les inconnues d'intégration, si le logiciel à développer utilise l'aspect réseau. En permettant de voir tout ce qui peut arriver et tous les scénarios possibles, l'équipe de conception pourra prendre des décisions face à certains problèmes.

Avec le développement d'une application dans un simulateur, les développeurs peuvent avoir une bonne idée de l'ampleur de l'implémentation réelle de l'application.

Dans certains cas, faire la trace des évènements se révèle trop fastidieuse et trop complexe pour être faite à la main. Dans ces cas, les simulateurs réseaux permettent de mieux visualiser et de mieux comprendre des protocoles ou des applications.

Lors de la mise en marché d'un protocole ou d'une application réseau, il se peut que ce protocole ou cette application ait un comportement inadéquat dans certaines situations. Lorsqu'il sera utilisé par un bon nombre d'entités, il se peut que ces défectuosités entraînent des bogues ou des problèmes sérieux aux architectures existantes. Ces « success disasters », comme la première version de http, sont déjà survenus par le passé mais tendent à disparaître grâce aux outils de développement de plus en plus efficaces et entre autre, grâce aux simulateurs.

4.1.4 Inconvénients des simulations réseaux

Un des inconvénients des simulations réseaux est qu'elles sont limitées à des architectures existantes. Si une architecture n'existe pas ou si les développeurs du simulateur réseau n'ont pas inclus une architecture (routeur, etc.), les utilisateurs ne peuvent pas l'utiliser. L'architecture doit donc être implémentée, si c'est possible.

Un autre inconvénient est qu'il est facile de tomber dans une trop grande simplification des différents problèmes. Les difficultés sont parfois difficiles à surmonter et prennent beaucoup de temps à résoudre. Étant donné l'écart entre les simulations et les situations réelles, il est facile de perdre de vue son objectif quant aux décisions à prendre. Les concepteurs doivent tester tous les cas et ne doivent pas se soustraire à un scénario particulier.

Lorsque l'on utilise des simulateurs, il est aussi facile de se restreindre à n'utiliser qu'un type d'architecture. Les concepteurs doivent respecter l'hétérogénéité qui existe au sein des réseaux et au niveau d'Internet. Les difficultés proviennent souvent d'architectures différentes de celles testées; respecter l'hétérogénéité permet donc de représenter un modèle plus près de la réalité.

4.1.5 Difficultés de représenter Internet

À cause de différents facteurs, la modélisation des situations réelles où des réseaux sont présents est souvent difficile à reproduire. Une bonne modélisation est celle qui saura intégrer les différences des réseaux existants et les conventions déjà établies pour créer un monde virtuel proche de la réalité.

Des facteurs comme les changements aux architectures et aux protocoles représentent un obstacle à une modélisation générique, utilisable dans tous les scénarios. L'Internet change très vite et les recherches dans le monde des réseaux sont nombreuses. Il est donc important d'effectuer une mise à jour régulière des simulateurs utilisés. Des nouveaux logiciels capables de bouleverser les réseaux existants peuvent représenter des inconnus pour l'avenir des réseaux.

Internet a été créé pour joindre des réseaux très hétérogènes. Des topologies différentes sont reliées entre elles et une interaction entre des piles de protocoles différentes sont possible (TCP/IP, IPX/SPX, Appletalk, etc). Les concepteurs utilisant des simulateurs doivent prendre en compte ces différences.

4.2 Les simulateurs réseaux

Deux simulateurs sont plus reconnus par la communauté de recherche en réseautique : Network Simulator, ou NS, et OPNET. Le tableau no 7 exprime les différences entre ces deux simulateurs.

	Network Simulator	**OPNET**
Langage	TCL, OTCL	ProtoC ~ C/C++
Entrée	Fichier texte passé en paramètre	Environnement graphique ou importation d'un fichier
Édition de noeud	Oui	Oui

Édition de réseau	Oui	Oui
Édition de comportement	Oui	Oui
Plate-formes	Unix, Linux, Windows	Linux, Windows 2000 et NT
Coût	Gratuit	Licence univ. gratuite

Tableau 7: Comparaison des deux principaux simulateurs réseaux

4.3 Simulateur utilisé

Pour les fins de mes recherches, j'ai choisi le simulateur Opnet. Le module Modeler étant développé pour la recherche et le développement, il a été choisi comme logiciel de développement pour l'évaluation de ma technique.

4.4 Implémentation

Pour bien commencer l'implémentation de ma technique dans le simulateur Opnet, je l'ai séparé en trois parties principales. La première étant l'apprentissage du simulateur Opnet, la deuxième le codage en C++ et la troisième, l'intégration du code dans le simulateur.

4.4.1 Étude du logiciel de simulation Opnet

Le logiciel de simulation Opnet est très vaste et très complexe. Plusieurs mois ont été nécessaires pour arriver à bien le comprendre. La partie qui apparaissait la mieux adaptée pour intégrer la technique était les *Generic Network Application*, ou GNA.

Les GNA permettent de faire un lien entre des définitions d'application générales configurables à l'aide de menus dans le logiciel et les diagrammes d'états définis avec de la programmation. Lorsque les GNA sont spécifiés avec des diagrammes d'état (Process

Models), les utilisateurs finaux peuvent utiliser les protocoles définis par le programmeur. De cette façon, l'utilisation et le paramétrage spécifique à une situation particulière deviennent facile.

4.4.2 Implémentation en C++

Dans le simulateur, le code ProtoC est utilisé. Comme il est l'équivalent du code C/C++ avec des mots clés, du code C/C++ peut être utilisé dans le simulateur. Comme l'environnement de développement VisualC++ permet de mieux déboguer un programme, j'ai donc choisi de prendre cette plate-forme en premier.

Le programme en C++ utilise plusieurs classes. En plus de deux classes pour les entités KDC et émetteur, le projet utilise une classe générateur pour la génération des nombres aléatoires. Ces nombres seront gérés par des lois de distributions dans le simulateur.

Le programme simule l'ajout et le retrait de participants à un groupe. La technique des arbres binaires hiérarchiques (HBT [12]) et ma technique (RS ou ROKEN-SKI) ont été codées pour les comparer entre elles. Le nombre de niveaux d'identifiants, le nombre de participants et le nombre de simulations peuvent être spécifiés en début de programme et le temps de l'ajout et du retrait sont tous gérés par des nombres aléatoires.

4.4.3 Implémentation sous OPNET Modeler

Le transfert de l'implémentation de ma technique à partir du code C++ de l'environnement Visual Studio vers le simulateur Opnet a été facile. Le simulateur Opnet

utilisant du C/C++ pour les instructions, le transfert des calculs et opérations a été aisé. La partie difficile à transférer est les interactions avec le système. Les demandes de saisies et d'affichage ne sont pas évidentes. Souvent, une seule façon est autorisée pour interagir avec le système.

La principale difficulté a été de trouver la façon de faire pour envoyer et recevoir des paquets de données entre deux entités. Après plusieurs mois de travail et plusieurs voies infructueuses de recherche de cette information, j'ai trouvé la façon de faire. Deux fonctions, une pour l'envoi et une pour la réception, ont été créées.

Ces méthodes sont génériques et peuvent être utilisées par n'importe quel objet réseau du simulateur. Ces fonctions utilisent la communication systèmes pour transférer les informations à la couche TCP/IP du simulateur. De cette façon, les paquets sont paramétrables et sont encapsulés dans des paquets IP et ne nécessitent pas de paquets spéciaux. Ces fonctions sont inclues en annexe.

4.4.3.1 Implémentation du KDC sous OPNET Modeler

Le diagramme d'état de l'entité « Client Server Manager » (le fichier clsvrmgr.pr.c) a été modifié pour permettre l'utilisation de ma technique avec le simulateur. Le gestionnaire de connexion client-serveur est devenu le gestionnaire des clés (l'entité KDC). Le diagramme d'état du gestionnaire est présenté à la figure 28. Dans le diagramme d'état, une dérivation a été créée pour permettre l'utilisation de la technique ROKEN-SKI. Une vérification est effectuée lorsqu'un paquet arrive au serveur. Le paquet est ouvert et si le

champ TYPE_DE_PAQUET contient des types de paquets utilisés par ma méthode, il est envoyé vers les états JOIN ou QUIT.

Pour permettre l'utilisation de ma technique et de permettre d'utiliser le simulateur Opnet de façon régulière, une condition a été introduite. L'utilisation de la méthode ROKEN-SKI avec le simulateur est limitée par l'existence du fichier `c:\rs.txt`. Si le fichier n'existe pas, le simulateur utilise les fichiers comme à l'habitude.

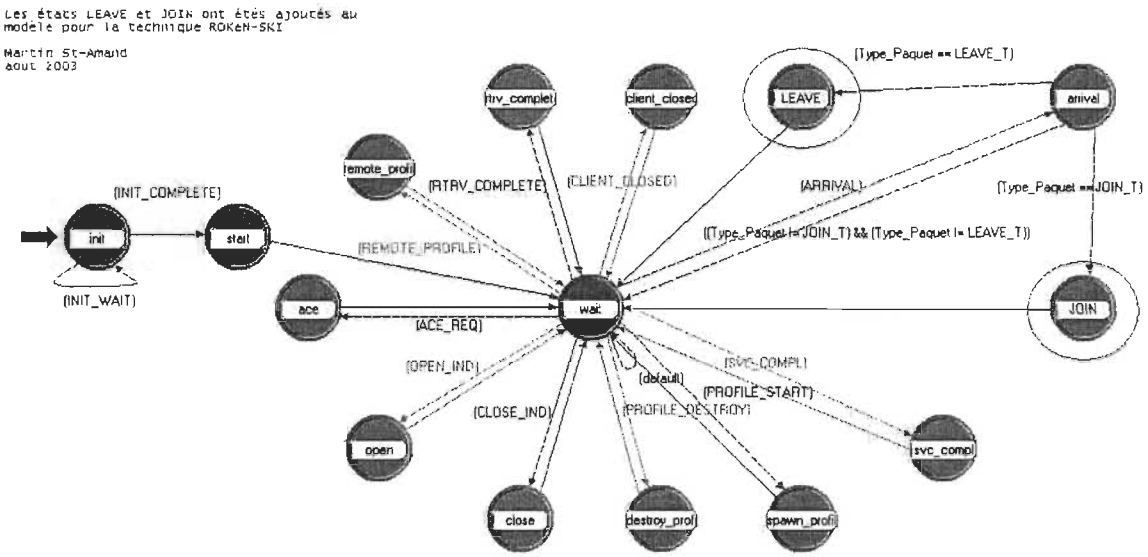


Figure 28: Diagramme d'état pour le serveur (KDC)

4.4.3.2 Implémentation de l'entité participant sous OPNET Modeler

Le diagramme d'état de l'entité «Client HTTP» (le fichier http_client.pr.c) a été modifié pour permettre l'utilisation de ma technique avec le simulateur. Le client de connexion HTTP est devenu l'entité participant. Le diagramme d'état de l'entité participant est présenté à la figure 29. Dans le diagramme d'état, une dérivation a été créée pour permettre l'utilisation de la technique ROKEN-SKI. Lors de la connexion, le diagramme envoie un paquet JOIN au KDC. Lorsque la session se termine, l'entité envoie un paquet QUIT au KDC.

Pour créer une connexion avec ROKEN-SKI, l'utilisateur du simulateur doit créer une demande d'application http et la simulation utilisera ma technique dans le simulateur. Pour permettre l'utilisation de ma technique chez les participants, la même condition qu'avec l'entité KDC doit être vérifiée. Si le fichier `c:\rs.txt` n'existe pas, le simulateur utilise les fichiers comme à l'habitude.

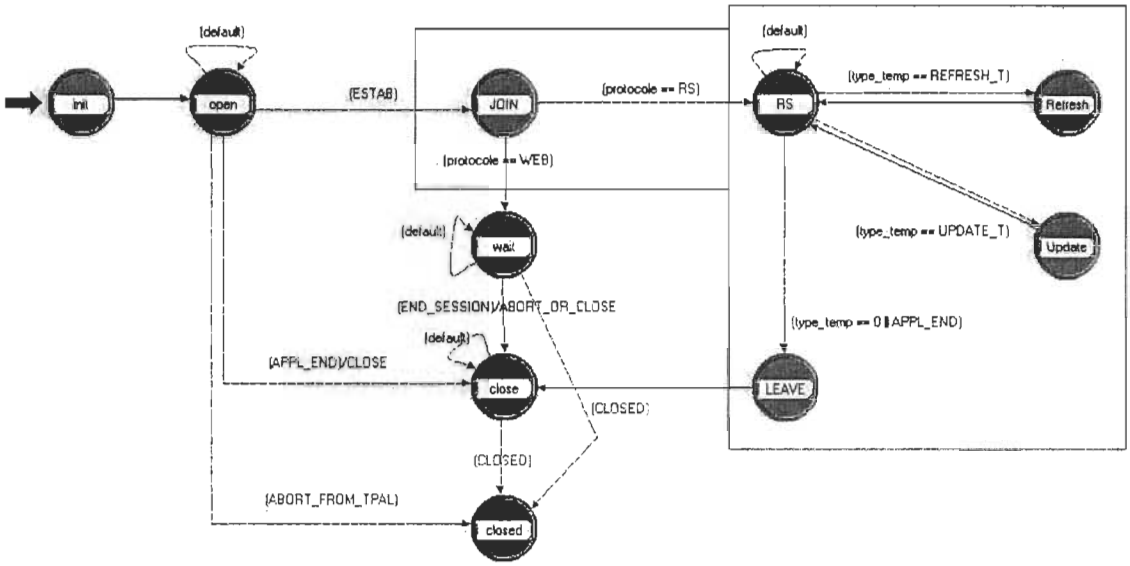


Figure 29: Diagramme d'état pour le client (participant)

4.5 Conclusion

Les simulations réseaux permettent donc de tester un protocole dans certaines conditions, tout en ayant un temps d'implémentation et un coût moindre qu'avec des tests effectués de manière classique. Lorsque les simulations sont bien effectuées, les résultats sont réalistes.

Le simulateur OPNET Modeler a été choisit. La méthode des GNA doit être utilisée pour bien implanter les protocoles de façon à respecter les spécifications de la pile TCP/IP. Les GNA sont configurables à l'aide de menus et permettent de la programmation à l'aide des diagrammes d'états. C'est donc grâce à ces GNA que j'ai pu implanter ma méthode et la méthode HBT pour les comparer.

Le chapitre suivant présente les différents résultats de simulation. Une comparaison entre les méthodes HBT [12][16] et ROKEN-SKI est détaillée.

CHAPITRE 5

RÉSULTATS

Dans ce chapitre, je présente les résultats obtenus avec les simulation effectuées. Deux types de simulations ont étées utilisées. Dans la première, j'ai utilisé une plate-forme C++ et dans la seconde, la plate-forme OPNET Modeler.

5.1 Résultats des simulations en C++

Pour bien vérifier le comportement de ma technique, j'ai utilisé un groupe de participants dont le nombre variait entre 500 et 30 000 participants. Pour assurer un bon affichage des résultats, seulement les groupes de 1000 à 30 000 seront affichés. Les simulations avec le code en C++, ont donné les résultats suivants.

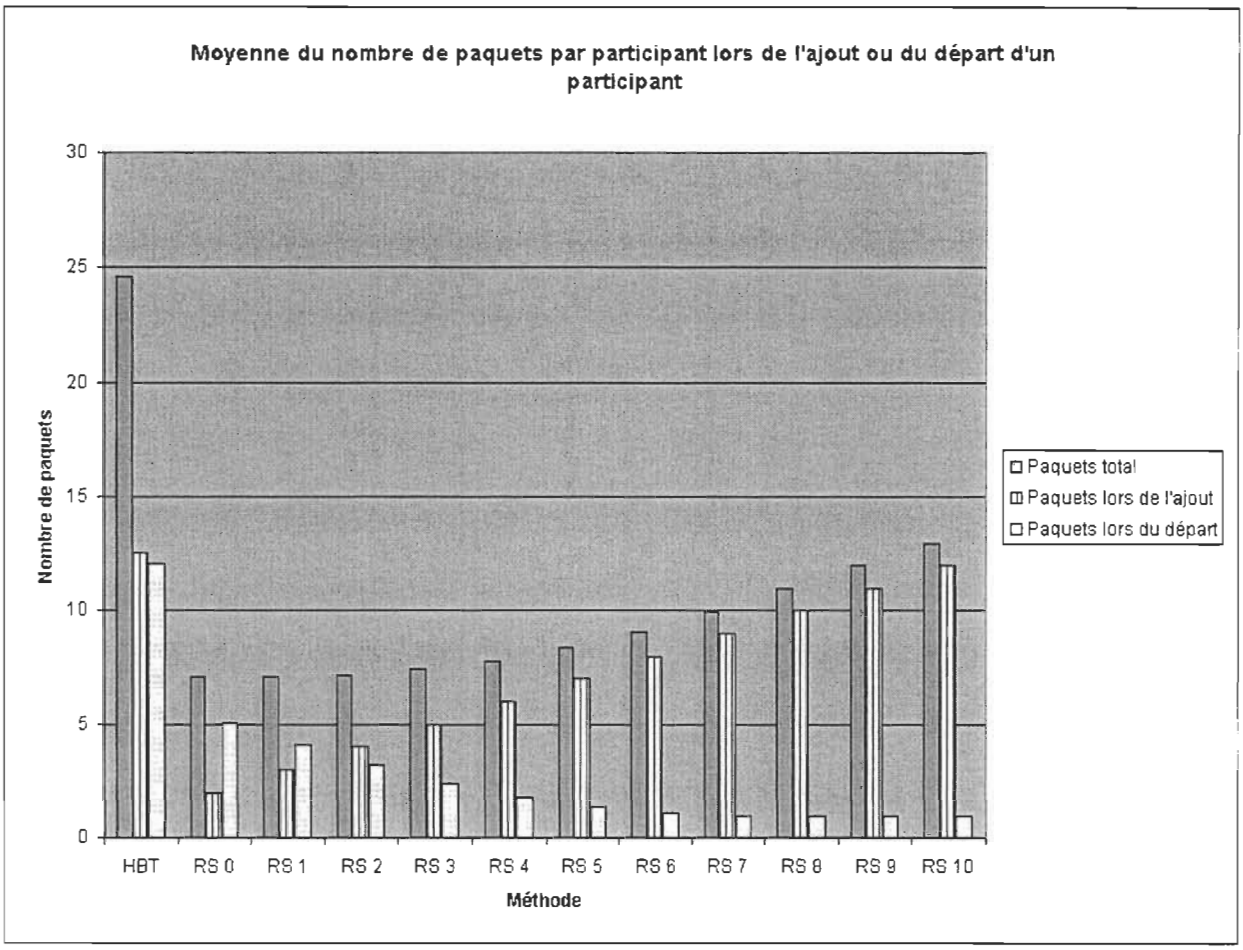


Figure 30: Nombre de paquets échangés lors de l'ajout et du départ d'un participant

Avec les résultats de la figure 30, il est clair que la méthode ROKEN-SKI :

1. utilise moins de paquets lors du départ d'un participant;

2. plus le nombre de niveaux augmente et plus la quantité de paquets échangés augmentent de plus en plus à partir de 5 niveaux.

Ces simulations en C++ permettent d'affirmer :

1. que dans l'intervalle de 2 et 5 niveaux, le nombre de paquets échangés lors de l'ajout et du départ sont mieux répartis;

2. la surcharge réseau sera moins grande si les deux sont égales.

Pour les simulations dans Opnet, la méthode ROKEN-SKI de niveau 3 sera utilisée et comparée à la méthode HBT.

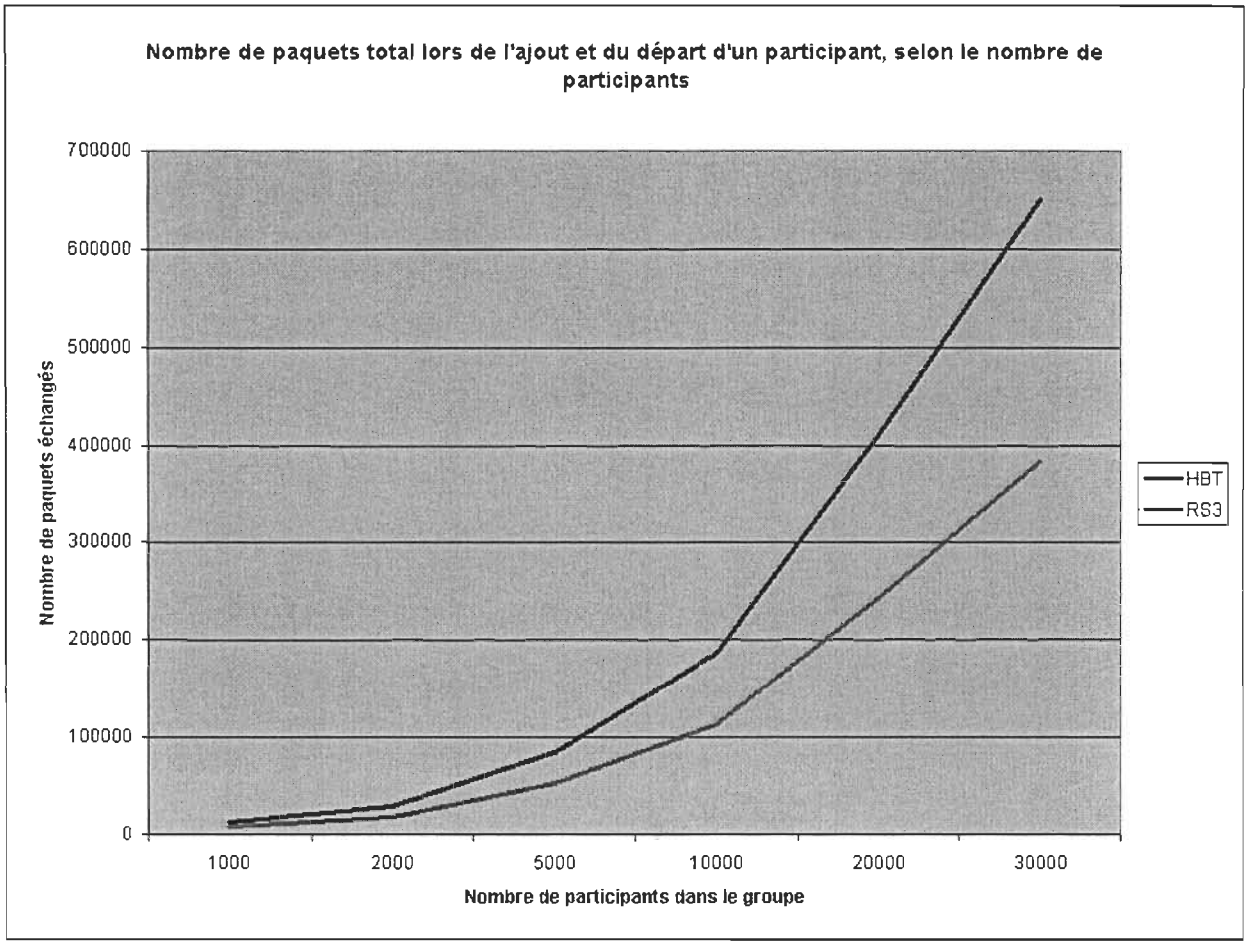


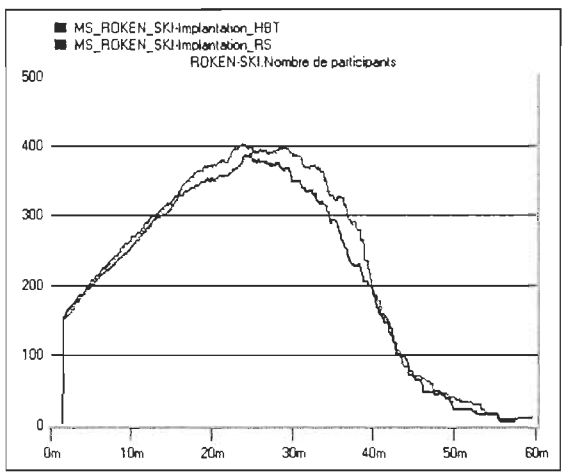
Figure 31: Nombre de paquets total selon le nombre de participants

Comme on peut le remarquer sur la figure 31, la méthode ROKEN-SKI est échelonnable. Même avec un grand nombre de participants, ma méthode conserve ses propriétés.

5.2 Résultats des simulations avec OPNET Modeler

5.2.1 Vérifications des paramètres de simulation

Pour s'assurer que le protocole HBT et la méthode ROKEN-SKI ont été utilisées dans des situations semblables, il fallait garder une trace des paramètres avec lesquels les simulations ont été effectuées. Les 4 prochains graphiques (figures 32, 33,34 et 35) affichent ces paramètres. **Il est à noter que l'axe des x représente le temps de la simulation et l'axe des y représente la fréquence**. Une implémentation parfaite aurait fait coïncider les deux courbes. Comme l'implémentation utilisée était bonne, il est donc difficile de distinguer les deux courbes.



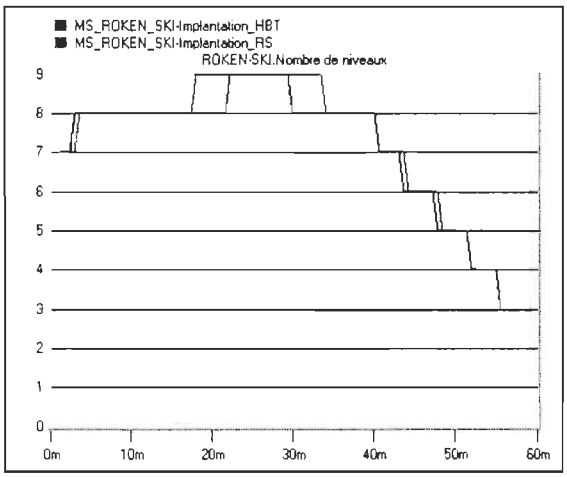
RS HBT

Figure 32: Nombre de participants selon le temps

Le nombre de participants a été définit par une loi normale de même paramètres dans les deux méthodes à comparer. Les deux méthodes ont le même nombre de participants tout au long des simulations. Le nombre de participants dans le groupe détermine le nombre

de niveaux dans l'arbre des clés. Le nombre de participants dans le groupe est illustré par la figure 32.

Le prochain graphique (figure 33) illustre les arbres des deux méthodes. Avec les différences au niveau du nombre de participants, il est normal de constater une différence du nombre de niveaux.



RS HBT

Figure 33: Nombre de niveaux de l'arbre des clés

Pour déterminer le moment auquel se connectera un participant, une loi normale a aussi été utilisée. Aussi, la même méthode a été utilisée pour la durée d'une session d'un participant dans le groupe. Le nombre d'ajouts au groupe et le nombre de départs des deux méthodes sont semblables et sont donc bien contrôlées.

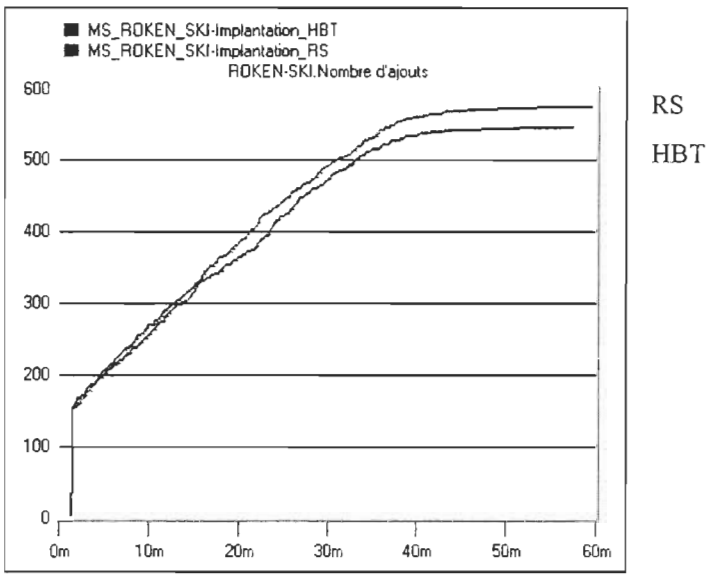


Figure 34: Nombre d'ajouts de participants

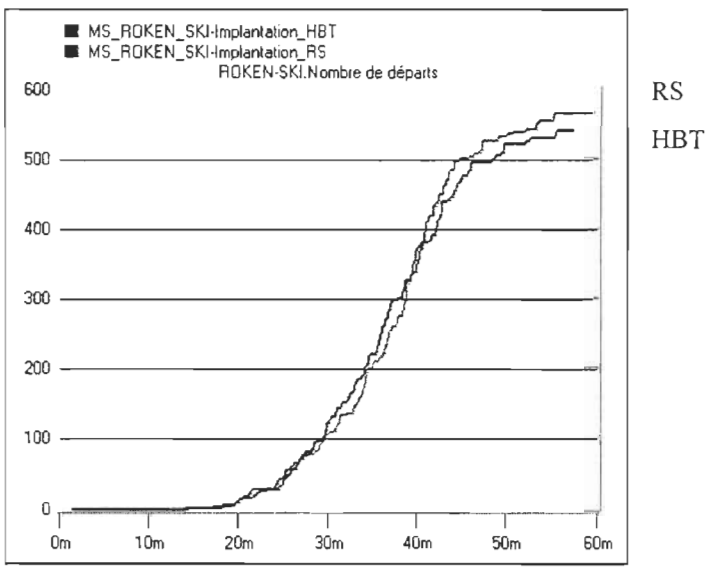


Figure 35: Nombre de départs de participants

Avec les graphiques 34 et 35, il est clair que les deux méthodes ont été utilisées dans des conditions semblables et peuvent donc être comparées entre elles.

5.2.2 Comparaisons entre la méthode HBT et ROKEN-SKI

5.2.2.1 Introduction

Plusieurs paramètres de comparaison existent pour comparer deux méthodes affectées au Multicast. Le premier est le nombre de paquets échangés, le deuxième est la demande en calcul et le troisième est le temps de connexion.

5.2.2.2 Comparaisons des paquets échangés

Pour le nombre de paquets échangés, ce nombre comprend le nombre de paquets Multicast et Unicast. Les paquets échangés, tel qu'illustré dans la figure 36, ont été capturés lors de l'ajout et du retrait de participants. Le nombre de paquets échangés lors de la diffusion d'informations n'étant pas nécessaire, il n'a pas été simulé.

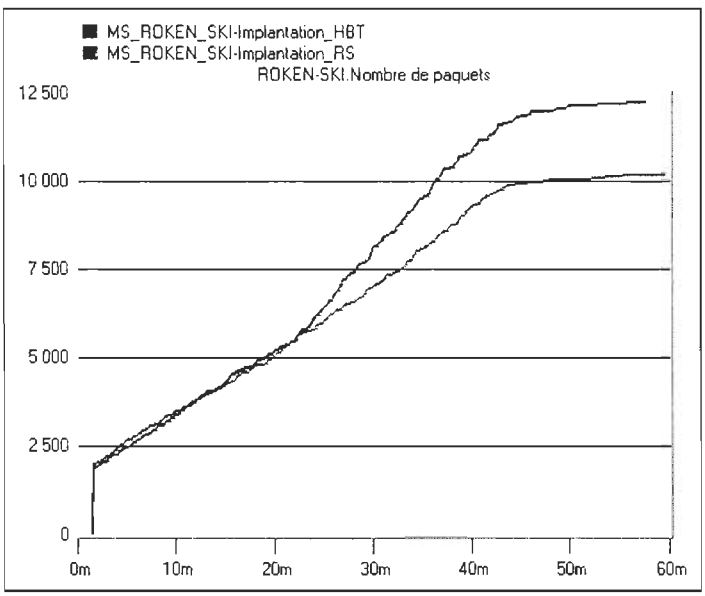


Figure 36: Nombre de paquets échangés

Lors de la simulation, le nombre de paquets échangés était semblable, jusqu'à un certain point. Mais, plus le nombre de participants est grand et plus la différence du nombre de paquets échangés est grande.

On peut aussi détailler cette comparaison avec la quantité de messages Unicast et Multicast échangés. Envoyer un paquet Unicast est très coûteux comparé à un message Multicast. Le but visé est de réduire le plus possible la quantité de paquets Unicast échangés.

Lors des simulations, il s'est avéré que le nombre de paquets Unicast était beaucoup plus grand lorsque la méthode HBT était utilisée, comme démontré à la figure 37. Ce résultat permet de dire que la méthode HBT est moins efficace lors de l'acheminement des clés que la méthode ROKEN-SKI, même si le nombre de paquets Multicast est plus grand avec la méthode ROKEN-SKI, comme démontré à la figure 38 les ressources réseaux demeurent moindres avec un paquet Multicast qu'avec un paquet Unicast.

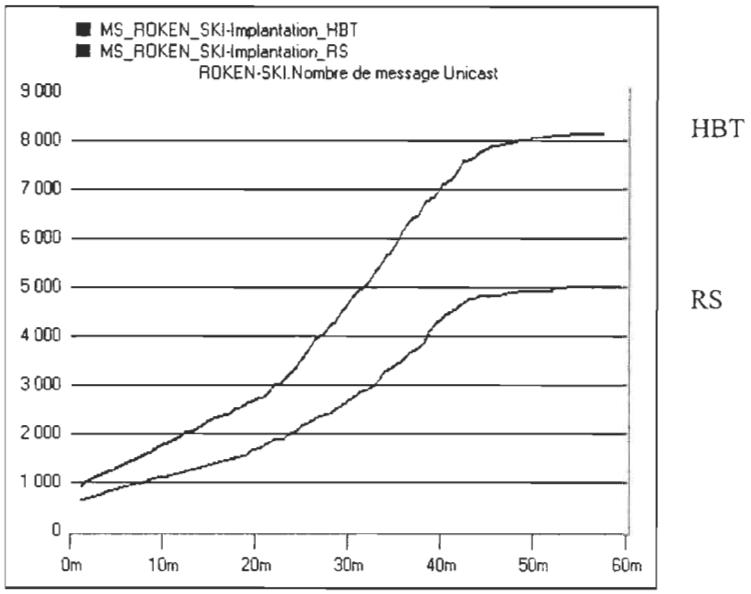


Figure 37: Nombre de paquets Unicast

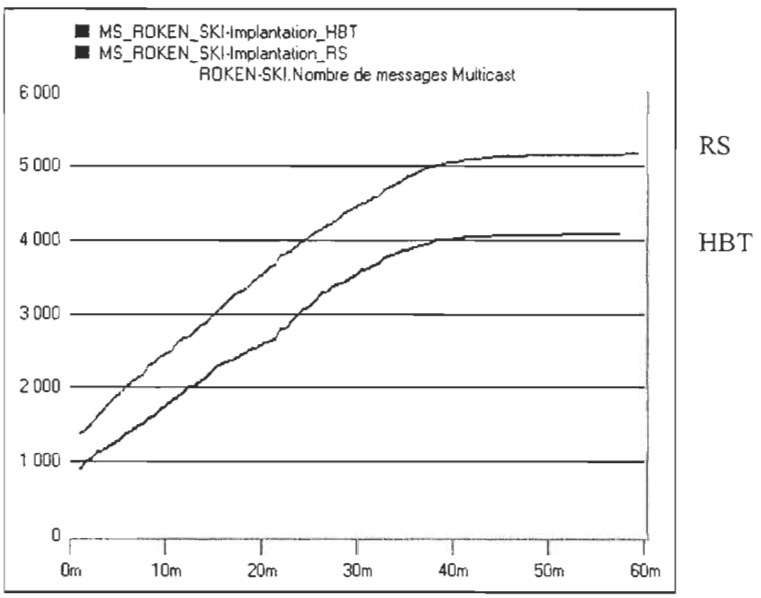


Figure 38: Nombre de paquets Multicast

Lors de l'ajout et du retrait d'un participant, il est important d'analyser ce qui se passe pour bien comprendre les enjeux. Un des aspects important de ma technique étant de

balancer les échanges de paquets entre les ajouts et départs, les graphiques 39 à 42 sont très intéressants et montrent la fiabilité de ma technique.

Lors de l'ajout d'un participant, le nombre de paquets échangés est supérieur avec la méthode HBT. Même si le nombre de paquets Unicast est moins grand, la somme des paquets Unicast et Multicast est plus importante qu'avec ma méthode. La figure 39 montre la quantité de paquets échangés lors de l'ajout.

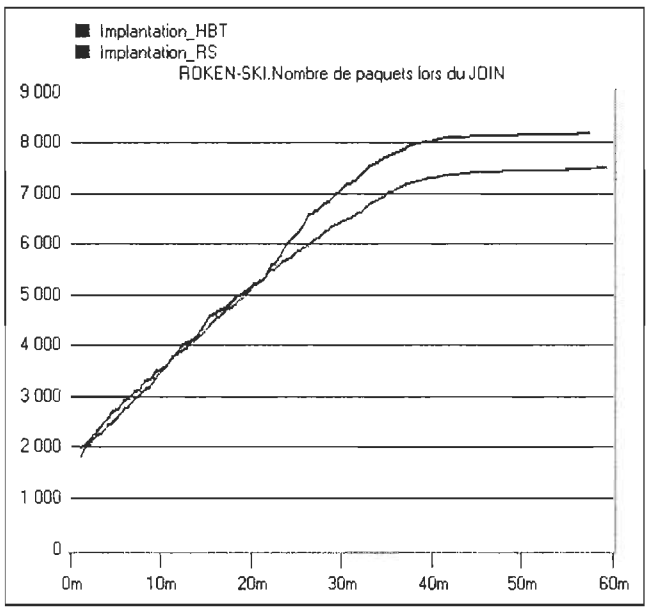


Figure 39: Nombre de paquets échangés lors de l'ajout

Le nombre de paquets Unicast lors de l'ajout d'un participant, tel qu'illustré à la figure 40, est plus grand avec ma méthode. Cette différence est due à l'envoi des identifiants lors de l'ajout d'un participant. Les identifiants reliés à celui-ci sont envoyés dans des paquets Multicast mais il doit en recevoir quelques-uns de façon Unicast. Les paquets Multicast échangés lors de l'ajout sont illustrés dans la figure 41.

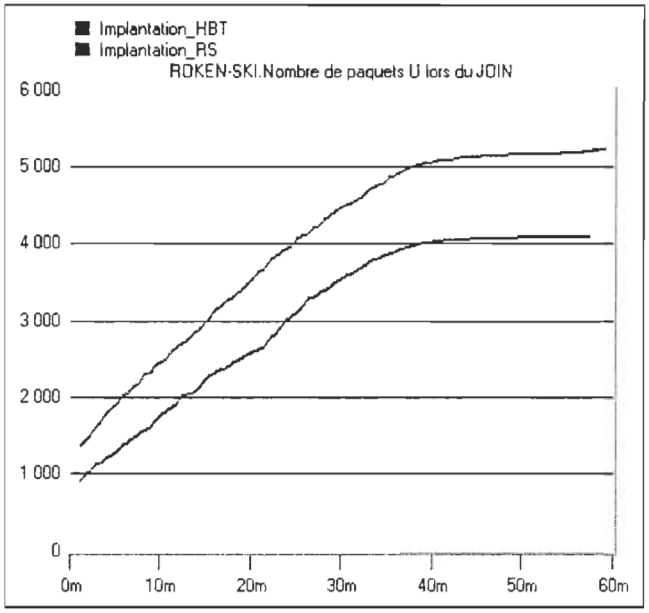


Figure 40: Nombre de paquets Unicast lors de l'ajout

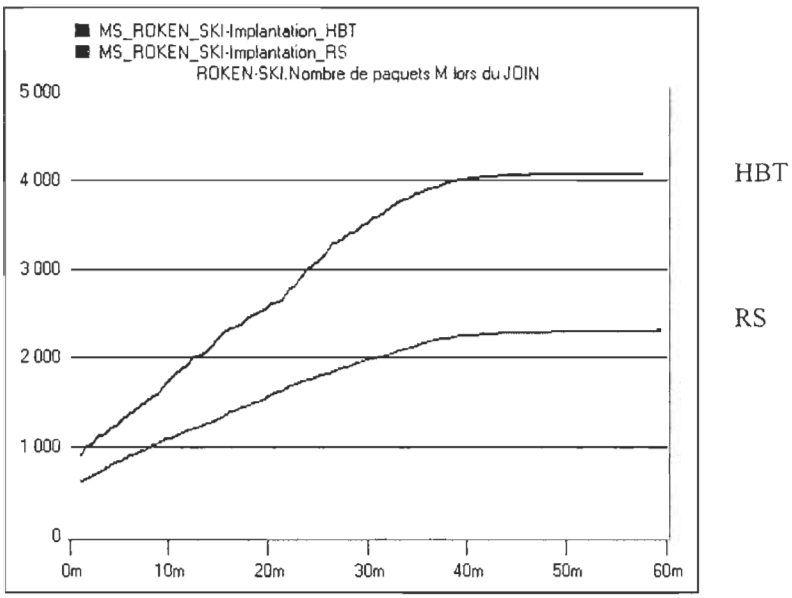


Figure 41: Nombre de paquets Multicast lors de l'ajout

Lors du départ d'un participant, le nombre total de paquets échangés pour changer les clés est moins grand avec la méthode ROKEN-SKI qu'avec la méthode HBT, tel que démontré dans la figure 42. Ceci permet de moins congestionner les réseaux utilisés et de moins surcharger l'entité KDC.

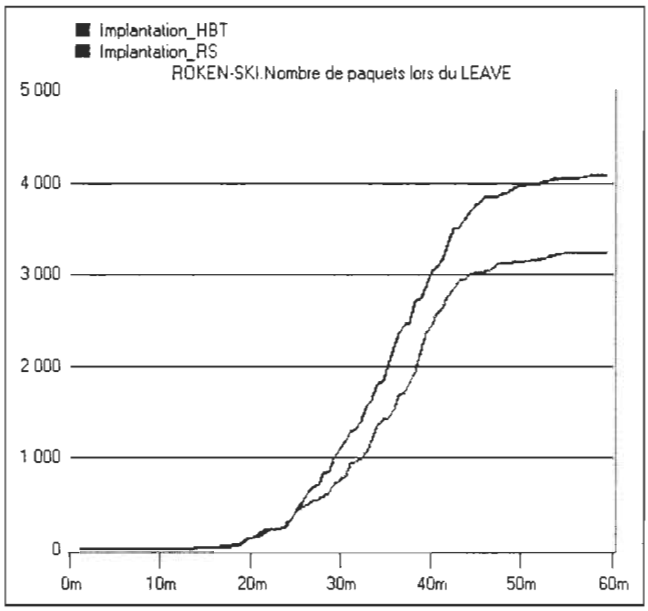
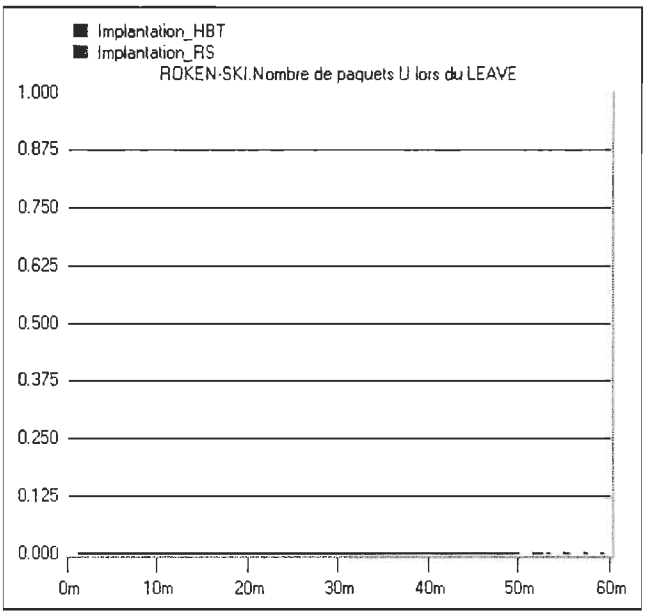


Figure 42: Nombre de paquets lors du départ

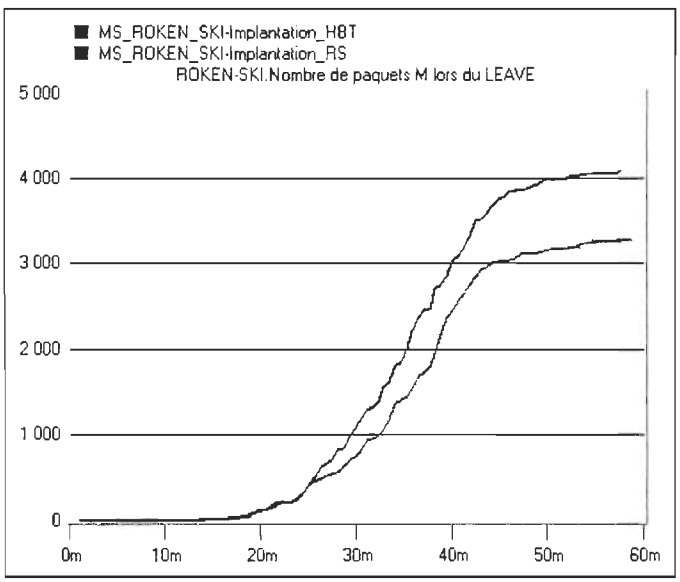
Le nombre de paquets Unicast envoyés lors du départ est zéro (voir figure 43). Le participant voulant quitter le groupe n'a pas d'informations à recevoir, que se soit avec une méthode ou l'autre. Donc, le nombre de paquets échangés lors du départ est égal au nombre de paquets Multicast échangés (voir la figure 44).

Les identifiants de ma méthode étant envoyés lors de l'ajout d'un participant, le nombre de paquets Multicast peut donc être diminué lors du départ.



HBT RS

Figure 43: Nombre de messages Unicast lors du départ



HBT

RS

Figure 44: Nombre de messages Multicast lors du départ

5.2.2.3 Comparaisons du temps de calcul

Lorsqu'une encryption est effectuée, l'information est transformée. Cette transformation nécessite beaucoup de calculs et beaucoup de ressources systèmes. Plus une méthode utilise l'encryption et plus elle est lente.

Avec ma méthode, j'utilise des modifications de clés déjà existantes sans avoir besoin de créer de nouvelles valeurs de clés. Cela réduit le temps de calcul du gestionnaire des clés. La figure 45 montre le nombre de nouvelles encryptions (génération de nouvelles clés et encodage) pour le même groupe étudié. Le nombre de nouvelles encryptions est beaucoup plus nombreux avec la méthode HBT, ce qui confirme qu'elle demande un plus grand temps de calcul et qu'elle est plus gourmande au niveau des ressources. Ma technique ROKEN-SKI vient donc améliorer cet aspect.

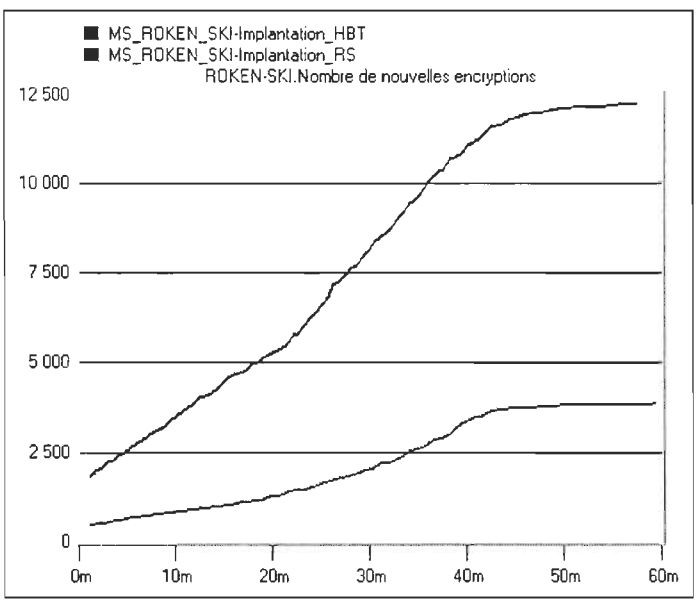


Figure 45: Nombre des nouvelles encryptions

Lors de l'ajout ou du départ d'un participant, le changement de la valeur d'une clé par une fonction à sens unique (un rafraîchissement) est peu coûteux, comparée à une nouvelle encryption.

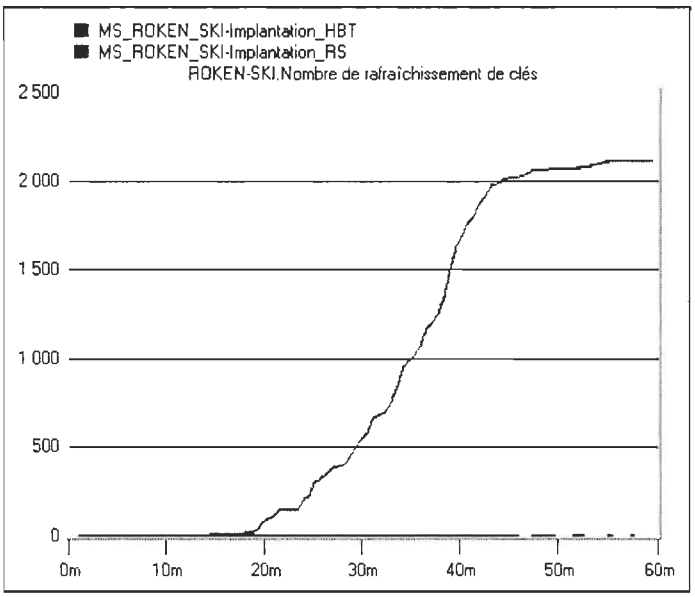


Figure 46: Nombre de rafraîchissement de clés

Les clés « glissées » (ou passées dans des fonctions de hachage) nécessitent la même quantité de mémoire et de calculs que des clés rafraîchies; elles sont obtenues par une deuxième fonction à sens unique. La méthode HBT n'utilise pas ces opérations tandis que la méthode ROKEN-SKI les utilisent beaucoup; les figures 45, 46 et 47 démontrent ce fait.

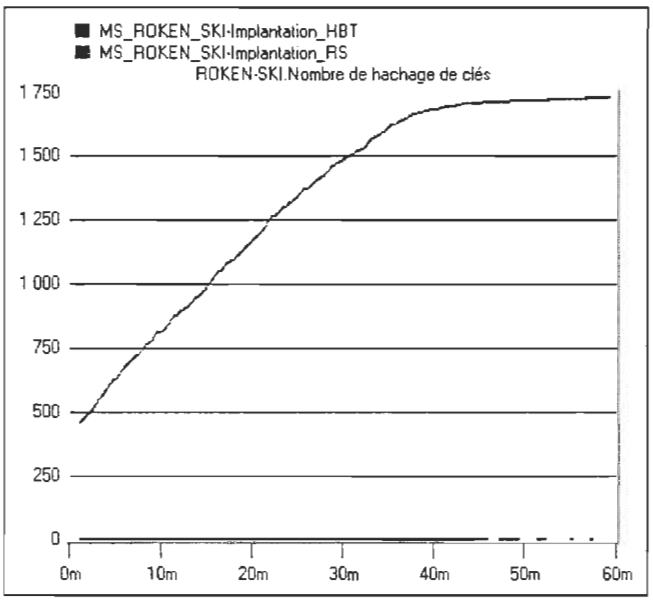


Figure 47: Nombre de hachage de clés

La quantité de nouvelles encryptions étant plus demandant au point de vue du temps de processeur, il est normal que la méthode l'utilisant le plus, la méthode HBT, soit celle qui utilise le plus le temps du processeur.

La modification d'une clé étant moins consommatrice que de nouvelles encryptions, la méthode ROKEN-SKI utilise moins de temps processeur que sa rivale.

Dans la figure 48, le temps de calcul de la méthode HBT est nettement supérieur à celui de la méthode ROKEN-SKI. Ceci découle directement de la quantité de nouvelles encryptions effectuées, puis de la quantité de clés modifiées.

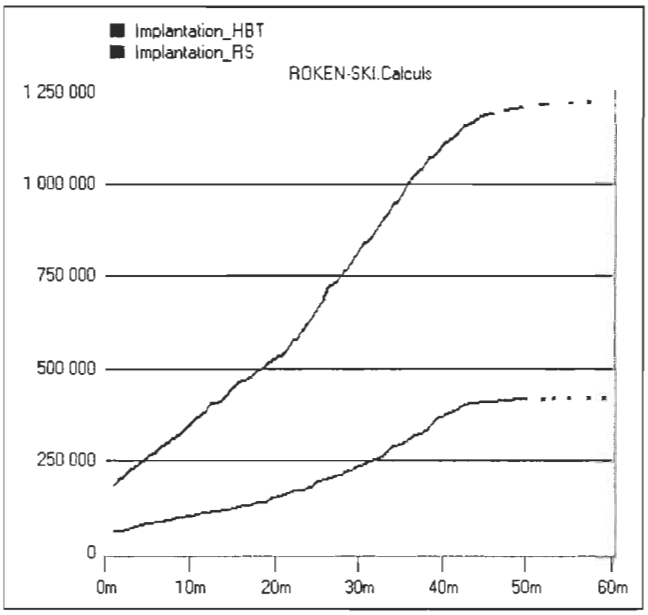


Figure 48: Temps de calcul

5.2.2.4 Comparaisons du temps de connexion

Lorsqu'un participant se joint au groupe, le temps entre sa demande d'ajout au groupe et le temps où il recevra les premières informations est le temps de connexion.

Dans la figure 49, le temps de connexion de la méthode HBT et de ma méthode est nul, sauf pour les valeurs représentées par la diagonale sur la figure 49. Les valeurs de la diagonales proviennent de données erronées des deux techniques. La technique utilisée pour recueillir les données de temps de connexion lors de la simulation est erronée.

Avec un nombre différent de paquets échangés lors des différentes opérations sur le groupe, la différence entre les deux méthodes aurait dû être plus grande et les résultats non nuls.

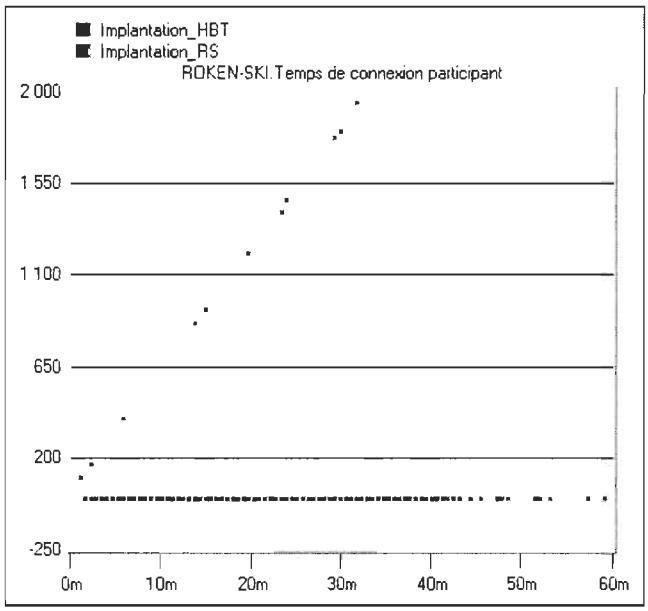


Figure 49: Temps de connexion

CHAPITRE 6

CONCLUSIONS

Les méthodes Multicast ont été créées, entre autre, pour réduire le nombre d'informations transmises sur les réseaux pour rejoindre plusieurs abonnés à la fois. Les techniques d'encryptions traditionnelles lorsqu'elles sont appliquées au Multicast contreviennent à cet avantage. En développant des méthodes régissant la communication Multicast encryptée, les chercheurs tendent vers un des premiers buts du Multicast.

Dans cet ouvrage, une nouvelle méthode de distribution des clés d'encryption ROKEN-SKI a été développée. D'après ses spécifications, elle visait à réduire le nombre de paquets utilisés lors de l'ajout et du départ d'un participant, tout en égalisant le rapport de paquets envoyés (ajout/départ) et tout en assurant une bonne compréhension de l'arbre des clés.

Ce mémoire n'est pas une solution à tous les problèmes qui existent présentement dans le secteur du Multicast encrypté. J'ai tenté d'apporter des solutions viables aux problèmes rencontrés par la communauté de recherche dans le domaine de l'échange des clés de cryptage.

Comme présenté au chapitre 5, la méthode ROKEN-SKI donne de meilleurs résultats que la méthode HBT. En effet, le nombre de paquet échangés est plus bas avec la méthode

ROKEN-SKI qu'avec la méthode HBT. Même si le nombre de paquets Unicast lors de l'ajout est plus important avec la méthode ROKEN-SKI, elle reste efficace.

Un de mes objectifs de départ était d'apporter une réduction des paquets tout en permettant d'équilibrer le nombre de paquets entre l'ajout et le départ d'un participant. D'après les données recueillies et démontré grâce à mes simulations, cet objectif a été atteint avec la méthode ROKEN-SKI avec 3 niveaux.

Non seulement la méthode utilise moins de paquets lors de l'ajout et du départ d'un participant, mais elle utilise moins d'encryptions. Donc en plus de décharger les architectures servant à assurer la communication Multicast, la méthode ROKEN-SKI allège les traitements au niveau du KDC.

La difficulté majeure avec le projet a été l'apprentissage du logiciel de simulations réseaux OPNET Modeler. Le temps d'apprentissage a été plus long que prévu et l'échéancier a été bouleversé. L'obstacle principal a été de trouver comment ajouter un protocole au simulateur. Après avoir exploré toutes les voies possibles, j'ai trouvé dans les Generic Network Applications (GNA) une facilité d'adaptation avec les nouvelles méthodes.

Les GNA sont complexes et difficiles à utiliser. Elles sont néanmoins très puissantes et peuvent permettre beaucoup de conditions génériques comme leur nom l'indique. Elles fonctionnent avec la pile TCP/IP ce qui est très important pour comparer les différentes recherches entre elles et elles se rapprochent bien de la réalité.

CHAPITRE 7

PERSPECTIVES

Je recommande aux futurs développeurs sur le logiciel de simulations réseaux de continuer à utiliser les GNA pour leurs nouveaux protocoles. Même si le temps d'apprentissage est plus grand, la facilité avec laquelle elles sont modifiables les rendent plus intéressantes à utiliser.

Les méthodes de distributions des clés dans le Multicast peuvent être encore améliorées. Avec une étude plus poussée des techniques d'encryptions, les chercheurs pourraient trouver une façon de réduire la quantité de clés qui sont transmises lors du départ d'un participant. Si une nouvelle méthode donnerait un indice aux participants adjacents à un participant quittant le groupe, sans compromettre la sécurité de la valeur de la clé, il serait facile de diminuer le nombre de paquets à un nombre très en deçà des méthodes actuelles.

Une piste serait d'utiliser une fonction à sens unique pour fabriquer des identifiants avec des clés déjà connues des participants. De cette façon le nombre de niveaux k utilisant des identifiants pourrait être plus grand, sans que le nombre de paquets transmis ne soit plus grand.

Lors de la création d'un identifiant associé à la clé 01001 et associé au participant 0100110, par exemple, la valeur de la clé 01001 sera classée dans le tableau des identifiants des participants connaissant cette clé. Avec une fonction à un sens, une

nouvelle valeur de la clé sera créée et la nouvelle valeur sera envoyée au nouveau participant par l'entité KDC.

La seule limite réside dans la probabilité de collusion entre participants. Le nombre de niveaux pourrait être choisit par l'initiateur du groupe. S'il désire un groupe totalement sécuritaire, il choisira un nombre de niveaux très bas et s'il désire être efficace, il choisira un nombre élevé de niveaux.

L'application de ma méthode aux réseaux sans fils et/ou Ad-hoc peut être risquée. La méthode de nomination des clés est statique et demande un environnement qui l'est aussi. Un des problèmes à résoudre serait de trouver une méthode dynamique qui viendrait utiliser l'arbre statique.

De plus, une technique pour transformer l'aspect centralisateur de ma méthode en un aspect distribué pourrait être très intéressante et plus performante. La séparation des utilisateurs en îlots ou sous-groupes pourrait être une alternative viable pour tout développement futur.

BIBLIOGRAPHIE

[1] S.E. Deering, "Host extensions for IP multicasting", RFC 1112, août 1989.

[2] B. Fenner, "IANA Considerations for IPv4 Internet Group Management Protocol (IGMP)", RFC 3228, février 2002.

[3] T. Henderson, "Latency and user behaviour on a multiplayer game server", Workshop on Networked Group Communications 2001, pages 1-13, novembre 2001.

[4] D. Choi et al., "Cost-optimal Dimensioning of a large scale video on demand system", International Workshop on Networked Group Communications 2002, pages 22-29, novembre 2002.

[5] Serge Courrier, "Multicast : l'alliance de la télé et du Web", SVM. Science & vie micro, no 159, pages 93-95, avril 1998.

[6] Y. Azar et al, "The Multicast Bandwidth advantage in serving a web site", International Workshop on Networked Group Communications 2001, pages 88-99, novembre 2001.

[7] Kenneth C. Miller, "Multicast: issues and applications", McGraw-Hill, 282 pages 1998.

[8] M. Hanley et al. "The reliable Multicast design space for bulk data transfer", RFC 2887, août 2000.

[9] H. Maisonniaux et P. Coquet, "La taxonomie du multipoint", École nationale supérieur de technologie, Paris, 1996.

[10] D. Maughan, M. Schertler, M. Schneider et J.Turner, "Internet Security Association and Key Management Protocol", Internet-draft, groupe de travail IPSEC, février 1997.

[11] W. Diffie et M. Hellman, "New Directions in Cryptography", IEEE

Transactions on Information Theory, IT-22, 6, pages 644-654, 1976.

[12] D. Wallner, E. Harder et R. Agee, "Key management for Multicast : issues and architecture", RFC 2627, NSA, juin 1999.

[13] Adrian Perrig, Dwan Song et J.D. Tygar. "ELK. A new protocol for efficient large-group key distribution", Proceedings of the IEEE Security and Privacy Symposim 2001, pages 247-262, mai 2001.

[14] J. Arkko, E. Carrara, F. Lindholm, M. Naslund et K. Norrman, "MIKEY: Multimedia Internet KEYing", INTERNET-DRAFT de MSEC, juillet 2002.

[15] H. Harney, C. Muckenhirn, et T. Rivers, "Group Key Management Protocol Specification", RFC 2093, septembre 1994.

[16] C. Wong, M. Gouda et S. Lam, "Secure Group Communications Using Key Graphs", (Apparu pour la première fois en 1998), IEEE/ACM Transactions on Networking, Vol.8 No.1, 2000.

[17] A. Fiat et M. Naor, "Broadcast Encryption, Advances in Cryptology" , CRYPTO Æ93 Proceedings, Lecture Notes in Computer Science, Vol. 773, pages 480 à 491, 1994.

[18] David A. McGrew et Alan T. Sherman, "Key establishment in large dynamic groups using one-way fuction trees", manuscript, mai 1998.

[19] Sandro Rafaeli, Laurent Mathy et David Hutchison. "EHBT: an efficient protocol for group key management", International Workshop on Networked Group Communications 2001, pages 159-171, 2001.

[20] R. Canetti et B. Pinkas. "A taxonomy of Multicast security issues", http://www.ietf.org/internet-drafts/draft-canetti-secure-Multicast-taxonomy-00.txt, mai 1998.

[21] M. Steiner, G. Tsudik et M. Waidner. "Cliques: A new approach to group key agreement", IEEE Transactions on Parallel and Distributed Systems, pages 380-387, 1998.

[22] S. Mittra, "Iolus: A framework for scalable secure Multicasting", Proceedings of ACM SIGCOMM '97, pages 277–288, Cannes, France, septembre 1997.

[23] Sandro Rafaeli et David Hutchingson, "Hydra: a decentralised group key management", Pittsburgh, Proceedings of WETICE 2002, 2002.

[24] Sally Floyd et Vern Paxson, "Difficulties in simulating the Internet", IEEE/ACM Transactions on networking, vol 9, no 4, pages. 392-403, Août 2001.

[25] A. Mankin et al. "IETF criteria for evaluating reliable Multicast transport and application protocols", RFC 2357, juin 1998.

[26] B. Whetten et al. "Reliable Multicast transport building blocks for one-to-many bulk-data transfert", RFC 3048, janvier 2001.

[27] L.Rizzo et L. Vicisano, "A reliable Multicast data Distribution Protocol based on software FEC techniques", Proceedings of the Fourth IEEE Workshop on the Architecture and Implementation of High Performance Communication Systems, juin 1997.

[28] S. Armstrong, A. Freier et K. Marzuilo, "Multicast Transport Protocol", RFC 1301, 1992.

[29] B. Amar Bensaber, "Étude des protocoles de communication de groupe: MTCP, une solution basée sur une extension de TCP", thèse de Doctorat, 142 pages, juillet 1998.

[30] S. Floyd, V. Jacobson et S. McCanne, "A reliable Multicast framework for lightweight sessions and application level framing", IEEE/ACM, Transactions on Networking, décembre 1997.

[31] L. Mathy et O. Bonaventure, "QoS Negociation for Multicast Communications" Lecture Notes in Computer Science 882, pages 199 - 218, Berlin, Allemagne, 1994.

[32] S. Fdida, J. F de Rezende, A. Mathy et D. Hutchinson, "M-Connection Service: A Multicast Service for Distributed Multimedia Applications", Proceedings of IFIP/IEEE International Conference on Distributed Platforms, Allemagne, février 1996.

[33] S. Armstrong, A. Freier et K. Marzuilo, "Multicast Transport Protocol", RFC 1301, 1992.

[34] Luigi Rizzo, "Pgmcc: A TCP-friendly Single-Rate Multicast Congestion Control Scheme", ACM SIGCOM, pages 17-28, Stockholm, Suède, août 2000.

[35] J. Byers, M. Luby et al, "FLID-DL: Congestion Control for Layered Multicast", International Workshop on Networked Group Communications 2000, pages 71-81, Palo Alto, USA, novembre 2000.

[36] Srinivasan Jagannathan, Kevin Almeroth et Anurag Acharya, "Topology Sensitive Congestion Control for Real-Time Multicast", Proceedings of The 10th International Workshop on Network and Operating System Support for Digital Audio and Video, Chapel Hill, North Carolina, USA, juin 2000.

[37] Dan Rubenstein, Nicholas F. Maxemchuk et David Shur, "A Centralized, TreeBased Approach to Network Repair Service for Multicast Streaming Media", Proceedings of IEEE NOSSDAV'00, Caroline du Nord, USA, juin 2000.

[38] T. Ballardie, "Scalable Multicast Key Distribution", RFC 1949 , mai 1996.

[39] R. Molva et A. Pannetrat, "Network security in the Multicast Framework", Networking and tutorials of LNCS 2497, pages 59-82, 2002.

[40] H. Harney, C. Muckenhirn. et T. Rivers, "Group Key Management Protocol Architecture", RFC 2094, septembre 1994.

[41] Marcel Waldvogel, Germano Caroni, Dan Sun, Nathalie Weiler et Bernhard Plattner. "The Versakey Framework: Versatile Group Key Management", IEEE Journal on Selected Areas in Communications, 7(8) , pages. 1614-1631, septembre 1999.

[42] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor et B. Pinkas. "Multicast Security: a taxonomy and some efficient construction", Proceedings of INFOCOM, New York, USA, mars 1999.

[43] R. Canetti et B. Pinkas. "A taxonomy of Multicast security issues", http://www.ietf.org/internet-drafts/draft-canetti-secure-Multicast-taxonomy-00.txt, mai 1998.

[44] G. Caronni, M. Waldvogel, D. Sun et B. Plattner. "Efficient security for large and dynamic Multicast groups", Proceedings of the IEEE 7th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE '98), juin 1998.

[45] D. M. Wallner, E. J. Harder et R. C. Agee. "Key management for Multicast: Issues and architectures", Internet-Draft (expired), juillet 1997. Archivé à http://www.tik.ee.ethz.ch/ ~mwa/Security/draft-wallner-key-arch-00.txt.

[46] Kin-Chin Chan et S.-H. Gary Chan, "Distributed Servers approach for Large-Scale Secure Multicast", IEEEjournal on selected areas in communications, vol 20, no 8, pages 1500-1510, octobre 2002.

[47] W. Chen et L. Dondeti, "Performance comparison of stateful and stateless group rekeying algorithms", International Workshop on Networked Group Communications, pages 12-19, octobre 2002.

[48] S. Banerjee et B. Bhattacharjee, "Scalable secure group communication ver IP Multicast", IEEE Journal on selected areas of communications, Vol 20, no. 8, octobre 2002.

[49] Multicast Security working group, description disponible au http://www.ietf.org/html.charters/msec-charter.html

[50] S. Kent et R. Atkinson, "Security Architecture for the Internet Protocol. Corporation for National Research Initiatives", RFC 2401, Reston, Virginia, USA, novembre 1998.

ANNEXE 1: CODE DE L'ENTITÉ KDC

Le listing dans le répertoire KDC sur le CD-ROM est le code qui permet au logiciel de simulation OPNET Modeler d'utiliser l'entité « KDC » pour exécuter une méthode de distribution de clés lors de l'ajout ou du départ d'un participant; les méthodes ROKEN-SKI ou Hierarchical Binary Trees peuvent être utilisées.

ANNEXE 2: CODE DE L'ENTITÉ PARTICIPANT

Le listing dans le répertoire PARTICIPANT sur le CD-ROM est le code qui permet au logiciel de simulation OPNET Modeler d'utiliser l'entité « Participant » pour effectuer une connexion et une déconnexion au KDC.

ANNEXE 3: PROJET DE SIMULATION

Le projet utilisé pour les simulations se trouve dans le répertoire PROJET sur le CD-ROM.

ANNEXE 4: PUBLICATIONS EFFECTUÉES AVEC CE SUJET DE MAÎTRISE

Martin St-Amand, Boucif Amar Bensaber et Dominique Seret, « A new solution for key distribution in secure Multicast », IADIS www/Internet conference, Portugal, Novembre 2003.

Martin St-Amand, Boucif Amar Bensaber et André Boumso, « Rokenski : A new pragmatic solution for key distribution in secure Multicast communications », Mediterranean Workshop on Ad-Hoc Networks, Tunisie, Juin 2003.

Martin St-Amand et Boucif Amar Bensaber, « A new solution for key distribution in secure Multicast communications », poster présenté lors de « Multiconference on Security and management », Las Vegas, USA, Juin 2003.

Martin St-Amand, « Une nouvelle méthode de transmission des clés de cryptage lors des communications Multicast », Congrès de l'ACFAS, Rimouski, Avril 2003.