

Implémentation d'un simulateur de traceroute.

Encadrant : Benoit Donnet (benoit.donnet@lip6)

Nombre d'Etudiants : Au moins 2.

Pré-Requis : Rigueur, Java, Java RMI

Description L'objectif du travail est d'implémenter en Java un simulateur de traceroute.

Traceroute est un outil réseau qui permet de découvrir le chemin qu'un paquet de données prend pour aller d'une machine S (la *source* ou *moniteur*) vers une machine D (la *destination*).

Traceroute fonctionne comme suit: la source du traceroute, S, envoie dans le réseau des paquets (ou *sondes*) en incrémentant pas à pas le champ *time-to-live* (TTL), le TTL étant un champ de l'en-tête du paquet IP. Le TTL a pour but d'indiquer combien de temps un paquet peut circuler dans le réseau. Chaque fois qu'un paquet entre dans un routeur, le routeur décrémente le TTL. Quand le TTL vaut 1, le routeur détermine que le paquet a consommé suffisamment de ressources dans le réseau, le jette et informe la source du paquet de l'incident en renvoyant un message ICMP *time exceeded*. En regardant l'adresse IP source du message ICMP, le moniteur peut apprendre l'adresse du routeur qui a jeté le paquet.

L'idée du travail est de simuler un tel comportement sur la base d'une architecture Client/Serveur. Le *client* est considéré comme étant la source du traceroute. Il veut connaître le chemin entre lui-même et une destination. Pour ce faire, il adresse des requêtes *traceroute* au serveur. Ces requêtes comprennent trois informations:

- l'adresse (ou l'identifiant) du client
- l'adresse (ou l'identifiant) de la destination
- la valeur du TTL.

A noter aussi que le client peut vouloir simplement connaître la longueur du chemin entre lui-même et une destination particulière. Par longueur de chemin, nous entendons le nombre de sauts.

De son côté, le *serveur* charge en mémoire une topologie (fichier XML ou fichier texte) représentant Internet. Il doit être capable d'accepter des requêtes de la part de clients. Pour chaque requête *traceroute*, il fournit comme réponse l'adresse (ou l'identifiant) de la machine (routeur intermédiaire ou destination) se situant à TTL sauts du client en direction de la destination. Pour des requêtes sur la longueur du chemin, il renvoie simplement le nombre de sauts entre le client et la destination.

Dans ce travail, il est demandé d'implémenter:

- Un module permettant de représenter en mémoire un réseau. Des informations sur ce réseau sont fournies en entrée sous la forme d'un fichier XML ou d'un fichier texte (au choix).
- Un module permettant de connaître la longueur du chemin entre un client et une destination (i.e. nombre de sauts).
- Un module permettant de répondre à des requêtes « *traceroute* ».
- Le serveur utilisant ces différents modules. Le serveur doit pouvoir gérer simultanément plusieurs requêtes. Les requêtes sont transmises du client au serveur sous la forme de requête Java RMI
- Un exemple de client faisant des requêtes au serveur.

Ce simulateur doit être implémenté en Java.

Références

- Java: <http://java.sun.com>
- Java RMI: <http://java.sun.com/docs/books/tutorial/rmi/index.html>

Implémentation de bibliothèques pour un traceroute IPv6.

Encadrant Benoit Donnet (benoit.donnet@lip6)

Nombre d'Etudiants : Au moins 2.

Pré-Requis : Rigueur, Java, C/C++, Java JNI, Intérêt pour IPv6

Description

L'objectif est d'étendre des bibliothèques existantes afin de permettre l'exécution de traceroute dans les réseaux IPv6.

Traceroute fonctionne comme suit: la source du traceroute, S, envoie dans le réseau des paquets (ou *sondes*) en incrémentant pas à pas le champs *time-to-live* (TTL), le TTL étant un champ de l'en-tête du paquet IP. Le TTL a pour but d'indiquer combien de temps un paquet peut circuler dans le réseau. Chaque fois qu'un paquet entre dans un routeur, le routeur décrémente le TTL. Quand le TTL vaut 1, le routeur détermine que le paquet a consommé suffisamment de ressources dans le réseau, le jette et informe la source du paquet de l'incident en renvoyant un message ICMP *time exceeded*. En regardant l'adresse IP source du message ICMP, le moniteur peut apprendre l'adresse du routeur qui a jeté le paquet.

Que se passe-t-il quand la sonde atteint finalement la destination? En fait, le comportement de la destination dépend du type de sonde qui est envoyé par la source. Typiquement, traceroute peut en utiliser trois types différents (au choix):

- UDP. Il s'agit d'un traceroute "classique". Les sondes sont des paquets UDP avec un numéro de port élevé (i.e. supérieur à 1024). Quand le paquet UDP arrive à la destination, celle-ci est supposée répondre avec un message ICMP *destination unreachable*. Le fait de mettre un numéro de port élevé offre une certaine probabilité qu'aucune application n'écoute sur ce port à la destination, ce qui générera un message ICMP *destination unreachable*.
- ICMP. Les sondes envoyées par la source sont des messages ICMP *echo request*. La destination est supposée répondre avec des messages *echo reply*.
- TCP. Les sondes envoyées sont des paquets TCP avec le flag *SYN* mis à 1. La destination est supposée répondre avec des paquets TCP dont le flag *SYN_ACK* est mis à 1. L'avantage du traceroute TCP est qu'il permet de passer outre la plupart des filtres de firewall. Le traceroute TCP suppose que les firewalls vont laisser entrer les paquets TCP si la destination écoute sur le port spécifié.

Il existe un projet open-source, *JSocket Wrench*, qui permet d'effectuer des traceroutes en Java. Malheureusement, les bibliothèques fournies par *JSocket Wrench* ne concernent que IPv4. Il est demandé, dans ce travail, d'étendre *JSocket Wrench* afin

- de pouvoir manipuler des paquets IPv6.
- de pouvoir manipuler des paquets ICMPv6.
- de pouvoir envoyer, dans le réseau, des paquets pour faire des traceroutes sur les réseaux IPv6.
- de pouvoir écouter les paquets ICMPv6 entrant.

Le développement des bibliothèques se fera principalement en Java. La manipulation des sockets fera appel à des commandes C/C++. Il n'est donc pas exclu de devoir développer des bibliothèques JNI pour permettre l'interaction entre Java et le C/C++.

Références

- *JSocket Wrench* R04: <http://jswrench.sourceforge.net/>
- Aide sur l'implémentation de traceroute: <http://cities.lk.net/trproto.html>
- IP protocols et format des paquets: <http://www.networksorcery.com/enp/topic/ipsuite.htm>
- RFC 768: User Datagram Protocol (<http://www.ietf.org/rfc/rfc768.txt>)
- RFC 793: Transmission Control Protocol (<http://www.ietf.org/rfc/rfc792.txt>)
- RFC 1883: Internet Protocol Version 6 (<http://www.ietf.org/rfc/rfc1883.txt>)
- RFC 1885: Internet Control Message Protocol Version 6 (<http://www.ietf.org/rfc/rfc1885.txt>)

Implémentation d'un traceroute TCP

Encadrant : Benoit Donnet (benoit.donnet@lip6.fr)

Nombre d'Etudiants : Au moins 2.

Pré-Requis Connaissance en C/C++, Intérêt pour l'implémentation, Rigueur

Description

L'objectif du travail est d'implémenter en C/C++ un *traceroute* basé sur des paquets TCP.

Traceroute est un outil réseau qui permet de découvrir le chemin qu'un paquet de données prend pour aller d'une machine S (la *source* ou *moniteur*) vers une machine D (la *destination*).

Traceroute fonctionne comme suit: la source du traceroute, S, envoie dans le réseau des paquets (ou *sondes*) en incrémentant pas à pas le champ *time-to-live* (TTL), le TTL étant un champ de l'en-tête du paquet IP. Le TTL a pour but d'indiquer combien de temps un paquet peut circuler dans le réseau. Chaque fois qu'un paquet entre dans un routeur, le routeur décrémente le TTL. Quand le TTL vaut 1, le routeur détermine que le paquet a consommé suffisamment de ressources dans le réseau, le jette et informe la source du paquet de l'incident en renvoyant un message ICMP *time exceeded*. En regardant l'adresse IP source du message ICMP, le moniteur peut apprendre l'adresse du routeur qui a jeté le paquet.

Que se passe-t-il quand la sonde atteint finalement la destination? En fait, le comportement de la destination dépend du type de sonde qui est envoyé par la source. Typiquement, traceroute peut en utiliser trois types différents (au choix):

- UDP. Il s'agit d'un traceroute "classique". Les sondes sont des paquets UDP avec un numéro de port élevé (i.e. supérieur à 1024). Quand le paquet UDP arrive à la destination, celle-ci est supposée répondre avec un message ICMP *destination unreachable*. Le fait de mettre un numéro de port élevé offre une certaine probabilité qu'aucune application n'écoute sur ce port à la destination, ce qui générera un message ICMP *destination unreachable*.
- ICMP. Les sondes envoyées par la source sont des messages ICMP *echo request*. La destination est supposée répondre avec des messages *echo reply*.
- TCP. Les sondes envoyées sont des paquets TCP avec le flag SYN mis à 1. La destination est supposée répondre avec des paquets TCP dont le flag SYN_ACK est mis à 1. L'avantage du traceroute TCP est qu'il permet de passer outre la plupart des filtres de firewall. Le traceroute TCP suppose que les firewalls vont laisser entrer les paquets TCP si la destination écoute sur le port spécifié.

Cependant, le comportement décrit plus haut est le cas idéal du traceroute. Un routeur le long du chemin peut ne pas répondre aux sondes pour différentes raisons. Par exemple, le protocole ICMP est désactivé dans le routeur ou bien il est surchargé. Afin d'éviter que la source n'attende un temps infini la réponse ICMP du routeur, la source active un temporisateur lors du lancement de la sonde. A l'expiration de ce timer, si aucun message ICMP n'a été reçu, alors, pour le TTL considéré, le routeur est étiqueté *non-répondant*.

Cependant, un problème survient quand c'est la destination qui ne répond pas aux sondes. Dans ce cas, la destination sera enregistrée comme non-répondante mais il est impossible de savoir si elle a été atteinte. Afin d'éviter d'inférer un chemin sans fin, une borne supérieure sur le nombre de machines non-répondante successives est utilisé. Une valeur typique pour cette borne est 5.

Dans ce travail, il est demandé d'implémenter une version particulière du traceroute TCP. Un des gros problèmes de traceroute est sa lenteur. A chaque TTL, la source envoie trois sondes et attend pour une éventuelle réponse. On peut augmenter la vitesse d'exécution du traceroute si on connaît la longueur du chemin. En effet, si on a connaissance du nombre de routeurs intermédiaires le long du chemin entre la source et la destination, on peut envoyer « en même

temps » tous les paquets. Si on n'arrive pas à connaître la longueur du chemin, alors on effectue un traceroute « classique », i.e. TTL par TTL.

Il est donc demandé d'implémenter:

- un module permettant de connaître la longueur du chemin entre la source du traceroute et la destination
- un module permettant d'envoyer simultanément plusieurs paquets dans le réseau, quelque soit le type de paquet (TCP, UDP, ICMP)
- un module permettant de construire des paquets TCP pour le traceroute TCP
- un module permettant d'écouter les paquets entrant (ICMP et TCP).
- une application utilisant ces différents modules.

Il est demandé d'implémenter cet outil en C/C++. Pour d'évidentes raisons de simplicité (polymorphisme, héritage, ...), il est conseillé d'utiliser un maximum le C++.

Références

- Aide sur l'implémentation de traceroute: <http://cities.lk.net/trproto.html>
- IP protocols et format des paquets: <http://www.networksorcery.com/enp/topic/ipsuite.htm>
- RFC 791: Internet Protocol (<http://www.ietf.org/rfc/rfc0791.txt>)
- RFC 768: User Datagram Protocol (<http://www.ietf.org/rfc/rfc768.txt>)
- RFC 792: Internet Control Message Protocol (<http://www.ietf.org/rfc/rfc792.txt>)
- RFC 793: Transmission Control Protocol (<http://www.ietf.org/rfc/rfc792.txt>)

Automatisation de mesures pour les réseaux ad hoc

Encadrant : Jérémie Leguay

Nombre d'étudiants demandés: 3

Description générale:

La réalisation de tests réels sur des plateformes sans-fil ad hoc s'avère souvent laborieuse du fait du manque d'outils appropriés. Le but de ce projet est de réaliser un outil déployant et réalisant automatiquement des scénarios de tests prédéfinis dans ce type d'environnement. Un scénario est défini ici par la séquence des connexions à établir et par la séquence des mesures à effectuer. Après collecte des mesures effectuées, l'outil réalisera un rapport synthétique présentant les résultats.

Pré requis :

- Etre familier avec Linux.
- Maîtrise d'un langage de script, TCSH si possible
- 3 portables.

Travail à réaliser :

- Identification d'un ou deux outils de mesure (délai, bande passante, ...).
- Conception/Réalisation de l'outil. Des scripts shell seront utilisés.
- Validation de l'outil sur un réseau ad hoc composé de 3 portables (prévoir l'installation d'un des protocoles de routage ad hoc OLSR ou AODV).

Applications ad hoc sur téléphone portable.

Encadrant : Jérémie Leguay

Nombre d'étudiants demandés : 2

Description générale :

Les applications devenant de plus en plus présente sur les appareils mobiles et conçues de manière décentralisée, des logiciels permettant d'échanger de l'information avec d'autres individus présents dans notre entourage sont tout à fait imaginable. Le but de ce projet est de se familiariser avec la plateforme J2ME, d'identifier les fonctionnalités de cette plateforme et de démontrer son utilisation de part le développement d'un outil de chat ad hoc entre téléphones portables équipés de la technologie Bluetooth.

Pré requis:

- 2 téléphones portables.
- JAVA

Travail à réaliser :

- Identification des possibilités de J2ME
- Conception de l'application
- Développement et démonstration

Sécurité des protocoles de routage ad hoc.

Encadrant : Jérémie Leguay

Nombre d'étudiants demandés : 3

Description générale :

De nombreux protocoles de routage pour les environnements ad hoc sans fil comme AODV ou OLSR ont fait l'objet d'implémentations. A l'heure actuelle aucune solution permettant de sécuriser ces protocoles vis-à-vis d'attaques comme l'injection de trafic de contrôle malicieux ou les dénis de service n'ont été intégrés au standard RFC. Le but de ce projet est donc d'identifier des implémentations d'extensions pour ces protocoles assurant certaines fonctionnalités de sécurité et d'en démontrer leur bon fonctionnement.

Pré requis :

- Notions de sécurité
- Etre familier avec Linux
- 3 portables.

Travail à réaliser :

- Identification des extensions de sécurité implémentées
- Choix de l'une d'elles et définition d'un scénario de démonstration (illustration via des attaques)
- Mise en place de la démonstration

Sur l'équité des courbes de remplissage dans le partage d'espaces d'adressage

Nombre d'étudiants demandés : 2

Réservé pour les étudiants de la SEMAINE du 6 février (GROUPE 1)

Encadrants : Marcelo Dias de Amorim (marcelo.amorim@lip6.fr) (*responsable*), Aline Carneiro Viana (aline.viana@irisa.fr) et Yannis Viniotis (candice@ncsu.edu)

Description générale

L'objectif futuriste du *pervasive computing*, aidé par la prolifération des dispositifs de communication sans fil, conduit à un changement révolutionnaire de notre société de l'information. Dans ce contexte, les utilisateurs auront la possibilité d'établir d'une façon spontanée des *réseaux auto organisables* (SONs). Les réseaux *ad hoc*, les réseaux de capteurs et les réseaux maillés sans fil sont des exemples de réseaux auto organisables. Plus particulièrement, le *routage* à large échelle dans les SONs constitue un vrai challenge en raison de la dynamique de la topologie et du manque d'infrastructure fixe. La stratégie de routage indirect basée sur DHT (table de hachage distribuée) est une proposition intéressante pour traiter le facteur d'échelle. Récemment, des nombreux protocoles implémentant ce type de routage et utilisant une architecture d'adressage géographique ont été proposés. Néanmoins, les modèles d'adressage et de localisation deviennent plus complexes à gérer dans les espaces multidimensionnels et exigent une association plus rapprochée entre les plans physique et logique. Dans ce contexte, le protocole Twins a été proposé.

Twins est une nouvelle architecture qui met en place un routage géographique indirect et un service de localisation efficace basé sur DHT. Son service de localisation est simple à gérer en raison de l'utilisation d'un espace unidimensionnel pour sa mise en place. Dans Twins, l'équivalence entre l'espace multidimensionnel, employé pour le routage, et l'espace unidimensionnel, utilisé pour la localisation, est obtenue par le biais de l'utilisation des *courbes de remplissage d'espace*. Les courbes de remplissage sont utilisées afin de réduire un problème de plusieurs dimensions en un problème d'une seule dimension : une ligne qui connecte tous les points de l'espace. Cette ligne simplifie la gestion de l'espace et rend facile son partage entre les nœuds du réseau.

Travail à réaliser

Le sujet principal de ce projet reposera sur l'analyse des courbes de remplissage d'espace dans le contexte de l'architecture Twins. Une large variété de courbes de remplissage a été proposée dans la littérature : *Hilbert*, *Z-order* et *Gray-coded*. Chacune de ces courbes présente différentes propriétés de localité (appelées *clustering*), qui affectent la gestion de l'architecture en question. Ainsi, les objectifs de ce projet sont :

- Une brève étude des propriétés liées aux différentes courbes de remplissage.
- L'implémentation et l'évaluation des ces propriétés dans le cas de l'architecture Twins. Plus spécifiquement, nous sommes intéressés par l'analyse de la distribution équitable de l'espace entre les nœuds.

Conditions requises Les candidats doivent avoir des connaissances solides en programmation (en particulier, C ou C++).

Références

- A. C. Viana, M. D. de Amorim, Y. Viniotis, S. Fdida, and J. F. de Rezende, "Twins: A Dual Addressing Space Representation for Self-Organizing Networks," to appear in *IEEE Transactions on Parallel and Distributed Systems*, December 2006.
- A. C. Viana, M. D. de Amorim, Y. Viniotis, S. Fdida, and J. F. de Rezende, "Easily-managed and topological-independent location service for self-organizing networks," *ACM MobiHoc*, Urbana-Champaign, IL, May 2005.

Implémentation d'un service de localisation géographique

Encadrant : Bamba Gueye (bamba.gueye@lip6.fr)

Nombre d'étudiants demandés : 2 ou 3

Description générale :

Nous nous proposons de fournir un service de la localisation géographique basé sur des mesures de RTT faites par des serveurs sondes vers un hôte cible. La technique qui sera implémentée se base sur la multilatération qui permet d'estimer une position géographique en utilisant un nombre suffisant de distances à partir de quelques points mobiles (comme GPS). Pour appliquer la multilatération dans l'Internet il faut transformer les mesures de délai en distance géographique.

Pour localiser un hôte, chaque serveur sonde calcule d'abord le RTT, entre lui et l'hôte cible, puis transforme ce délai en distance géographique. Ainsi chaque serveur sonde S_i (hôte référence) estime que l'hôte cible T se trouve quelque part à l'intérieur du cercle C_{iT} centré en S_i et de rayon R_{iT} . Par exemple si nous avons K serveurs sondes, nous avons $C_T = \{ C_{1T}, C_{2T}, \dots, C_{KT} \}$ cercles. L'unique région R, si elle existe, formée par l'intersection de l'ensemble des cercles $C_{iT} \in C_T$ contient l'estimation de localisation de la cible.

Pour trouver l'estimation de localisation de la cible T, il faudra estimer la région R. Pour ce faire, le centroïde du polygone inscrit dans cette région R sera choisi comme localisation de l'hôte cible. Les sommets du polygone seront formés par les points d'intersection des cercles C_{iT} intérieurs à l'ensemble des cercles.

Pré-requis : Language C

Travail à réaliser :

On considéra comme connu la distance géographique entre les serveurs sondes et la cible, la localisation géographique des serveurs sondes.

Il faudra implémenter:

- le processus de localisation de la cible par chaque serveur sonde.
- la recherche de la région R qui représente la zone d'intersection de tous les cercles ayant comme
- centre le serveur sonde S_i et de rayon la distance géographique vers la cible.
- la recherche des sommets du polygone inscrit à l'intérieur de cette région R
- la recherche de la localisation géographique de la cible qui sera le centroïde de ce polygone.

Liens complémentaires:

Pour une bonne compréhension de la méthodologie vous pourriez visiter les liens ci-dessous:

http://rp.lip6.fr/~gueye/articles/ToN_CBG.pdf (plus complet)

<http://rp.lip6.fr/~gueye/articles/CFIP05.pdf>

Ce service a été déjà implémenté dans un autre langage, R, par conséquent les algorithmes sont disponibles.

Traces IPv6 Mobile

BUT : obtenir des traces sur les échanges de messages qui apparaissent dans le cadre de l'utilisation d'IPv6 Mobile pour les phases de mobilité:

- Agent Discovery
- Enregistrement
- Déconnexion
- ...

Encadrante : Anne Fladenmuller, Anne.Fladenmuller@lip6.fr

Tâches :

- Ecrire un court tutorial sur la mobilité dans IPv6 et les versions de IPv6 disponibles.
- Mettre en place votre plateforme.
- Définir l'architecture dont vous avez besoin pour visualiser ces échanges de trames et de paquets
- Définir les scénarios de test.

Architecture

5 ordinateurs munis de deux cartes réseau connectés sur deux LAN parallèles.

MODALITES

Avant de débiter les projets vous devrez présenter au plus tard la semaine qui précède votre semaine de développements un pré-rapport spécifiant :

- une architecture « minimale » de plateforme dont vous aurez besoin pour réaliser ces tests.
- Les échanges de trames ou paquets que vous voudrez mettre en évidence pendant le projet.
- Le tutorial.

Ce pré rapport devra être validé avant le début des tests et implémentations.

A la fin de la semaine du projet devront être rendus :

- le rapport contenant l'ensemble des traces définies (pré-rapport).
- une version des traces, ceci permettant de visualiser avec Ethereal les échanges de trames/paquets/datagrammes.
- Une explication des filtres à appliquer à Ethereal sur les différentes traces pour visualiser les échanges pertinents de trames.

Ce sujet est proposé pour 1 groupe de 4 étudiants

Mise en évidence du fonctionnement d'un réseau local

BUT : Configurer un réseau local afin de mettre en évidence le fonctionnement des serveurs d'un réseau.

Encadrante : Anne Fladenmuller, Anne.Fladenmuller@lip6.fr

Lieu du projet : Campus de Jussieu.

Architecture :

Vous disposerez de 5 ordinateurs munis de cartes 802.11.

TACHES :

Fournir une base de traces qui met en évidence le fonctionnement :

- D'un ou plusieurs serveurs DHCP (proxy DHCP)
 - Attribution d'adresses lorsqu'il y a plusieurs serveurs DHCP sur un LAN
 - De quel serveur provient l'adresse (proxy ou non)
 - Visualiser les différentes informations qui peuvent être communiquées en même temps que l'adresse
 - ...
- D'un serveur DNS
 - Echanges entre site primaire / secondaire
 - DNS dynamique
 - Requêtes itératives à un site distant
 -
- Mise en évidence du fonctionnement du NAT
 - Transfert de paquets mettant en évidence le changement de ports et d'adresses IP
 - ...
- Les modes infrastructure et ad hoc des cartes sans fil :
 - Connexion au point d'accès/authentification
 - Échange de paquets avec des seuils de fragmentation différents entre l'AP et les émetteurs récepteurs,

Prérappel :

- Définir une architecture pour mettre en place cet ensemble de traces
- Définir des scénarios permettant de visualiser ces fonctionnalités et valider vos choix

MODALITES

Avant de débiter les projets vous devrez présenter un pré-rapport précisant les traces que vous voudrez identifier pour mettre en évidence le fonctionnement de l'ensemble de ces services. Ce pré rapport devra être validé avant la semaine de tests et d'implémentations.

A la fin de la semaine devront être rendus :

- le rapport contenant les scénarios pour mettre en évidence le bon fonctionnement des différents services et les fonctions intéressantes de chacun (pré-rapport).
- une version électronique des traces, qui pourront être visualisées avec Ethereal ainsi qu'un rapport sur ce quelles mettent en évidence.

Ce sujet est proposé pour 1 groupe de 4 étudiants

Développement d'un outil de détection d'attaques

BUT : Développer un outil de détection d'attaques sur un réseau local

Encadrantes : Bénédicte Le Grand, Anne Fladenmuller

Lieu du projet : Campus de Jussieu

Architecture :

Vous disposerez de 5 machines sur un réseau local.

TACHES :

Prérappel :

- Lister les attaques réseau possibles sur un réseau (exemple : déni de service, recherches de vulnérabilités)
- Lister les applications/outils disponibles pour effectuer ces attaques (Linux)
- Définir une architecture permettant de mettre en évidence ce type d'attaques
- Définir les attaques que vous souhaiteriez détecter

Développement

- Implémenter une application qui affichera des alertes lorsque les attaques ou des comportements anormaux sur le réseau seront détectés.
- Rendre le rapport contenant une analyse de la plateforme, des difficultés rencontrées et des traces

MODALITES

Avant de débiter les projets vous devrez présenter un pré-rapport précisant les traces que vous voudrez identifier pour mettre en évidence le fonctionnement de l'ensemble de ces services. Ce pré rapport devra être validé avant la semaine de tests et d'implémentations.

A la fin de la semaine d'implémentation devront être rendus :

- le rapport contenant les scénarios pour mettre en évidence le bon fonctionnement des différents services et les fonctions intéressantes de chacun (pré-rapport).
- une version électronique des traces, qui pourront être visualisées avec Ethereal ainsi qu'un rapport sur ce quelles mettent en évidence.

Ce sujet est proposé pour 1 groupe de 4 étudiants

Agenda partagé utilisant l'IPv6 et le concept multicast

BUT : Créer une mini-application permettant la visualisation et la gestion de l'emploi du temps d'un groupe de personnes à caractéristiques hétérogènes.

- Partager des tâches de l'agenda d'une personne avec tout le monde
- Partager des tâches de l'agenda d'une personne avec un groupe fermé avec mot de passe partagé
- Faire cohabiter plusieurs groupes de personnes différentes avec des possibilités de visualisation différentes, p.ex. « les profs » voient l'emploi du temps « des étudiants ». Les étudiants voient si un prof est libre à un moment donné, mais ne voient pas la raison de son indisponibilité. Quand un nouvel événement est proposé les participants doivent confirmer leur disponibilité à la personne qui propose

Encadrant : Konstantin Kabassanov (mail: Konstantin.Kabassanov@lip6.fr)

Tâches :

- Ecrire un cahier des charges.
- Découper le travail en sous-tâches disjointes.
- Définir plusieurs groupes multicast, prévoir le procédé de confirmation en mode connecté (donc forcément tcp unicast), ainsi que le cryptage basique par mot de passe (on ne gère pas comment le mot de passe est fourni)
- Se répartir le travail.
- Implémentation
- Définir des scénarios de test.

ARCHITECTURE

N ordinateurs banalisés (où $N > 1$;)

MODALITES

Avant de débiter les projets vous devrez présenter au plus tard la semaine qui précède votre semaine de développements un pré-rapport spécifiant :

- Le cahier de charge.
- Le découpage du projet en sous-projet et la répartition des personnes.
- La conception de l'application.

Ce pré rapport devra être validé avant le début des implémentations et des tests.

A la fin de la semaine du projet devront être rendus :

- Le pré-rapport.
- Les sources de l'application.
- Un mini-manuel d'utilisation.

Ce sujet est proposé pour 1 groupe de 4 étudiants

Application web permettant l'authentification automatique au travers d'un portail captif IPv6

BUT : Créer une application sur PDA permettant de rentrer un login et un mot de passe pour effectuer l'authentification de l'utilisateur sur un portail captif web ipv6. Bonus, intégrer un réflecteur DNS qui traduit les requêtes DNS effectuée en local en requêtes DNS encapsulées dans de l'IPv6.

- Créer une interface pour rentrer les données de connexion
- Etablir la connexion en https avec les serveurs web
- Effectuer l'authentification
- Maintenir la connexion
- Traduire toute requête DNS IPv4 envoyée sur l'adresse 127.0.0.1 du PDA en requête IPv6 valide vers un serveur DNS donné, récupérer le résultat et le renvoyer à l'application l'ayant demandé. Attention les communications DNS s'effectuent en UDP, il faudra donc maintenir des connexions virtuelles pour pouvoir renvoyer le résultat à la bonne application.

Encadrant : Konstantin Kabassanov (mail: Konstantin.Kabassanov@lip6.fr)

Tâches :

- Ecrire un cahier des charges.
- Découper le travail en sous-tâches disjointes.
- Se répartir le travail.
- Implémentation.
- Définir des scénarios de test séparés pour l'authentification et pour le translateur DNS.
- Définir des scénarios de test pour l'ensemble de l'application.

ARCHITECTURE

2 ordinateurs sous Windows XP avec Visual Studio et/ou Embedded C++
1 PDA sous Windows Mobile 2003.

MODALITES

Avant de débiter les projets vous devrez présenter au plus tard la semaine qui précède votre semaine de développements un pré-rapport spécifiant :

- Le cahier de charge.
- Le découpage du projet en sous-projet et la répartition des personnes.
- La conception des 2 sous-applications.
- Le plan d'intégration.

Ce pré rapport devra être validé avant le début des implémentations et des tests.

A la fin de la semaine du projet devront être rendus :

- Le pré-rapport.
- Les sources de l'application.
- Un mini-manuel d'utilisation.

Ce sujet est proposé pour 1 groupe de 4 étudiants

Comparaison du multicast en IPv6 : PIM6-SM vs PIM6-SSM

BUT : obtenir des traces sur le trafic multicast IPv6 dans le cas de PIM-SM et dans le cas de PIM-SSM (implique les protocoles de gestion de groupe MLD et MLDv2)

- Communications entre les récepteurs et les routeurs de bordure
- Communications entre les routeurs de bordure et le RP (rendez-vous point) quand elles existent
- Communications entre la source et les récepteurs ou la source et le RP

Encadrant : Konstantin Kabassanov (mail: Konstantin.Kabassanov@lip6.fr)

Tâches :

- Ecrire un court tutoriel sur le multicast IPv6 et les versions de MLD et PIM implémentées.
- Mettre en place votre plateforme.
- Définir l'architecture dont vous avez besoin pour visualiser ces échanges de trames et de paquets
- Définir les scénarios de test.

ARCHITECTURE

5 ordinateurs dont :

- 2 ordinateurs avec une seule carte réseau
- 1 ordinateur avec 2 cartes réseau
- 2 ordinateurs avec 3 cartes réseau

MODALITES

Avant de débiter les projets vous devrez présenter au plus tard la semaine qui précède votre semaine de développements un pré-rapport spécifiant :

- Une architecture « minimale » de plateforme dont vous aurez besoin pour réaliser ces tests.
- Les échanges de trames ou paquets que vous voudrez mettre en évidence pendant le projet.
- Le tutoriel.

Ce pré rapport devra être validé avant le début des tests et implémentations.

A la fin de la semaine du projet devront être rendus :

- le rapport contenant l'ensemble des traces définies (pré-rapport).
- une version des traces, ceci permettant de visualiser avec Ethereal les échanges de trames/paquets/datagrammes.
- Une explication des filtres à appliquer à Ethereal sur les différentes traces pour visualiser les échanges de trames pertinents.

Ce sujet est proposé pour 1 groupe de **4 étudiants**

Implémentation d'un outil d'analyse de trace NS-2

Encadrants : Luigi Iannone (luigi.iannone@lip6.fr)

Nombre d'Etudiants : 2

Prérequis : Connaissance environnement Unix (BSD)

Description

NS-2 est un des simulateurs les plus répandu dans le monde de la recherche en réseaux de communications. Il a le gros avantage d'être « open source » ce que permet a chacun de implémenter des nouvelles solutions à n'importe quel niveau de la pile protocolaire.

Typiquement une simulation NS-2 produit deux type de fichiers : un fichier d'animation NAM et un fichier de trace qui contient aussi tous les événement qui ce produisent pendant une simulation. Cependant, il n'existe pas un vrai outil qui permet d'analyser cette trace et donner des statistiques de base.

Le but de ce projet est de réaliser un programme d'analyse des traces NS-2 et qui fournit des statistiques de base, de préférence de façon graphique.

Le travail doit se dérouler avec une première phase de apprentissage du simulateur et du format du fichier trace. Ensuite, il faut décider les statistiques de base à fournir.

La dernière phase consiste a implémenter une petit outil d'analyse capable de analyser n'importe quel trace NS-2 et fournir les statistiques de base décide dans la première phase.

Le langage de programmation peut être choisi par les étudiants, mais le logicielle doit être structuré et facilement extensible. Il est fortement conseillé Perl/Tk.

Références

- NS-2 : <http://www.isi.edu/nsnam/ns/>

Implémentation de Wimax sous NS

Encadrants Luigi Iannone (luigi.iannone@lip6.fr) , Benoit Donnet (benoit.donnet@lip6.fr)

Nombre d'Etudiants : Au moins 2.

Pré-requis Intérêt pour Wimax, Langage C/C++

Description

L'objectif du travail est d'implémenter, dans le simulateur NS, une pile permettant de gérer Wimax. Wimax (Worldwide Interopability for Microwave Access) est une norme technique basée sur le standard de transmission radio 802.16, validé en 2001 par l'organisme international de normalisation IEEE.

Le Wimax est développé par le consortium Wimax Forum, qui rassemble aujourd'hui plus de 200 industriels, FAI et opérateurs téléphoniques.

Le standard 802.16a, validé fin 2002, permet d'émettre et recevoir des données dans les bandes de fréquences radio de 2 à 11Ghz, avec un débit maximum de 70 mégabits par seconde et ce, sur une portée de 50km. En pratique, cependant, cela permet d'atteindre les 12 mégabits par seconde sur une portée de 20km.

Après les liaisons fixes de point à point (par exemple, domicile à borne de connexion), le Wimax devrait évoluer vers la mobilité (portable à borne de connexion, ou autre portable, par exemple).

Un des atouts de Wimax est un mécanisme d'allocation de bande passante à la demande (Grant/Request Access). Alors que le Wi-Fi souffre parfois de collision entre paquets et du surcroît de trafic qui en découle, Wimax permet une allocation de bande passante à chaque utilisateur en fonction de ses besoins. Par exemple, si un utilisateur demande à faire une vidéoconférence avec une excellente qualité, l'opérateur lui attribue une priorité haute afin que la transmission soit la plus fluide possible.

Un autre atout de Wimax est son débit et sa portée. Il devrait permettre d'atteindre et fournir de l'accès Internet à des régions difficilement accessible (pour des raisons de coût, par exemple) par des moyens plus conventionnels.

Cependant, afin de tester Wimax, nous avons besoin de le modéliser dans des outils classiques. Il existe plusieurs façons de le faire:

- Implémentation dans MatLAB d'un module Wimax
- Implémentation d'une pile Wimax dans le simulateur NS

Dans ce travail, nous proposons aux étudiants de développer dans NS une pile Wimax. Il est demandé de fournir, au minimum, une implémentation permettant de faire tourner des simulations basiques sous NS.

Références

- Le simulateur NS-2: <http://www.isi.edu/nsnam/ns/>
- Le Wimax Forum: <http://www.wimaxforum.org/home/>
- IEEE 802.16: <http://www.ieee802.org/16/>

Étude des performances des réseaux sans fil

BUT : Mettre en évidence l'impact des communications sans fil sur les performances du trafic utilisateur.

Nombre d'étudiants : 4

Encadrante : Anne Fladenmuller (Anne.fladenmuller@lip6.fr)

TACHES

Différents aspects peuvent être montrés :

- les notions d'interférences entre points d'accès communiquant sur le même canal,
- l'impact du signal sur bruit sur les débits de transmission (fonction de la distance du portable à l'AP),
- la chute du débit d'une connexion sur un réseau ad hoc multi-sauts,
- ...

Vous devrez réaliser des expérimentations pour mettre en évidence ces comportements et réfléchir à d'autres cas de figure qui peuvent induire des baisses de performances pour le trafic utilisateur.

ARCHITECTURE

Vous disposerez de 3 portables munis de cartes WIFI pour réaliser des tests mettant en évidence le comportement des réseaux sans fil et un AP.

MODALITES

Avant de débiter les projets vous devrez présenter au plus tard la semaine qui précède votre semaine de développements un pré-rapport spécifiant :

- Le cahier de charge.
- Le découpage du projet en sous-projet et la répartition des personnes.
- La conception de l'application.

Ce pré rapport devra être validé avant le début des implémentations et des tests.

A la fin de la semaine du projet devront être rendus :

- Le pré-rapport.
- Les sources de l'application.
- Un mini-manuel d'utilisation.