الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Master

Filière Télécommunication

Spécialité Réseaux & Télécoms

Présenté par

Karim Mohamed Adlane

&

Boudilmi Souad

Interception d'une connexion P2P dans un réseau d'entreprise

« BitTorrent Vs E-Donkey »

Proposé par : Mr. M. MEHDI

Année Universitaire 2020-2021

En guise de reconnaissance, nous tenons à témoigner nos sincères remerciements à toutes les personnes qui ont contribués de près ou de loin au bon déroulement de notre mémoire de fin d'étude et à l'élaboration de ce travail.

Tout d'abord nous voudrions présenter nos sincères remerciements à notre encadreur « Mr. MEHDI Merouane », et nous voudrions également lui témoigner notre gratitude pour son soutien et ses conseils qui nous ont été très précieux afin de mener notre travail à bon port, Merci.

Nos vifs remerciements vont également au prestigieux membre du jury et à tout le staff du département de Génie Electronique de l'université de SAAD DAHLAB Blida 1, très spécialement nos professeurs durant tout ce cycle de cinq ans.

Enfin, nous remercions chaleureusement nos chers parents pour leur soutien et leur patience, qui ont étaient toujours là dans les moments difficiles.

Dans l'impossibilité de citer tous les noms, nos sincères remerciements vont à tous ceux et celles, qui de près ou de loin, ont permis par leurs conseils et leurs compétences la réalisation de ce mémoire.

ملخص:

في السنوات الأخيرة، إكتسبت شبكات الند للند شعبية في شكل تطبيقات مشاركة الملفات مثل'' مي تورنت '' و'' أ ميول''. مع هذه الشعبية، تأتي الآثار الأمنية، ونقاط الضعف، وتشبع النطاق الترددي. في هذه المذكرة، ندرس الإطار الذي بنيت عليه معظم شبكات الند للند، ونقوم بفحص حزم حركة مرور الشبكة من أجل استخراج البصمات الرقمية لهذين التطبيقين، وذلك بفضل برنامج تحليل الحزم '' وايرشارك''. سندرس كيفية الكشف عن استخدام كل من بروتوكولات '' بيت تورنت'' و '' اي دونكي'' في شبكة شركة بنظام كشف التسلل " سنورت ". سمح لنا هذا الحل بإكتشاف إستخدام شبكات الند للند بشكل فعال للغاية.

كلمات المفاتيح: شبكة الند للند، أي دونكي، بيت تورنت، تحليل حركة المرور، نظام كشف التسلل، سنورت.

Résumé :

Ces dernières années, les réseaux pair à pair (P2P) ont gagnés en popularité sous la forme d'applications de partage des fichiers tel que ''µtorrent" et "A-Mule". Cette popularité s'accompagne d'implications, de vulnérabilités en matière de sécurité et saturation de bande passante. Dans ce mémoire, nous examinons le cadre sur lequel la plupart des réseaux Peer to Peer sont construits, on va faire une inspection des paquets de trafic réseau afin d'extraire les empreintes numériques de ces deux applications, grâce au logiciel d'analyse du paquets "Wireshark". Nous examinons comment peut-on détecter l'utilisation des deux protocoles "BitTorrent'" et "E-Donkey" dans un réseau d'entreprise a l'aide d'un système de détection d'intrusion "Snort". Cette solution nous a permis de détecter l'utilisation du Peer to Peer de manière très efficace.

Mots clés : Réseau pair à pair, E-Donkey, BitTorrent, Analyse du trafic, empreintes numériques systèmes de détections d'intrusion, Snort.

Abstract :

In recent years, peer to peer (P2P) networks have gained popularity in the form of file sharing applications such as utorrent and A-Mule. With this popularity comes security implications, vulnerabilities, and bandwidth saturation. In this thesis, we examine the framework on which most Peer to Peer networks are built, we will do an inspection of the network traffic packets in order to extract the digital prints of these two applications, and with using packet analysis software. '' Wireshark ''. We are examining how to detect the use of both BitTorrent and edonkey protocols in a corporate network with a "Snort" intrusion detection system. This solution allowed us to detect the use of P2P very effectively

Keywords : Peer to Peer network, E-Donkey, BitTorrent, Traffic analysis, digital prints, intrusion detection systems, Snort

Liste des abréviations

ADSL : Asymetric Digital Subscriber Line.

BASE : Basic Analysis and Security Engine.

CAN : Content Adressable Network

CMD : Command prompt.

DDOS : Distributed Denial of Service.

DOS : Denial of Service.

DHCP : Dynamic Host Configuration Protocol.

DHT : Distributed Hash Table.

Ed2k : EDonkey2000 network.

FTP : File Transfer Protocol.

HIDS : Host Intrusion Detection Systems.

HTTP : HyperText Transfer Protocol.

HTTPS : HyperText Transfer Protocol Secure.

ICMP : Internet Control Message Protocol.

ID : Identifier.

IDS : Intrusion Detection Systems.

IP : Internet Protocol.

IPV4 : Internet Protocol version 4.

IPV6 : Internet Protocol version 6.

KAD : Kademlia.

MD4 : Message Digest 4.

MP3 : MPEG Audio Layer 3.

MSG : Message.

NAT : Network Address Translation.

NIDS : Network intrusion detection systems.

OSI : Open Systems Interconnection.

Liste des abréviations

PAT : Port Address Translation

PCAP : Packet capture.

P2P : Peer to Peer.

QOS : Quality Of Service

RC4 : Rivest Cipher 4.

RES : Respond.

REQ : Request.

SGBD : Système de Gestion de Base de Données.

SID : Service Identifier.

TCP : Transmission Control Protocol.

TLS : Transport Layer Security.

TTL : Time To Live.

UDP : User Datagram Protocol.

VPN : Virtual Private Network.

Table des matières

Chapitre II : Classification des réseaux Peer to Peer

Chapitre III : Inspection des paquets P2P

Liste des figures

Liste des tableaux

Introduction générale

En quelques années seulement, le monde de l'Internet a connu un développement fulgurant, que ce soit chez les particuliers ou dans le domaine professionnel offrant aux utilisateurs du globe un moyen rapide et efficace pour partager des informations. Ce monde est en effervescence autour d'un phénomène portant le nom de Peer to Peer qui constitue une part significative du trafic total dans de nombreux réseaux.

A l'arrivé de ce dernier, il n'est plus rare de voir des échanges de plusieurs Giga octets de données sur Internet à travers des petites machines. Il ne se limite plus aujourd'hui au partage de fichiers, mais aussi au travail collaboratif...etc.

Beaucoup de problèmes se posent devant ces avantages. Les gens qui utilisent une application Peer to Peer, ils consommeront une énorme quantité de bande passante aussi ces applications sont sujettes à une large diffusion des logiciels piratés, même le taux d'épidémie de virus sur le réseau est très élevé et très fréquemment.

E-Donkey et BitTorrent sont l'un des protocoles de partage de fichiers Peer to Peer les plus populaires, accessible par tout le monde et offrant aux utilisateurs une méthode simple et facile pour le partage et téléchargement des fichiers.

L'utilisation de ces derniers dans un réseau d'entreprise est mal identifiée pour les internautes, car ils exposent l'entreprise à divers risques de sécurité et à des problèmes judiciaires. Pour cela les organisations et les entreprises commencent à prêter attention au risque d'utilisation du réseau Peer to Peer dans leurs réseaux.

Afin de surveiller le réseau et détecter l'utilisation du Peer to Peer, les systèmes de détection d'intrusion réseau (NIDS) ont été proposé comme solution par l'identification des signatures des protocoles par rapport à un flux temps réel.

Pour toutes ces raisons on cherche à mettre en place une solution fiable basée sur le NIDS contre l'utilisation de ces protocoles.

A cet effet nous allons aborder le phénomène du Peer to Peer avec un regard technique sur les deux protocoles E-Donkey et BitTorrent, leurs principes et les différents

risques qu'ils peuvent causer. Par la suite on proposera une technique de détection de l'utilisation de ces derniers en passant par les étapes suivantes :

01- Analyse profonde du trafic de réseau à l'aide de l'analyseur de paquets ''Wireshark''.
02- Extraction des empreintes P2P.
03- Implémentation des signatures au niveau de l'IDS ''Snort'' afin de valider l'interception des utilisateurs P2P.

Dans notre premier chapitre nous aborderons les généralités sur les réseaux informatiques, puis à travers le chapitre II nous allons voir de près le monde du Peer to Peer, ses avantages et ses inconvénients. Dans le chapitre III nous procédons à l'étude minutieuse d'analyse des paquets Peer to Peer pour l'extraction des empreintes numérique. Enfin, le dernier chapitre nous permettra de démontrer la fiabilité de ces signatures à travers l'implémentation de ces derniers dans l'IDS.

Chapitre I

Généralités sur les réseaux informatique

I.1. Introduction

Avec l'arrivée de l'Internet et son rôle actif croissant dans l'économie et la société, offrant aux utilisateurs du monde entiers un moyen rapide et efficace pour le partage des informations, l'informatique réseau n'a eu de cesser de trouver des innovations pour exploiter toutes les ressources afin d'atteindre cet objectif.

Dans ce chapitre nous allons voire quelques notions sur les réseaux informatique avec un aperçu sur les différentes couches du modèle TCP/IP, ensuite en présentant les types des réseaux informatiques, ainsi que la présentation de base du réseau Peer to Peer.

Et enfin nous citerons brièvement quelques champs d'application de ce dernier.

I.2. Définition des réseaux informatiques

Un réseau informatique est la base d'un réseau de communication, c'est un ensemble d'équipements informatiques reliés entre eux grâce à des supports de transmissions afin de pouvoir permettre aux utilisateurs du réseau d'échanger des informations et partager les différentes ressources matérielles et logicielles.

I.3. Modèle TCP/IP

TCP / IP signifie (Transmission Control Protocol / Internet Protocol) il est nommé d'après deux protocoles principaux (TCP et IP). C'est un système d'architecture de serveur en couches dans lequel chaque couche est définie selon une fonction spécifique à exécuter. Il a pour but d'acheminer les données entre l'émetteur et le destinataire, largement utilisé dans l'architecture Internet actuelle.

Le modèle TCP / IP comporte quatre couches qui travaille en collaboration pour transmettre les données d'une couche a une autre. Ces couches sont très similaires aux couches du modèle OSI.

On peut noter la différence entre les deux modèles par le schéma suivant :

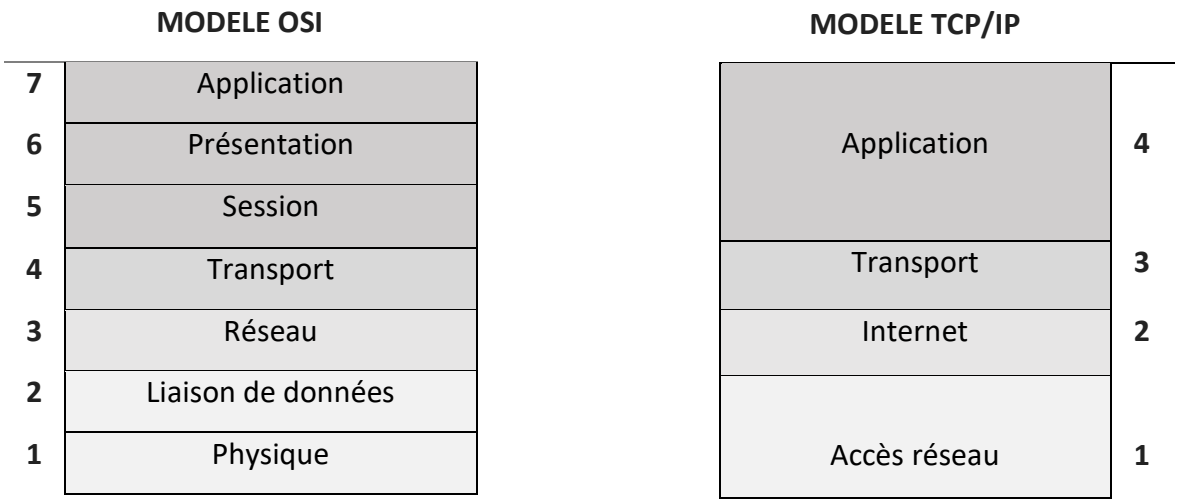


Figure I.1: Modèle OSI et TCP/IP

I.3.1. Couche Accès réseau

Les protocoles de cette couche permettent au système de fournir des données aux autres appareils sur un réseau directement connecté. Il définit comment utiliser le réseau pour transmettre un datagramme IP. Contrairement aux protocoles de niveau supérieur, les protocoles de la couche d'accès réseau doivent connaître les détails du réseau sous-jacent (sa structure de paquets, son adressage, etc.) pour former correctement les données transmises afin de respecter les contraintes du réseau [1].

I.3.2. Couche Internet

Elle traite des données sous forme de datagrammes ou de paquets de données. Cette couche effectue principalement l'adressage logique des paquets de données en y ajoutant l'adresse IP (Internet Protocol), l'adressage IP peut être effectué soit en utilisant le protocole Internet version 4 (IPv4) ou le protocole Internet version 6 (IPv6).

La couche Internet effectue également le routage des paquets de données en utilisant les adresses IP [1].

I.3.3. Couche de transport

Elle traite des données sous la forme de segments de données. Elle effectue principalement la segmentation des données reçues des couches supérieures. Elle est responsable du transport des données et de l'établissement de la communication entre la

couche application et les couches inférieures. Cette couche facilite la communication et la livraison sans erreur des données [1].

La couche Transport facilite le contrôle de l'encombrement à l'aide des protocoles suivants :

i- TCP

TCP signifie ‘’Transmission Control Protocol’’. C'est un protocole orienté connexion. Il effectue le séquençage et la segmentation des données. Il effectue également un contrôle de flux et d'erreurs dans la transmission de données. Il existe une fonction d'accusé de réception dans TCP pour les données reçues. C'est un protocole lent mais fiable. Il convient aux éléments de données importants et non en temps réel.

ii- UDP

UDP signifie ‘’User Datagram Protocol’’. C'est un protocole qui n'effectue pas de contrôle de flux et d'erreurs lors de la transmission de données. Il n'y a pas de fonction d'acquittement dans UDP pour les données reçues. C'est un protocole rapide mais peu fiable. Il convient aux éléments de données en temps réel.

TCP	UDP
Connexion orientée	Sans connexion
Fiable	Non fiable
Lent	Plus rapide
Frais généraux d'en-tête plus élevés	Frais généraux d'en-tête inférieur
Mécanismes complets de vérification des erreurs	Vérification des erreurs très basique
Séquençage des paquets de données	Pas de séquençage des paquets de données
Retransmission des paquets perdus	Pas de retransmission

Tableau I-1: Comparaison entre TCP et UDP

I.3.4. Couche d'application

La couche Application dans le modèle TCP / IP équivaut aux trois couches supérieures (couche Application, Physique et Session) du modèle OSI. Elle traite la communication de l'ensemble du message des données. La couche Application fournit une interface entre les services réseau et les programmes d'application. Elle fournit principalement des services aux utilisateurs pour travailler sur le réseau. Par exemple, transfert de fichiers, navigation Web, etc. Cette couche utilise tous les protocoles de niveau supérieur comme HTTP, HTTPS, FTP,...etc. [1].

I.4. Architecture réseau

L'architecture de réseau informatique définit le cadre physique et logique d'un réseau informatique. Elle décrit comment les ordinateurs sont organisés dans le réseau. Les composants de l'architecture réseau comprennent le matériel, les logiciels, les supports de transmission (filaires ou sans fil), la topologie du réseau et les protocoles de communication.

Les deux types d'architectures réseau utilisées sont :

I.4.1. Client / serveur

L'architecture client/serveur désigne un mode de communication entre plusieurs composants d'un réseau. Client-Serveur est l'un des styles architecturaux distribués les plus connus, composé de deux composants, le fournisseur et le consommateur.

Le fournisseur est un serveur qui fournit une série des services ou des ressources consommés par le client.

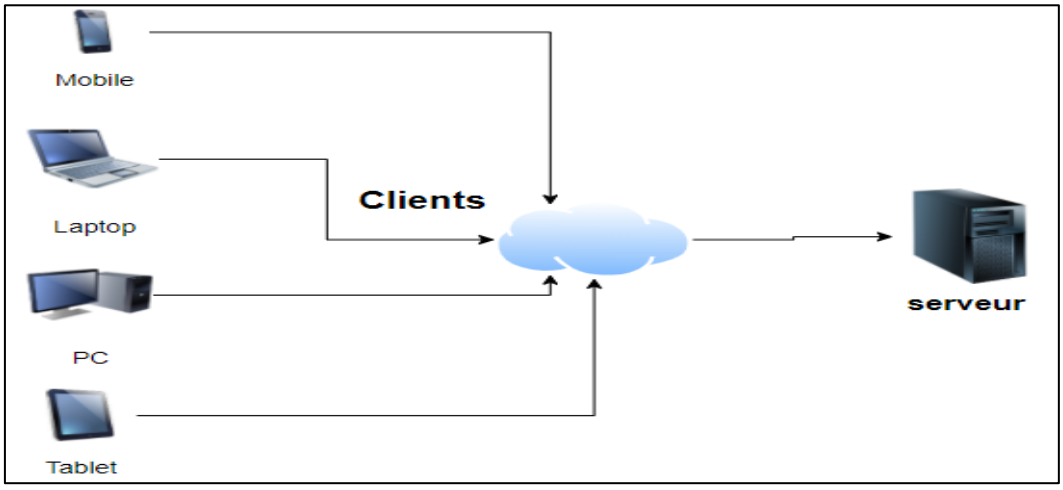


Figure I.2: Modèle de réseau client – serveur

I.4.2. Peer to Peer

Un réseau informatique Peer to Peer fait référence à un réseau de partage de toutes sortes de ressources informatiques qui n'a pas de client ou de serveur fixe, mais une série des nœuds qui se comportent simultanément en tant que clients et serveurs par rapport aux autres nœuds du réseau. On peut considérer que chaque utilisateur d'un Peer to Peer est à la fois un client et serveur.

Les utilisateurs Peer to Peer stockent des fichiers sur leurs ordinateurs et l'application Peer to Peer permet aux autres utilisateurs de télécharger ces fichiers sur leurs ordinateurs.

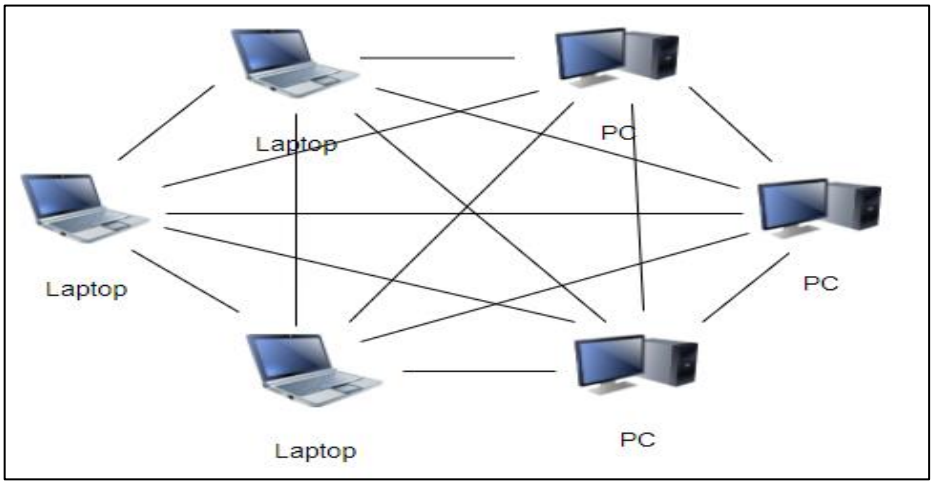


Figure I.3: Modèle de réseau Peer to Peer

I.5. L'historique de réseau Peer to Peer

Le Peer-Peer : Un concept redécouvert par Napster

Fin 1998, Shawn Fanning, un étudiant américain passionné d'informatique décide de quitter l'université et se lance dans l'écriture d'un logiciel pour permettre l'échange de fichiers musicaux. La raison d'être de ce logiciel repose sur le constat suivant : rechercher des MP3 sur les moteurs de recherche habituels conduit à une perte de temps énorme et les réponses sont souvent inappropriées. Après quelques mois de travail acharné, une première version du logiciel est disponible. Fanning décide de tester une première version le 1er juin 1999 et appelle son logiciel Napster (son pseudo sur Internet). Le logiciel qui ne devait être testé que par quelques-uns de ses amis remporte un succès des plus rapides [2].

La fin de Napster

Tombent alors les premières interdictions de la part des universités : les étudiants l'utilisent tellement qu'ils saturent les bandes-passantes. Les groupes de musique demandent à ce qu'on protège leurs droits. En janvier 2001, le verdict de la 9e cour d'appel de San Francisco tombe : Napster viole la loi sur les droits d'auteurs et devra cesser dans un bref délai l'échange gratuit de fichiers musicaux MP3. C'est une victoire importante pour les maisons de disques, même si elles n'obtiennent pas la fermeture immédiate de Napster [2].

L'arrivée de nouveaux logiciels : le pur Peer-Peer

Devant les menaces de la justice pour violation des droits d'auteurs puis l'interruption du service sur décision judiciaire, de nouveaux logiciels similaires apparaissent. Mais au lieu de faire appel à un serveur centralisé indexant les ressources pour mettre en relation les usagers, ils transfèrent cette fonction d'annuaire à des milliers d'ordinateurs par le monde. Le principe du pur Peer to Peer est né [2].

I.6. Caractéristique de Peer to Peer

Les systèmes Peer to Peer possèdent des caractéristiques avantageuses par rapport aux autres systèmes basés sur le paradigme Client/serveur. Pour cette raison, ils sont très populaires. Même s'il n'y a pas, à notre connaissance, un système Peer to Peer possédant toutes les caractéristiques suivantes :

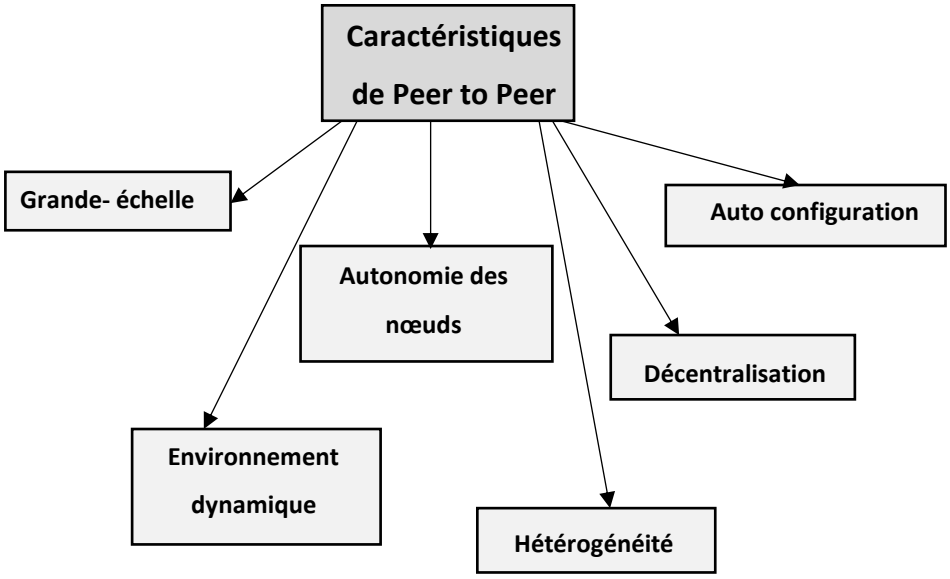


Figure I.4: Caractéristiques de Peer to Peer

- **Grande- échelle**

Il s'agit de faire coopérer un grand nombre des nœuds (jusqu'à des milliers ou des millions) pour partager leurs ressources tout en maintenant une bonne performance des systèmes [3].

- **Autonomie des nœuds**

Chaque nœud gère ses ressources d'une façon autonome. Il décide quelle partie de ses données à partager. Il peut se connecter ou/et se déconnecter à n'importe quel moment [3].

- **Environnement dynamique**

À cause de l'autonomie des nœuds, chaque nœud peut quitter le système à n'importe quel moment ce qui fait disparaître ses ressources du système. De nouvelles ressources peuvent être ajoutées au système lors de la connexion des nouveaux nœuds [3].

- **Hétérogénéité**

À cause de l'autonomie des nœuds possédant des architectures matérielles et/ou logicielles hétérogènes, les systèmes Peer to Peer doivent posséder des techniques convenables pour résoudre les problèmes liés à l'hétérogénéité de ressources [3].

- **Décentralisation**

Le fait que chaque nœud gère ses propres ressources permet d'éviter la centralisation de contrôle [3].

- **Auto configuration**

Puisque les systèmes Peer to Peer sont souvent déployés sur l'Internet, la participation d'un nouveau nœud à un système Peer to Peer ne nécessite pas une infrastructure coûteuse [3].

I.7. Avantages du modèle Peer to Peer

Le Réseau Peer to Peer a des fonctionnalités qui le distinguent des autres architectures d'Internet. Parmi ces avantages, nous avons mentionné :

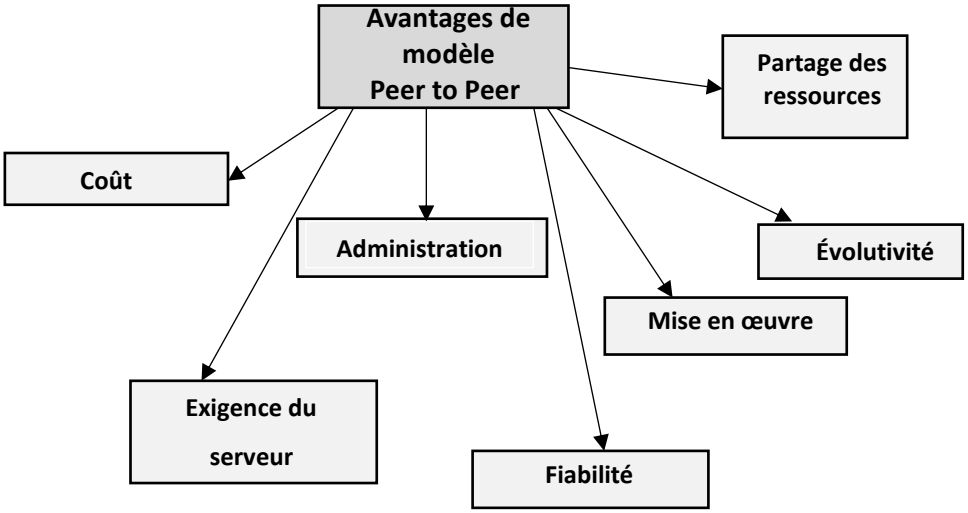


Figure I.5: Avantages de Peer to Peer

- **Coût**

Le coût global de création et de maintenance d'un réseau Peer to Peer est relativement peu coûteux [4].

- **Fiabilité**

Le réseau Peer to Peer ne dépend pas d'un système centralisé. Ce qui signifie que les ordinateurs connectés peuvent fonctionner indépendamment les uns des autres [4].

- **Mise en œuvre**

Il est généralement facile de configurer un réseau Peer to Peer ne nécessitant aucune connaissance avancée. Seul un concentrateur ou un commutateur est nécessaire pour la connexion [4].

- **Évolutivité**

Le réseau Peer to Peer possède l'une des meilleures fonctionnalités d'évolutivité. Même si des clients supplémentaires sont ajoutés, les performances du réseau resteront les mêmes [4].

- **Administration**

Aucun administrateur réseau spécialisé n'est nécessaire puisque tous les utilisateurs ont le droit de gérer leur propre système [4].

- **Exigence du serveur**

Dans la mise en réseau Peer to Peer, chaque ordinateur connecté agit comme un serveur et un poste de travail. Par conséquent, il n'est pas nécessaire d'utiliser un serveur dédié [4].

- **Partage des ressources**

Dans le réseau Peer to Peer, les ressources sont partagées de manière égale entre tous les utilisateurs. Ce réseau Peer to Peer peut être utilisé pour localiser et télécharger facilement des fichiers en ligne [4].

I.8. Problèmes de réseau Peer to Peer

I.8.1. Sécurité réseaux

Peer to Peer est devenu une méthode de mise en réseau populaire qui augmente la robustesse et l'évolutivité. Cependant, les modèles informatiques Peer to Peer distribués et décentralisés présentent de sérieux problèmes de sécurité.

- **DOS**

Les réseaux Peer to Peer demandant énormément de ressources et impliquant des milliers d'utilisateurs, ils peuvent être détournés pour lancer des attaques par dénis de services (DoS) et dénis de services distribués (DDoS). Les réseaux Peer to Peer peuvent être utilisés pour lancer des attaques distribuées vers une cible. Par exemple, un serveur eDonkey (ou un tracker BitTorrent) spécialement préparé peut donner les mêmes adresses cibles à tous les clients qui vont se connecter à eux.

- **Attaque Sybil**

L'attaque Sybil est une attaque visant à éviter les systèmes de partages basés sur la réputation. Elle consiste à voler une identité (possédant beaucoup de crédit) ou à se forger de nombreuses identités pour casser le système de réputation [6].

- **Attaque Eclipse**

L'attaque Eclipse vise principalement les réseaux Peer to Peer structurés. Dans ces réseaux, chaque nœud maintien des pointeurs vers ses voisins. L'attaque consiste alors à contrôler plusieurs pairs influant sur le réseau (une grande partie des voisins), pour ne pas rediriger le trafic vers les bons pairs et donc modifier l'utilisation normal du routage [6].

I.8.2. Consommation de la bande passante

Parmi les grands inconvénients de la technologie Peer to Peer on a le problème de consommation de bande passante, non seulement pour les téléchargements mais aussi pour le partage des fichiers.

Certains protocoles qui sont basé sur cette technique consommée jusqu'au 100% de la bande passante (BitTorrent, EDonkey…), car ils utilisent des techniques de téléchargement permet de télécharger et partager des grands fichiers rapidement.

I.8.3. Copyright

Peer to Peer est une technologie puissante qui a de nombreuses utilisations. Elles peuvent être utilisés pour partager et échanger de la musique, des films, des logiciels et d'autres matériels électroniques. L'utilisation de réseaux Peer to Peer pour télécharger ou partager du matériel protégé par des droits d'auteur, peut violer les droits des titulaires de droits d'auteur. Ce qui conduit à un suivi judiciaire.

I.9.Technique d'anonymat

I.9.1. Proxy

Un serveur proxy est un service qui fonctionne comme un relais entre le client et le serveur, c'est un hôte à double hébergement avec deux adresses IP réseau. L'adresse du côté sortant est celle que voit Internet. Les proxys sont souvent utilisés en conjonction avec la traduction d'adresses réseau (NAT), qui masque les adresses IP des utilisateurs sur le réseau interne.

Les serveurs proxy anonymes permettent aux utilisateurs de surfer sur le Web et de garder leur adresse IP privée, il peut gérer, filtrer le flux des donnes entrant/ sortant de réseau.

I.9.2. VPN

Un VPN, "virtual private network" en anglais ou réseau privé virtuel en français, fait en sorte que votre ordinateur ne puisse plus être suivi.

Voici comment cela fonctionne : au lieu de se connecter à votre fournisseur d'accès internet, votre ordinateur se connecte à un serveur VPN, via une connexion sécurisée et

cryptée. Ensuite, le serveur du VPN contacte le site internet sur lequel vous souhaitez aller. Les informations concernant votre visite y sont enregistrées, comme d'habitude, mais avec l'adresse IP du serveur VPN comme référence, et non pas celle de votre ordinateur.

Les VPN, en résumé, vous mettent à l'abri des regards indiscrets quand vous surfez sur internet. Les serveurs VPN sont extrêmement puissants, et sont capables de protéger des centaines d'utilisateurs en même temps.

I.10. Conclusion

Au cours de ce premier chapitre on a présenté globalement des généralités autour les réseaux informatiques. Tout d'abord on a défini la notion de réseau informatique. On a vu aussi la conception du modèle TCP/IP avec ces quatre couches et les protocoles TCP et UDP. Ensuite, on a présenté les deux réseaux client-serveur et le Peer to Peer avec son historique, ces caractéristiques, ces avantages et ces inconvénients.

Le prochain chapitre va nous permettre de voir de près le Peer to Peer.

Chapitre II

Classification des réseaux Peer to Peer

II.1. Introduction

Le réseau informatique Pair à Pair (en anglais Peer to Peer) devient la plus grande source de téléchargement au monde. Ce réseau est basé sur des nouvelles techniques de transfert de données. On va présenter dans ce chapitre les différentes architectures Peer to Peer existantes, avec leurs principes de fonctionnement. Notre focalisation se basera sur les deux protocoles de partage Peer to Peer, E-Donkey et BitTorrent, et pour finir une synthèse sera présentée sur les logiciels d'échange de fichier Peer to Peer.

II.2. Classification des réseaux Peer to Peer

En général, Il est également possible de regrouper ces architectures en deux grandes catégories souvent utilisées pour classifier les réseaux Peer to Peer :

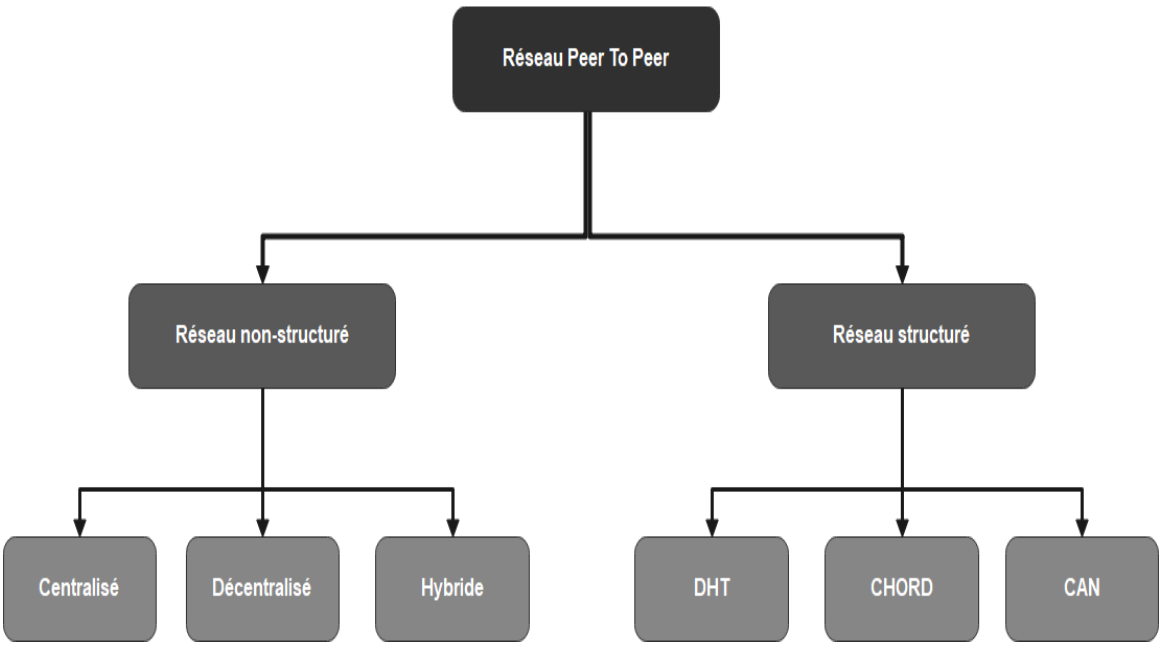


Figure II.1: Classifications des réseaux Peer to Peer

II.2.1. Réseau Peer to Peer non-structuré

Réseau simple et facile à mettre en œuvre, toutes les Peers sont indépendantes mais la propagation des requêtes se fait de manière automatique par inondation en interrogeant toutes les Peers du réseau jusqu'à obtention du le Peer concernée. Le temps

de localisation des Peers est grand à cause de l'absence de critère de répartition des données sur les Peer [7].

On peut classifier cette catégorie en 3 sous-catégories :

i. Réseau Peer to Peer centralisé

La première génération des réseaux de partage, consiste à utiliser un serveur central qui gère toutes les tâches :

- L'identité des utilisateurs.
- L'indexation des Fichiers.
- Les recherches et la mise en relation des Peers (le serveur donne à chaque client la localisation du fichier recherché).

L'échange des fichiers se fait directement entre les Peers sans passer par le serveur. Ce système permet de centraliser l'index mais les fichiers sont décentralisés [8].

❖ **Principe de Fonctionnement :**

Dans ce type de réseau, si un utilisateur veut :

- **Joindre le réseau :** il doit s'enregistrer au niveau du serveur.
- **Partager un fichier :** l'utilisateur doit se déclarer au serveur central.

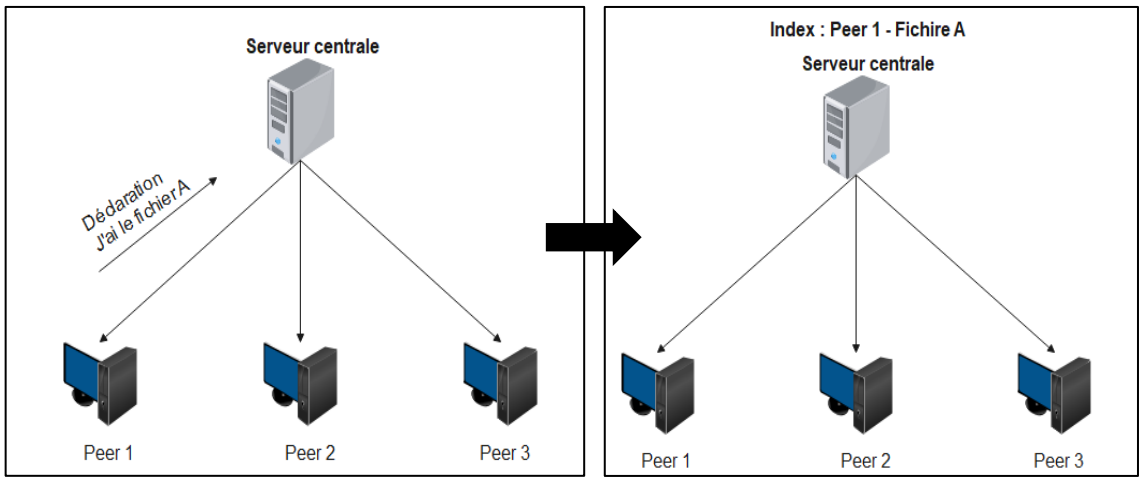


Figure II.2: Partage d'un fichier – Peer to Peer centralisé

- **Télécharger un fichier :** Pour obtenir un fichier, il suffit juste d'interroger l'index central, qui va envoyer une adresse IP.

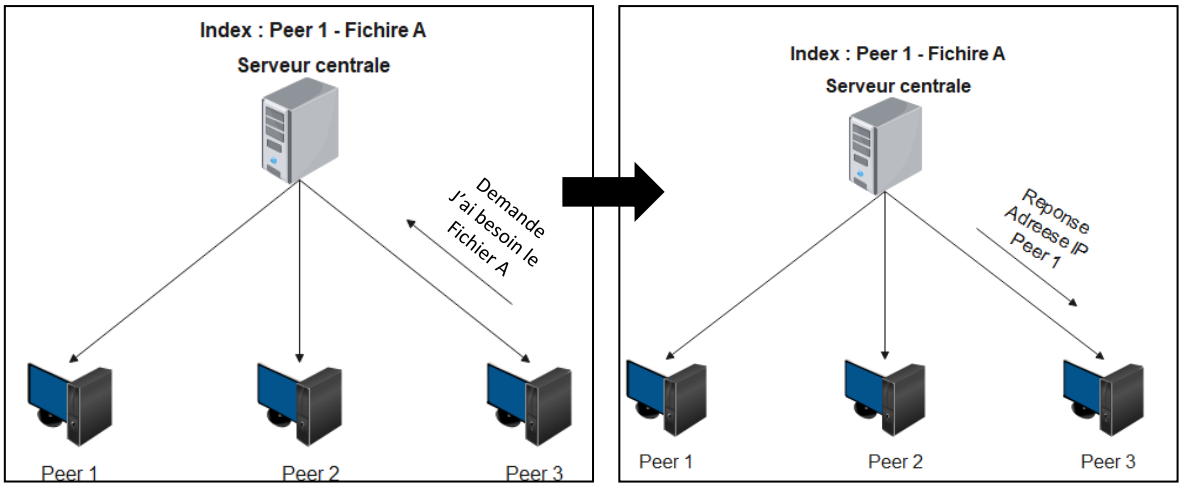


Figure II.3: Recherche d'un fichier – Peer to Peer centralisé

Par la suite, il faut utiliser un logiciel de partage (Napster, Audiogalaxy…) qui va faire la liaison entre les deux Peers sans la nécessité du serveur.

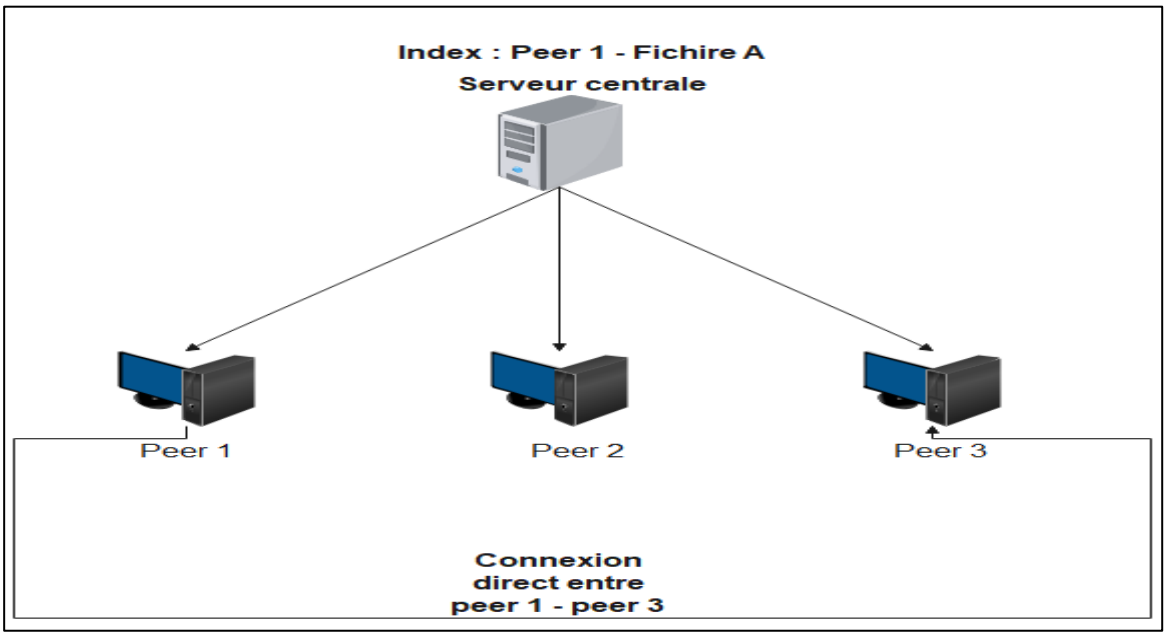


Figure II.4: Téléchargement de fichier – Peer to Peer centralisé

Le serveur central fonctionne comme un annuaire (son rôle est terminé une fois l'adresse IP trouvée). Cependant sans le serveur central, le réseau ne peut pas fonctionner.

Dans le but d'améliorer la performance de ce système, il est possible de mettre en place un groupe de serveur, sachant que l'accès aux données partagées dans chaque serveur est totalement transparent pour les autres utilisateurs.

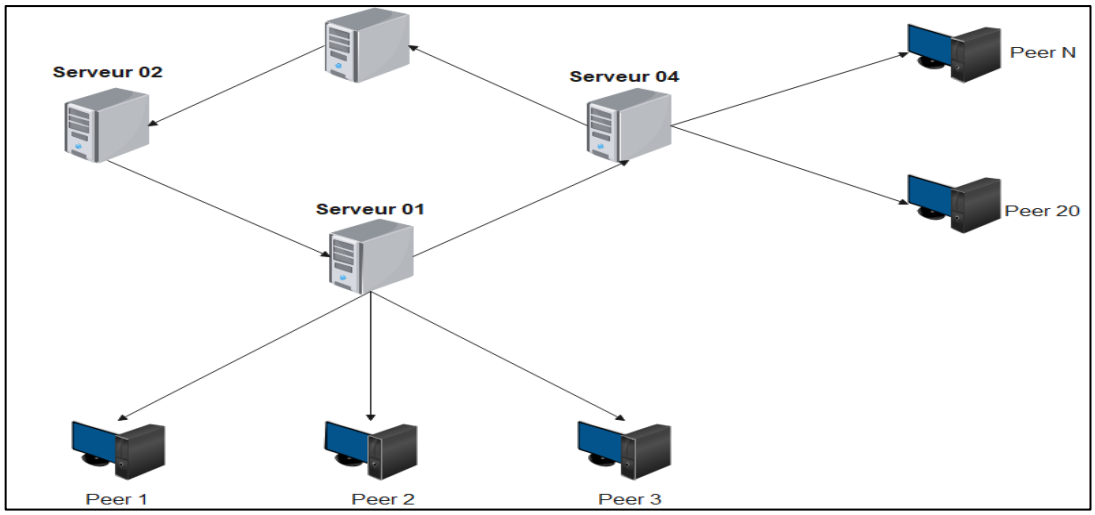


Figure II.5: Technique d'amélioration de système Peer to Peer centralisé

Le réseau Peer to Peer centralisé assure un volume de trafic réduit car la communication entre les Peers dépend du besoin seulement, par contre il est très faible en terme de sécurité car il suffit de bloquer le serveur central pour bloquer tous les autres utilisateurs, ainsi toutes les adresses IP sont visibles au niveau du serveur, donc il suffit juste d'accéder au serveur pour voir toutes les informations qui concernent les clients.

ii. Réseau Peer to Peer décentralisé (purs)

Les Réseaux décentralisés dits « purs », représente la 2eme génération des réseaux non-structurés. Dans ce système, toutes les composantes du réseau sont égales et jouent le même rôle. Chaque élément joue à la fois le rôle du client et le rôle de serveur [8].

Servent : Serveur + client

La différence entre ce modèle et le modèle de 1ère génération est l'absence du serveur central.

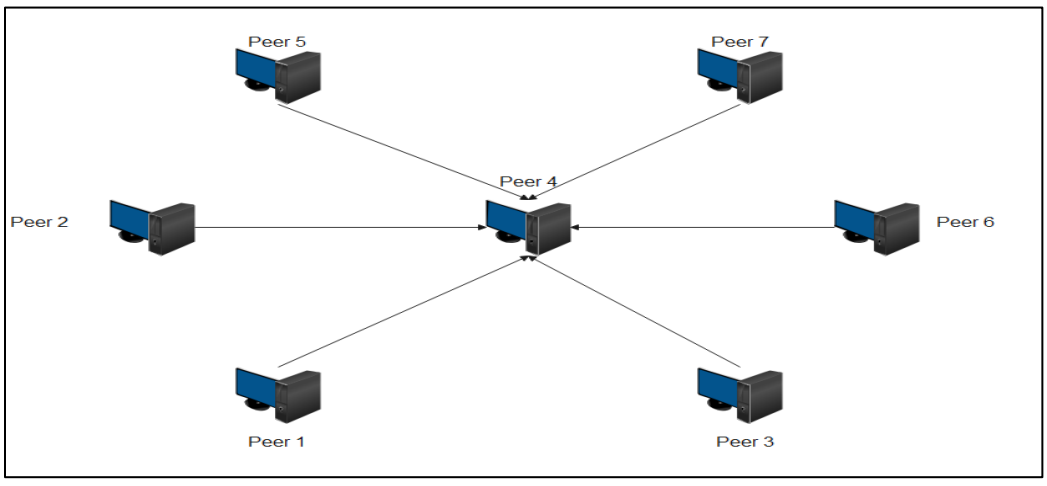


Figure II.6: Architecture Peer to Peer décentralisé

❖ **Principe de fonctionnement :**

Pour bien comprendre le principe de fonctionnement de ce réseau, on va prendre le réseau **Gnutella** comme un exemple d'étude, d'où le client a pour but de :

- **Joindre le réseau :** il faut connaître au moins un servent déjà présent sur le réseau, par la suite la découverte des autres servent ce fait à l'aide de Ping.
- **Trouver le premier servent :** en utilisant la base des données (http://gnutellahosts.com).

Les Peers qui reçoivent un ping, doivent répondre avec une (ou plusieurs) requête(s) qui indique leur adresse IP, numéro de port ainsi que le nombre et la quantité de données partagées.

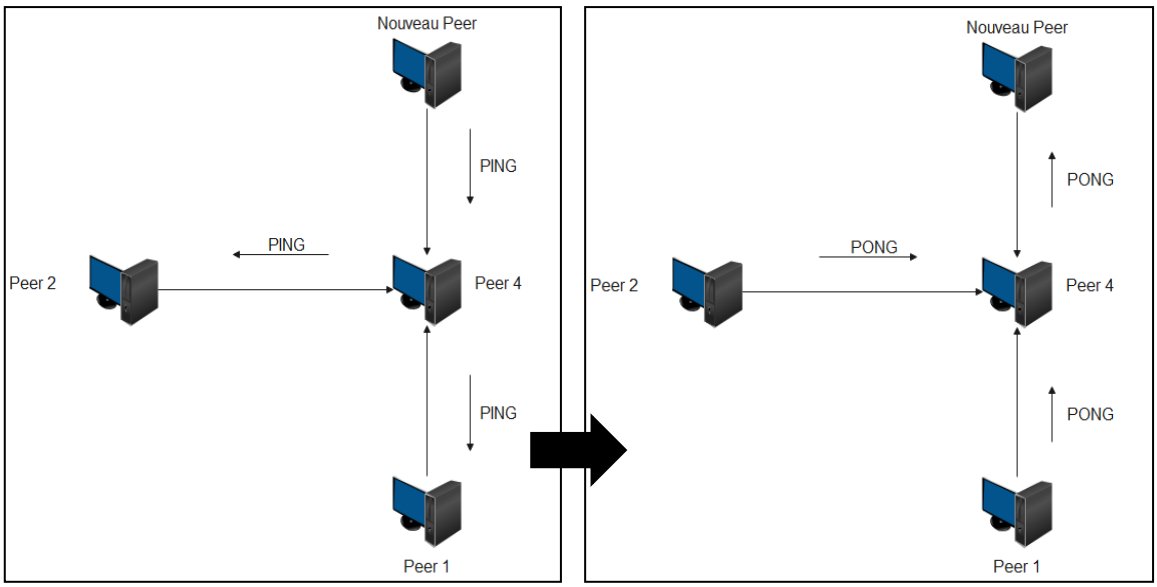


Figure II.7: Connexion d'un Peer / Peer to Peer décentralisé

- **Échange des fichiers :** Si un servent cherche à télécharger un fichier dans le réseau, il envoie une requête **QUERY** à ses voisins.
  - Si le voisin détient le fichier, il va répondre avec une requête **QUERYHIT**.
  - Sinon, il diffuse la requête Query à ses voisins.
- Cette requête est contrôlée par la valeur de **TTL** (time to live).

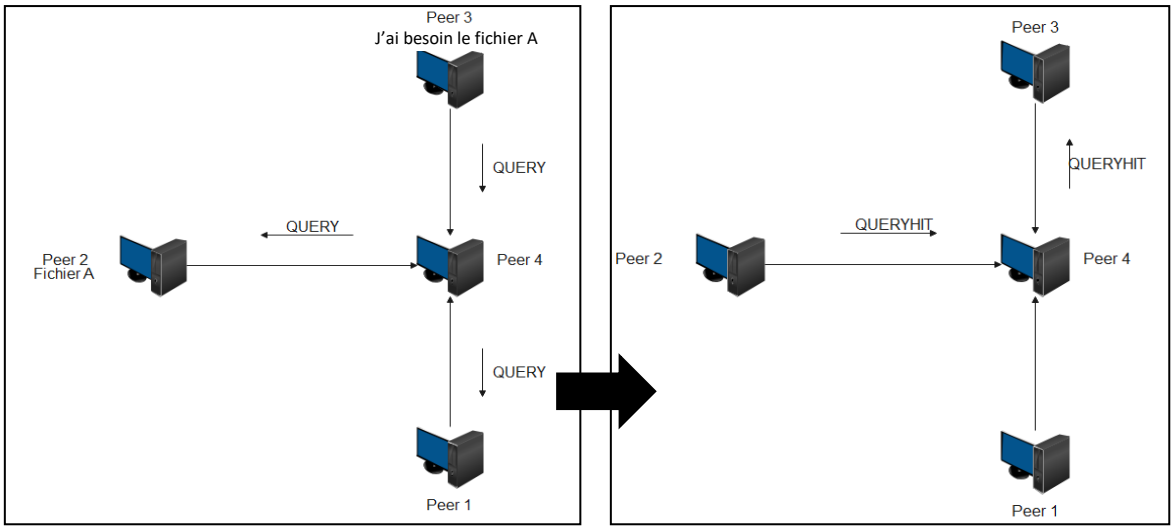


Figure II.8: Recherche des fichiers / Peer to Peer décentralisé

Lorsque le servent localise le fichier (d'après **QUERYHIT**), il envoie une **requête GET** vers le Peer qui contient le fichier. Par la suite, il va établir une connexion entre les deux Peers et commence le téléchargement.

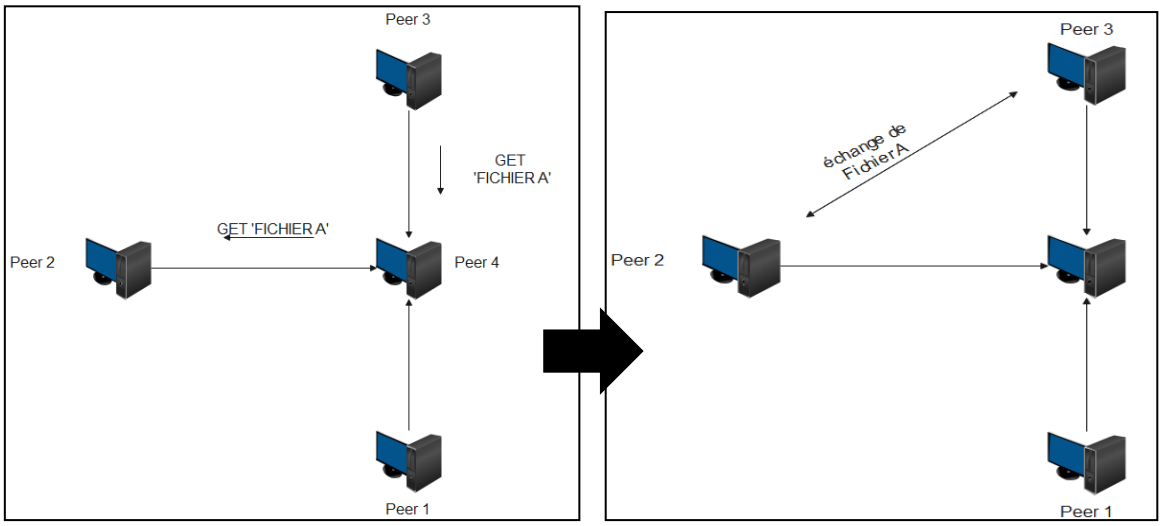


Figure II.9: échange des fichiers / Peer to Peer décentralisé

Le mécanisme de ce réseau permet de voir une grande tolérence aux pannes, très grande capacité, et facilité de jondre le réseau. Par contre il consomme énormément de bande passante et diminue le niveau de sécurité du réseau.

iii. Réseau Peer to Peer hybride

Ce système regroupe les techniques utilisées dans les deux architectures précedantes, sauf que le regroupement des Peers se fait en fonction du débit :

- **Super-Peer :** Un Peer avec un haut débit.
- **Peer :** Chaque terminal informatique dispose d'une connexion bas débit.

Chaque regroupement de Peers est géré par un Super-Peer et chaque Super-Peer indexe les fichiers des Peers bas débit qui lui sont rattachés (le même fonctionnement du serveur central – système centralisé). Les Supers-Peers connectés entre eux avec une liaison haut débit, fonctionnent toujours comme des systèmes décentralisés .

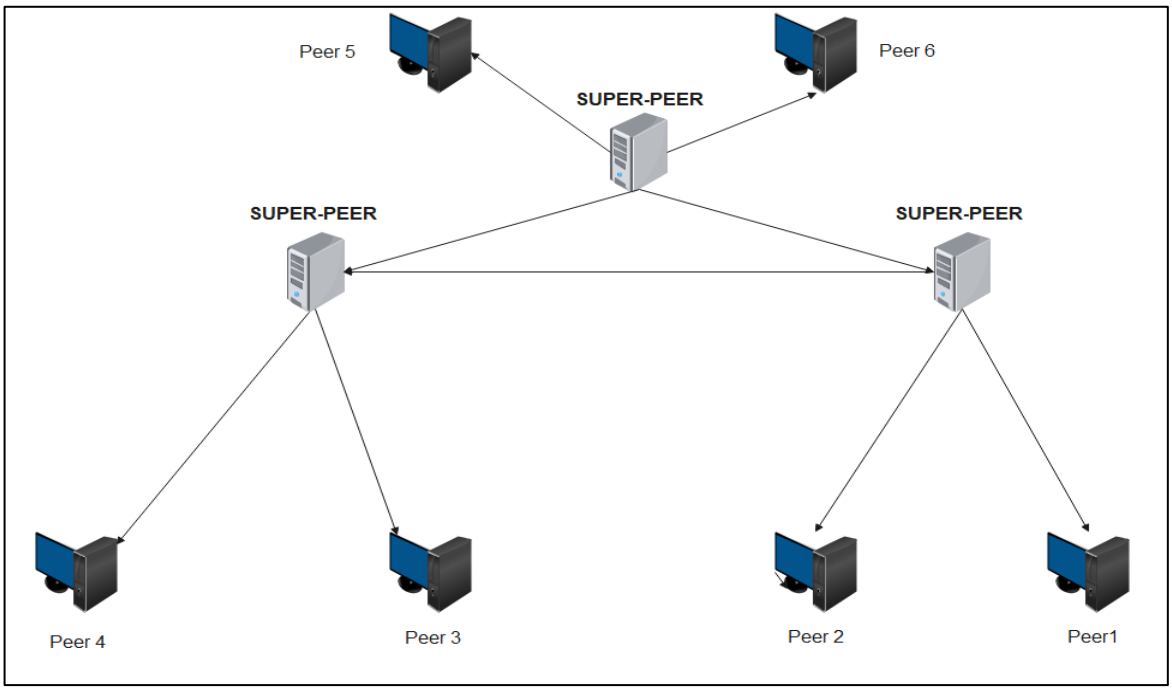


Figure II.10: Architecture hybride

❖ **Principe de fonctionnement :**

C'est un modèle qui fonctionne suivant deux méthodes en même temps, le but reste le même mais le principe change, Pour :

- **Joindre le réseau** : Il suffit d'installer un logiciel de partage (Kazaa par exemple).
- **Trouver un fichier** : Le client envoi une demande de fichier au Super-Peer le plus proche, qui va à son tour partager la demande sur tout le réseau.

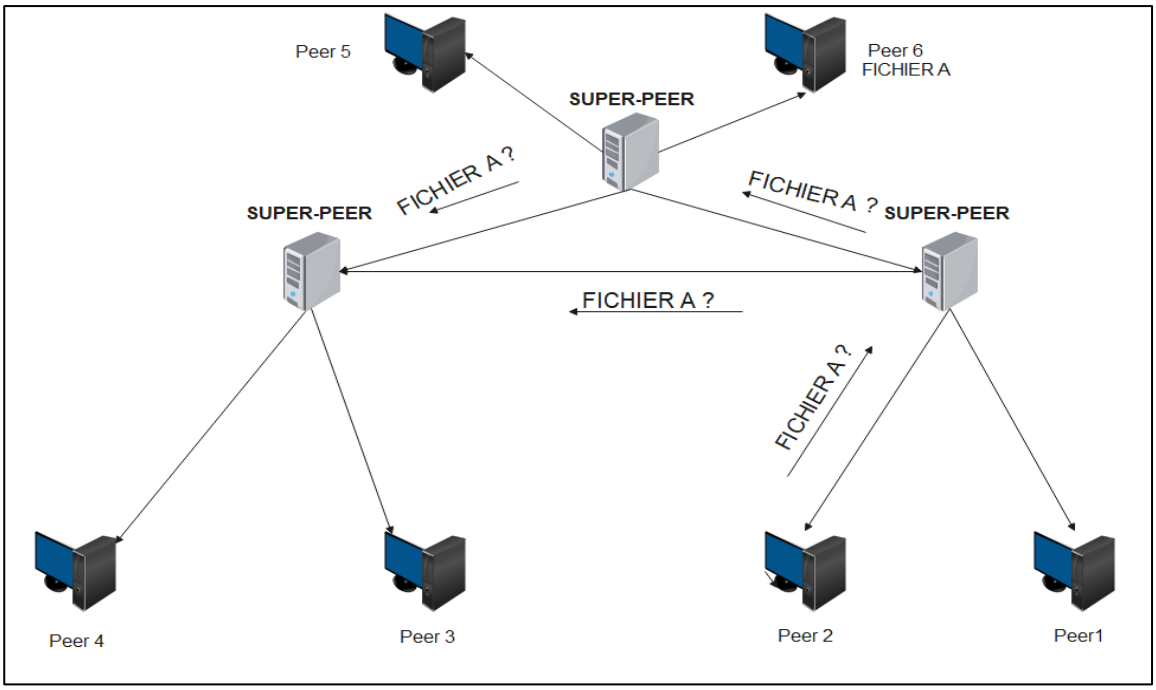


Figure II.11: Recherche d'un fichier / Peer to Peer Hybride

Une fois que le super-Peer recoit une réponse par les autres super-Peer, il informera le Peer cherchant la localisation du fichier, ensuite le Peer va établir une connexion directe avec l'hôte du fichier.

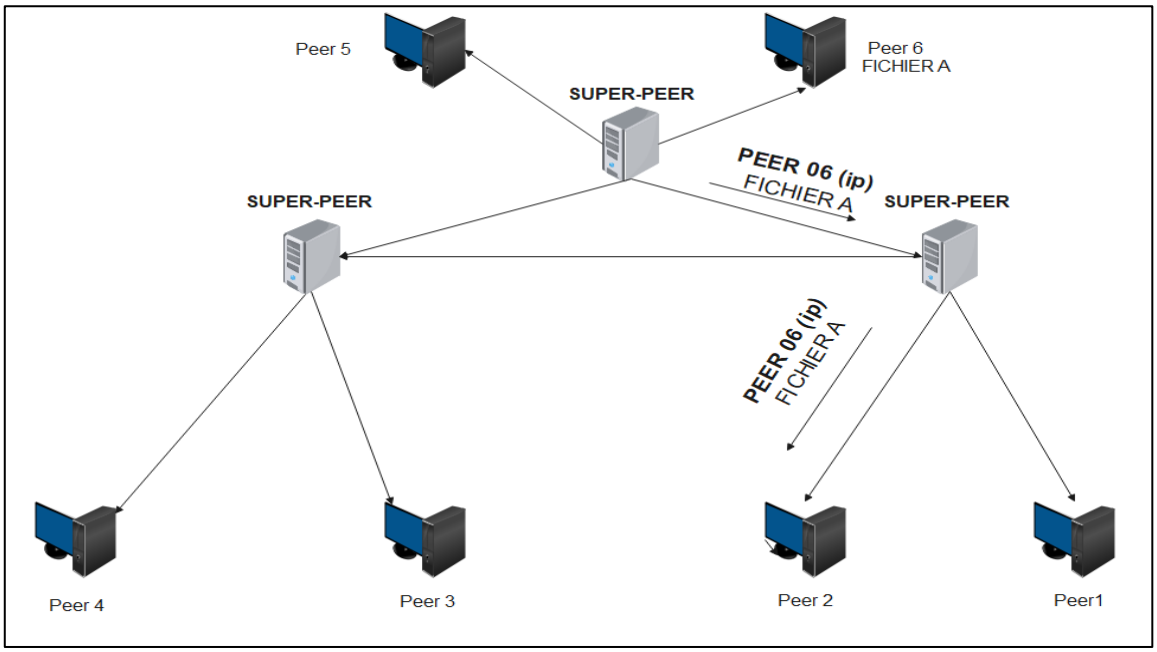


Figure II.12: Réponse de recherche d'un fichier / Peer to Peer hybride

Une fois la connexion établie, le transfert du fichier entre les deux Peers commence, puis la connexion est interrompue à la fin du transfert.

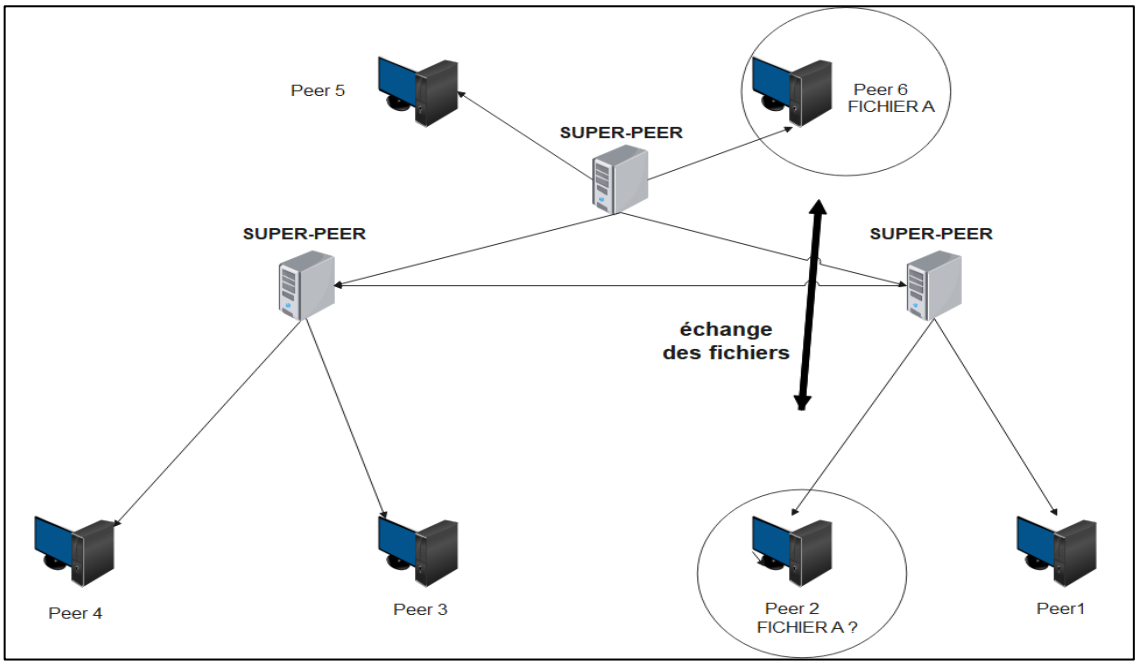


Figure II.13: Téléchargement d'un fichier / Peer to Peer hybride

Le réseau hybride a pour but de combiner les avantages des architectures centralisées et décentralisées pour avoir un réseau tolérant aux pannes avec une grande capacité et faible consommation de la bande passante.

iv. Téléchargement multiple

Le problème des connexions ADSL est leurs débits asymétriques. Le débit de sortie des clients (UPLOAD) est de 10 à 20 fois plus faible que leur débit de téléchargement (DOWNLOAD). Ce faible débit interrompe l'ensemble du réseau, car il est indépendamment de sa vitesse de connexion. Un ordinateur sera limité par le débit imposé par le Peer qui envoie des données. Pour cela, la notion du téléchargement multiple a été introduite pour compenser cette asymétrie.

C'est un système de téléchargement des fichiers qui permet d'optimiser le temps de téléchargement. il permet à un client de télécharger les fichiers en petits morceaux à partir de plusieurs Peers contrairement à l'ancien système. Le client ne peut télécharger le fichier qu'à partir du Peer source disposant du fichier complet [4].

Le schéma ci-dessous montre clairment le fonctionnement du réseau de téléchargement multiple :

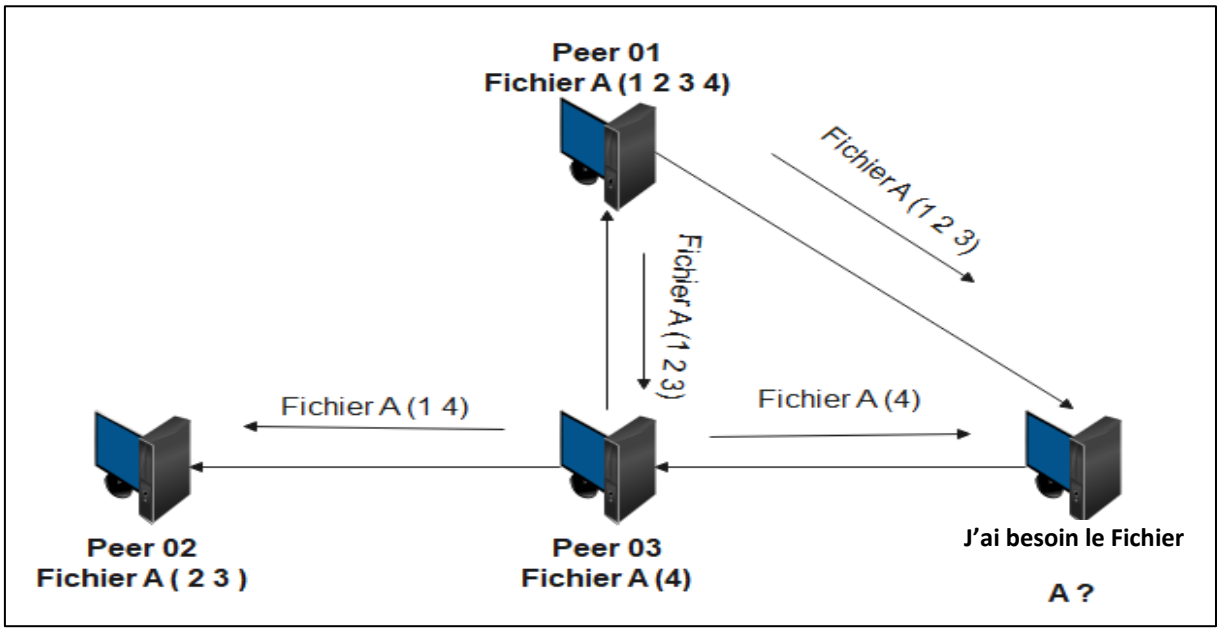


Figure II.14: Réseau de téléchargement multiple

Le Peer 01 possède la totalité du fichier (1 2 3 4) que les Peers 02, 03 et 04 veulent télécharger. Les Peers 02 et 03 contiennent déjà des morceaux du fichier A, ils peuvent télécharger le reste des morceaux à partir du Peer 01 et mutuellement partager les données relatives au Peer 04.

Avec ce mécanisme, on ne peut jamais avoir une saturation de bande passante au niveau du Peer 01(Théoriquement), à cause du partage des tâches entre les différents Peers.

Cela permet, d'éviter partiellement le problème de connexion ADSL avec les réseaux Peer to Peer, grâce à l'équilibrage des capacités UPLOAD et DOWNLOAD.

Cette technique garantit l'exploitation maximale de la vitesse du réseau, qui permet d'avoir un téléchargement rapide avec une grande diversité des ressources, d'un autre côté l'inconvénient majeur de cette technique est la bande passante, surtout dans les réseaux d'entreprises & universitaires. D'une part, toute la bande passante est utilisée par un seul logiciel (de téléchargement multiple), d'autre part la sécurité des réseaux devient faible.

II.2.2. Réseau Peer to Peer Structuré

Les réseaux Peer to Peer structurés sont basés sur l'établissement d'une DHT (Distributed Hash Table) permettant de "placer" les nouveaux Peers aux seins du réseau. Chaque Peer reçoit une liste des voisins avec lesquels il pourra communiquer. Il s'agit ici des voisins "logiques", ceux-ci pouvant se trouver à l'autre bout du monde.

De plus, chaque Peer est responsable d'une partie spécifique du contenu du réseau. Ils sont en général identifiés par un couple clé/valeur qui permet de réaliser des recherches avec un nombre de messages croissant de façon logarithmique.

i. Table de Hachage Distribuée

Le principe des tables de hachage distribuées consiste à utiliser une fonction de hachage « h », que l'on peut appliquer à la fois aux adresses IP des Peers et aux chaines ASCII identifiant une ressource disponible sur le réseau. Le résultat est un nombre aléatoire codé sur « m bits », « m » est choisi suffisamment grand pour que la probabilité d'une collision soit suffisamment petite. Typiquement m = 160, et on dispose donc de 2^{160} identificateurs [6].

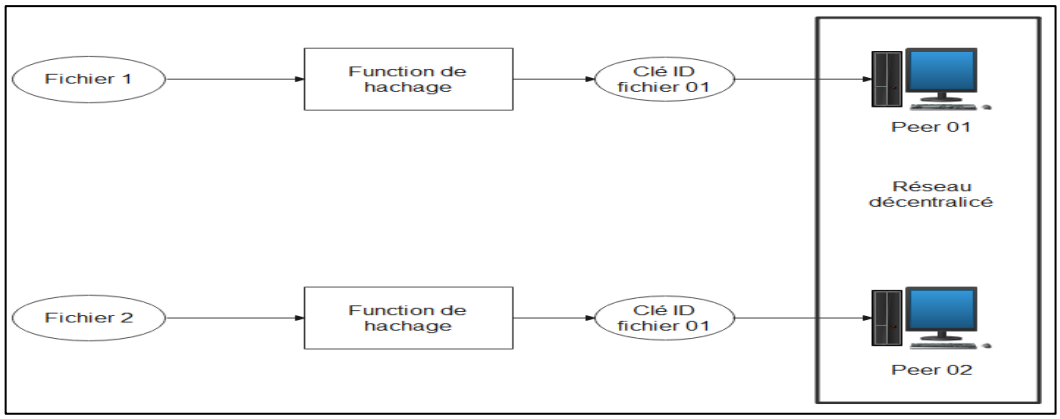


Figure II.15: Principe de hachage distribué

Parmi les algorithmes structurés [6] :

- CAN (Topologie en D-dimension)
- CHORD (Topologie anneau)
- Pastry (Topologie en Anneau)
- KADEMLIA (Topologie en Arborescence)

ii. Réseau KADEMLIA

Le réseau KAD est un réseau Peer to Peer basé sur le DHT qui implémente le protocole KADEMLIA. L'accès au réseau KAD est fourni via le client E-Mule, qui peut également se connecter au réseau E-Donkey2.

Il spécifie la structure du réseau et l'échange d'informations à travers la consultation des nœuds. Les nœuds KADEMLIA communiquent entre eux en utilisant UDP. Un réseau virtuel ou superposition est formé par les nœuds participants où chaque nœud est identifié par un numéro « l'ID du nœud ». Ce dernier sert non seulement d'identificateur, mais l'algorithme KADEMLIA l'utilise pour localiser des valeurs (généralement des hachages ou de mots-clés). En fait, le nœud stocke les informations sur l'emplacement actuel du fichier ou de la ressource [9].

Les techniques utilisées dans le réseau structuré améliorent les performances des réseaux Peer to Peer car ils diminuent le temps de recherche et ils permettent de localiser les fichiers très rapidement, ainsi la démonisation des charges sur les Peers.

II.3. Synthèse des architectures Peer to Peer

Réseau Peer to Peer	Structuré	Non-structuré
Autonomie	Faible	Forte
Efficacité	Forte	Faible
Qualité de réponse	Forte	Faible
Tolérance aux pannes	Forte	Forte
Capacité	Forte	Forte

Tableau II-1: Synthèse des architectures Peer to Peer

Le bilan montre clairement qu'il n'y a pas encore de système parfait absolu, c'est pour cette raison que la recherche dans ce domaine est encore très active.

Chaque architecture a sa propre technique avec des différentes avantages, donc pour regrouper le maximum d'avantages il faut chercher à regrouper les techniques des deux architectures, pour cela les deux protocoles BitTorrent et E-Donkey font une

révolution dans le domaine de téléchargement Peer to Peer car il combine le réseau non-structuré décentralisé et les réseaux structurés. Cette combinaison entre ces deux derniers mets ces deux protocoles en tête des protocoles Peer to Peer les plus performante.

II.4. Réseau de partage E-Donkey

Le réseau E-Donkey est un réseau de partage de fichiers Peer to Peer hybride décentralisé. le serveur ne partage aucun fichier, il gère uniquement la distribution des informations et travaille comme plusieurs dictionnaires centreaux qui contiennent les informations sur les fichiers partagés et l'emplacement des clients respectifs.

Le protocole E-Donkey utilise des hachages MD4 de 16 octets seulement pour identifier un fichier indépendamment de son nom de fichier. Il permet de télécharger tous types de fichiers (Video,musique…) [9].

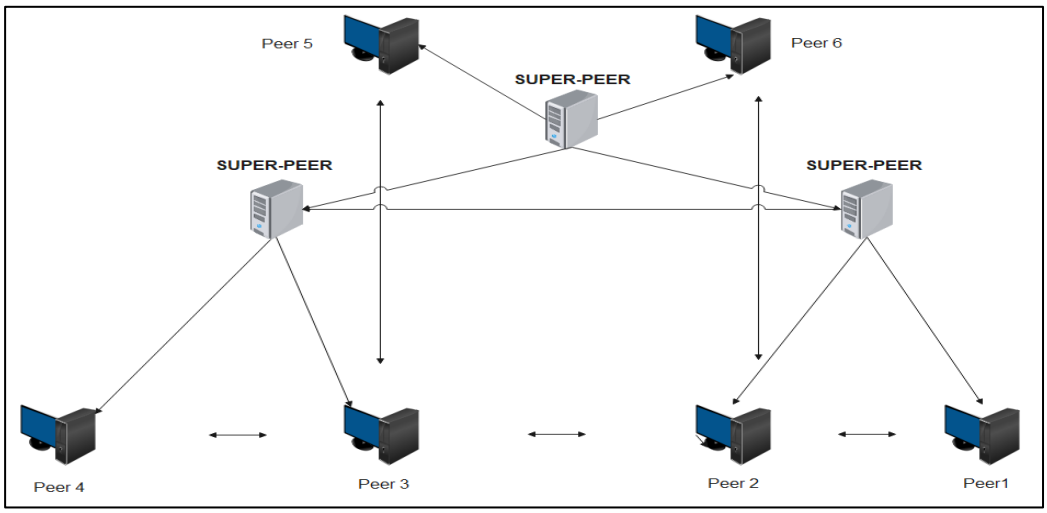


Figure II.16 : Réseau E-Donkey

II.4.1. Composant du réseau E-Donkey

❖ **Serveur :**

Les serveurs sont les points d'entrée du réseau, Ils permettent d'indexer l'ensemble des fichiers des clients connectés et permettent aux Peers de se faire connaître sur le réseau.

❖ **Client :**

C'est l'élément le plus important dans le réseau, car il assure le fonctionnement du Peer to Peer. Pour devenir un client edonkey il faut utiliser l'un des programmes clients (E-Mule, A-Mule, Edonkey2000…) ce dernier assure les tâches suivantes :

- Recherche les fichiers (type, nom, date…)
- Envoie les fichiers
- Connecte au serveur.
- Télécharge les fichiers.
- Gère l'utilisation de la bande passante.

II.4.2. Principe de fonctionnement

❖ **Connexion au serveur :**

Une fois l'installation du logiciel client terminée, le Peer cherche a établir une connexion avec le serveur. Ce dernier distribuer un des deux ID pour identifier le client correspondant à l'adresse IP du client avant la connexion établie :

- **HighID** : Recommendé car il permet une connexion facile.
- **LOWID** : Cet ID est assigné au client quand le serveur n'a pas réussi à initier une connexion avec lui.

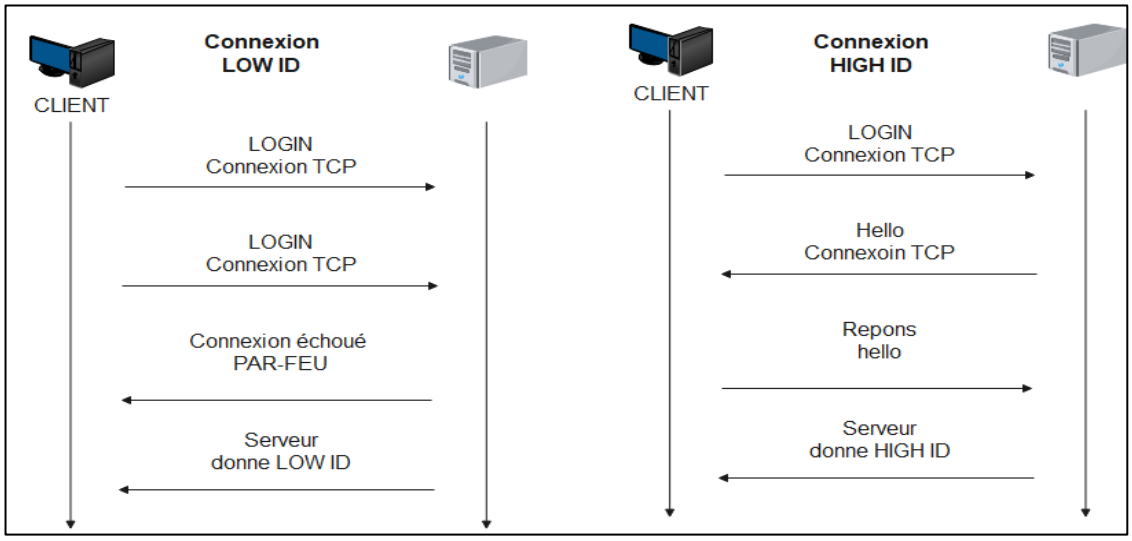


Figure II.17: Connexion Low/high ID

Lorsqu'un LOWID est obtenue même un inconvénient majeur fais surface, on ne pourra jamais communiquer avec les autres serveurs (les serveurs n'échangent pas entre eux les LowID) [10].

❖ **Messages principaux entre le client et leur serveur :**

Lorsque le client établit une connexion TCP avec un serveur, il commence par l'envoie et la réception des messages de mise à jour d'information. Le client commence à envoyer une liste de ses fichiers partagés et demande ensuite une liste des serveurs. Le serveur répond avec un message qui contient son statut, sa version et une liste des serveurs.

Par la suite, le client demande les sources des différents fichiers qu'il est en train de télécharger. Le serveur y répond par une suite de message contient les sources de chaque fichier.

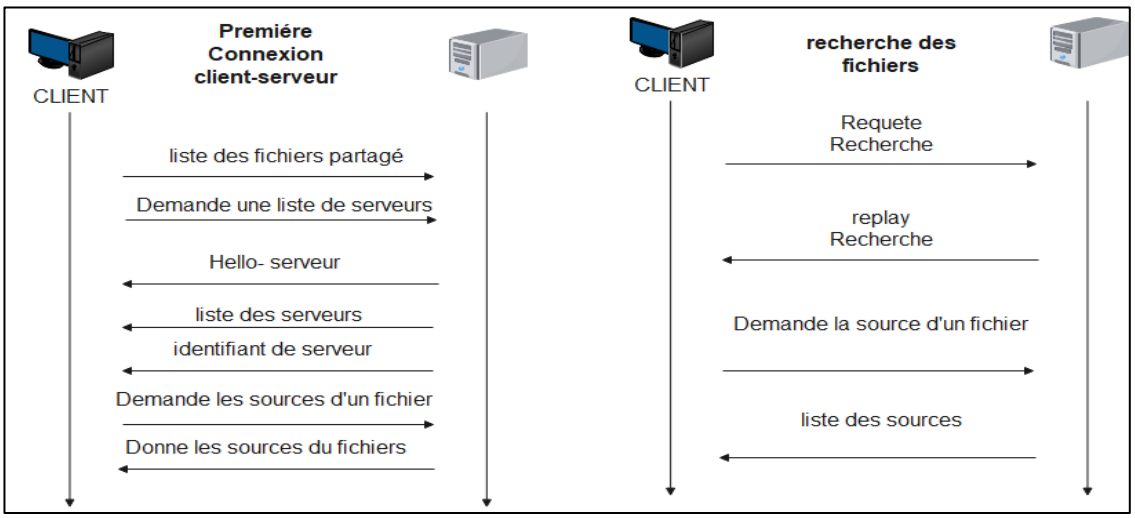


Figure II.18: Première connexion entre le client-serveur

❖ **File ID :**

Le File ID est utilisé pour identifier d'une façon unique les fichiers sur le réseau. Chaque fichier est identifié par un hash sur 128 bits calculé par le client et basé sur le contenu du fichier (cet hash est calculé par l'algorithme MD4). Il est divisé en parties de 9,28Mo dont on calcule pour chacune d'elle la somme MD4. Le hashset est l'ensemble des hash combinés, pour trouve le File hash on applique le MD4 sur le hashset.

❖ **Recherche de fichier :**

Le client E-Donkey peut effectuer une recherche du fichier par son nom. Ce dernier est traduit par le serveur en liste de mots clés, va ensuite comparer les mots clés aux noms des fichiers disponibles et répond en envoyant la liste des fichiers correspondants. On filtre la recherche avec plusieurs autres paramètres comme le type du fichier, la taille minimale,

la taille maximale et l'extension du fichier. Les recherches peuvent être effectuées sur le serveur auquel le client est connecté (TCP), mais il est également possible d'interroger les autres serveurs d'une manière non connectée (UDP).

- **Liste d'attente de téléchargment :**

Lorsqu'un client souhaite télécharger une partie du fichier situé au niveau d'un autre client, il est placé dans une file d'attente de téléchargement de ce dernier. Si la file est vide le Peer commence immédiatement le téléchargement. Sinon, des clients sont déjà en attente, et donc placé dans la queue. La file d'attente est de taille finie et il est possible que le client soit rejeté parce qu'elle est pleine.

Des messages sont émis en UDP périodiquement entre les Peers pour connaître leur position dans la file d'attente. Il y a 3 types de réponses à ce message :

- Le client est dans la file.
- La file est pleine.
- Le fichier n'existe pas.

Le client supporte un système de crédit dans le but d'encourager les utilisateurs à partager leurs fichiers. Plus le client envoi des données à d'autres clients, il aura une augmentation du crédit chez ces derniers, et il avance dans la file d'attente.

Malgré la grande popularité de ce protocole durant les années 2000-2009 il présente plusieurs failles et problèmes. L'un de ces derniers est au niveau du client car beaucoup de personnes utilisent un routeur pour se connecter à internet, ils ne disposent alors que d'une seule adresse publique, celle du routeur. Il faut donc rediriger les ports TCP et UDP de eMule vers le bon client (PAT). Dans le cas où cette redirection n'est pas effective, le client pourra uniquement se connecter en LOWID et ainsi il ne va pas être privilégié pour le partage.

Pour cela les gens cherchent de basculent vers l'utilisation d'un nouveau réseau simple et rapide, c'est le réseau BitTorrent.

II.5. Réseau de partage BitTorrent

Réseau décentralisé qui reprend le principe du téléchargement multiple qui repose complétement sur le protocole BitTorrent. Le nouveau mécanisme de ce protocole est basé sur la coopération entre les Peers qui permet de distribuer des fichiers de tailles importantes par des petits serveurs.

Chaque Peer est responsable de maximiser son propre taux de téléchargement en contactant des Peers appropriés. Des Peers avec des taux de téléchargement élevés seront également en mesure de télécharger des données avec des vitesses élevées, ce qui pose le problème majeur du téléchargement torrent ‘’l’occupation de la bande passante’’. Les Peers sont toujours en mode de fonctionnement (UPLOAD) même si le téléchargement est clos. En effet, les Peers ne font pas partie d'un réseau global, mais ils sont regroupés par fichier, il existe un réseau autour de chaque fichier «.torrent ». [9]

II.5.1. Les composants du réseau torrent

- **Peer :** Une machine qui utilise un logiciel client BitTorrent, auquel d'autres clients se connectent et échangent des données (Torrent).
- **Seed :** Une machine qui contient un fichier complet qui va être partager au Peers.
- **Leech :** C'est un utilisateur qui télécharge sans jamais proposer ses fichiers aux autres utilisateurs.
- **Swarm :** Un groupe de Seeds et de Peers partageant le même torrent.
- **Tracker :** Un serveur qui garde la trace des Peers et Seeds dans un Swarm. Il n'a pas de copie du fichier lui-même, mais il permet de gérer le processus de transfert de fichiers. N’importe qui, est susceptible de créer son propre tracker.
- **Scrape :** C'est quand un client envoie une demande au tracker pour obtenir des informations sur les statistiques de fichier torrent, comme avec qui il partage le fichier.
- **Torrent :** Généralement, l'instance d'un fichier ou d'un groupe de fichiers distribués via BitTorrent.

- **Fichier torrent (MetaData) :** Un fichier qui décrit le ou les fichiers en cours de distribution, où trouver les pièces et d'autres informations nécessaires à la distribution du fichier.

- **Bloc (Fragment) :** Un bloc est un morceau de fichier. Lorsqu'un fichier est distribué via BitTorrent, il est divisé en plus petits morceaux, ou blocs. En général, la taille du bloc est de 256 Ko, plus les fragments sont petits, plus le téléchargement sera rapide.

❖ **Création d'un fichier torrent :**

La création de (fichier) torrent se fait de manière simple à l'aide d'outils tel que ''torrenteditor, Maketorrent…'', il suffit de choisir :

- l'emplacement & Nom de fichier.
- Le Tracker (lien/IP).
- La Taille du fragment en général 256 Ko.

Enfin, pour amorcer le partage, il suffit d'utiliser un client BitTorrent comme si on souhaitait télécharger le fichier. Le client vérifiera que le fichier que l'on souhaite télécharger est le même que celui présent sur notre disque. Il se connectera alors au tracker et commencera le partage du fichier. On sera alors considéré comme le Seed original et les autres clients pourront se connecter à nous pour télécharger le fichier.

II.5.2. Principe de fonctionnement

Le client qui veut télécharger un fichier à l'aide de ce protocole il doit installer au moins un client du protocole BitTorrent (BitTorrent, uTorrent, Vuze…), par la suite il surfe sur le web pour trouver le lien du fichier MetaData (.torrent). Ce fichier contient toutes les informations nécessaires pour établir une connexion entre le client et le tracker qui va gérer le téléchargement par la suite. Après l'établissement de la connexion, le tracker commence à communiquer avec le client avec les requêtes ''GET HTTP'' . Ces requêtes contiennent les informations nécessaires pour commencer le téléchargement et l'échange des fichiers.

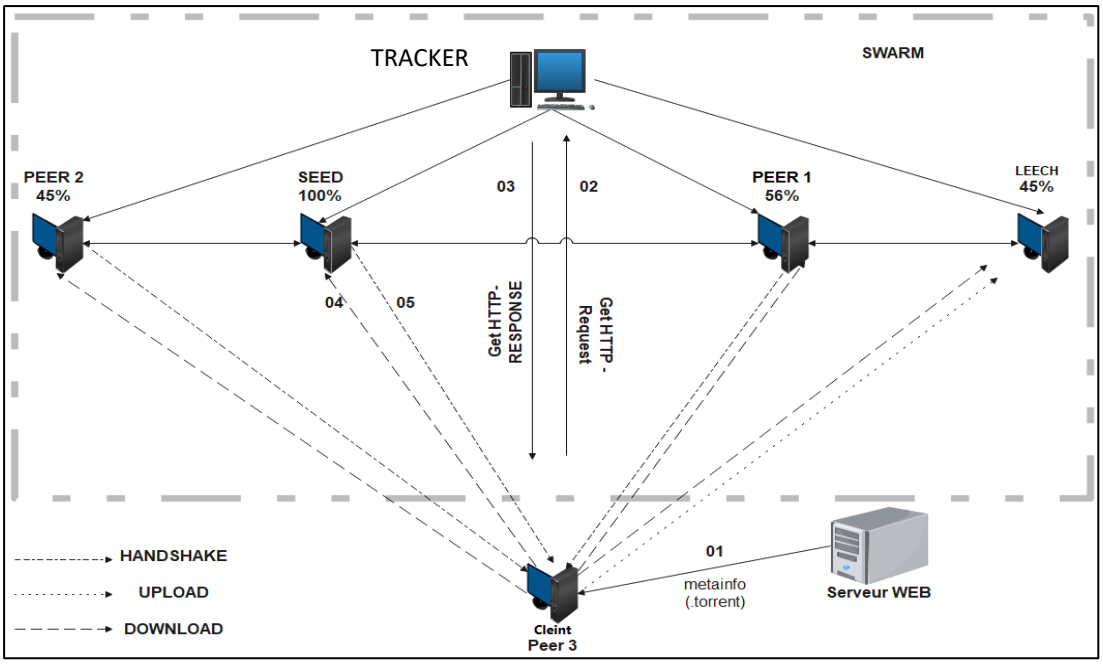


Figure II.19: Principe de fonctionnement – Protocole BitTorrent

❖ **Structure d'un fichier MetaData (.torrent) :**

Un fichier torrent contient des informations concernant le fichier à télécharger et le tracker, comme montre le tableau ci-dessous :

Anoncer	URL DE TRACKER
Info	informations sur les fichiers
Name	Nom du fichier torrent
files	description des fichiers
Lenght	taille d'un fichier (bayte)
Path	chemin d'un fichier
Pièce lenght	La taille de chaque partie du fichier, issue de la fragmentation.
Pièces	Chaîne constituée de la concaténation de tous Valeurs de hachage SHA1 de 20 octets, un par pièce (encodé en binaire brut)

Tableau II-2: Clés principales d'un fichier MetaData (.torrent)

❖ **Trouver le Tracker :**

Une fois que le client obtient le fichier Metadata, il peut par la suite se connecter au tracker par des requet **http GET** vers son URL/IP, afin d'avoir des informations sur la localisation du fichier.

❖ **Connexion au tracker :**

Tout d'abord, Le client envoie une requête **« HTTP GET anounce »** vers le tracker, cette requête contient :

Info_hash	Hachage SHA1 de 20 octets de la valeur de la clé info du fichier MetaData
Peer_id	Chaîne de 20 bits utilisée comme identifiant unique pour le client, générée par le client au démarrage
Port	Numéro de port sur lequel le client écoute. Les ports réservés pour BitTorrent sont généralement 6881-6999
IP	Adresse IP du client

Tableau II-3: Clés principales – requête http tracker

Le tracker répond avec une liste des Peers qui permet au client de participer au réseau torrent. Cette requête contient différents informations (voire table II-4) :

Intervalle	Intervalle à attendre avant de contacter le tracker
Peers	Liste des Peers, chaque peer avec une clé.
Ip	Adresse IP de client (Peer)
Peer id	Identifiant de client (Peer)
Port	Port de client (Peer)

Tableau II-4: Clés principales – réponse http tracker

- **Développement du Protocole BitTorrent**

L'architecture classique du réseau BitTorrent basée sur un fichier MetaData gérée par un seul tracker central, afin de faciliter les taches du Peer une extension importante vise à supprimer ce point faible du protocole a était mise en place, le client implémente alors une table DHT pour identifier et stocker la liste des différents clients partageant le fichier qui l'intéresse, cette extension permet aux clients BitTorrent de faire des connexions sans tracker.

On pourra aussi noter la présence d'une extension permettant le WebSeeding autorisant un serveur HTTP à servir de "Seed" pour les téléchargements et ainsi combiner le protocole BitTorrent avec le protocole HTTP.

Côté sécurité, l'extension des connexions cryptées permet la création de liaisons cryptées entre les Peers, protégeant ainsi le contenu de nos messages des regards extérieurs. Le cryptage RC4 obscurcit **non seulement l'en-tête mais le flux entier.**

II.6. Les logiciels Peer to Peer les plus utilisés

II.6.1. A-Mule

Logiciel le plus réputé des clients du réseau E-Donkey aujourd'hui. Les codes sources de ce logiciel sont publics (disponible sur **sourceforge.net**). Le fait qu'il soit open source fait apparaître un certain nombre de versions enrichies ou « mods », qui sont censées apporter de meilleures performances et d'autres options. A-Mule propose en plus une extension du protocole E-Donkey conçu pour la sécurité et l'utilisation du protocole UDP [12].

L'application a cependant un atout supplémentaire, elle est proposée sur de nombreux systèmes d'exploitation.

II.6.2. µTorrent

Est un client Peer to Peer très léger compatible avec le protocole BitTorrent, un protocole de téléchargement des plus populaires, conçu pour la distribution de fichiers à très haut débit. µTorrent est très utilisé car il est très léger et extrêmement facile à utiliser.

La version actuelle de µTorrent pour Windows est la ''3.5.5'', mise en ligne le 30/06/2020. Cette nouvelle "build" de la version 5.5 corrige différents soucis de fonctionnement et de sécurité, change l'onglet Vitesse en graphique dans le panneau de détail et ajoute la recherche intelligente de torrents [13].

II.6.3. Synthèse des logicielles Peer to Peer

Client	µtorrent	A-Mule
Protocole	BitTorrent	E-Donkey
Extension	DHT	KADEMLIA
Cryptage	Oui	Non
Fil d'attente	Non	Oui
Navigation	Non	Oui
Consommation	Élevé	Moyenne
Popularité	Grande	Petite
Sécurité	Forte	Faible
Mise en place	Facile	Difficile

Tableau II-5: Synthèse des logicielles Peer to Peer

II.7. Les risques de l'utilisation du réseau Peer to Peer

Le système utilisé par les torrents favorise la prolifération des différents logiciels malveillants comme les logiciels espions ou les **ransomwares**. Si un utilisateur dont l'ordinateur est infecté par un virus, télécharge le fichier saint et devient un serveur, les ordinateurs qui le téléchargeront après lui seront infectés également. L'utilisateur prend donc le risque d'être la cible du virus ou des logiciels malveillants. Ainsi que le risque que l'appareil soit infecté à cause des torrents potentiellement dangereux et les données qui peuvent être volées, tel que les données bancaires, mots de passe, ou tout autre donnée.

Le réseau torrent est considéré comme un réseau ouvert, n'importe qui peut joindre le réseau sans aucun contrôle. Chaque client risque d'être victime d'attaques réseaux comme l'attaque DOS/DDOS qui cible dernièrement les réseaux Peer to Peer en premier lieu grâce à leur facilité d'atteindre les machines cibles et les utilisées par la suite comme des botnets. Sans oublier les sanctions pénales qui peuvent aller de la sommation à un séjour en prison avec une amende élevée. En France par exemple, l'article L335-2 du code de la propriété intellectuelle dispose d'une peine maximale de 300 000 euros d'amende et de 3 ans d'emprisonnement. Il est donc impératif d'être conscient avant l'utilisation des réseaux Peer to Peer [11].

II.8. Conclusion

Dans ce chapitre, on a vu les différentes architectures du réseau Peer to Peer, ainsi que les différentes techniques et algorithmes pour avoir un téléchargement rapide et fiable. D'une manière générale, les Peer to Peer actuels souffrent de nombreux problèmes tels que les fichiers truqués, virus et autres. Cependant, la plupart des applications Peer to Peer actuellement existantes supportent des utilisations illégales : des crimes, propagande, influence sur les réseaux universitaires, entreprises … L'utilisation du Peer to Peer comporte donc de nombreux risques, mais parallèlement à ces attaques, les défenses et protections se développent.

C'est dans l'optique de cette problématique, et à l'aide de cette étude détaillée du réseau Peer to Peer plus précisément les caractéristiques des protocoles BitTorrent et E-Donkey et leurs principes de fonctionnement, que nous abordons de proposer une méthode fiable de détection de l'utilisation de ces deux protocoles.

Chapitre III

Inspection des paquets P2P

III.1. Introduction

D'après ce qu'on a vu précédemment, les clients du réseau Peer to Peer tel que µTorrent et AMule sont très utilisés pour le partage et le téléchargement des fichiers. Cependant, l'utilisation de ces deux clients dans un réseau de campus universitaire ou d'entreprise peut exposer ce dernier à divers risques. Pour cela, les entreprises commencent à prêter attention au risque de l'utilisation du Peer to Peer dans leurs réseaux.

La détection d'utilisation de ces deux clients nécessite une inspection profonde de paquets, afin d'extraire les signatures qui permettent de faire la différence entre un trafic Peer to Peer et un trafic normal.

A cet effet une analyse détaillée du trafic sera élaborée, afin de mettre en évidence la différence qui permet de distinguer le trafic des application Peer to Peer.

III.2. Plan de travail

Afin de mieux comprendre le fonctionnement du ces protocoles, on va faire une implémentation réelle par l'installation des clients de ces deux derniers au niveau de deux PC différents dans un réseau local.

- Le travail se déroulera suivant deux volets :

- ✓ **1ère étape : Inspection profonde des paquets (Analyse).**

Après l'installation des clients, l'étape qui suit est le téléchargement d'un fichier quelconque au niveau des machines qui contiennent ces clients. On va utiliser par la suite la surveillance du réseau pour collecter les traces du trafic circulant entre le réseau d'entreprise et le reste de l'internet. Le réseau se connecte à ses FAI via un routeur frontalier utilisé pour le trafic sortant et entrant.

L'installation du logiciel ''WireShark'' va nous permettre au niveau du serveur proxy, d'exploiter les paquets entrants /sortants au cours du téléchargement afin d'extraire les signatures pour chaque protocole.

✓ **2-ème étape : Implémentation des empreintes numériques.**

Pour finir, on va implémenter ces empreintes numériques sous forme de règles dans un système d'intrusion (SNORT) qui sera installé au niveau du serveur proxy.

III.3. Architecture de travail

La figure ci-dessous (Figure III.1) montre bien l'architecture de notre travail :

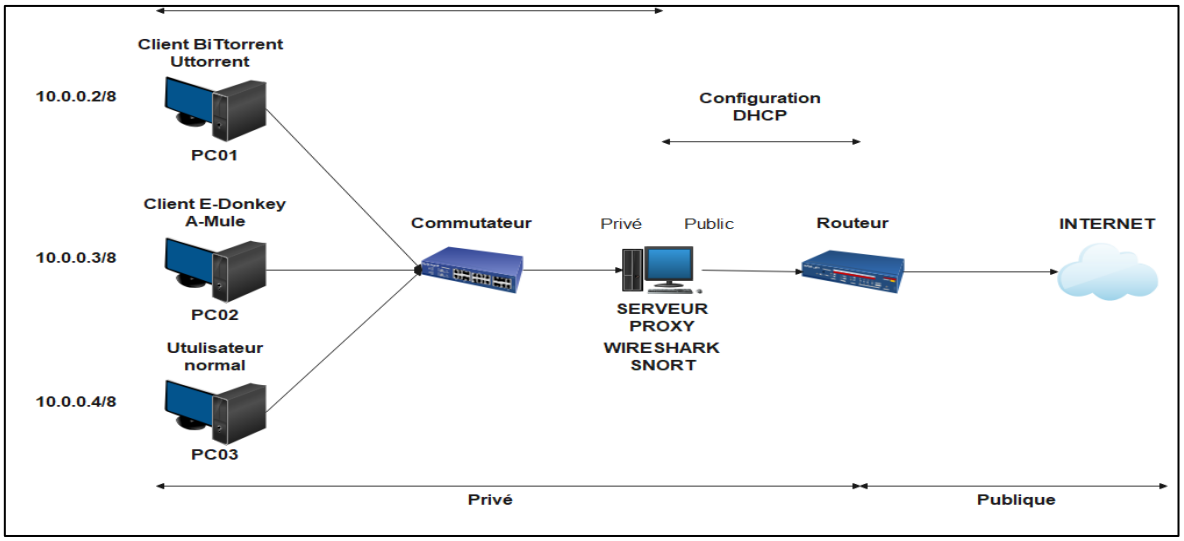


Figure III.1: Architecture de Travail - Laboratoire

- Le serveur proxy à deux interfaces réseau :
  - Interface publique : Configuration automatique
  - Interface privée : Adresse IP 10.0.0.1 dans la plage 10.0.0.0/8.
  - Logiciel installé : Wireshark / Snort.
  - Système d'exploitation : Windows 10
- Les clients :
  - Client 01 : Adresse IP 10.0.0.2 | Passerelle 10.0.0.1
    - Logiciel installé : µTorrent.
    - Système d'exploitation : Windows 10
  - Client 02 : Adresse IP 10.0.0.3 | Passerelle 10.0.0.1
    - Logiciel installé : AMule.
    - Système d'exploitation : Windows 10
  - Client 03 : Adresse IP 10.0.0.4 | Passerelle 10.0.0.1
    - Utilisateur normal.
    - Système d'exploitation : Windows 10

III.4. Outil d'analyse & capture « Wireshark »

Wireshark est un analyseur de trafic réseau, avec une interface graphique qui nous permettra de capturer les trames qui entrent et sortent de notre serveur, puis de les disséquer et d'étudier leur contenu et les afficher d'une manière détaillée.

Wireshark utilise la même bibliothèque de capture de paquets (libpcap) que d'autres renifleurs populaires tels que TCPDUMP, bien qu'il soit capable de lire de nombreux autres types de format de capture. C'est également un logiciel de distribution gratuit qui peut fonctionner sur différentes plateformes (Windows, Linux et Mac ...). Mais, probablement, le plus remarquable est son interface graphique et la puissance de capacité du filtrage des paquets affichés [14].

Parmi ces fonctionnalités les plus intéressantes :

- Capture les données des paquets en direct (temps réel) depuis une interface réseau.
- Affichage et enregistrement des paquets avec des informations de protocole très détaillées.
- Recherche et filtrage des paquets sur plusieurs critères.
- Affichage colorisé des paquets en fonction des filtres.
- Affichage du contenu du paquet pour toutes les couches du réseau.
- Exportation de certains ou tous les paquets dans un certain nombre de formats de fichiers de capture....

➢ L'interface graphique de l'utilisateur du Wireshark se présente comme suit après la capture des paquets :

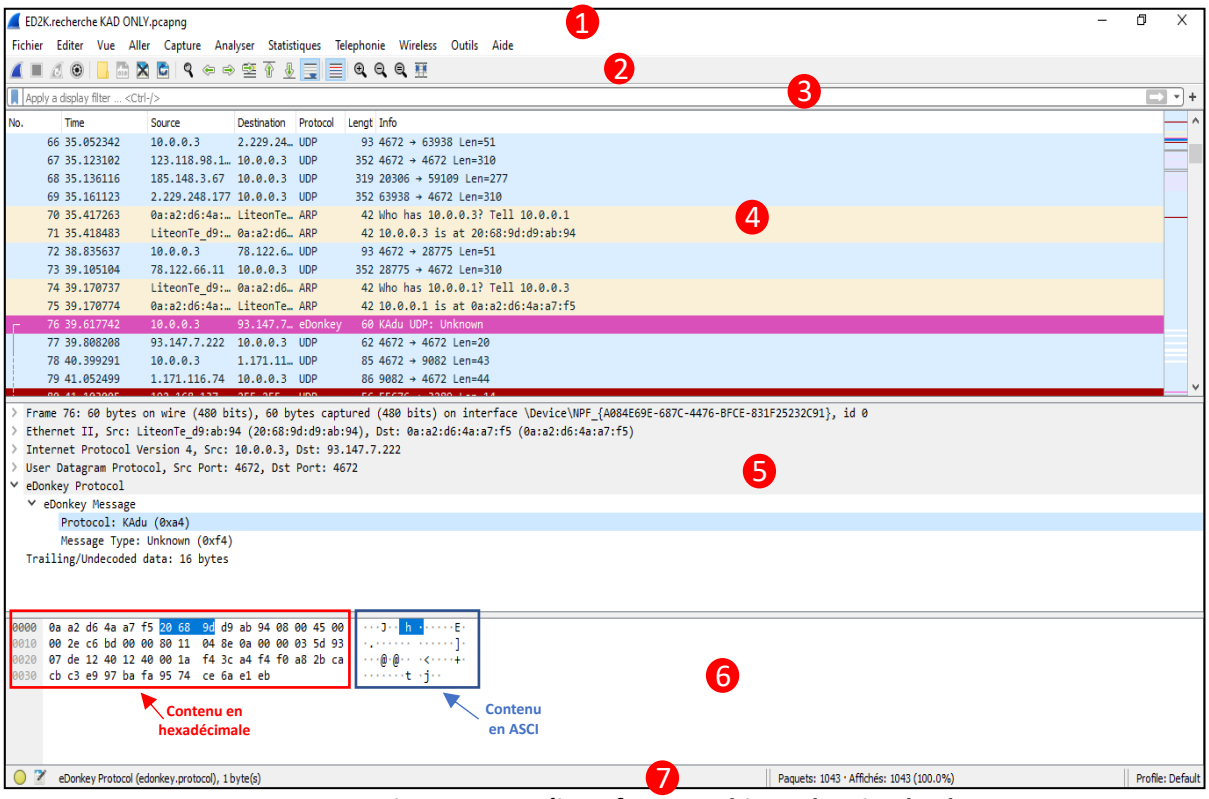


Figure III.2 : L'interface graphique de wireshark

01- Le menu : Est utilisé pour démarrer les actions.

02- La barre d'outils principale : Fournit un accès rapide aux éléments fréquemment utilisés à partir du menu.

03- La barre d'outils de filtrage : Permet aux utilisateurs de définir des filtres d'affichage pour filtrer les paquets affichés.

04- Le volet de la liste des paquets : Affiche un résumé de chaque paquet capturé. En cliquant sur les paquets dans ce volet, vous contrôlez ce qui est affiché dans les deux autres volets.

05- Le volet des détails des paquets : Affiche plus en détail de paquet sélectionné dans le volet de la liste des paquets.

06- Le volet d'octets de paquet : Affiche les données du paquet sélectionné dans le volet de la liste de paquets et met en surbrillance le champ sélectionné dans le volet de détails des paquets.

07- La barre d'état : Affiche des informations sur l'état actuel du programme et les données capturées.

III.5. La structure d'analyse

L'inspection approfondie des paquets va nous permettre de déceler les deux protocoles à savoir BitTorrent et E-Donkey à travers l'étude de deux logiciels A-Mule et µTorrent.

III.5.1. A-Mule

La structure d'analyse du trafic pour le client AMule est divisée en trois parties :

- Établissement de connexion.
- Accomplissement d'une recherche d'un fichier.
- Téléchargement d'un fichier.

a. Etablissement de connexion :

Pour établir une connexion vers le serveur E-Donkey avec le client A-Mule, il faut tout d'abord cliquer sur l'icône **Réseaux** pour afficher la liste des serveurs disponibles. Le client A-Mule affiche la liste des serveurs avec leurs caractéristiques sur son interface graphique où on peut trouver : le nom du serveur, son adresse IP et Port, le temps de réponse, nombre d'utilisateurs, nombre de fichiers etc. ... Comme afficher ci-dessous.

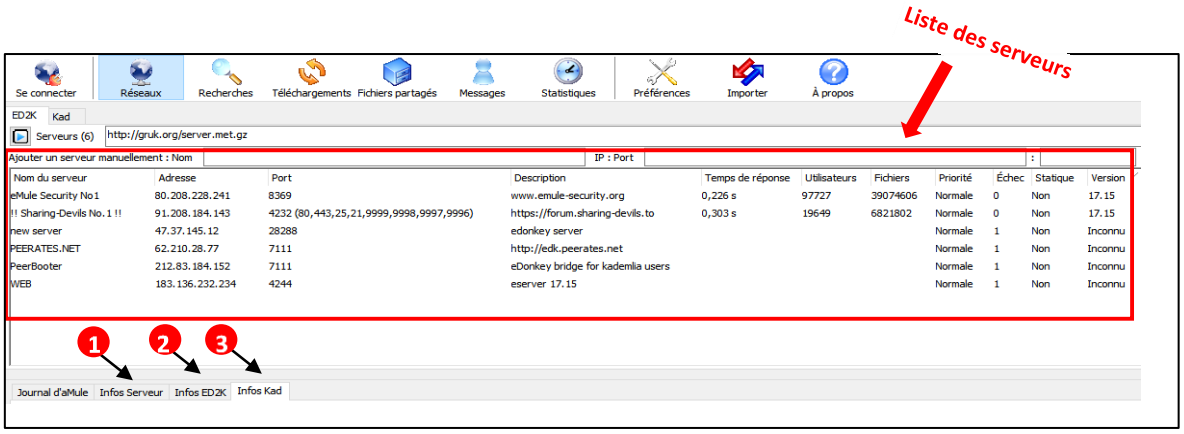


Figure III.3 Etablissement de connexion vers le serveur E-Donkey

Toutes les informations qui concernent le client et le serveur sont affichées dans la zone info comme suit :

- **Zone 1 :** C'est la zone qui décrit la description détaillée du serveur.
- **Zone 2 :** Elle contient toutes les informations qui concerne la connexion Ed2k.
- **Zone 3 :** Elle contient toutes les informations qui concerne la connexion KAD.

- Une fois la connexion établie, A-Mule rapporte un **ID** et indique le type d'ID que nous avons dans le globe qui apparait dans le coin inferieure droit, dans cette capture on voit plus clairement qu'on a un **LowID**, ce qui signifie que le port TCP est bloqué ou n'est pas accessible (La cause de cela peut être trouvée en présence du pare-feu). On peut aussi noter la présence des informations du réseaux KADEMLIA comme le statut du réseau, les ports, l'adresse IP publique du client, et d'autre paramètres supplémentaires concernant les autres utilisateurs.

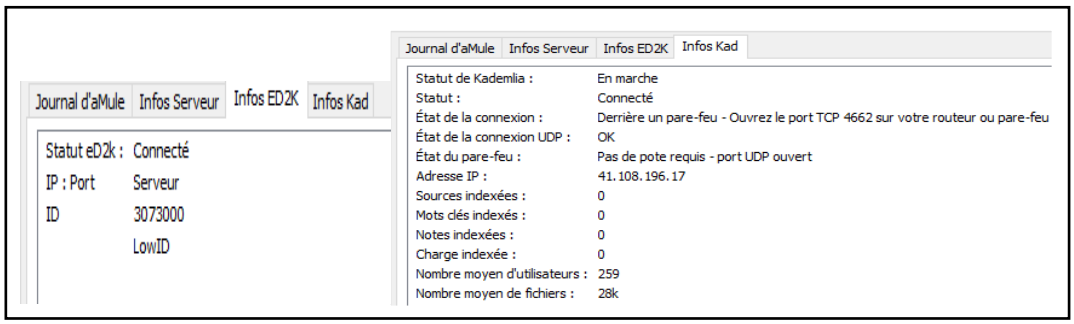


Figure III.4: Statut de connexion – client A-Mule

b. Accomplissement d'une recherche d'un fichier

L'intégration d'un moteur de recherche dans le client A-Mule offre un choix de recherche des fichiers selon trois types de connexion différents (Locale -Globale - KAD).

Une fois que vous êtes connecté, dans la barre supérieure cliquez sur le bouton **Recherches** pour accéder à la section de recherche. A chaque recherche, un onglet différent s'ouvrira.

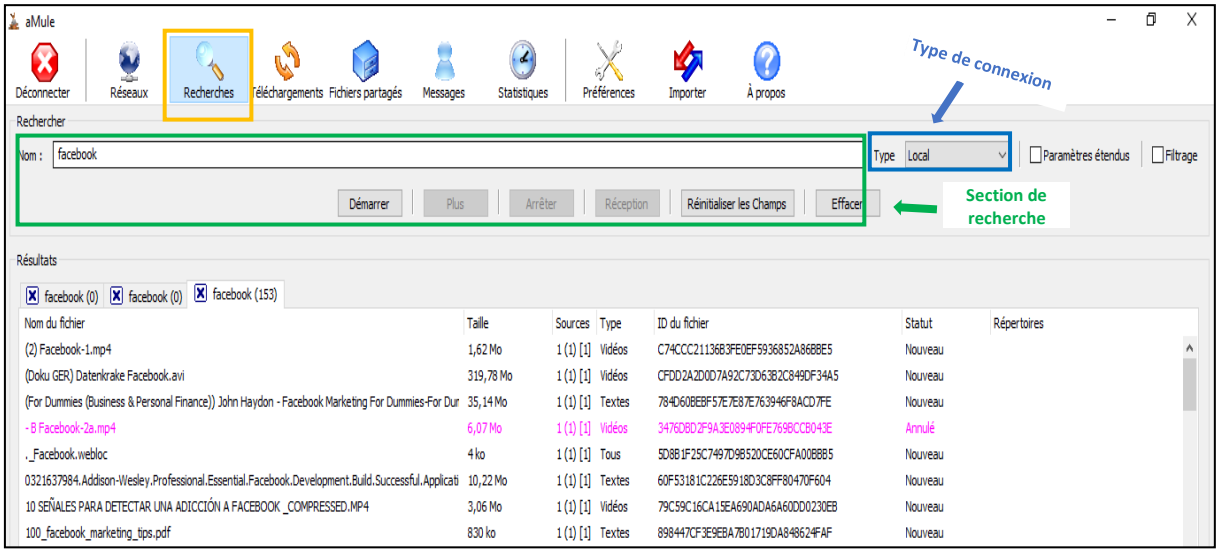


Figure III.5 Résultats de recherche avec le client E-Donkey

c. Téléchargement du fichier

Dans les résultats de la recherche, on double-clique sur le fichier souhaité et le téléchargement commence. Ensuite, on clique sur l'icône **Téléchargement** pour accéder à la section de téléchargement, où nous verrons la progression de chacun. Qui nous indiquera le nombre de sources ou de clients auxquels nous sommes connectés.

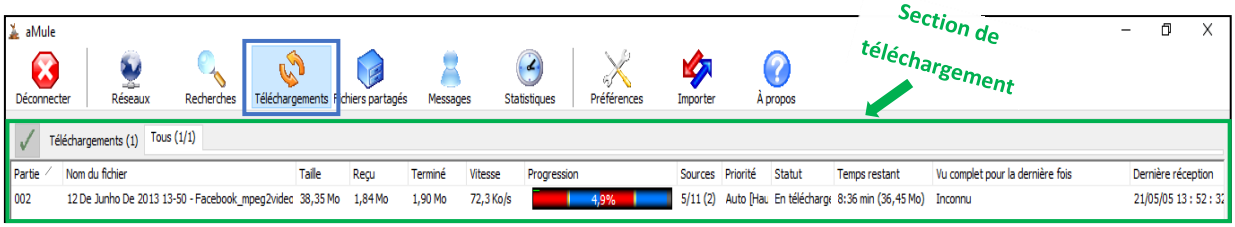


Figure III.6 : Progression de téléchargements du fichier avec le client E-Donkey

Note : il existe une liste d'attente derrière chaque fichier à télécharger et les clients doivent passer obligatoirement par cette liste, si la liste est vide le téléchargement commencera directement.

III.5.2. µTorrent

Le fonctionnement de ce client est complétement différent car l'utilisateur doit chercher lui-même le fichier à télécharger afin d'obtenir le fichier MetaData qui ressemble à ce fichier, après il lance le téléchargement dans µTorrent indépendamment. Donc les états de fonctionnement de ce client sont :

1. Obtention du fichier MetaData.
2. Exécution du fichier MetaData : trouver le tracker.
3. Etablissement de connexion Peer – Peer.
4. Mode crypté : Forcé.

La 1 ère étape à faire est de se connecter aux sites web offrant des téléchargements de fichier torrent, voici l'exemple de trois sites torrents différents :

- https://LimeTorrents.com
- https://RARBG.com
- https://OxTorrent.com

On laisse chaque téléchargement jusqu'à la fin, après on refait le même procédé avec le mode crypté. Cette méthode nous permettra d'étudier tous les cas possibles du fonctionnement de µTorrent.

a. Obtention du fichier MetaData

Lorsqu'un utilisateur cherche à utiliser le client µTorrent pour télécharger un fichier, il faut tout d'abord obtenir le fichier MetaData, ce dernier est la première étape du téléchargement µTorrent.

- Exemple d'un fichier MetaData obtenu à partir de « LimeTorrents »

Figure III.7 : Téléchargement du fichier MetaData site « Limetorrents »

b. Exécution du fichier MetaData : Etablissement de connexion Peer – Tracker

Après avoir récupéré le fichier MetaData, on cherche à trouver les autres Peers qui partagent le fichier. Pour cela, le client va envoyer des requêtes **HTTP GET** vers le tracker qui va gérer la connexion par la suite. Il suffit juste d'exécuter le fichier MetaData pour établir la connexion au tracker, on peut de même limiter le taux de consommation de la bande passante, voir les trackers qui vont gérer ce téléchargement, activer ou désactiver le DHT, et aussi d'autres configurations de téléchargement.

La touche **Propriétés** nous permettra d'avoir d'autre options comme montre la figure ci-dessous :

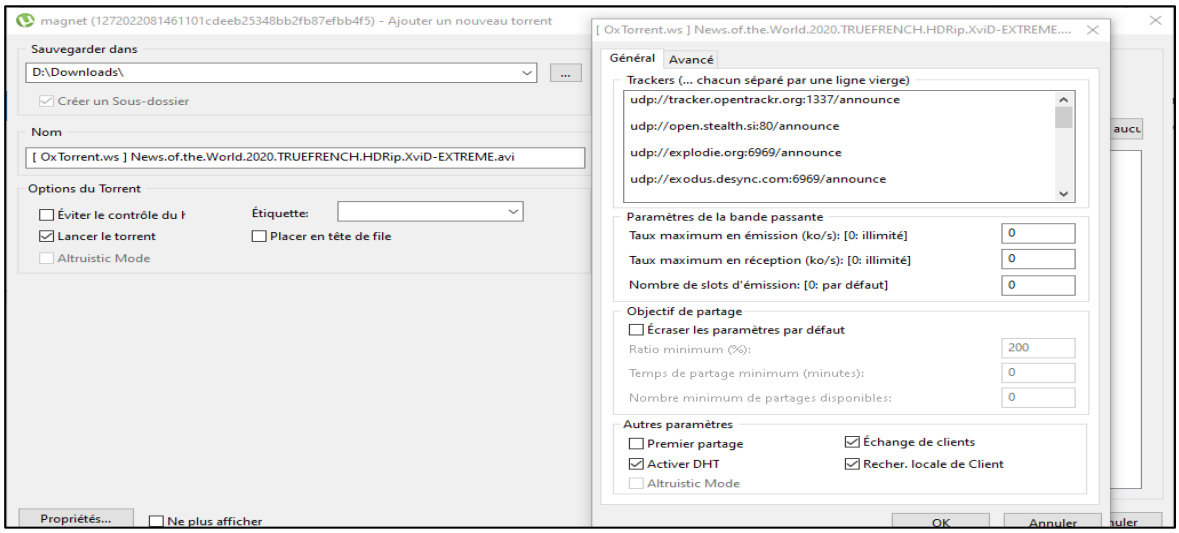


Figure III.8 : Fenêtre de configuration du téléchargement torrent

c. Etablissement de connexion Peer – Tracker

Une fois que le téléchargement commence, on retrouve au niveau du logiciel toutes les informations qui concernent le fichier et leurs sources (voir figure III.9)

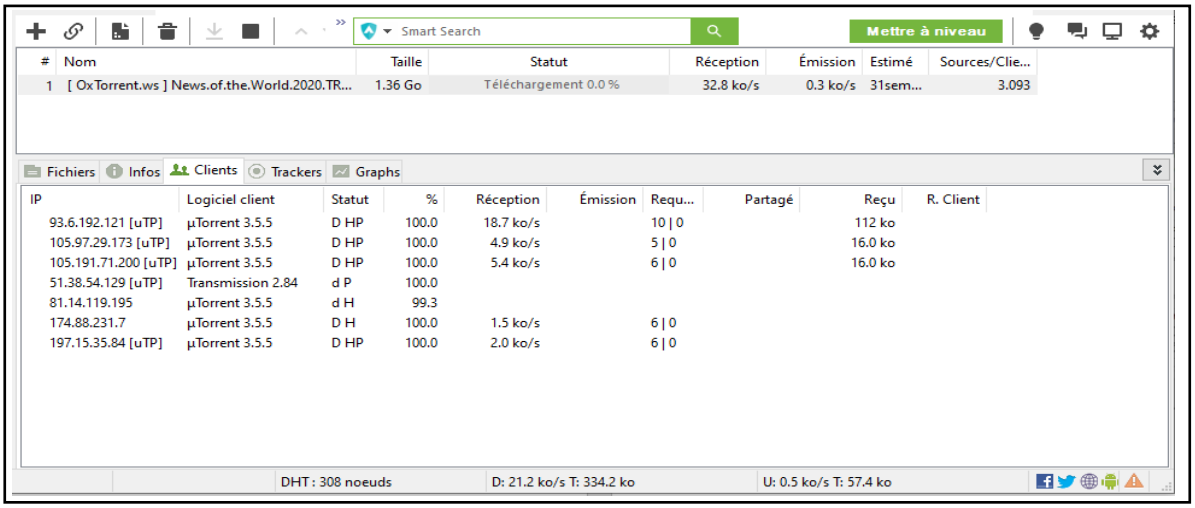


Figure III.9 : Interface graphique du logiciel µTorrent

d. Mode crypté : Forcé

Généralement l'acheminement de paquets entre clients se fait en clair, mais il contient des extensions de cryptage qui permet de chiffrer la communication entre les Peers pour protéger nos messages contre les regards extérieurs. Vu que les Peers en mode crypté ne peuvent communiquer qu'entre eux seulement, les sources des fichiers seront limitées.

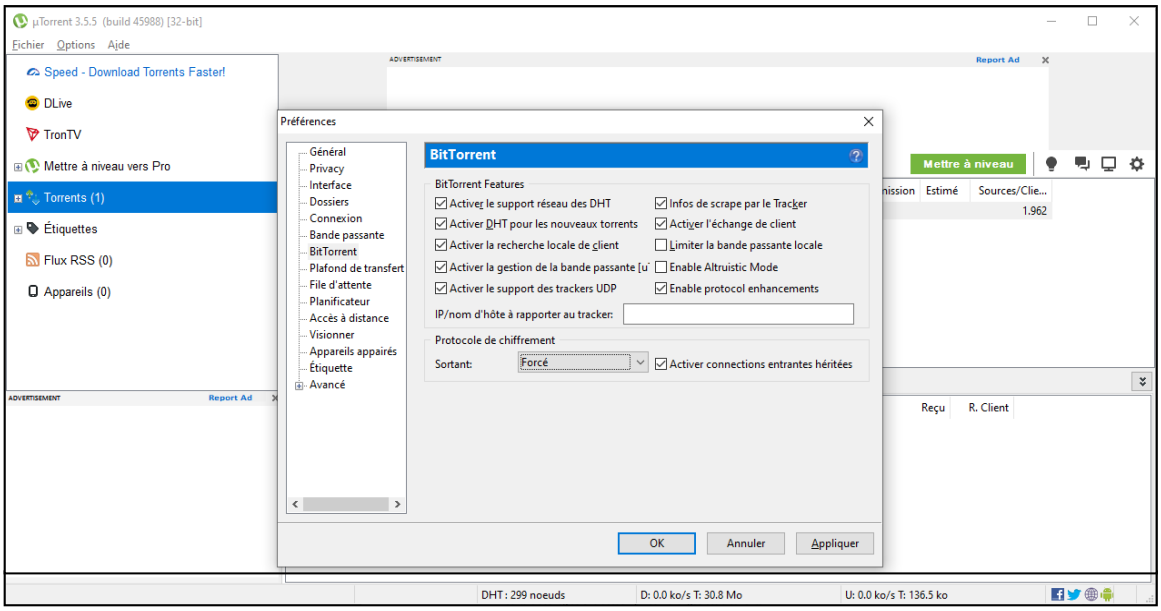


Figure III.10: Extensions du protocole BitTorrent sous µTorrent

III.6. L'analyse du trafic

On va effectuer une analyse approfondie des paquets pour chaque état de fonctionnement précédent afin d'extraire les empreintes numériques, cette dernière nous permettra d'identifier l'utilisation de chaque Protocole.

III.6.1. A-Mule

a. Etablissement de connexion :

Voici un extrait des paquets reçus lors d'une connexion à base d'un client E-Donkey (voire figure III.11).

No.	Time	Source	Destination	Protocol	Length	Info
77	32.466787	10.0.0.3	59.126.251.46	eDonkey	167	eDonkey TCP: Hello
82	33.127163	59.126.251.46	10.0.0.3	eDonkey	170	eDonkey TCP: Hello Answer
84	33.282754	10.0.0.3	59.126.251.46	eDonkey	65	eMule Extensions TCP: Second Identification State
88	33.844041	59.126.251.46	10.0.0.3	eDonkey	88	eMule Extensions TCP: Hello, eMule Extensions TCP: Second Identification State
91	33.900927	10.0.0.3	59.126.251.46	eDonkey	68	eMule Extensions TCP: Unknown
94	34.355990	59.126.251.46	10.0.0.3	eDonkey	137	eMule Extensions TCP: Public Key
95	34.410249	10.0.0.3	59.126.251.46	eDonkey	162	eMule Extensions TCP: Hello, eMule Extensions TCP: Public Key
99	35.025006	10.0.0.3	59.126.251.46	eDonkey	109	eMule Extensions TCP: Signature
100	35.277777	59.126.251.46	10.0.0.3	eDonkey	109	eMule Extensions TCP: Signature
122	49.566192	10.0.0.3	91.208.184.143	eDonkey	48	eDonkey UDP: Server Status Request
126	55.221325	10.0.0.3	212.83.184.152	eDonkey	48	eDonkey UDP: Server Status Request
137	60.737926	10.0.0.3	183.136.232.234	eDonkey	48	eDonkey UDP: Server Status Request
147	66.255303	10.0.0.3	62.210.28.77	eDonkey	48	eDonkey UDP: Server Status Request
172	70.676874	10.0.0.3	47.37.145.12	eDonkey	48	eDonkey UDP: Server Status Request

Figure III.11: Paquets E-Donkey capturés lors d'établissement de connexion

Lors du lancement de l'application A-Mule, on constate l'envoie de plusieurs paquets UDP et TCP. Ces derniers sont la première étape de fonctionnement A-Mule, qui nous permettrons en suite le téléchargement et le partage des fichiers. Si on veut identifier l'utilisation de protocole E-Donkey, il faudra étudier et analyser ces requêtes en détail.

D'après la figure (III.11) l'établissement de la connexion avec le serveur « 59.126.251.46 » commence par une requête '' **Hello client** ''. Le contenu de cette requête est détaillé dans la figure ci-dessous :

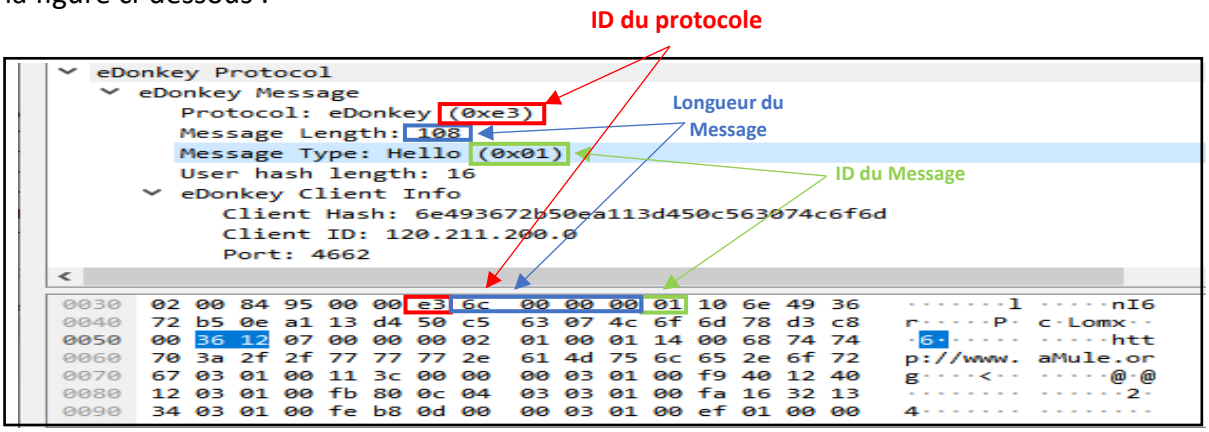


Figure III.12 : Capture du paquet (E-Donkey-Hello)

- Le serveur répond à cette requête par ''**Hello answer''**

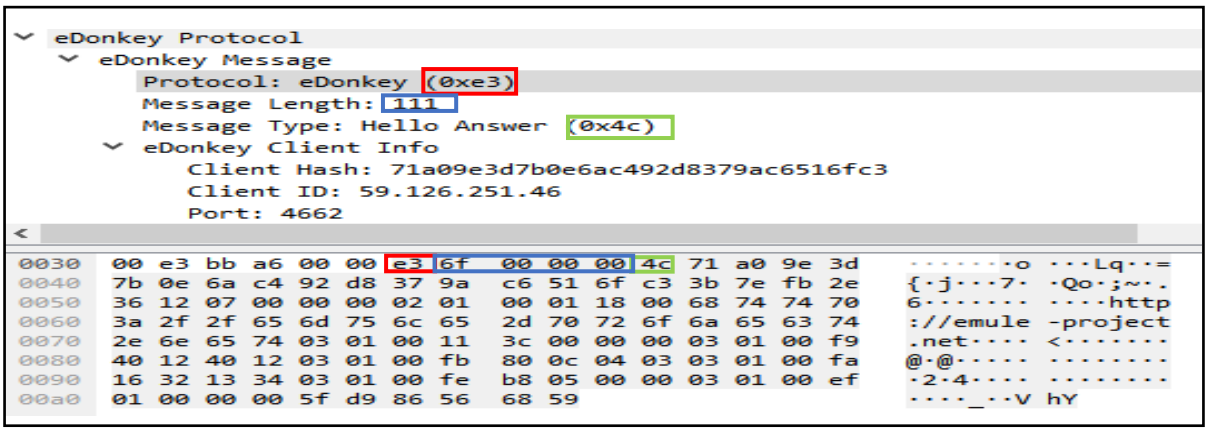


Figure III.13 : Capture du paquet (server hello answer)

- Après l'établissement de connexion, le serveur E-Donkey et le client commence l'échange des paramètres supplémentaires qui concerne l'indentification et les options de partage.

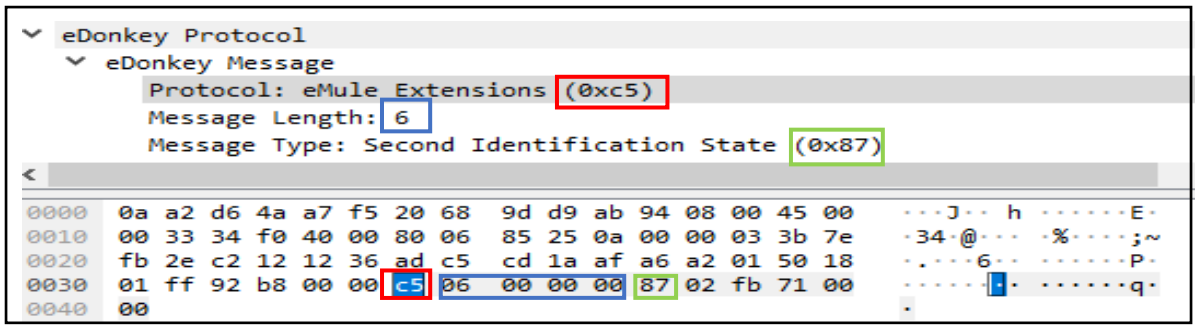


Figure III.14 : Capture du paquet (Second identification state)

- Cette requête représente le deuxième état d'identification avec le client qu'il envoie pour s'assurer de la communication avec le serveur. Par la suite le serveur répond par une requête qui contient les deux informations précédentes « **Hello & second identification state** »

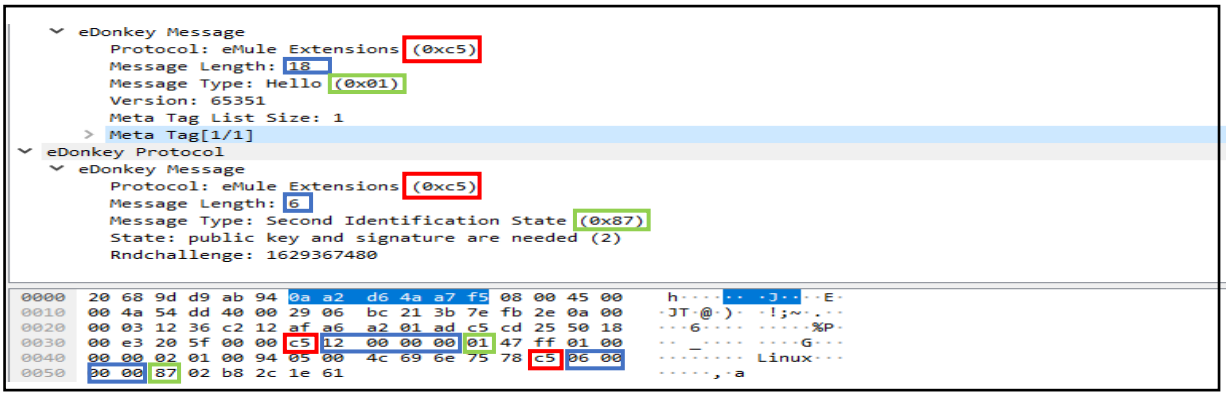


Figure III.15 : Capture du paquet (Second identification state response)

- Le serveur propose une clé publique au client par une requête « **Public -key** », pour indexer l'ID du client au niveau du serveur.

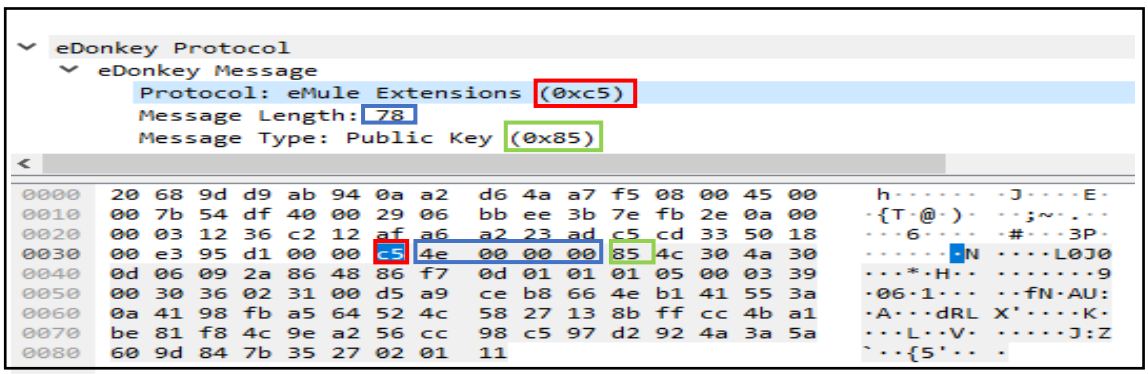


Figure III.16: Capture du paquet (Public Key)

- Le client répond par une requête contenant la confirmation de cette clé :

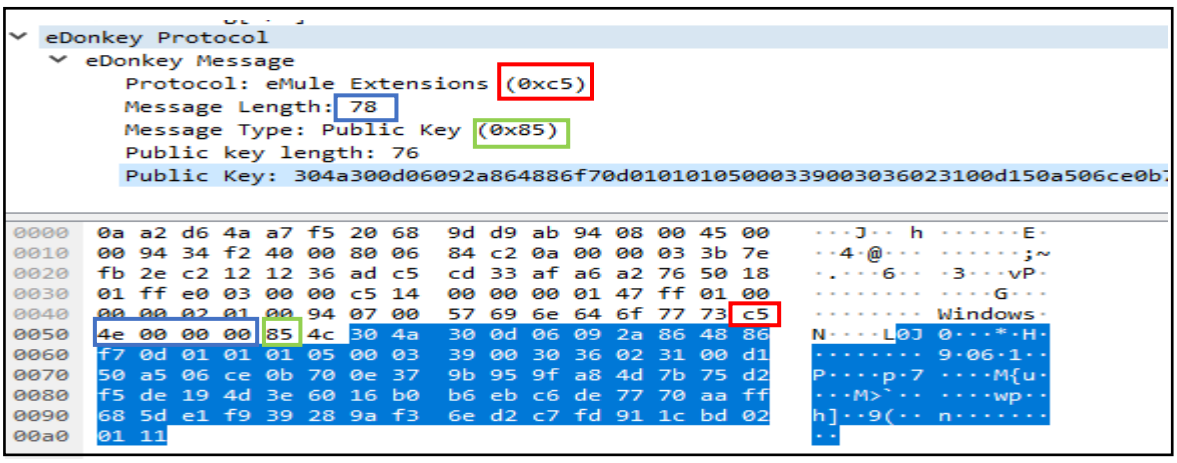


Figure III.17 : Capture du paquet (Public Key response)

- Pour terminer l'identification, le client propose au serveur une signature spéciale qui identifiera les téléchargements ou le partage de ce client, par une requête **Signature** :

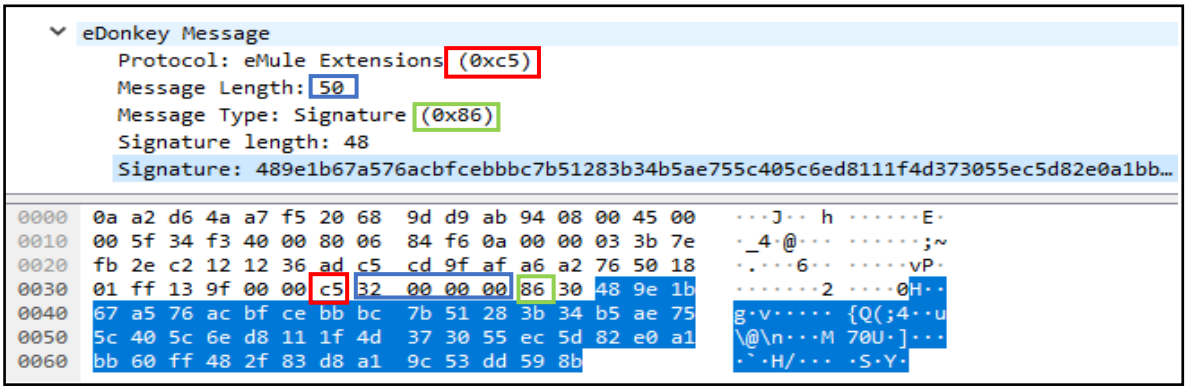


Figure III.18: Capture du paquet (Signature request)

- Le serveur répond par une requête contenant la signature qui identifiera par la suite toutes les taches de ce client :

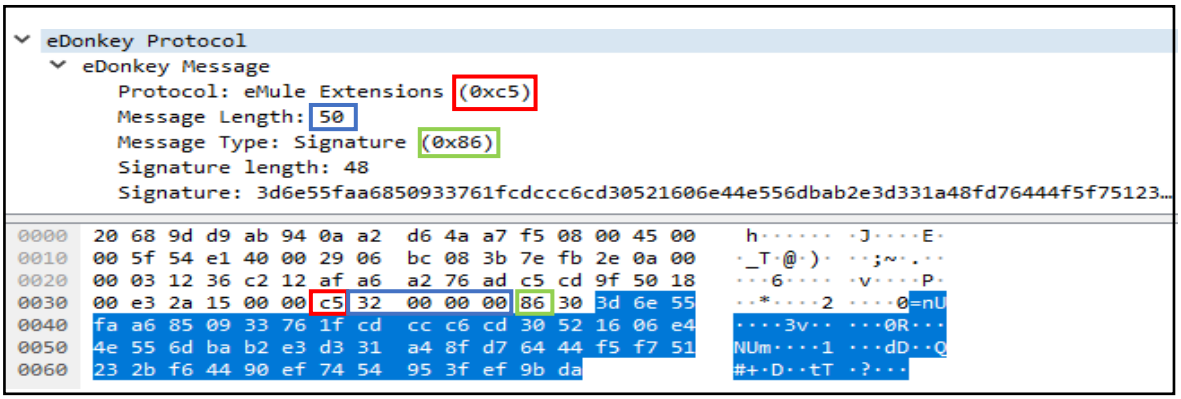


Figure III.19: Capture du paquet (Signature response)

- Le client établit automatiquement des connexions UDP avec les serveurs de ce réseau afin de faire constamment des mises à jour d'état de ces derniers. La figure (III.20) montre clairement la demande des états des serveurs à partir du client a l'aide de la requête « **Server status request** » :

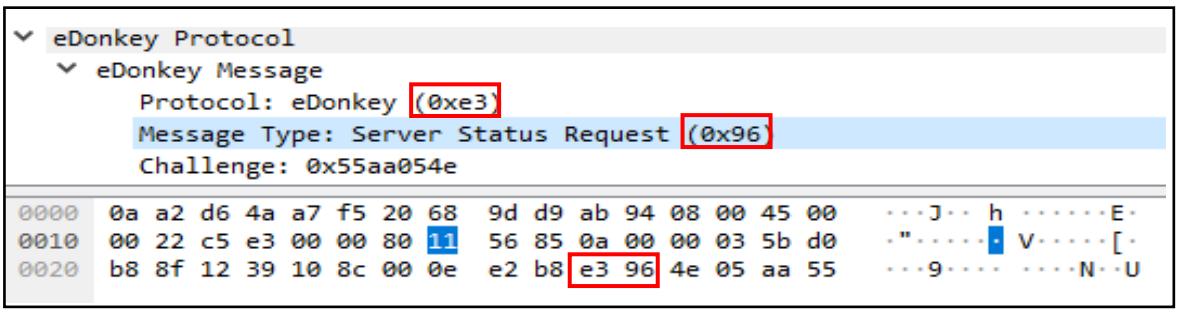


Figure III.20: Capture du paquet (Server Status Request)

b. Accomplissement d'une recherche d'un fichier

Une fois la connexion avec le serveur E-Donkey établit, le client peut faire une recherche au niveau de A-Mule. Les résultats d'analyse obtenu pendant cette étape sont représentés ci-dessous :

No.	Time	Source	Destination	Protocol	Length	Info
102	32.101501	10.0.0.3	80.208.228.241	eDonkey	62	eDonkey UDP: Reask File Ping
103	32.461742	80.208.228.241	10.0.0.3	eDonkey	914	eDonkey UDP: Search File Results
104	32.461850	80.208.228.241	10.0.0.3	eDonkey	899	eDonkey UDP: Search File Results
105	32.466836	80.208.228.241	10.0.0.3	eDonkey	859	eDonkey UDP: Search File Results
106	32.496497	80.208.228.241	10.0.0.3	eDonkey	925	eDonkey UDP: Search File Results
107	32.767910	80.208.228.241	10.0.0.3	eDonkey	843	eDonkey UDP: Search File Results

Figure III.21 : Paquets E-Donkey capturés lors d'une recherche A-Mule N°01

> Note : Toutes les connexions reposent sur l'UDP seulement car le client est situé derrière un Pare-Feu, donc il est identifié par un LOWID.

- Le client envoie une requête au serveur « 80.208.228.241 », de type « **Reask File Ping** » afin de localiser le fichier, cette requête est représentée comme suit

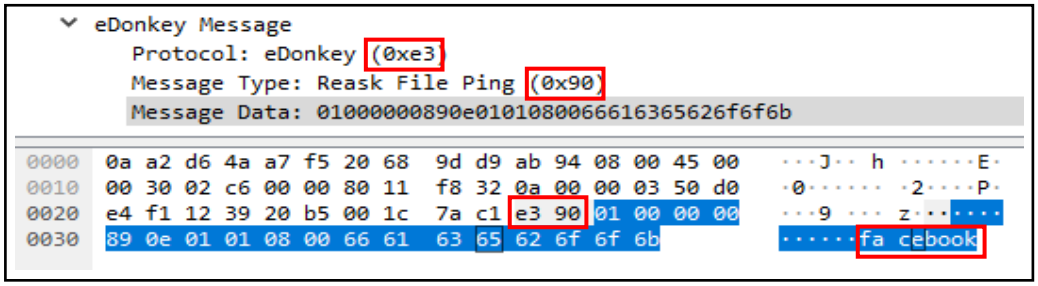


Figure III.22: Capture du paquet (Reask file ping)

- Le contenu de la recherche est clairement visible dans le paquet (figure III.22), le serveur répond par une requête « **Search file resultes** » comme suit :

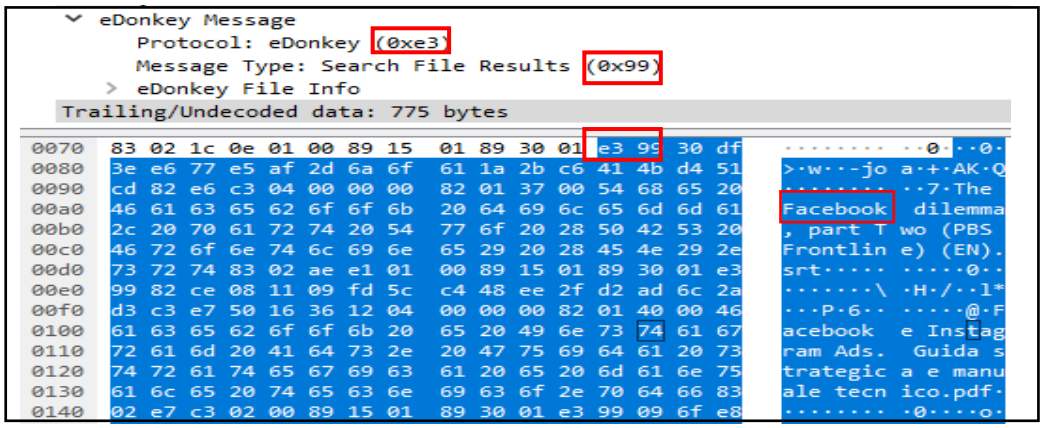


Figure III.23: Capture du paquet (Search file resultes)

- Le client A-Mule utilise la requête « **search File** » afin de demander des informations concernent la recherche, cette requête est diffusée vers tous les serveurs, uniquement le serveur principal du réseau E-Donkey y répond dû au LOWID.

100	31.929819	10.0.0.3	212.83.184.152	eDonkey	55 eDonkey UDP: Search File
110	32.945184	10.0.0.3	62.210.28.77	eDonkey	55 eDonkey UDP: Search File
111	33.788911	10.0.0.3	47.37.145.12	eDonkey	55 eDonkey UDP: Search File
112	34.601616	10.0.0.3	183.136.232.234	eDonkey	55 eDonkey UDP: Search File

Figure III.24 : Paquets E-Donkey capturés lors d'une recherche A-Mule N°02

c. Téléchargement des fichiers

Le téléchargement à l'aide de A-Mule est basé entièrement sur le protocole KADEMLIA, ce dernier fonctionne suivant un mécanisme basé sur UDP permettant d'effectuer des téléchargements même si les ports TCP sont bloqués.

Les requêtes principales de ce protocole sont représentées dans la figure suivante :

No.	Time	Source	Destination	Protocol	Length	Info
7	4.641334	10.0.0.3	1.85.248.135	eDonkey	64	Kademlia UDP: KADEMLIA2_HELLO_REQ
10	5.095082	1.85.248.135	10.0.0.3	eDonkey	80	Kademlia UDP: KADEMLIA2_HELLO_RES
11	5.469505	10.0.0.3	1.85.248.135	eDonkey	77	Kademlia UDP: KADEMLIA2_REQ
12	6.015294	1.85.248.135	10.0.0.3	eDonkey	111	Kademlia UDP: KADEMLIA2_RES
98	31.254452	10.0.0.3	220.190.11.227	eDonkey	64	Kademlia UDP: KADEMLIA2_HELLO_REQ
190	88.574234	10.0.0.3	58.253.40.192	eDonkey	64	Kademlia UDP: KADEMLIA2_HELLO_REQ
214	98.658998	10.0.0.3	117.62.48.69	eDonkey	77	Kademlia UDP: KADEMLIA2_REQ
457	108.050690	10.0.0.3	106.84.201.217	eDonkey	77	Kademlia UDP: KADEMLIA2_REQ
2808	135.292005	10.0.0.3	183.167.174.45	eDonkey	64	Kademlia UDP: KADEMLIA2_HELLO_REQ
2850	135.765195	183.167.174.45	10.0.0.3	eDonkey	80	Kademlia UDP: KADEMLIA2_HELLO_RES
2881	136.119968	10.0.0.3	183.167.174.45	eDonkey	77	Kademlia UDP: KADEMLIA2_REQ
2917	136.560219	183.167.174.45	10.0.0.3	eDonkey	111	Kademlia UDP: KADEMLIA2_RES
3084	138.573953	10.0.0.3	106.91.101.108	eDonkey	64	Kademlia UDP: KADEMLIA2_HELLO_REQ
3929	148.684609	10.0.0.3	119.119.167.201	eDonkey	64	Kademlia UDP: KADEMLIA2_HELLO_REQ

Figure III.25: Paquets E-Donkey capturés lors du téléchargement d'un fichier

D'après la figure (III.25) on peut constater que le fonctionnement du protocole KADEMLIA est basé sur 4 types de requête principales :

- **KADEMLIA_hello_REQ**

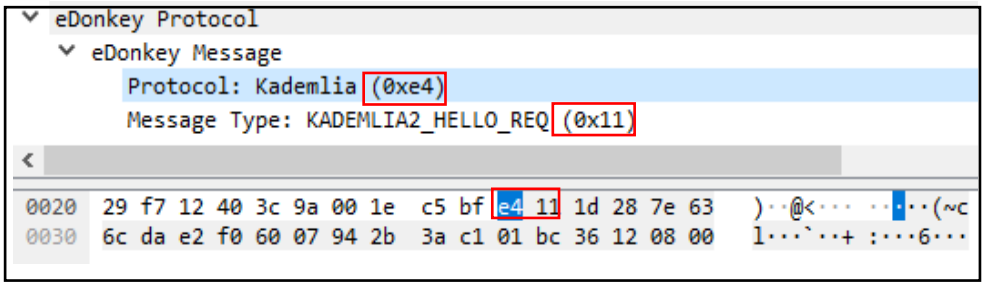


Figure III.26: Capture du paquet (KADEMLIA_hello_REQ)

- **KADEMLIA_hello_RES**

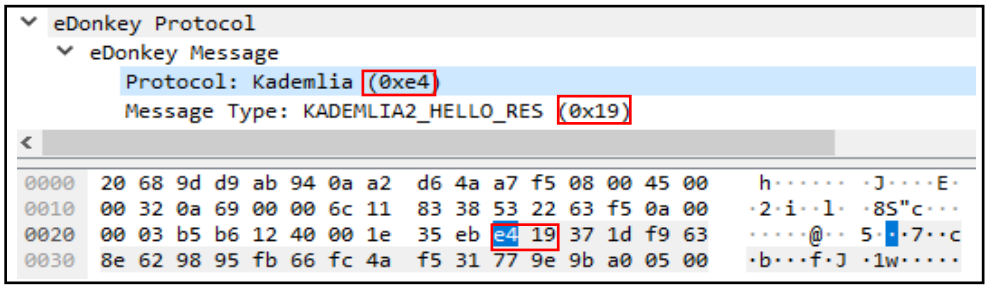


Figure III.27: Capture du paquet (KADEMLIA_hello_RES)

- **KADEMLIA2_REQ**

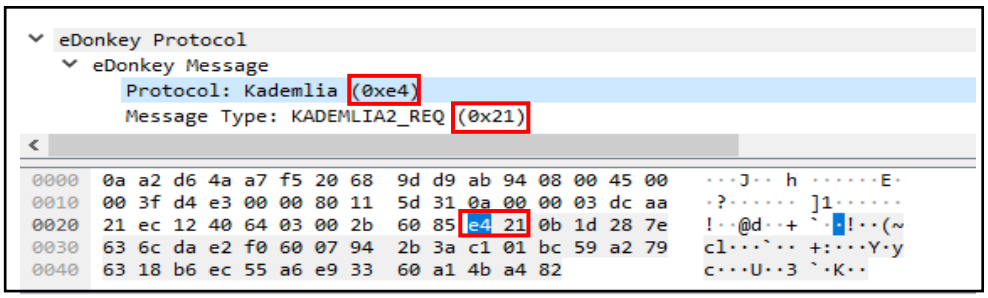


Figure III.28: Capture du paquet (KADEMLIA2_REQ)

- **KADEMLIA2_RES**

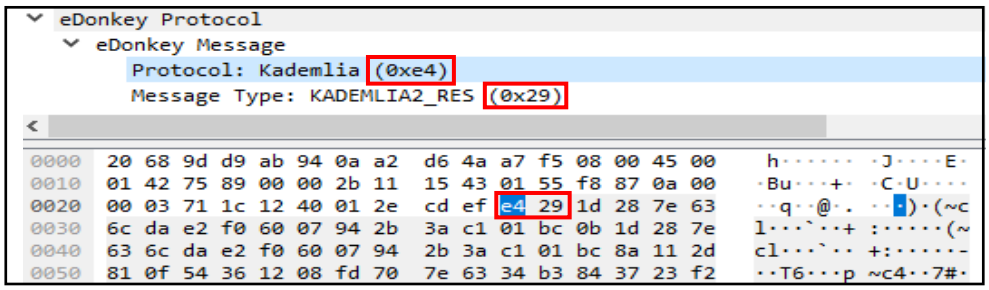


Figure III.29:Capture du paquet (KADEMLIA2_RES)

La mise à jour des nœuds et les Peers du protocole KADEMLIA se fait de manière périodique. A l'aide de la requête suivante :

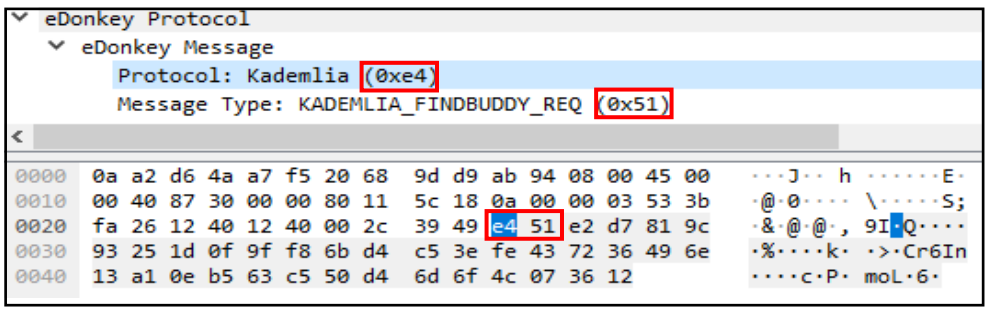


Figure III.30 : Capture du paquet (KADEMLIA_FINDBUDDY_REQ)

Après plusieurs essais on déduit qu'il n'y a pas de nouveauté dans les requêtes entre le client et les serveurs. Les requêtes principales du client A-Mule sont représentés dans notre étude précédente, sauf qu'il existe d'autres éléments supplémentaires que l'on citera à la fin de ce chapitre.

Note : Il est possible d'effectuer le téléchargement à l'aide du protocole E-Donkey uniquement avec les requêtes « GET SOURCES » qui est l'équivalent de « KADEMLIA2_REQ »

III.6.2. µTorrent

a. Obtention du fichier MetaData :

L'utilisation du protocole http permet l'obtention du fichier MetaData pareille à celui de la navigation normale, sauf que les requêtes **HTTP GET** sont destinés cette fois pour un fichier torrent. La figure (III.31) montre clairement les requêtes HTTP échangées lors de la première connexion pour l'obtention du fichier « .torrent » :

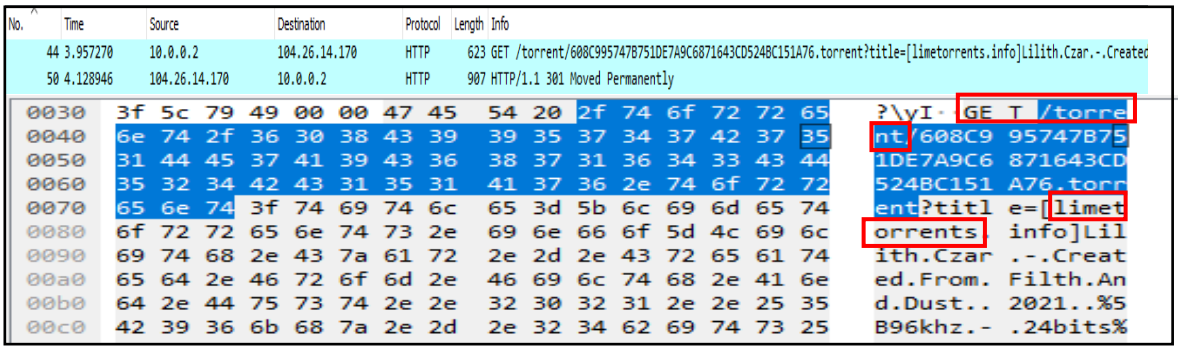


Figure III.31: Contenu de la requête http GET TORRENT

- Le client demande au serveur web « 104.26.14.170 » des informations pour le fichier « .torrent ».
- Cette demande est envoyée une fois que l'utilisateur veut télécharger le fichier .torrent.

b. Exécution du fichier MetaData : trouver le tracker

Jusqu'à présent on n'a pas commencé le téléchargement, une fois l'exécution du fichier MetaData est fait le client BitTorrent commence l'établissement de la connexion avec le tracker. Les paquets capturés dans cette étape sont représentés dans la figure (III.32) :

No.	Time	Source	Destination	Protocol	Length	Info
567	16.749134	10.0.0.2	172.67.140.164	HTTP	250	GET /scrape?info_hash=%19dAq%cd%ba%89%f7~%
589	17.207289	172.67.140.164	10.0.0.2	HTTP	59	HTTP/1.1 200 OK (text/html)
737	20.232945	10.0.0.2	104.21.3.146	HTTP	431	GET /announce?info_hash=%19dAq%cd%ba%89%f7
791	20.613462	104.21.3.146	10.0.0.2	HTTP	59	HTTP/1.1 200 OK (text/plain)
1458	26.170246	10.0.0.2	54.37.106.164	HTTP	431	GET /announce?info_hash=%19dAq%cd%ba%89%f7
1514	26.422944	54.37.106.164	10.0.0.2	HTTP	408	HTTP/1.1 307 Temporary Redirect

Figure III.32: Requêtes principales Peer-Tracker – http GET

- Le client contacte un nouveau serveur, mais cette fois-ci il contacte le tracker « 172.67.140.164 » avec la requête **GET SCRAPE** qui contient l'ID du fichier « info_hash » pour avoir des informations concernant le fichier « .torrent » et tous les trackers hébergés.

- Le contenu de requête **http GET SCRAPE** est clairement visible dans la figure ci-dessous :

```
0000  0a a2 d6 4a a7 f5 08 ed  b9 28 a2 aa 08 00 45 00   ···J···· ·(····E·
0010  00 e5 77 b7 40 00 80 06  7a 90 0a 00 00 02 63 53   ··w·@··· z·····cS
0020  9a 76 c6 3e 00 50 dd 2b  b5 de 05 1c 13 c0 50 18   ·v·>·P·+ ·····P·
0030  40 42 f9 2d 00 00 47 45  54 20 2f 73 63 72 61 70   @B·-··GE T /scrap
0040  65 3f 69 6e 66 6f 5f 68  61 73 68 3d 25 36 30 25   e?info_h ash=%60%
0050  38 63 25 39 39 57 47 25  62 37 51 25 64 65 7a 25   8c%99WG% b7Q%dez%
0060  39 63 68 71 64 25 33 63  25 64 35 25 32 34 25 62   9chqd%3c %d5%24%b
0070  63 25 31 35 25 31 61 76  20 48 54 54 50 2f 31 2e   c%15%1av  HTTP/1.
0080  31 0d 0a 48 6f 73 74 3a  20 74 72 61 63 6b 65 72   1··Host:  tracker
0090  2e 74 66 69 6c 65 2e 63  6f 0d 0a 55 73 65 72 2d   .tfile.c o··User-
00a0  41 67 65 6e 74 3a 20 75  54 6f 72 72 65 6e 74 2f   Agent: u Torrent/
00b0  33 35 35 28 31 31 31 39  31 35 39 31 38 29 28 34   355(1119 15918)(4
00c0  35 39 36 36 29 0d 0a 41  63 63 65 70 74 2d 45 6e   5966)··A ccept-En
00d0  63 6f 64 69 6e 67 3a 20  67 7a 69 70 0d 0a 43 6f   coding:  gzip··Co
00e0  6e 6e 65 63 74 69 6f 6e  3a 20 43 6c 6f 73 65 0d   nnection : Close·
00f0  0a 0d 0a                                          ···
```

Figure III.33: Contenu de la requête http GET SCRAPE

- Le tracker « 127.67.140.164 » répond avec une liste des trackers disponible, le client envoie des requêtes **http GET ANOUNCE** avec le même ID « info_hash » vers tous les trackers disponibles.

- Le contenu de requête « **http GET ANOUNCE** » est représenté dans la figure suivante :

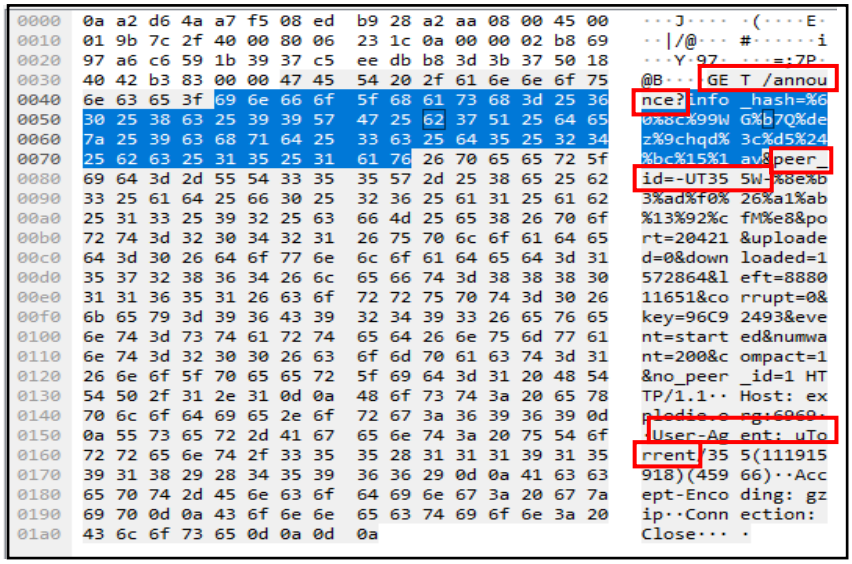


Figure III.34: Contenu de la requête http GET ANOUNCE

- Cette requête regroupe toutes les informations nécessaires pour établir la connexion avec le tracker.

- Une fois l'un de ces trackers « **104.21.3.146** » répond avec la requête « **http/1.1 200 OK** » le client établira une connexion. La réponse du tracker permet au client d'établir des connexions avec les Peers qui partage le fichier, cette réponse est comme suit :

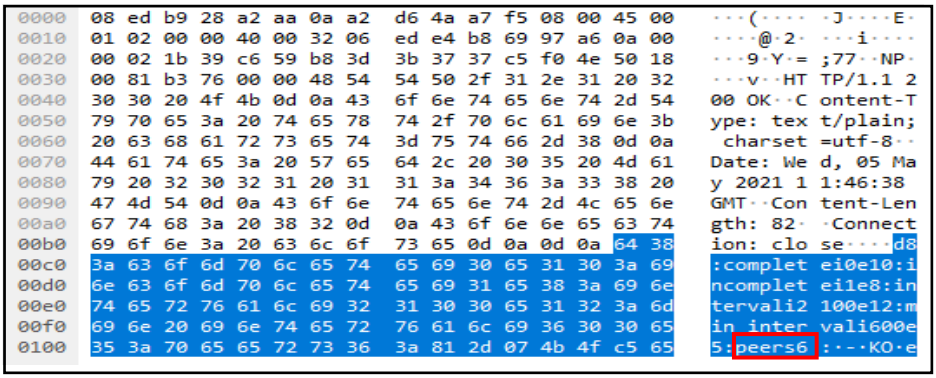


Figure III.35: Contenu de la réponse http GET ANOUNCE

Note : Jusqu'à cette étape aucune interaction du protocole BitTorrent est notée, seul le protocole http qui prend en charge l'établissement de la connexion avec le tracker.

c. Etablissement de la connexion Peer – Peer

Une fois que l'utilisateur a une connexion avec les Peers, le premier message qu'il envoie devrait être **BitTorrent HANDSHAKE,** le protocole BitTorrent établit cette connexion comme la figure le montre :

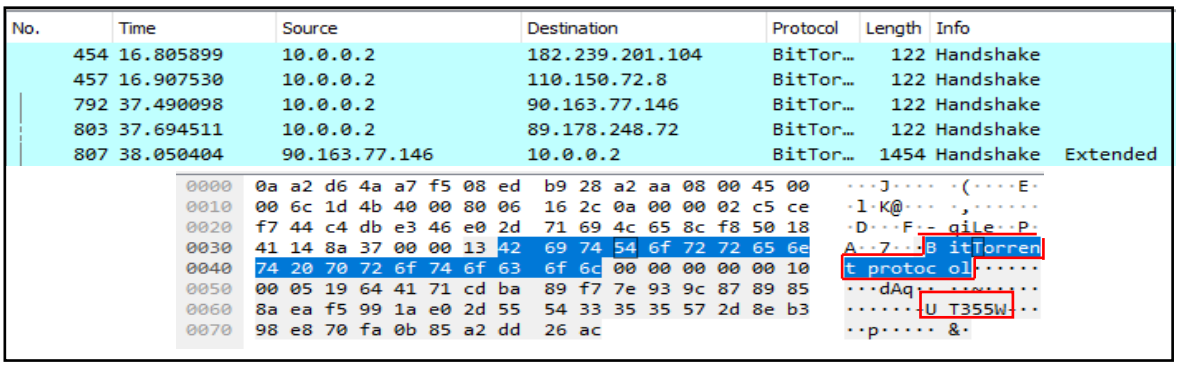


Figure III.36:BitTorrent HANDSHAKE

- On remarque que la requête **HANDSHAKE** contient des informations dédiées au protocole BitTorrent et le Client (Peer ID).

Note : Cette étape sera répétée avec tous les Peers qui étaient envoyés par le tracker.

- La réponse de cette requête est **HANDSAKE EXTENDED,** qui est représentée dans la figure (III.37) :

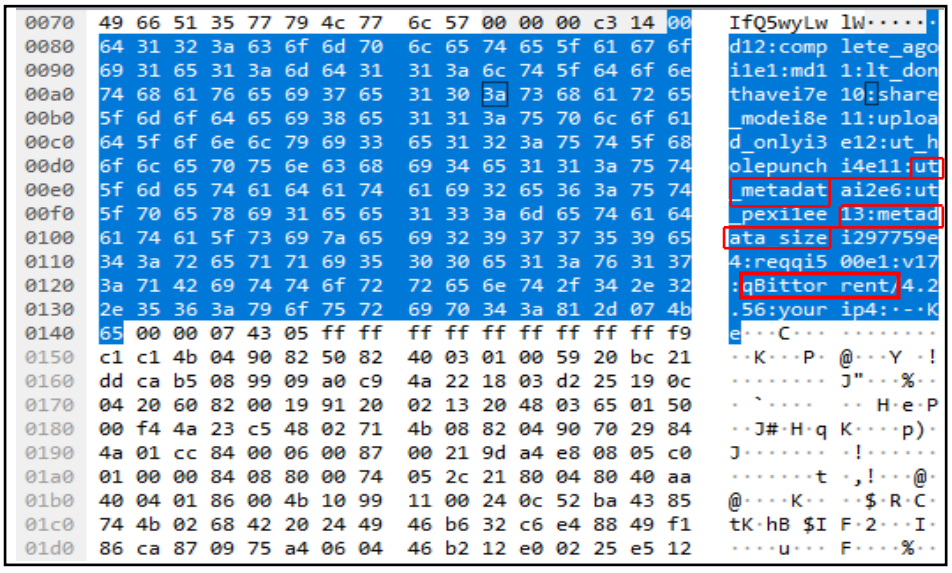


Figure III.37: Réponse BitTorrent HANDSHAKE

- D'après la figure (III.37), on constate que la requête **BitTorrent handshake** dédie pour renseigner le client sur les paramètres des Peers, comme la taille du fichier qui

contient, le mode de fonctionnement « upload ou bien download », le logiciel qui l'utilise (QBitTorrent) avec sa version.

- Ensuite, le client joint le **SWARM** de téléchargement, et commence à faire la négociation avec les autres Peers concernant les ports et les pièces disponibles /restants.

> Note : D'une manière générale le fonctionnement de µTorrent nécessite deux protocoles principales « BitTorrent, http ».

- Parmi les extensions du protocole BitTorrent on a le DHT, ce dernier a pour but d'identifier les Peers rapidement. Il peut être détecté dans les tests effectués par le **ping DHT** qui implémente dans le protocole UDP :

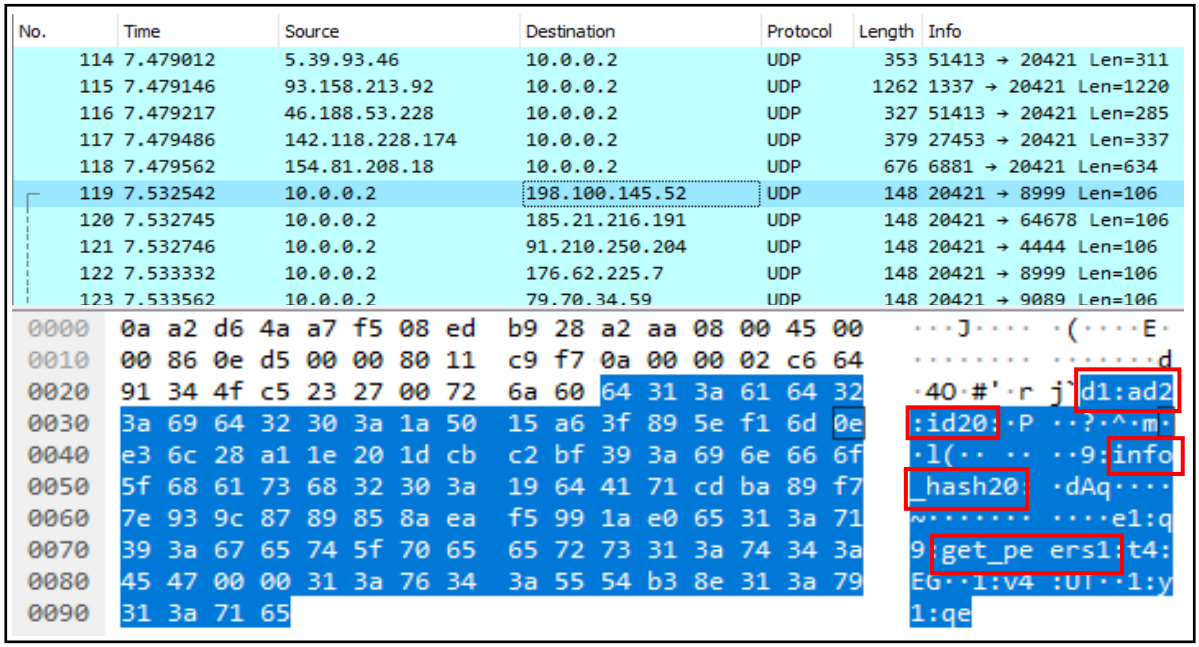


Figure III.38: Ping DHT – Peers

- La figure (III.38) illustre clairement cette apparence. Le ping est représenté par la chaine « **d1 :ad2 :id20** », informations de torrent par « **info_hash20** », et la demande des Peers par « **get_peers1** ».

- Autre utilisation de DHT pour pinger le tracker, cette fois la requête contient seulement un indice qui concerne le tracker « **ANNOUNCE** ».

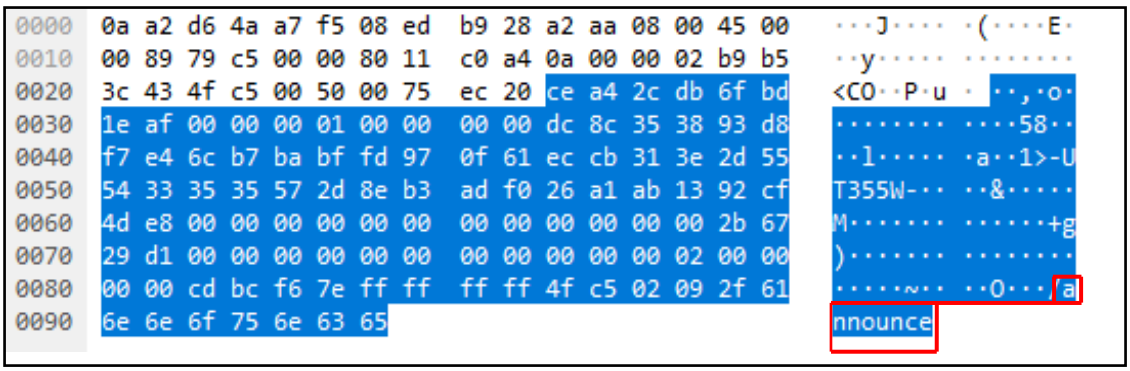


Figure III.39: Ping DHT - Tracker

d. Mode crypté : Forcé

Les signatures du protocole BitTorrent trouvées jusqu'à maintenant sont toutes en mode claire, une fois le mode crypté activé l'analyse de trafic devient comme la figure (III.40) nous montre :

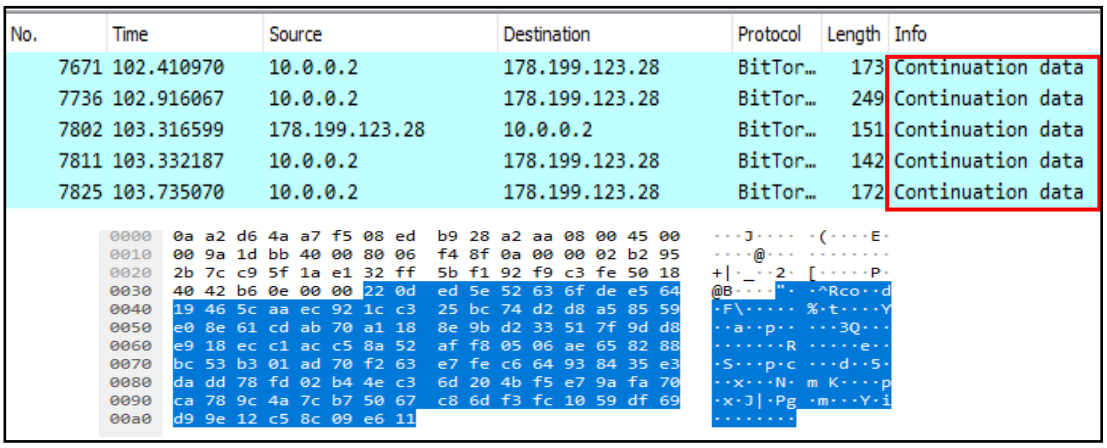


Figure III.40: BitTorrent HANDSHAKE – Crypté

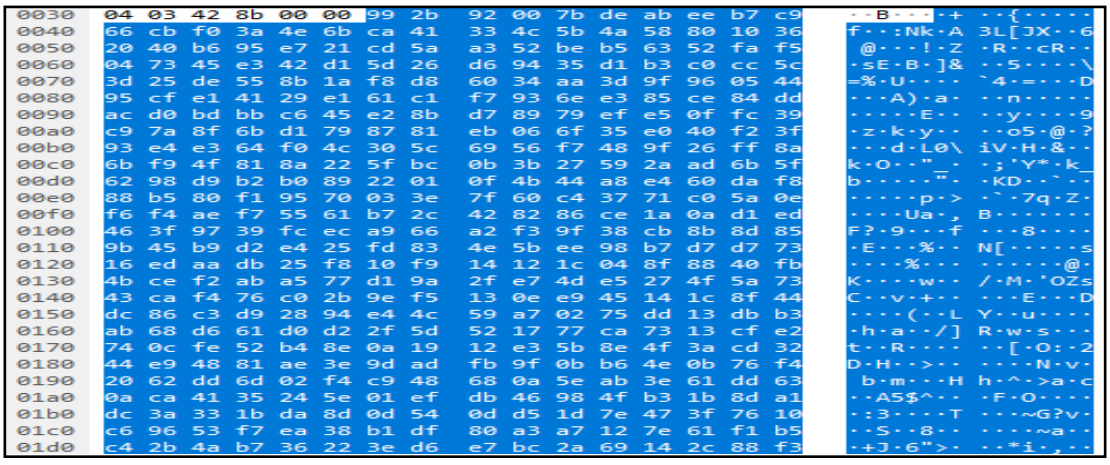


Figure III.41: Réponse de BitTorrent HANDSHAKE – Crypté

- Tous les échanges Peer to Peer qui sont basé sur BitTorrent seront chiffrés, mais comme nous l'avons vu précédemment ou on a deux autres protocoles qui rentrent

en jeux dans la structure du fonctionnement BitTorrent, le protocole http et UDP ces deux derniers fonctionnent toujours en claire même si le cryptage BitTorrent est activé.

Solution proposée : On va focaliser la détection de la connexion torrent cryptée sur le fonctionnement du protocole http et UDP.

III.7. Résultat d'analyse : Signatures d'applications

III.7.1. AMule

Notre structure d'extraction des empreintes basée sur 3 champs dans le paquet E-Donkey capturé :

- **Protocole** - Un ID de protocole à un octet - **0xE3** pour E-Donkey, **0xC5** pour E-Mule, **0xE4** pour KADEMLIA.
- **Taille** – les nombre d'octet entre le ID de protocole et le type de message de ce protocole.
- **Type** – Un octet unique - un ID de message unique.

A travers notre analyse, on déduit le tableau des signatures du protocole E-Donkey représenté ci-dessous :

TYPE DE MSG	Length	Identifiant	Transport protocole
Edonkey HELLO	XX XX XX XX	E3 01	TCP
Edonkey Hello answer	XX XX XX XX	E3 4C	TCP
Emule extensions Second identification state	XX XX XX XX	C5 87	TCP
Emule extensions Public key	XX XX XX XX	C5 85	TCP
Emule extensions Hello answer	XX XX XX XX	C5 4C	TCP
Emule extensions Signature	XX XX XX XX	C5 86	TCP
Edonkey Get server info		E3 A2	UDP
Edonkey Server statut		E3 97	UDP
Edonkey Server Statut req		E3 96	UDP
KADEMLIA-KADEMLIA2_HELLO_REQ		E4 11	UDP
KADEMLIA-KADEMLIA2_HELLO_RES		E4 19	UDP
KADEMLIA-KADEMLIA2_REQ-FIND NODE		E4 21 0B	UDP
KADEMLIA-KADEMLIA2_REQ-FIND VALUE		E4 21 02	UDP
KADEMLIA-KADEMLIA2_RES		E4 29	UDP
KADEMLIA-KADEMLIA2_REQ		E4 21	UDP
KADEMLIA-KADEMLIA_FINDBUDDY_REQ		E4 51	UDP
Edonkey Get Sources		E3 9a	UDP
eDonkey-Search File		E3 98	UDP
eDonkey-Search File Results		E3 99	UDP
eDonkey-Reask File Ping		E3 90	UDP

Tableau III-1: Empreintes numériques du protocole E-Donkey

III.7.2. µTorrent

Les résultats d'analyse des signatures du protocole BitTorrent sont résumés dans le tableau suivant :

N°	Nom	Contenu	Transport protocole
01	GET TORRENT	GET /torrent	TCP
02	GET SCARPE	GET /scrape?info_hash	TCP
		User-Agent: uTorrent	TCP
03	GET/ ANOUNCE	GET /announce?info_hash	TCP
		User-Agent: uTorrent	TCP
05	HADSHAKE	BitTorrent protocol	TCP
06	HANDSHAKE EXTENDED	ut_metadata	TCP
		metadata_size	TCP
08	DHT Ping Peer	d1:ad2:id20	UDP
		info_hash20	UDP
		get_peers1	UDP
09	DHT Ping Tracker	/announce	UDP

Tableau III-2: Empreintes numériques du protocole BitTorrent

Note : La méthode d'analyse et d'extraction est différente, chaque protocole est identifie par une méthode, A-Mule par les captures des identifiantes des paquets et µTorrent par la capture du contenu du paquet.

III.8. Conclusion

A travers ce chapitre nous avons analysé le trafic Peer to Peer venant des applications µTorrent et AMule, l'étude de cette analyse nous a permis d'extraire des empreintes numériques qui identifient l'utilisation des protocoles BitTorrent et E-Donkey, afin de les implémenter dans un système de détection d'intrusion IDS.

Dans le prochain chapitre, nous allons procéder à faire l'implémentation de ces empreintes dans l'IDS ''Snort'' et tester grâce à deux scenarios la validité de ces empreintes numériques.

Chapitre IV

Implémentation et validation des signatures

IV.1. Introduction

L'IDS est un système de détection d'intrusion (en anglais Intrusion Détection System). C'est un logiciel conçu à l'origine pour repérer des activités anormales ou suspectes sur une cible analysée. Dans ce sens, l'administrateur système responsable peut gérer la découverte d'un « intrus » grâce au fait que lorsqu'une tentative d'intrusion est détectée, une alerte est émise.

Au cours de ce dernier chapitre, on va étudier les systèmes de détection d'intrusions SNORT dans le but de créer des règles qui nous permettrons de tester la fiabilité des signatures extraites précédemment pour la détection d'utilisation des protocoles E-Donkey et BitTorrent.

IV.2. Système de détection d'intrusion SNORT

Il existe trois grands types de systèmes de détection d'intrusions (IDS) bien distincts :

- Les NIDS (Network Based Intrusion Detection System).
- Les HIDS (HostBased Intrusion Detection System).
- Les IDS hybrides à la fois NIDS et HIDS.

Snort est un IDS ou un système de détection d'intrusion basé sur un réseau (NIDS). Il est open sources et fonctionne sous les plates-formes Windows, Linux.... Il met en œuvre un moteur de détection d'attaques et d'analyse des ports qui permet d'enregistrer, d'alerter et de répondre à toutes les anomalies préalablement définies telles que des modèles qui correspondent à des attaques, des balayages, des tentatives d'exploitation de toute vulnérabilité, l'analyse des protocoles connus, etc... Et tout cela en temps réel [15].

Figure IV.1: Logo de Snort IDS [16]

IV.3. Modes de fonctionnements de Snort

- **Mode « écoute » :** il permet d'exécuter Snort en mode renifleur et de surveiller les paquets que le système IDS perçoit ("snort -v")
- **Mode « log de paquets »** : Le journal des paquets permet l'archivage des paquets circulant sur le réseau IDS. Grâce à ses arguments, il permet des opérations intéressantes pour limiter la taille des fichiers logs à certains critères, comme une plage d'adresses IP.
- **Mode « détection d'intrusion »** : Le mode IDS permet d'adopter un certain comportement dans le cas où des chaînes dans les paquets interceptés sont détectées (en cascade). Selon les règles spécifiées dans les fichiers de l'extension "rules".

IV.4. Les règles de Snort

Snort se caractérise par son utilisation d'un langage simple et puissant pour la description des règles. Et lors du développement de ces règles il faut retenir certains nombres de directives simples.

- L'ensemble des règles doit tenir sur une seule ligne.
- Les règles sont divisées en deux parties.

L'en-tête	L'option de règle

Figure IV.2: Structure de base des règles de détection d'intrusions

La structure générale de la règle Snort est représentée comme suit :

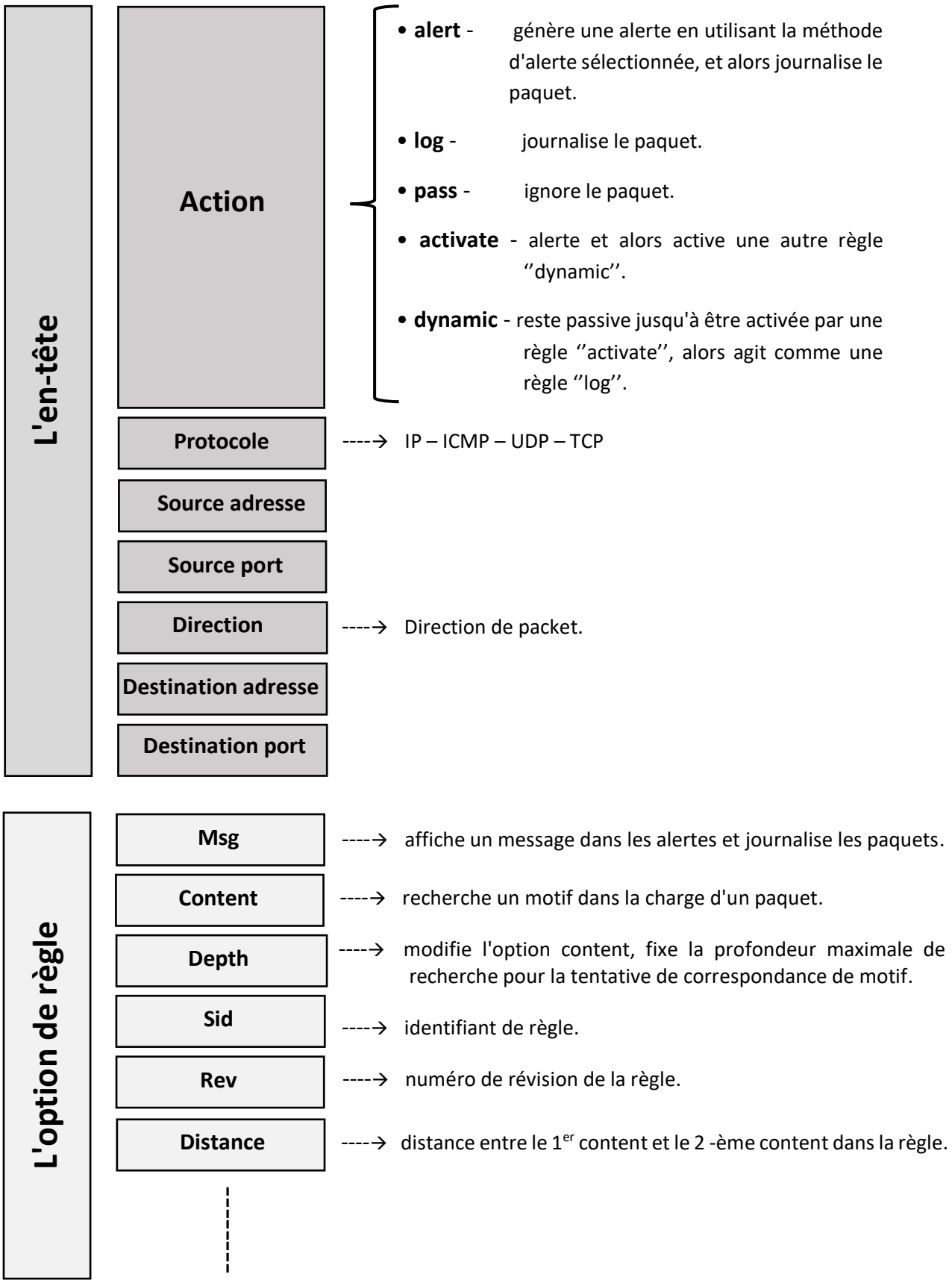


Figure IV.3: Structure générale d'une règle de détection d'intrusions

IV.5. Création des règles à partir des empreintes extraites

La structure de création des règles de détection de ces deux protocoles est basée complètement sur le protocole de transport en premier lieu. Tel mentionné précédemment, chaque protocole est identifié par une méthode différente.

A) A-Mule

La création de la règle dépend du protocole de transport utilisé, TCP ou bien UDP.

Règle N° 01 : E-Donkey-Hello

* La capture de ce paquet est affichée dans la figure suivante :

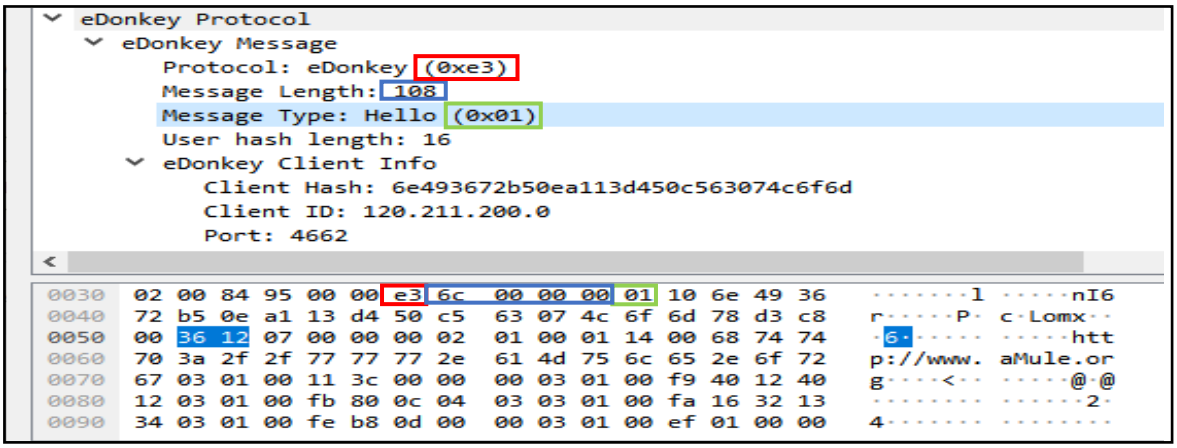


Figure IV.4 : Capture du paquet (E-Donkey Hello)

* L'ID du protocole E-Donkey est « E3 », l'ID du message Type est « 01 » et entre les deux, quatre octets du message **Length** sont présents. Dans les règles d'identifications des paquets du protocole E-Donkey, l'option de distance va être utilisée pour présenter les quatre octets mentionnés précédemment.

ID PROTOCOLE	DISTANCE				ID MESSAGE
XX	Xx	Xx	Xx	Xx	**XX**

Tableau IV-1: structure de la création des règles E-Donkey TCP

*** Donc nous aurons la règle suivante :**

```
alert tcp any any -> any any (msg:"possibilité d'utilisation d'Emule: eDonkey-Hello ";
content:"|E3|"; depth:1; content:"|01|"; depth:1;distance:4; sid:1000000; rev:1;)
```

Règle N° 02 : eDonkey-Hello answer

```
alert tcp any any -> any any (msg:"possibilité d'utilisation d'Emule: eDonkey-Hello answer "; content:"|E3|"; depth:1; content:"|4C|"; depth:1;distance:4; sid:1000001; rev:1;)
```

Règle N° 03 : Emule extensions-Second identification state

```
alert tcp any any -> any any (msg:"possibilité d'utilisation d'Emule: Emule extensions-Second identification state "; content:"|C5|"; depth:1; content:"|87|"; depth:1;distance:4; sid:1000002; rev:1;)
```

Règle N° 04 : Emule extensions-Public key

```
alert tcp any any -> any any (msg:"possibilité d'utilisation d'Emule: Emule extensions-Public key "; content:"|C5|"; depth:1; content:"|85|"; depth:1;distance:4; sid:1000003; rev:1;)
```

Règle N° 05 : Emule extensions-Signature

```
alert tcp any any -> any any (msg:"possibilité d'utilisation d'Emule: Emule extensions-Signature "; content:"|C5|"; depth:1; content:"|86|"; depth:1;distance:4; sid:1000004; rev:1;)
```

Règle N° 06 : Emule extensions-Hello answer

```
alert tcp any any -> any any (msg:"possibilité d'utilisation d'Emule: Emule extensions-Hello answer ";content:"|C5|"; depth:1; content:"|4C|"; depth:1; distance:4; sid:1000005; rev:1;)
```

Règle N° 07 : eDonkey-Get server info

Dans les requêtes UDP on va se focaliser seulement sur la suite des octets de l'ID du protocole et du message, sans besoin d'utiliser l'option de **DISTANCE** dû au fait qu'ils sont déjà concaténés.

```
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: eDonkey-Get server info "; content:"|E3 A2|";depth:2; sid:1000006; rev:1;)
```

Règle N° 08 : eDonkey-Server statut

```
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: eDonkey-Server statut "; content:"|E3 97|";depth:2; sid:1000007; rev:1;)
```

Règle N° 09 : eDonkey-Server Statut req

```
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: eDonkey-Server Statut req  "; content:"|E3 96|";depth:2; sid:1000008; rev:1;)
```

Règle N° 10 : Kademlia-KADEMLIA2_HELLO_REQ

```
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: Kademlia-KADEMLIA2_HELLO_REQ "; content:"|E4 11|";depth:2; sid:1000009; rev:1;)
```

Règle N° 11 : Kademlia-KADEMLIA2_HELLO_RES

```
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: Kademlia-KADEMLIA2_HELLO_RES "; content:"|E4 19|";depth:2; sid:1000010; rev:1;)
```

Règle N° 12 : Kademlia-KADEMLIA2_REQ-FIND NODE

```
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: Kademlia-KADEMLIA2_REQ-FIND NODE "; content:"|E4 21 0B|";depth:3; sid:1000011; rev:1;)
```

Règle N° 13 : Kademlia-KADEMLIA2_REQ-FIND VALUE

```
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: Kademlia-KADEMLIA2_REQ-FIND VALUE "; content:"|E4 21 02|";depth:3; sid:1000012; rev:1;)
```

Règle N° 14 : Kademlia-KADEMLIA2_RES

```
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: Kademlia-KADEMLIA2_RES "; content:"|E4 29|";depth:2; sid:1000013; rev:1;)
```

Règle N° 15 : Kademlia-KADEMLIA2_REQ

```
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: Kademlia-KADEMLIA2_REQ"; content:"|E4 21|";depth:2; sid:1000014; rev:1;)
```

Règle N° 16 : Kademlia-KADEMLIA_FINDBUDDY_REQ

```
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: Kademlia-KADEMLIA_FINDBUDDY_REQ "; content:"|E4 51|";depth:2; sid:1000015; rev:1;)
```

Règle N° 17 : eDonkey-Get sources

```
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: eDonkey-Get sources"; content:"|E3 9A|";depth:2; sid:1000016; rev:1;)
```

Règle N° 18 : eDonkey-Search File

```
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: eDonkey-Search File"; content:"|E3 98|";depth:2; sid:1000017; rev:1;)
```

Règle N° 19 : eDonkey-Search File Results

```
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: eDonkey-Search File Results"; content:"|E3 99|";depth:2; sid:1000018; rev:1;)
```

Règle N° 20 : Reask File Ping

```
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: Reask File Ping "; content:"|E3 90|";depth:2; sid:1000019; rev:1;)
```

a) µTorrent

Règle N° 01 : handshake

Les règles de détection de ce protocole dépendent seulement du contenu, ou la structure des règles répétée est la même mais le contenu est différent.

```
alert tcp any any -> any any (msg: "Possibilité d'utilisation de uTorrent: handshake "; content:"BitTorrent protocol"; sid: 200001 ; rev:1;)
```

Règle N° 02 : handshake extension

```
alert tcp any any -> any any (msg: "Possibilité d'utilisation de uTorrent: handshake extended "; content:"ut_metadata"; content:"metadata_size" ; sid: 2000002 ; rev:1;)
```

Règle N° 03 : Get torrent

```
alert tcp any any -> any any (msg: "Possibilité d'utilisation de uTorrent: Get torrent ";
content:"GET /torrent"; sid: 2000003 ; rev:1;)
```

Règle N° 04 : get scrape

```
alert tcp any any -> any any (msg: "Possibilité d'utilisation de uTorrent: get scrape ";
content:"GET /scrape?info_hash" ;content:"User-Agent: uTorrent"; sid:2000004 ; rev:1;)
```

Règle N° 05 : get anounce?info_hash

```
alert tcp any any -> any any (msg: "Possibilité d'utilisation de uTorrent: get anounce";
content:"GET /announce?info_hash"; content:"User-Agent: uTorrent";sid:2000005 ; rev:1;)
```

Règle N° 06 : DHT Ping Peer

```
alert udp any any -> any any (msg: "Possibilité d'utilisation de uTorrent: DHT Ping Peer ";
content:"d1:ad2:id20"; content:"info_hash20";content:"get_peers1"; sid:2000006 ; rev:1;)
```

Règle N° 07 : DHT Ping Tracker

```
alert udp any any -> any any (msg: "Possibilité d'utilisation de uTorrent: DHT Ping Tracker ";
content:"/announce"; sid:2000007 ; rev:1;)
```

- Par la fin, **20 règles** pour E-Donkey et **07 règles** pour BitTorrent ont été créés. Il nous reste donc seulement l'implémentation de ces derniers au niveau de l'IDS Snort.

IV.6. Implémentation des signatures d'applications

- L'implémentation des signatures se fait directement par la copie de ces différentes règles au niveau du fichier au niveau du fichier « local.rules.txt ».

- Le chemin d'emplacement de ce fichier est :

C:\Snort1\rules\local.rules

```
#------------Edonkey rules
alert tcp any any -> any any (msg:"possibilité d'utilisation d'Emule: eDonkey-Hello "; content:"|E3|"; depth:1; content:"|01|"; depth:1;distance:4; sid:1000000; rev:1;)
alert tcp any any -> any any (msg:"possibilité d'utilisation d'Emule: eDonkey-Hello answer "; content:"|E3|"; depth:1; content:"|4C|"; depth:1;distance:4; sid:1000001; rev:1;)
alert tcp any any -> any any (msg:"possibilité d'utilisation d'Emule: Emule extensions-Second identification state "; content:"|C5|"; depth:1; content:"|87|"; depth:1;distance:4; sid:1
alert tcp any any -> any any (msg:"possibilité d'utilisation d'Emule: Emule extensions-Public key "; content:"|C5|"; depth:1; content:"|85|"; depth:1;distance:4; sid:1000003; rev:1;)
alert tcp any any -> any any (msg:"possibilité d'utilisation d'Emule: Emule extensions-Signature "; content:"|C5|"; depth:1; content:"|86|"; depth:1;distance:4; sid:1000004; rev:1;)
alert tcp any any -> any any (msg:"possibilité d'utilisation d'Emule: Emule extensions-Hello answer ";content:"|C5|"; depth:1; content:"|4C|"; depth:1; distance:4; sid:1000005; rev:1;)
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: eDonkey-Get server info "; content:"|E3 A2|";depth:2; sid:1000006; rev:1;)
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: eDonkey-Server statut "; content:"|E3 97|";depth:2; sid:1000007; rev:1;)
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: eDonkey-Server Statut req  "; content:"|E3 96|";depth:2; sid:1000008; rev:1;)
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: Kademlia-KADEMLIA2_HELLO_REQ "; content:"|E4 11|";depth:2; sid:1000009; rev:1;)
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: Kademlia-KADEMLIA2_HELLO_RES "; content:"|E4 19|";depth:2; sid:1000010; rev:1;)
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: Kademlia-KADEMLIA2_REQ-FIND NODE "; content:"|E4 21 0B|";depth:3; sid:1000011; rev:1;)
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: Kademlia-KADEMLIA2_REQ-FIND VALUE "; content:"|E4 21 02|";depth:3; sid:1000012; rev:1;)
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: Kademlia-KADEMLIA2_RES "; content:"|E4 29|";depth:2; sid:1000013; rev:1;)
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: Kademlia-KADEMLIA2_REQ"; content:"|E4 21|";depth:2; sid:1000014; rev:1;)
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: Kademlia-KADEMLIA_FINDBUDDY_REQ "; content:"|E4 51|";depth:2; sid:1000015; rev:1;)
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: eDonkey-Get sources"; content:"|E3 9A|";depth:2; sid:1000016; rev:1;)
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: eDonkey-Search File"; content:"|E3 98|";depth:2; sid:1000017; rev:1;)
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: eDonkey-Search File Results"; content:"|E3 99|";depth:2; sid:1000018; rev:1;)
alert udp any any -> any any (msg:"possibilité d'utilisation d'Emule: eDonkey-Reask File ping"; content:"|E3 90|";depth:2; sid:1000019; rev:1;)
```

Figure IV.5: L'implémentation des règles A-Mule dans le fichier local.rules

```
#------------Bittorrent rules
alert tcp any any -> any any (msg: "Possibilité d'utilisation de uTorrent: handshake "; content:"BitTorrent protocol"; sid:100000001 ; rev:1;)

alert tcp any any -> any any (msg: "Possibilité d'utilisation de uTorrent: handshake extended "; content:"ut_metadata"; content:"metadata_size" ; sid:100000002 ; rev:1;)

alert tcp any any -> any any (msg: "Possibilité d'utilisation de uTorrent: Get torrent "; content:"GET /torrent"; sid:100000003 ; rev:1;)

alert tcp any any -> any any (msg: "Possibilité d'utilisation de uTorrent: get scrape "; content:"GET /scrape?info_hash" ;content:"User-Agent: uTorrent"; sid:100000004 ; rev:1;)

alert tcp any any -> any any (msg: "Possibilité d'utilisation de uTorrent: get anounce"; content:"GET /announce?info_hash"; content:"User-Agent: uTorrent";sid:100000005 ; rev:1;)

alert udp any any -> any any (msg: "Possibilité d'utilisation de uTorrent: DHT Ping Peer "; content:"d1:ad2:id20"; content:"info_hash20";content:"get_peers1"; sid:100000006 ; rev:1;)

alert udp any any -> any any (msg: "Possibilité d'utilisation de uTorrent: DHT Ping Tracker "; content:"/announce"; sid:100000007 ; rev:1;)
```

Figure IV.6: L'implémentation des règles µTorrent dans le fichier local.rules

IV.7. Test de fonctionnement

Pour tester le fonctionnement de nos règles, on va tout d'abord lancer Snort en utilisant la commande suivante :

```
C:\Users\PFE>cd ..

C:\Users>cd ..

C:\>cd snort1

C:\Snort1>cd bin

C:\Snort1\bin>snort -i1 -c c:\snort1\etc\snort.conf -l c:\snort1\log -K ascii -i
```

```
        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.8.6-ODBC-MySQL-FlexRESP-WIN32 GRE (Build 38)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/snort/snort-team
           Copyright (C) 1998-2010 Sourcefire, Inc., et al.
           Using PCRE version: 7.4 2007-09-21
           Using ZLIB version: 1.2.3

           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 1.12  <Build 18>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_DCERPC  Version 1.1  <Build 5>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
Not Using PCAP_FRAMES
```

Figure IV.7: Lancement de Snort - CMD

- Une fois le message « **Not Using PCAP-FRAMES** » est affiché, Snort sera en mode fonctionnement.
- Après le lancement de Snort, il faut lancer l'interface graphique « Base » pour visualiser les alertes en temps réel.

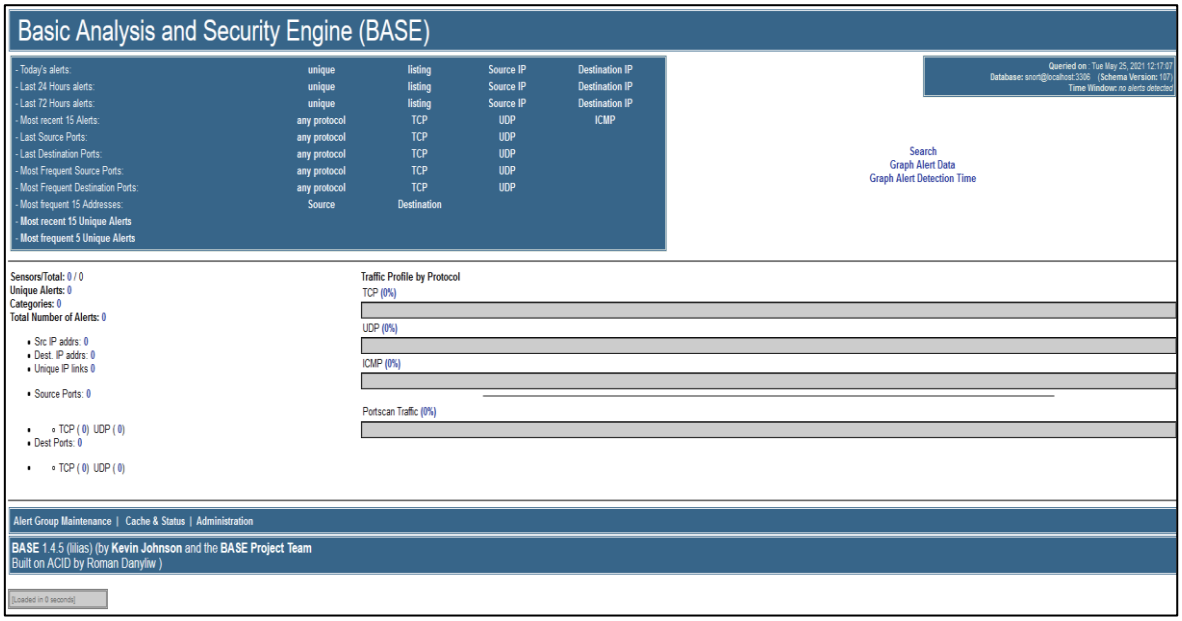


Figure IV.8: Interface graphique BASE

- Pour cela on va effectuer un test au niveau de notre réseau laboratoire qui est constitué de trois clients et un serveur, en activant la détection Snort sur le trafic des clients venant de A-Mule et µTorrent, Snort nous donne la main de consulter toutes les statistiques du réseau comme le nombre total des paquets traités, les paquets rejetés, les paquets retransmirent, ainsi que le temps total de fonctionnement.

```
*** Caught Int-Signal
Run time prior to being shutdown was 879.285000 seconds
database: Closing connection to database "snort"
===============================================================================
Packet Wire Totals:
   Received:       217874
   Analyzed:       217821 (99.976%)
    Dropped:           50 (0.023%)
Outstanding:            3 (0.001%)
```

Figure IV.9: Statistiques des paquets analysés

01- A-Mule

Une fois que le client utilise A-Mule, Snort lance des alertes UDP/TCP. Ces alertes sont représentées dans la figure suivante :

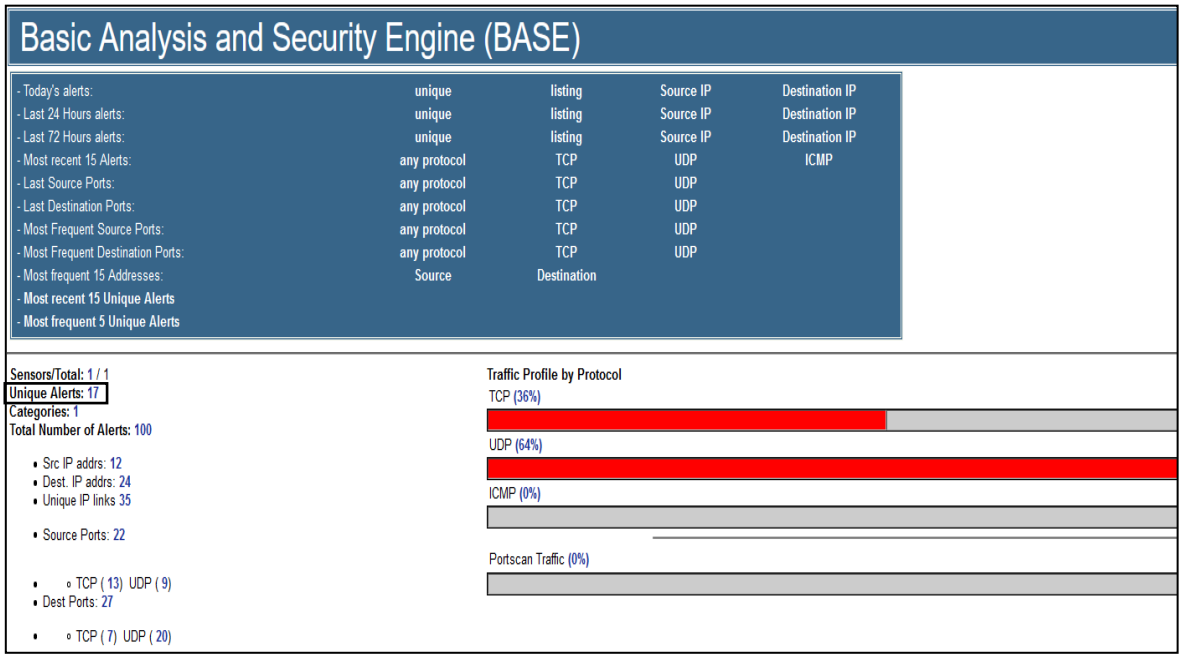
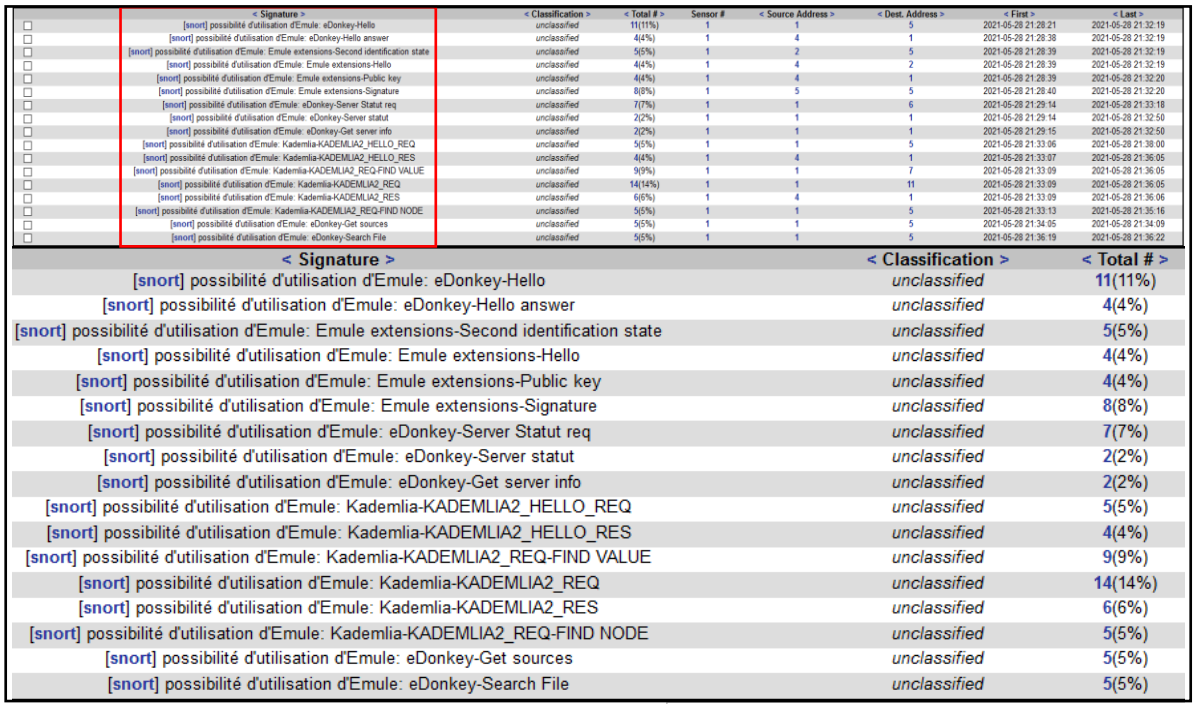


Figure IV.10: Détection de client A-Mule

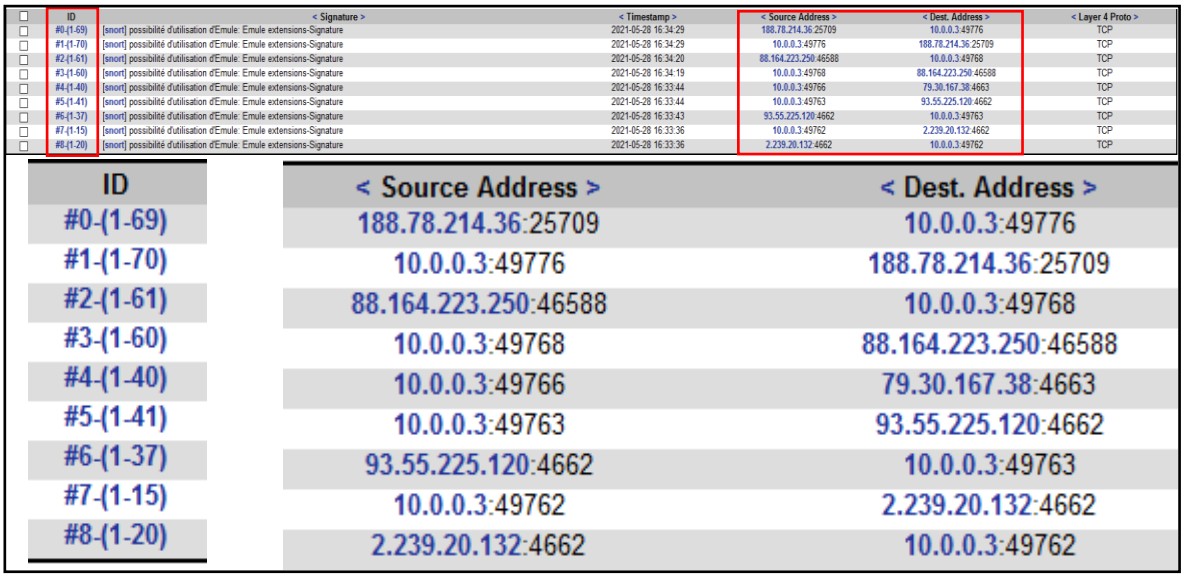
Le lien « **Unique alerts** » nous permet de consulter ces règles comme affiché dans la figure (IV.11) :



< Signature >	< Classification >	< Total # >
[snort] possibilité d'utilisation d'Emule: eDonkey-Hello	unclassified	11(11%)
[snort] possibilité d'utilisation d'Emule: eDonkey-Hello answer	unclassified	4(4%)
[snort] possibilité d'utilisation d'Emule: Emule extensions-Second identification state	unclassified	5(5%)
[snort] possibilité d'utilisation d'Emule: Emule extensions-Hello	unclassified	4(4%)
[snort] possibilité d'utilisation d'Emule: Emule extensions-Public key	unclassified	4(4%)
[snort] possibilité d'utilisation d'Emule: Emule extensions-Signature	unclassified	8(8%)
[snort] possibilité d'utilisation d'Emule: eDonkey-Server Statut req	unclassified	7(7%)
[snort] possibilité d'utilisation d'Emule: eDonkey-Server statut	unclassified	2(2%)
[snort] possibilité d'utilisation d'Emule: eDonkey-Get server info	unclassified	2(2%)
[snort] possibilité d'utilisation d'Emule: Kademlia-KADEMLIA2_HELLO_REQ	unclassified	5(5%)
[snort] possibilité d'utilisation d'Emule: Kademlia-KADEMLIA2_HELLO_RES	unclassified	4(4%)
[snort] possibilité d'utilisation d'Emule: Kademlia-KADEMLIA2_REQ-FIND VALUE	unclassified	9(9%)
[snort] possibilité d'utilisation d'Emule: Kademlia-KADEMLIA2_REQ	unclassified	14(14%)
[snort] possibilité d'utilisation d'Emule: Kademlia-KADEMLIA2_RES	unclassified	6(6%)
[snort] possibilité d'utilisation d'Emule: Kademlia-KADEMLIA2_REQ-FIND NODE	unclassified	5(5%)
[snort] possibilité d'utilisation d'Emule: eDonkey-Get sources	unclassified	5(5%)
[snort] possibilité d'utilisation d'Emule: eDonkey-Search File	unclassified	5(5%)

Figure IV.11: Liste des alertes lancées par Snort – Client A-Mule

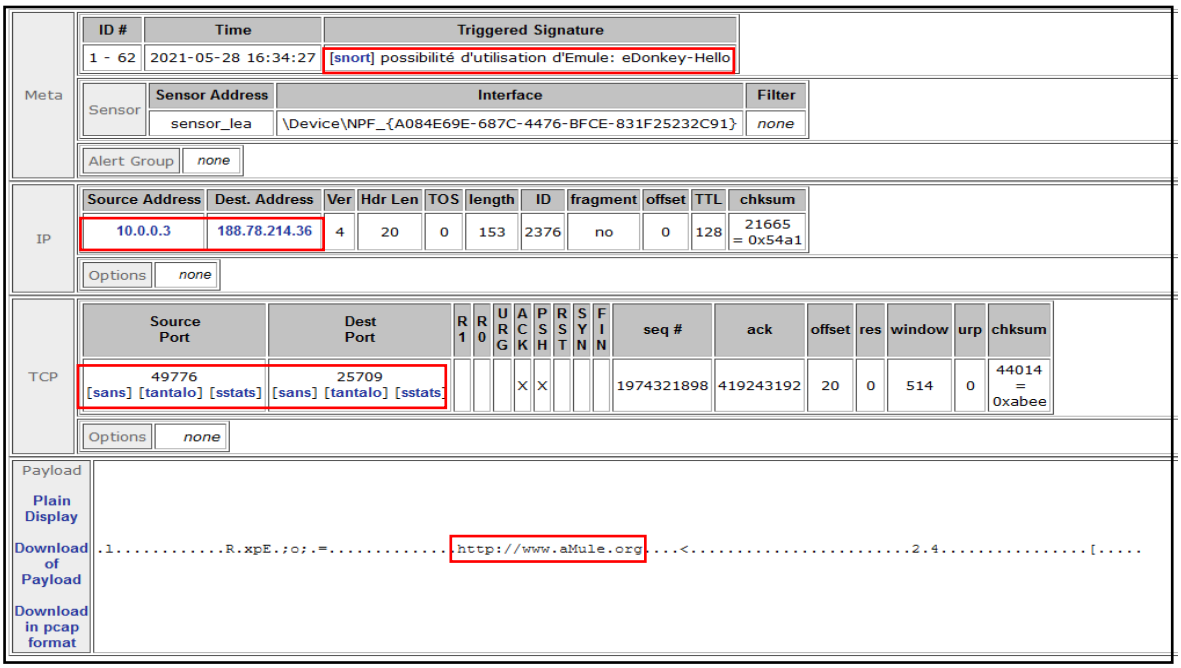
- On peut voir des informations détaillées sur les alertes de chaque règle dans la colonne « **Total** », comme suit :



ID	< Source Address >	< Dest. Address >
#0-(1-69)	188.78.214.36:25709	10.0.0.3:49776
#1-(1-70)	10.0.0.3:49776	188.78.214.36:25709
#2-(1-61)	88.164.223.250:46588	10.0.0.3:49768
#3-(1-60)	10.0.0.3:49768	88.164.223.250:46588
#4-(1-40)	10.0.0.3:49766	79.30.167.38:4663
#5-(1-41)	10.0.0.3:49763	93.55.225.120:4662
#6-(1-37)	93.55.225.120:4662	10.0.0.3:49763
#7-(1-15)	10.0.0.3:49762	2.239.20.132:4662
#8-(1-20)	2.239.20.132:4662	10.0.0.3:49762

Figure IV.12: Alertes lancées par la règle « edonkey-Hello »

- Cette capture nous montre clairement les adresses IP et les Ports (Source et destination), le temps de déclenchement d'alerte, le protocole et la signature. En cliquant sur l'ID on aura des informations détaillées qui concernent le paquet capturé.
- On peut même consulter le contenu des paquets, et voir la correspondance du contenu avec la règle :



Meta

ID #	Time	Triggered Signature
1 - 62	2021-05-28 16:34:27	[snort] possibilité d'utilisation d'Emule: eDonkey-Hello

Sensor	Sensor Address	Interface	Filter
	sensor_lea	\Device\NPF_{A084E69E-687C-4476-BFCE-831F25232C91}	none

Alert Group: none

IP

Source Address	Dest. Address	Ver	Hdr Len	TOS	length	ID	fragment	offset	TTL	chksum
10.0.0.3	188.78.214.36	4	20	0	153	2376	no	0	128	21665 = 0x54a1

Options: none

TCP

Source Port	Dest Port	R1	R0	URG	ACK	PSH	RST	SYN	FIN	seq #	ack	offset	res	window	urp	chksum
49776 [sans] [tantalo] [sstats]	25709 [sans] [tantalo] [sstats]				X	X				1974321898	419243192	20	0	514	0	44014 = 0xabee

Options: none

Payload

Plain Display | Download of Payload | Download in pcap format

.l.....R.xpE.;o;.=.....http://www.aMule.org....<.....2.4.....[.....

Figure IV.13: Contenu du paquet edonkey Hello – Snort

- Après la consultation du contenu, on peut confirmer que ce trafic venant de l'application A-Mule par le nom du domaine « https://www.aMule.org » est l'indice qui prouve la source du trafic, et confirme aussi la fiabilité de nos règles.
- Les alertes générées par chaque règle sont présentées comme suit :

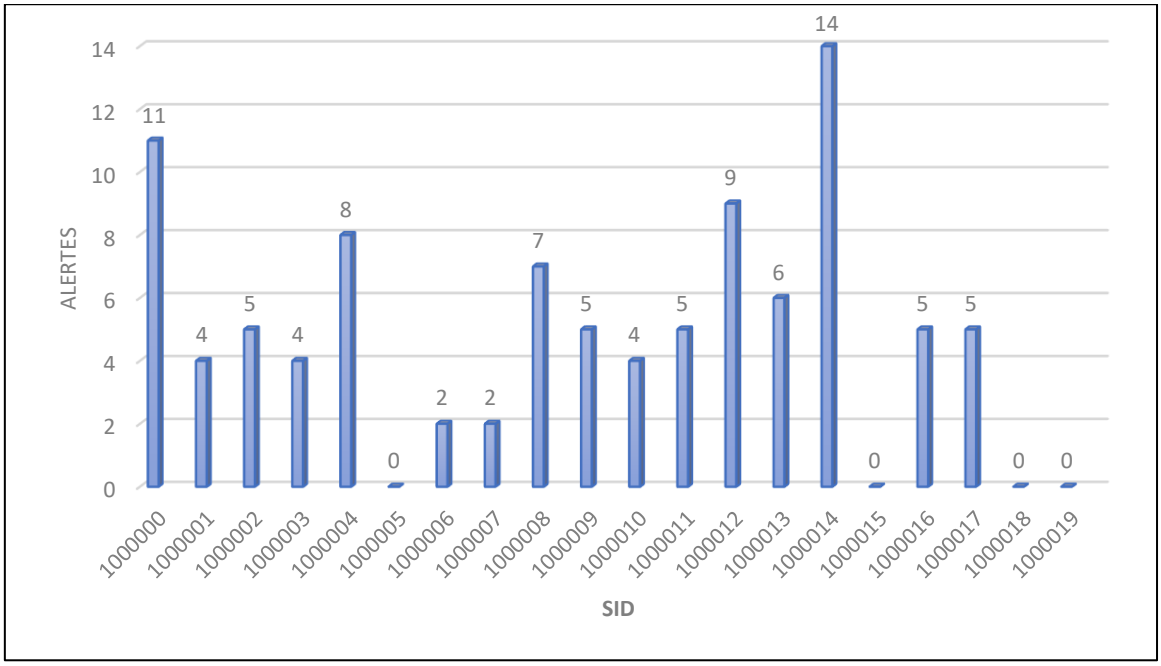


Figure IV.14:Colonnes graphiques des résultats de détection E-Donkey

Remarques :

- On remarque que Snort a détecté l'utilisation de l'application A-Mule, ou il a lancé 100 alertes qui correspondent à nos règles.
- Les alertes sont lancées en temps réel.
- On peut noter l'absence de 4 règles, qui sont basées sur des identifiants des messages de fonctionnement E-Donkey supplémentaire.
- D'après la figure (IV.14) des alertes, on note que la présence du protocole KADMELIA est clair. Un grand nombre de requêtes capturées sont des requêtes KAD dont le client A-Mule se base principalement pour la recherche et le téléchargement des fichiers.

02- µTorrent :

- Une fois que le client lance le téléchargement torrent, l'interface graphique de Snort affiche ces alertes :

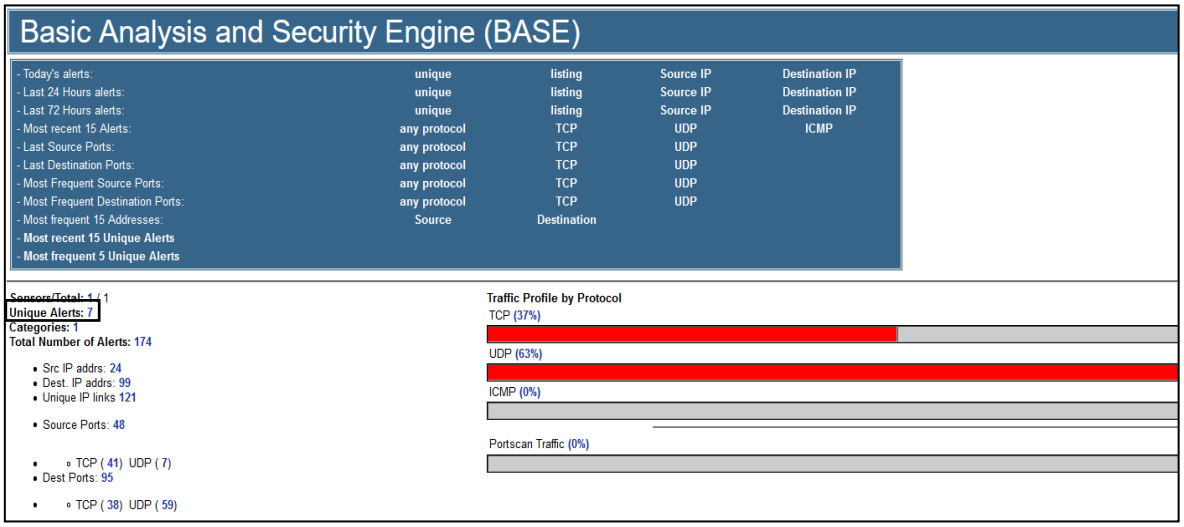
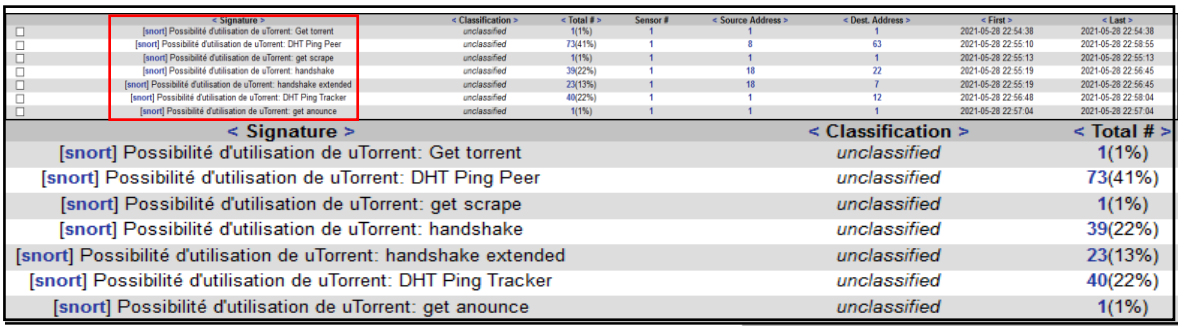


Figure IV.15: Détection de client µTorrent

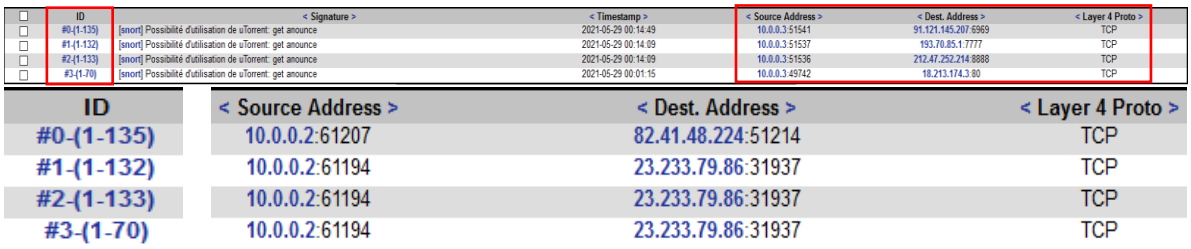
- Si on clique sur le lien « Unique alerte » on obtient les règles qui génèrent toutes les alertes, comme mentionné dans la figure (IV.16) :



< Signature >	< Classification >	< Total # >
[snort] Possibilité d'utilisation de uTorrent: Get torrent	unclassified	1(1%)
[snort] Possibilité d'utilisation de uTorrent: DHT Ping Peer	unclassified	73(41%)
[snort] Possibilité d'utilisation de uTorrent: get scrape	unclassified	1(1%)
[snort] Possibilité d'utilisation de uTorrent: handshake	unclassified	39(22%)
[snort] Possibilité d'utilisation de uTorrent: handshake extended	unclassified	23(13%)
[snort] Possibilité d'utilisation de uTorrent: DHT Ping Tracker	unclassified	40(22%)
[snort] Possibilité d'utilisation de uTorrent: get anounce	unclassified	1(1%)

Figure IV.16:Liste des alertes lancée par Snort – Client µTorrent

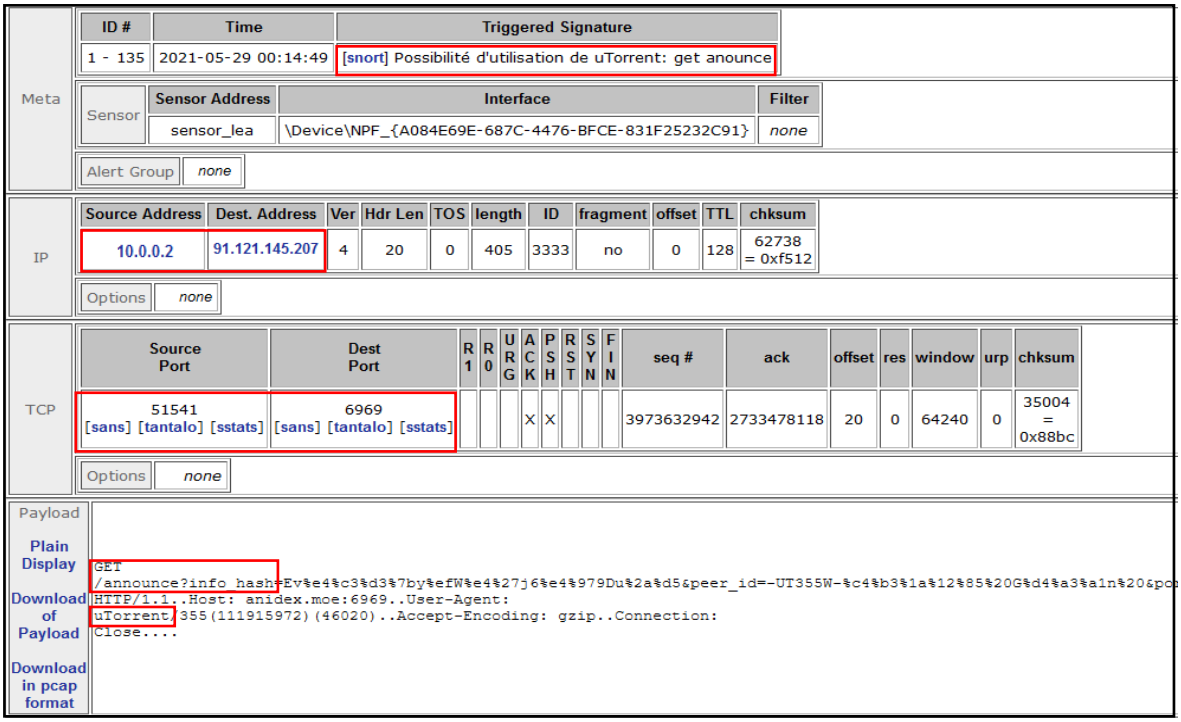
- Chaque signature lance au minimum une alerte, ce qui veut dire que toutes les règles sont juste. En cliquant sur le nombre total des alertes lancées par la règle « get announce» on trouve :



ID	< Source Address >	< Dest. Address >	< Layer 4 Proto >
#0-(1-135)	10.0.0.2:61207	82.41.48.224:51214	TCP
#1-(1-132)	10.0.0.2:61194	23.233.79.86:31937	TCP
#2-(1-133)	10.0.0.2:61194	23.233.79.86:31937	TCP
#3-(1-70)	10.0.0.2:61194	23.233.79.86:31937	TCP

Figure IV.17:Alertes lancées par la règle « get announce »

- On clique sur l'ID de l'alerte, on trouve :



Meta	ID #	Time	Triggered Signature
	1 - 135	2021-05-29 00:14:49	[snort] Possibilité d'utilisation de uTorrent: get anounce

Sensor	Sensor Address	Interface	Filter
	sensor_lea	\Device\NPF_{A084E69E-687C-4476-BFCE-831F25232C91}	none

Alert Group: none

IP	Source Address	Dest. Address	Ver	Hdr Len	TOS	length	ID	fragment	offset	TTL	chksum
	10.0.0.2	91.121.145.207	4	20	0	405	3333	no	0	128	62738 = 0xf512

Options: none

TCP	Source Port	Dest Port	R 1	R 0	U R G	A C K	P S H	R S T	S Y N	F I N	seq #	ack	offset	res	window	urp	chksum
	51541 [sans] [tantalo] [sstats]	6969 [sans] [tantalo] [sstats]				X	X				3973632942	2733478118	20	0	64240	0	35004 = 0x88bc

Options: none

Payload

Plain Display

Download of Payload

Download in pcap format

```
GET
/announce?info_hash=Ev%e4%c3%d3%7by%efW%e4%27j6%e4%979Du%2a%d5&peer_id=-UT355W-%c4%b3%1a%12%85%20G%d4%a3%a1n%20&po
HTTP/1.1..Host: anidex.moe:6969..User-Agent:
uTorrent/355(111915972)(46020)..Accept-Encoding: gzip..Connection:
Close....
```

Figure IV.18: Contenu de requête get announce– Snort

- Cette requête http contient des informations concernant le tracker, l'application client, l'ID du fichier (voir chapitre III). Toutes ces informations sont clairement visibles dans la case du contenu apparent dans la figure (IV.18). Cela nous permet de dire que ces alertes correspondent au trafic torrent.
- Toutes les alertes générées sont présentées dans la figure suivante :

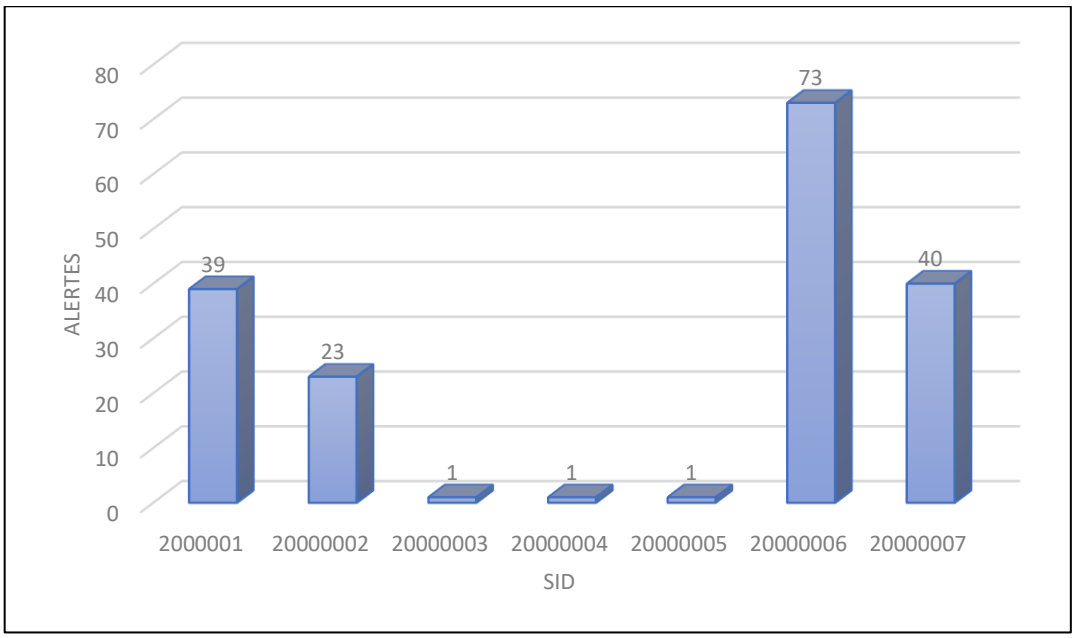


Figure IV.19:Colonnes graphiques des résultats de détection µTorrent

Remarques :

- On remarque que Snort a détecté l'utilisation de l'application µTorrent ou il a lancé 174 alertes qui correspondent à nos règles.
- On remarque aussi que le nombre d'alertes générées par la règle « DHT Ping Peer » est très grand et cela est totalement normal parce que cette règle génère une alerte sur chaque requête envoyer vers les Peers.
- Toutes nos règles déclenchent des alertes qui sont représentées dans la figure (IV.19).
- Le fonctionnement du Protocole BitTorrent se base complétement sur la requête « BitTorrent Handshake ».
- Le Protocole BitTorrent suit une structure de fonctionnement basé en premier lieu sur le protocole http.

Note : jusqu'à présent le test est effectué en mode clair sans aucun cryptage.

02.A- µTorrent (Mode crypté) :

On va refaire la même procédure de téléchargement, mais cette fois-ci avec l'activation du mode crypté, les alertes affichées par Snort sont :

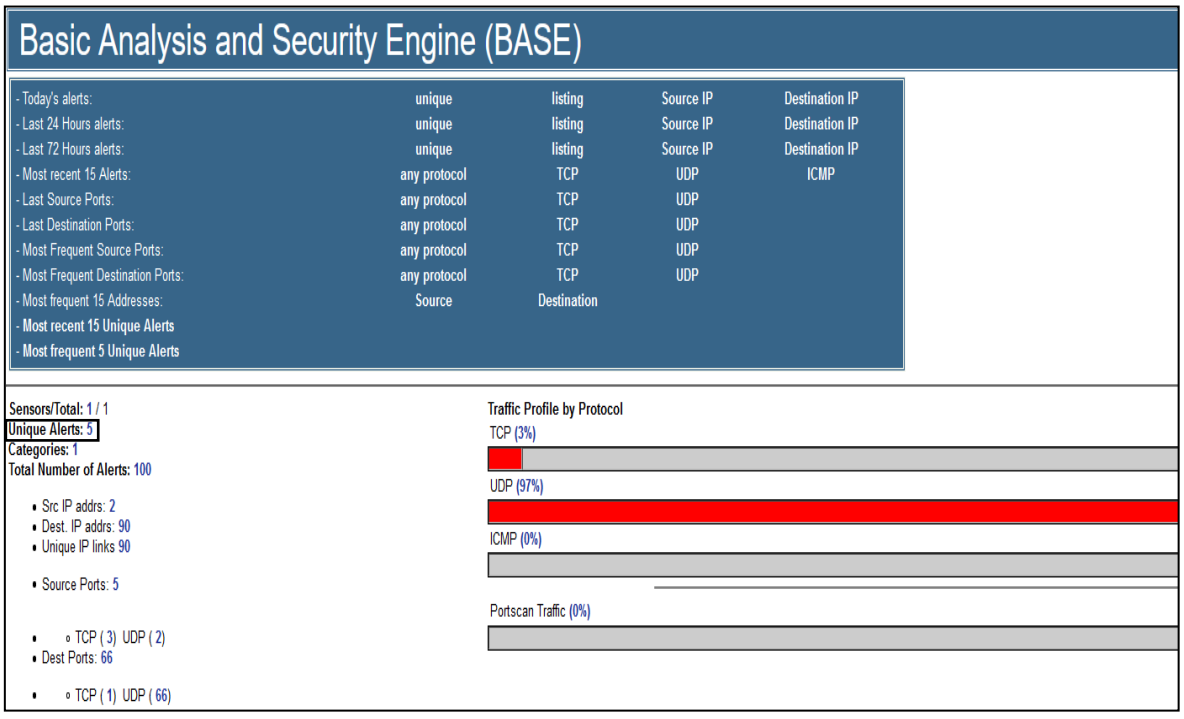
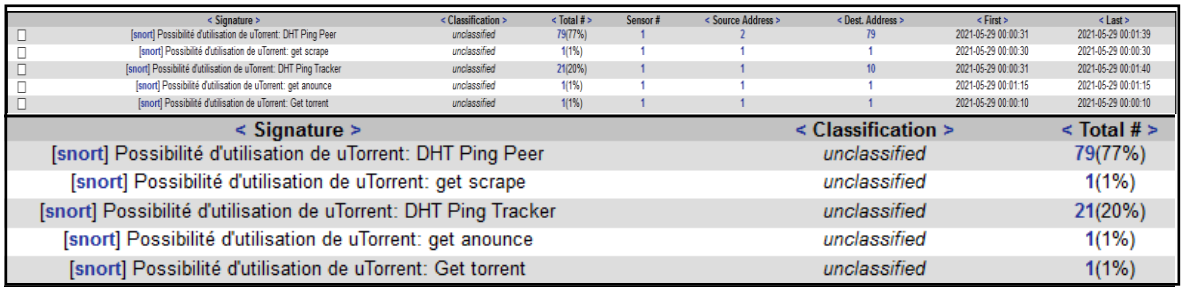


Figure IV.20: Détection du client µTorrent - Mode crypté

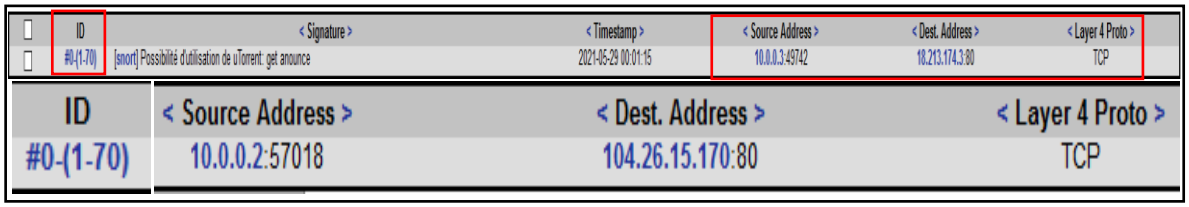
- Snort détecte l'utilisation de l'application µTorrent en mode crypté, il lance 100 alertes qui correspondent à nos 5 règles.
- Le lien « **Unique alerts** » nous permet de consulter les 5 alertes lancées par Snort comme affiché dans la figure (IV.21) :



< Signature >	< Classification >	< Total # >
[snort] Possibilité d'utilisation de uTorrent: DHT Ping Peer	unclassified	79(77%)
[snort] Possibilité d'utilisation de uTorrent: get scrape	unclassified	1(1%)
[snort] Possibilité d'utilisation de uTorrent: DHT Ping Tracker	unclassified	21(20%)
[snort] Possibilité d'utilisation de uTorrent: get announce	unclassified	1(1%)
[snort] Possibilité d'utilisation de uTorrent: Get torrent	unclassified	1(1%)

Figure IV.21: Liste des alertes lancées par Snort – Client µTorrent

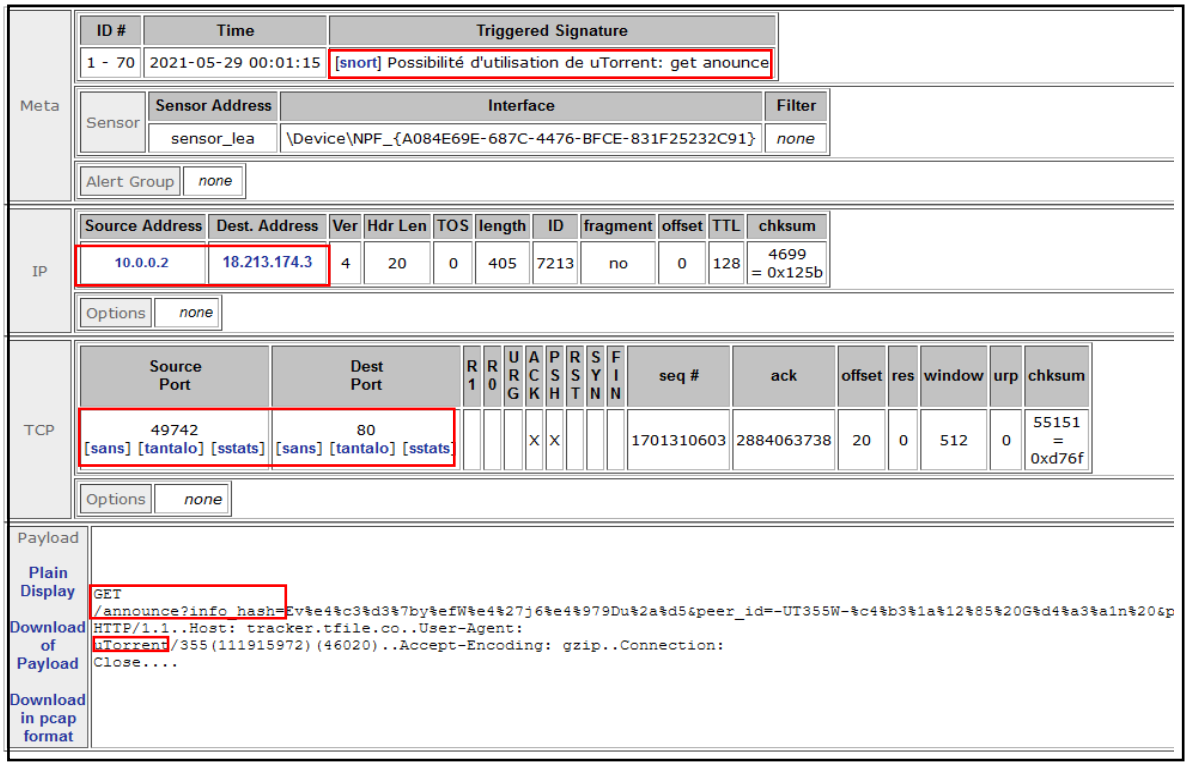
- En cliquant sur le nombre total des alertes lancées par la règle « **get announce** » on trouve :



ID	< Signature >	< Timestamp >	< Source Address >	< Dest. Address >	< Layer 4 Proto >
#0-(1-70)	[snort] Possibilité d'utilisation de uTorrent: get announce	2021-05-29 00:01:15	10.0.0.2:57018	104.26.15.170:80	TCP

Figure IV.22: Les alertes lancés par la règle « get announce »

- En cliquant sur l'ID de l'alerte, on trouve :



Meta

ID #	Time	Triggered Signature
1 - 70	2021-05-29 00:01:15	[snort] Possibilité d'utilisation de uTorrent: get announce

Sensor	Sensor Address	Interface	Filter
	sensor_lea	\Device\NPF_{A084E69E-687C-4476-BFCE-831F25232C91}	none

Alert Group: none

IP

Source Address	Dest. Address	Ver	Hdr Len	TOS	length	ID	fragment	offset	TTL	chksum
10.0.0.2	18.213.174.3	4	20	0	405	7213	no	0	128	4699 = 0x125b

Options: none

TCP

Source Port	Dest Port	R 1	R 0	U R G	A C K	P S H	R S T	S Y N	F I N	seq #	ack	offset	res	window	urp	chksum
49742 [sans] [tantalo] [sstats]	80 [sans] [tantalo] [sstats]				X	X				1701310603	2884063738	20	0	512	0	55151 = 0xd76f

Options: none

Payload

Plain Display | Download of Payload | Download in pcap format

```
GET
/announce?info_hash=Ev%e4%c3%d3%7by%efW%e4%27j6%e4%979Du%2a%d5&peer_id=-UT355W-%c4%b3%1a%12%85%20G%d4%a3%a1n%20&p
HTTP/1.1..Host: tracker.tfile.co..User-Agent:
uTorrent/355(111915972)(46020)..Accept-Encoding: gzip..Connection:
Close....
```

Figure IV.23:Contenu du paquet http GET announce – Snort

- En trouvant les informations du client BitTorrent, une garantie de trafic venant de l'application µTorrent est assuré.

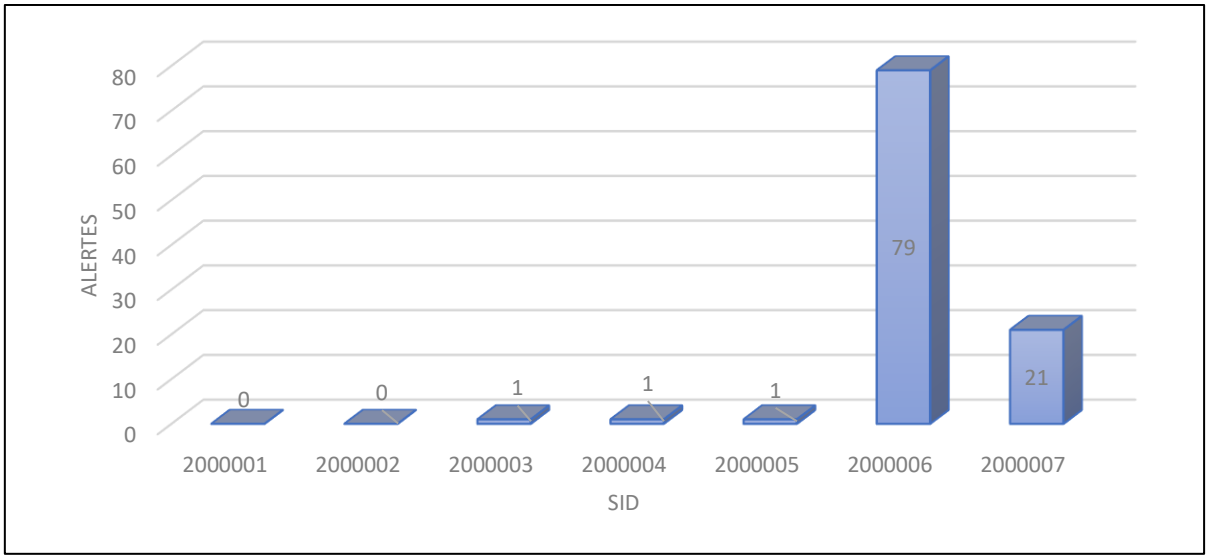


Figure IV.24: Colonnes graphiques des résultats de détection µTorrent crypté

- On peut noter clairement l'absence des alertes déclenchées par les deux règles « Handshake » « Handshake Extended » du protocole BitTorrent à cause du cryptage.

IV.8. Test de fiabilité

- Après la confirmation du fonctionnement de nos règles, on doit maintenant confirmer leurs fiabilités. Pour cela, deux aspects principaux de la détection sont évoqués :
    - Le faux négatif : L'IDS ne détecte pas l'intrusion.
    - Le faux positif : L'IDS lance l'alerte sans aucune intrusion réelle.
- Notre intérêt à nous se dirige vers le deuxième aspect, où on va effectuer le test principal dans un réseau de simulation de réseau d'entreprise pendant 4 jours, en mettant Snort en fonctionnement sur l'interface publique du serveur proxy qui capte le trafic venant des clients en temps réel. On pourra donc consulter et suivre les résultats de cette expérience à l'aide de l'interface BASE.
- L'architecture du test est montrée dans la figure (IV.25) ci-dessous :

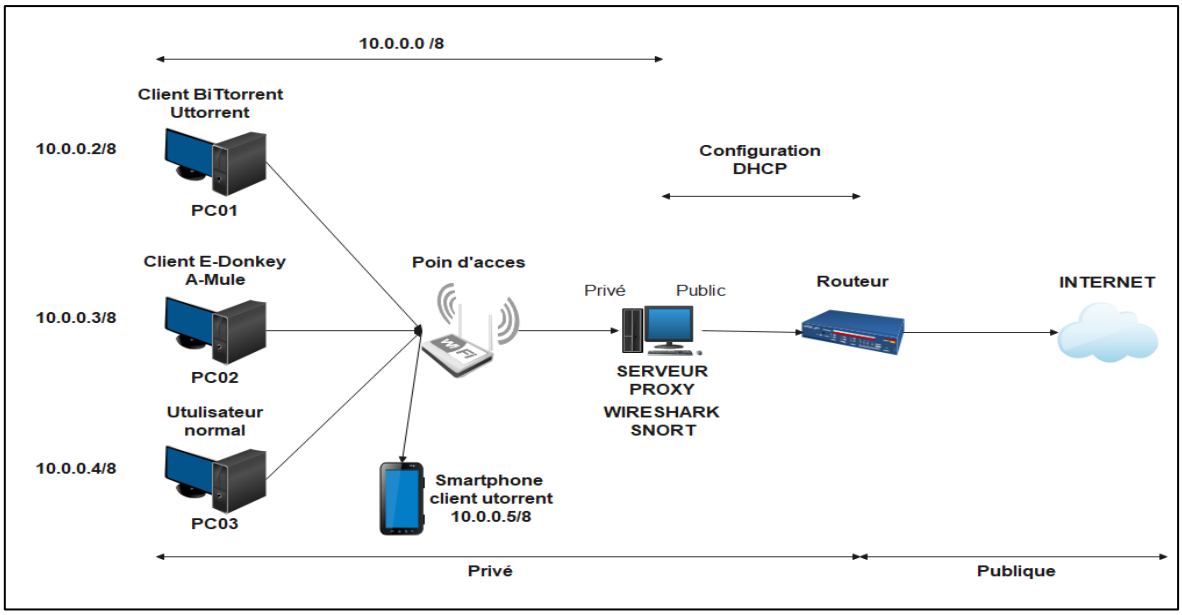


Figure IV.25: Architecture du réseau d'entreprise

- Pour garantir la fiabilité des règles, un test sans aucune utilisation des clients Peer to Peer dans le réseau n'est réalisé. Les résultats se présentent comme ceci :

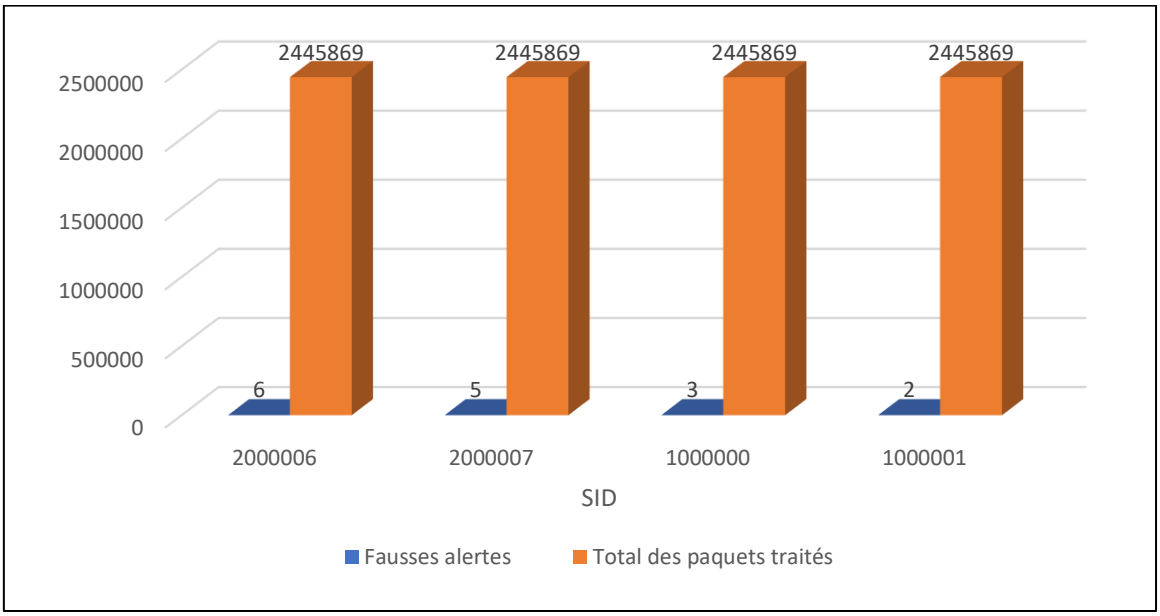


Figure IV.26: Colonnes graphiques des résultats des fausses alertes

- Taux acceptable remarqué dans la figure IV.26 où quatre règles seulement parmi les vingt-sept ont généré des fausses alertes.
- Fausses alertes causées par la similitude de contenu, ainsi que le taux d'erreur de détection du Snort.

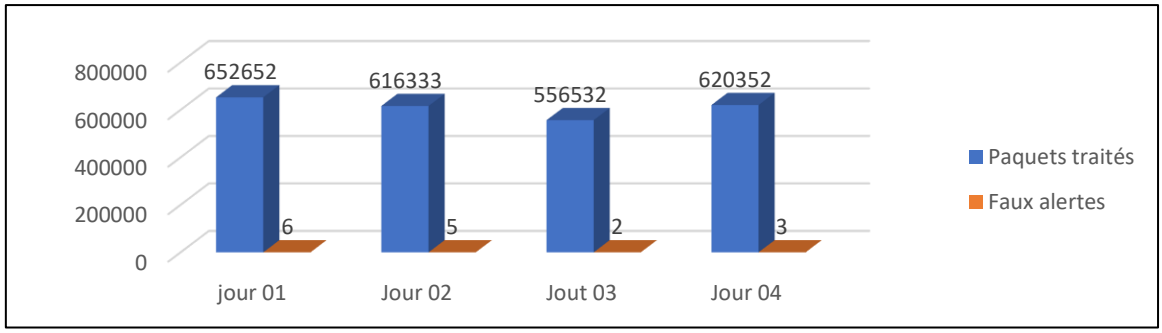


Figure IV.27: Colonnes graphiques des résultats – test réel

- A travers les résultats fournis précédemment, on va classer chacune de ces règles selon leur taux de fiabilité comme montrer ci-dessous :

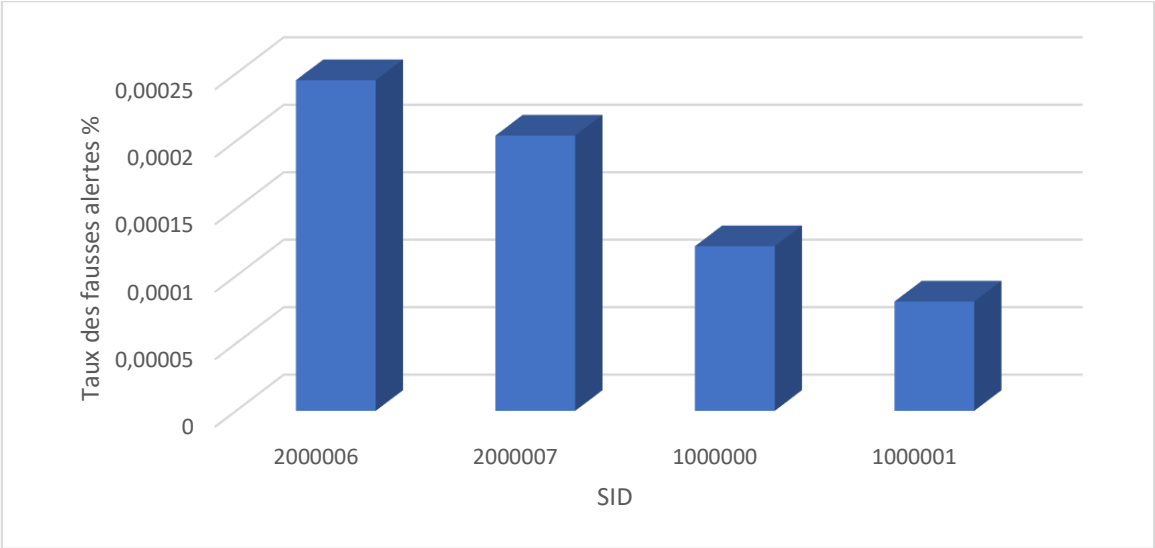


Figure IV.28 : Classification des règles - fausses alertes

- On peut clairement constater le changement du taux des fausses alertes après chaque essai, avec une moyenne acceptable de **0,000643%.** Les **4** règles précédentes génèrent **16** fausses alertes sur **2445869** paquets traités pendant les 4 jours, la fiabilité de nos règles est donc confirmée.
- La constatation sur les quatre règles nous ont permis d'observer une similitude avec les liaisons ping effecteur pour atteindre n'importe quels serveurs, ainsi que pour la connexion serveur hello qui est très souvent élaborer dans une connexion TLS.

Note : Il existe toujours un taux de faux positifs sur ce type d'alerte, il est donc nécessaire de vérifier le contexte de l'alerte afin de déterminer si l'on a bien affaire à une véritable tentative d'utilisation du réseau Peer to Peer.

7.3. Résultats :

- ✓ Détection de l'utilisation du trafic Peer to Peer basé sur le protocole E-Donkey et BitTorrent.
- ✓ Détection de la connexion BitTorrent cryptée, à l'aide des requêtes principales d'établissement d'une connexion torrent « **http GET TORRENT/SCRAPE/ANNOUNCE »**.
- ✓ Utilisation du DHT qui est considérée comme une extension supplémentaire du protocole BitTorrent. On recommande donc d'éviter son utilisation pour une quelconque détection du mode crypté.
- ✓ Détection de la connexion µTorrent à partir du smartphone.
- ✓ Implication d'alertes faites par toutes les règles implémentées.
- ✓ Crédibilité de toutes les règles, avec de différents taux de fiabilités pour chacune.

IV.9. Méthodes supplémentaires de protection

Pare-feu (firewall)

Une autre méthode supplémentaire de protection contre les risques d'utilisation du réseau Peer to Peer est le PARE-FEU. C'est un outil de protection basé sur le blocage des Ports par défaut des applications (automatiquement le blocage du trafic). L'emplacement du PARE-FEU sera entre l'ensemble du réseau interne et le serveur proxy, cela va nous permettre de diminuer la charge sur notre serveur. Les Ports par défaut de chaque protocole sont :

Protocole	Ports par défaut
E-Donkey	TCP – [4660 – 4669]
	UDP – [4670 – 4679]
BitTorrent	TCP – [6681 – 6999]

Tableau IV-2: Ports par défaut des applications E-Donkey et BitTorrent

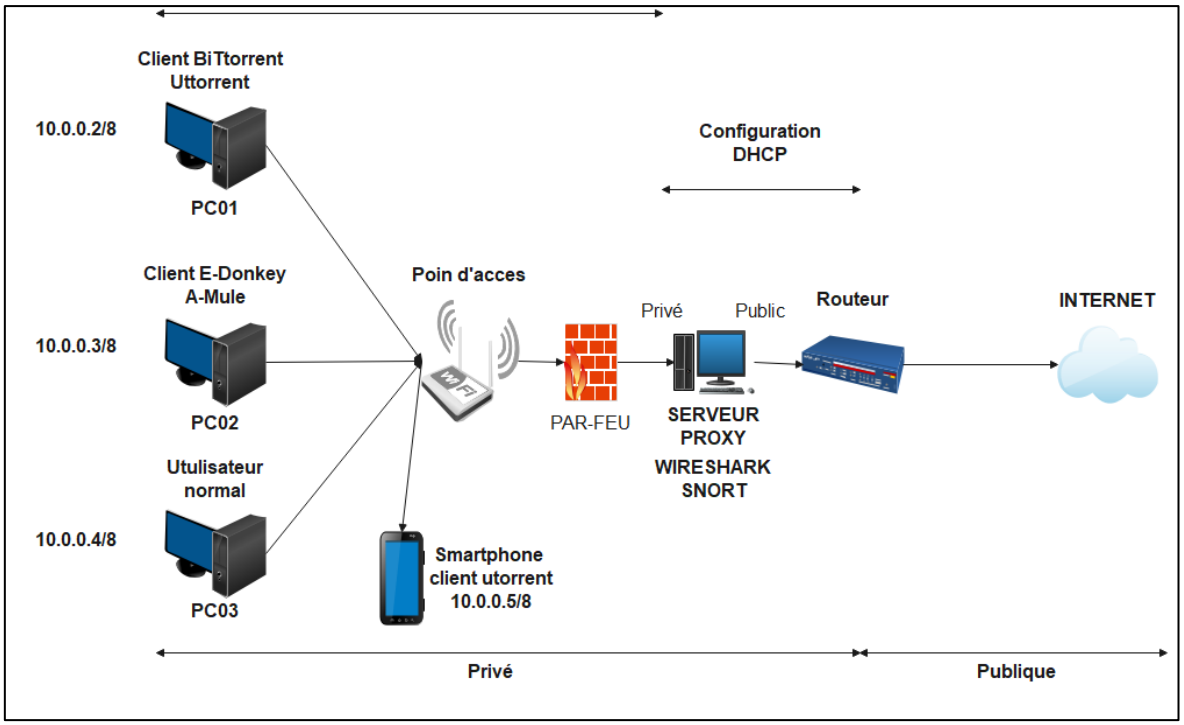


Figure IV.28: Amélioration de la protection par le PARE-FEU

- Le changement des ports de fonctionnement des clients permet d'éviter cette étape de protection, pour cela il est recommandé d'éviter l'utilisation de cette méthode uniquement.

IV.10. Conclusion

Nous avons acquis durant ce chapitre certaines connaissances liées à l'utilisation de Snort en tant que système de détection d'intrusion réseau sous Windows, à l'aide des différents identifiants extraits du troisième chapitre où on a créé des règles de détection pour le Protocole E-Donkey et BitTorrent. Et afin de bien gérer la détection, on a utilisé l'interface graphique BASE avec ses nombreuses fonctionnalités qui nous facilitent le suivi du trafic analysé.

Comme tous les systèmes de sécurité informatique, il existe toujours un taux de faux positifs dans les systèmes de détection d'intrusion. Le test du laboratoire nous a permis d'atteindre un taux de détection de 100%. En revanche, le test effectué dans un environnement réel affiche un taux de fausses alertes de 0,000643% généré par certaines règles, nous permettons de confirmer qu'il n'y a pas de détection fiable à 100% dans la pratique.

Conclusion générale

La nouvelle technologie offerte par les réseaux Peer to Peer facilite les tâches et améliore les méthodes d'échange et partage de fichiers. Par contre, ce réseau contient pas mal d'inconvénient qui ne peuvent pas être pris à la légère tel que : les virus (ex : En 2015, 10% des malwares étaient propagés via des applications P2P), les attaques informatiques (Dos et DDos) etc... Pour cela, le contrôle et le suivi de son utilisation au sien des entreprises et universités est essentiels.

Ce travail est principalement axé sur la sécurité d'un réseau d'entreprise contre l'utilisation de deux protocoles Peer to Peer : « E-Donkey » et « BitTorrent ». C'est dans ce cadre-là qu'on a proposé une solution dans une architecture client-serveur constituée de trois utilisateurs qui passent par un serveur proxy où « Wireshark » y est installé afin de faire une analyse profonde du trafic pour récupérer les empreintes numériques dédies aux Peer to Peer. Par la suite, on a conçu une liste de règles qu'on a implémenté dans un détecteur d'intrusion réseau « Snort » au niveau du serveur proxy pour remédier à ce problème.

Le test de notre solution est divisé en deux scénarios, le premier est effectué au niveau de notre laboratoire afin de valider le bon fonctionnement de nos règles, où les tests étaient très concluants puisque nous avons pu détecter toute les utilisations de l'application A-Mule et µTorrent avec un taux de 100%.

L'efficacité de ces règles est le but de notre deuxième scénario, effectué dans un environnement réel où on a mis en place une architecture client-serveur, mais cette fois-ci sans aucune utilisation des clients Peer to Peer afin de simuler un réseau d'entreprise dans lequel on a implémenté des règles au niveau du serveur proxy. En effectuant ce test-là pendant 4 jours, on a constaté au final que ce système a engendré des fausses alertes pour quatre règles seulement parmi les vingt-sept. Malgré ce faible taux de fiabilité presque nul, on a bel et bien pu valider l'efficacité de nos règles.

Afin d'améliorer la détection du protocole E-Donkey et BitTorrent dans une entreprise, on propose ce qui suit :

- ✓ Utilisation d'un HIDS au niveau des machines d'employées, avec des règles spéciales contre l'utilisation de ces applications.
- ✓ Utilisation d'un NIDS dans le réseau, avec des règles constituées de signatures qui identifient l'emploi de ces protocoles.
- ✓ Mise à jour fréquente de la base des signatures de NIDS.
- ✓ Sensibilisation contre les menaces venant des applications Peer to Peer.
- ✓ Mise en place d'un serveur annuaire interne pour gérer les profils d'employés (ex : Active Directory)
- ✓ Activation du QoS en mettant à la quarantaine le trafic Peer to Peer.

Pour finir, on peut dire que la recherche dans ce domaine est encore très active, surtout que cette technologie est devenue la plus grande source de téléchargement de fichiers.

Bibliographie et Webographie

[1] Cisco « Cisco Certified Network Associate 1 » version 6 : Notions de base sur les réseaux.

[2] Nathalie Budan, Benoit Tedschi, Stéphane Vaubourg, 'Nouvelles Technologies Réseau', 2003.

[3] Didi Souheyla, Cherifi Sabah, 'Mise au point d'une application de téléchargement et de communication en Peer-to-Peer', 2015.

[4] Patrick MARLIER, « Sécurité du Peer-to-Peer », article LABO-ASSO.

[5] Jean-Marier Flaus, 'Cybersécurité des systemes industriels', editions ISTE, 2019.

[6] Asma DAOUDI, « Détection de l'attaque Eclipse dans le réseau eDonkey », Mémoire de Magister en informatique, Université Abderahmane Mira de Béjaïa 2007.

[7] Fabrice Schuler, « Etude et utilisation des technologies des P2P », Publié le 28 février 2005 - Mis à jour le 14 février 2020. Récupéré en janvier 2021 sur www.schuler.developpez.com.

[8] KADDOUR Halima, « Mise en œuvre d'une application SMA Dans les réseaux P2P », mémoire de Master, Université Abou Bakr Belkaid– Tlemcen 2015.

[9] Kevin koo, « Understanding of BitTorrent Protocol », FORENSIC INSIGHT; DIGITAL FORENSICS COMMUNITY IN KOREA. Récupéré en janvier 2021 sur www.forensicinsight.org.

[10] Oliver Heckmann, Axel Bock, Andreas Mauthe, Ralf Steinmetz « The eDonkey File-Sharing Network », Multimedia Kommunikation (KOM) Technische Universitat Darmstadt ¨ Merckstr. 25, 64293 Darmstadt,

[11] 'BitTorrent a gagné contre eMule'. Récupéré en janvier 2021 sur : https://www.numerama.com/magazine/27507-bittorrent-a-gagne-contre-emule-le-p2p-a perdu.html.

[12] 'E-Mule', récupéré en Mars 2021 sur : https://www.emule-project.net

[13] 'utorrent', récupéré en Mars 2021 sur : https://www.phonandroid.com

[14] 'About Wireshark' récupéré en Mars 2021 sur : https://www.wireshark.org/

[15] Illir KADRIU : 'Utilisation de Snort dans une PME', Haute école de gestion Genève, édition, 2019.

[16] 'Logo Snort ' récupéré en Mai 2021 sur : https://www.snort.org.