UNIVERSITÉ DU QUÉBEC À MONTRÉAL

MÉTHODE CO-DESIGN (LOGICIEL/MATÉRIEL) D'IDENTIFICATION ET D'AUTO CLASSIFICATION DES PROTOCOLES DE HAUT NIVEAU.

MÉMOIRE PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE SYSTÈME
EN VUE DE L'OBTENTION DU DIPLÔME DE
MAÎTRISE EN SCIENCES

PAR
RÉJEAN LEPAGE

DÉCEMBRE 2002

LES REMERCIEMENTS

Je tiens à remercier le professeur Omar Cherkaoui, mon directeur de recherche, qui m'a permis d'accéder à la maîtrise, qui a su juger de mes compétences, qui m'a fait confiance et qui m'a fourni les ressources et les connaissances nécessaires à la réussite.

Je tiens à remercier le professeur Yvon Savaria, mon co-directeur de recherche mais aussi mon patron au travail depuis les 7 dernières années. Il m'a permis de faire parti de sa recherche et ce conjointement à mon travail.

Je tiens à remercier Tien-Hung Bui pour avoir été mon correcteur, pour avoir pris le temps de comprendre et de stimuler le raisonnement et parce qu'il a su mettre en mots des idées parfois confuses et complexes.

Je voudrais aussi souligner la participation des étudiants de PFE. Chi-Tam Pham qui a travaillé à la validation de l'architecture avec les outils Seamless et Renoir de Mentor Graphics. Ainsi que Alexandre Boissy qui a participé à la revue de littérature sur la classification des paquets. Finalement, le dernier mais non le moindre, Jean-Marc Tremblay pour avoir démarré et inspiré le projet sur les processeurs réseaux au GRM.

Présenté par :

Réjean Lepage,
Candidat à la M.Sc. en Informatique systèmes, UQAM
Rejean.Lepage@polymtl.ca

Membres du jury :

Omar Cherkaoui, Ph.D.
Directeur, Professeur titulaire, UQAM
Cherkaoui.Omar@uqam.ca

Yvon Savaria ing, Ph.D.
Codirecteur, Professeur titulaire, École Polytechnique
Yvon.Savaria@polymtl.ca

Guy Bois, Ph.D.
Professeur agrégé, École Polytechnique
Guy.Bois@polymtl.ca

El Mostapha ABOULHAMID, Ph.D.
Professeur, Université de Montréal
em.aboulhamid@umontreal.ca

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ÉQUATIONS

ACRONYMES

Tableau 0.1 – Acronymes

ARM	(Système de prototypage SoC ; nom du fabricant.)
ASIC	Application Specific Integrated Circuits
CAM	Content Addressable Memory
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
FDDI	Fiber Distributed Data Interface
FPGA	Field Programmable Gate Array
GRM	Groupe de Recherche en Microélectronique, École Polytechnique de Montréal
H.323	Norme de l'ITU-T
IEEE-1394	Standard de L'IEEE avec Plusieurs Noms: Apple – Firewire, Sony: i.Link
IEEE-802.3	Standard de L'IEEE réseau Ethernet
IP	Internet Protocol
IPX	(Protocole de transport de la compagnie Novell)
ISO	International Standard Organization
ITU	International Télécommunications Union
OSI	La norme « *Open System Interconnections* » Interconnexion de systèmes ouverts.
PCM	Codage de données PCM pulse code modulation
QdS / QoS	Qualité de Service
RSVP	ReSerVation Protocol
RTCP	Real-Time Control Protocol
RTP	Real-Time transport Protocol
RTSP	Real-Time Streaming Protocol
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SoC	System on Chip
TCAM	Ternary Content Addressable Memory
TCP	Transmission Control Protocol
UDP	User Datagram Packet
VHDL	VHSIC: Very High Speed Integrated Circuit; Hardware Description Language
VoIP	Voice Over Internet Protocol

RÉSUMÉ

Le processus d'identification de protocoles est en général simple pour les premières couches. Par exemple, pour la couche liaison des données, il suffit de lire le contenu d'un paquet d'une interface réseau connue et de comparer certains champs de son en-tête à des valeurs connues. On considère qu'il y a identification du protocole lorsqu'il y a une correspondance entre ces valeurs. La situation se complique lorsque le débit de traitement est très élevé et que l'on veut identifier les protocoles des couches supérieures.

Nous proposons une méthode d'identification par apprentissage applicable à la classification des paquets de haut niveau. L'analyse des données, jusqu'au niveau application de l'OSI, est nécessaire dans de nombreux contextes, par exemple : pour assurer la qualité de service ou pour l'interception des paquets lors de l'écoute électronique et diverses applications en sécurité informatique. La création de signatures générées à partir de cette analyse servira à notre identification. L'analyse du contenu des paquets sera accomplie pour identifier des paquets tel que ceux appartenant au protocole RTP qui est utilisé lors des sessions multimédias sur les réseaux IP.

Il sera démontré qu'en comparant 6 ou 12 octets seulement, il est possible d'identifier environ 95% des paquets, tel que RTP, dans un flot contenant de la vidéo numérique.

Certaines méthodes ainsi qu'une architecture logicielle/matérielle accélérant les points sensibles seront proposées et modélisées.

CHAPITRE I

1 INTRODUCTION

L'évolution du monde des réseaux de télécommunications a entraîné une hétérogénéité au niveau des réseaux informatiques. Les moyens de communication entre ces différents réseaux sont les protocoles de communication. Un protocole de communication spécifie la façon dont la communication des données (transmission et réception) se fait dans ce réseau.

Pour des réseaux homogènes, la communication est simple et normale et ne requiert aucun intermédiaire relatif aux protocoles, puisqu'ils sont de même type. Par contre, pour des réseaux hétérogènes, il y a nécessité de les adapter, pour qu'ils puissent se comprendre, à savoir, lire et écrire des données compréhensibles pour chacun d'entre eux. C'est dans ce cadre que se situe la notion de convertisseur de protocoles génériques, qui est l'ensemble dans lequel notre projet prend part.

Dans le cadre de ses activités de recherche, le projet « netpro » du GRM « Groupe de recherche en microélectronique » de l'École Polytechnique, vise la conception d'un convertisseur de protocoles. La conversion de protocoles ne représente pas en elle-même une tâche complexe, mais c'est le contexte dans lequel elle se produit qui la rend complexe. En réalité, la complexité de cette dernière s'amplifie avec les contraintes suivantes :

- Un débit élevé de données à traiter arrivant à un réseau,
- Tous les protocoles doivent être reconnus ou identifiés avant d'être convertis ou traités,
- La reconfigurabilité lors de l'arrivée d'un nouveau protocole,

D'après notre revue de littérature, il existe déjà des convertisseurs de protocoles, mais ces derniers n'offrent pas, en général, un bon compromis entre la flexibilité, les coûts non récurrents et la rapidité de conversion.

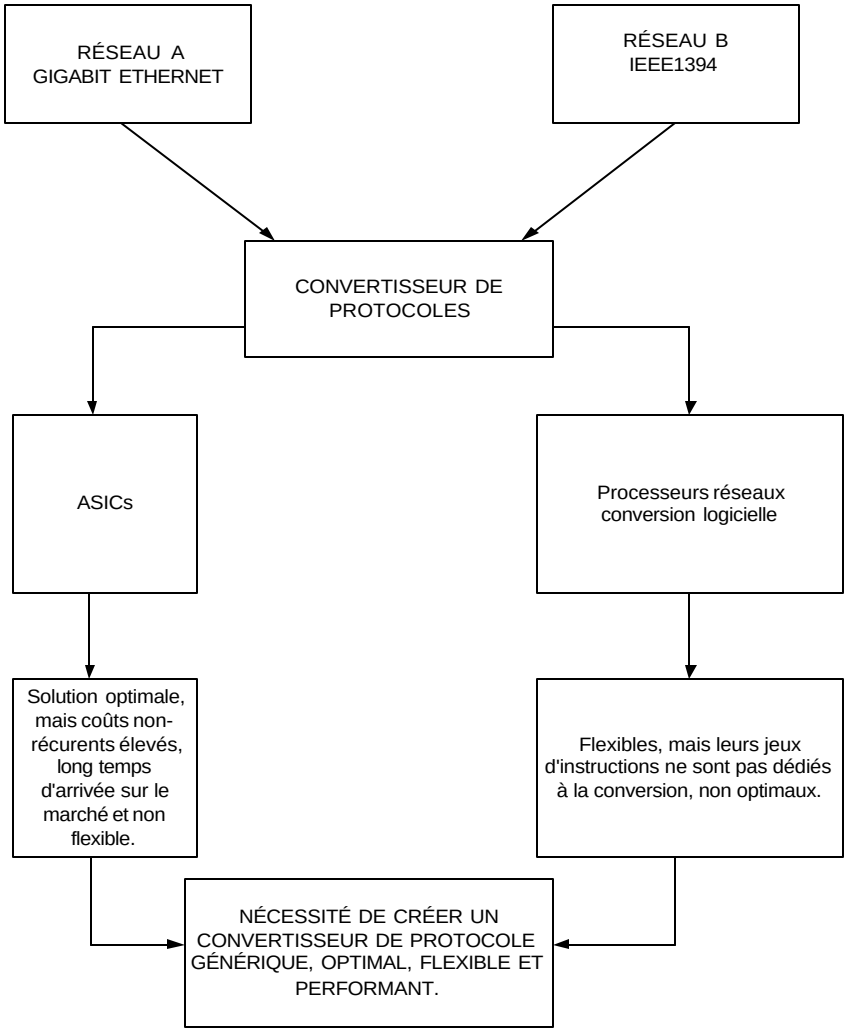

Figure 1.1 - Nécessité du convertisseur de protocole.

La Figure 1.1 présente un exemple qui illustre la situation actuelle en ce qui concerne les convertisseurs de protocoles. Cette figure situe les convertisseurs de protocoles entre deux réseaux différents. Elle démontre deux moyens de fabrication de ce type de processeur avec certains avantages et désavantages de chacune des solutions. Soit à gauche, en

matériel (ASIC), soit à droite, en logiciel. Après examen de cette situation, nous pouvons voir l'utilité en ce qui concerne la mise au point d'un convertisseur de piles de protocoles adaptable et reconfigurable pour permettre de gérer l'arrivée d'un nouveau protocole.

Pour ajouter un autre degré de difficulté, il peut être question de conversion d'empilement de protocoles. En effet, il est très souvent possible qu'une trame de données soit composée d'un ensemble de protocoles différents, appartenant chacun à une couche spécifique du modèle OSI.

1.1 Formalisation du problème

Le but de œtte recherche est de contribuer à définir l'architecture d'un convertisseur de protocoles et de développer une méthode d'identification des protocoles de tous les niveaux du modèle OSI. Il est surtout nécessaire d'accélérer l'identification des protocoles de haut niveau. Le but de cette accélération logicielle ou matérielle est de trouver un moyen d'identifier ou de classifier rapidement les protocoles. Elle vise la réduction du temps requis pour identifier les cas fréquents, pour ainsi allouer plus de temps à la conversion de protocoles. Elle permettra le fonctionnement fluide de la conversion en temps réel fonctionnant avec les débits du convertisseur.

Le problème n'est pas d'identifier les protocoles de bas niveau parce qu'ils sont annoncés par l'en-tête de la trame de la couche inférieure. Il est d'identifier le protocole de niveau application, car ces protocoles ne sont pas annoncés par un champ dans un en-tête des paquets. Nous devons faire une analyse du contenu des données de l'application pour récupérer l'information pertinente à la reconnaissance des prochains paquets. La recherche sera axée sur l'identification des protocoles de transport de la

vidéo numérique et sera conciliable avec le projet de convertisseur de protocoles du GRM supporté par la compagnie GENNUM.

L'analyse du contenu en temps réel des données peut être difficile à réaliser quand les débits sont élevés. Il est possible, et peut-être nécessaire, d'utiliser des accélérateurs matériels. La justification d'utiliser des accélérateurs devra se faire en calculant le temps requis pour identifier les trames et mettre à jour les signatures. Si le temps de traitement en logiciel est trop long, il sera peut-être question d'utiliser des co-processeurs matériels. Il sera même possible de classifier les protocoles à l'intérieur de sessions de données existantes.

À travers ce projet de recherche, des méthodes et des algorithmes d'identification de protocoles seront expliquées. Une méthode dynamique capable de préparer l'anticipation des flots à haut débit identifiera les protocoles et les sessions multimédias.

Nos recherches conduisent à la proposition d'une nouvelle méthode pour identifier ou classifier les protocoles à l'aide d'accélérateurs matériels. L'utilisation d'une CAM modifiée, présentée dans [USP], sera envisagée.

1.2 Plan des chapitres

Le chapitre 2 a comme objectif l'explication des normes, des protocoles et des concepts nécessaires à la compréhension de la recherche. Il sera une référence sur les parties importantes des protocoles nécessaires à la compréhension des notions de base en microélectronique.

Le chapitre 3 a comme objectif général de définir le contexte du projet existant et les architectures développées qui indiquent les limites de ce projet. Ce chapitre contient une revue de littérature sur les méthodes de traitement des paquets et expose les sujets associés à l'interconnexion de réseaux.

Le chapitre 4 a comme objectif d'expliquer les méthodes d'identification existantes. Une analyse statistique du contenu du réseau sera réalisée. Il a comme objectif spécifique l'explication des procédés d'identification. Il en démontre la complexité et donne des résultats de profilage pour la méthode proposée.

Le chapitre 5 justifie une implémentation matérielle par l'analyse des fonctions requises en évaluant le débit réel du réseau. Il vise à établir les limites d'une l'implémentation entièrement logicielle qui conduirait à une incapacité de traiter des débits élevés.

Le chapitre 6 explique et définit l'architecture développée. Il compare les performances et quantifie l'accélération du traitement en développant un modèle de calcul et en estimant la charge du réseau.

La conclusion met en valeur les contributions scientifiques de ce mémoire et propose des idées pour la poursuite de la recherche.

CHAPITRE II

2 ÉTUDE DES PROTOCOLES ET NOTIONS DE BASE

La présentation d'éléments nécessaires à la réalisation du mémoire et à sa compréhension est essentielle, ce sont les notions de base sur lesquelles se fondent des affirmations formulées dans les chapitres suivants de ce mémoire.

Dans ce chapitre des concepts nécessaires à cette recherche seront présentés, tels que:

- notre position par rapport au modèle de référence OSI,
- les protocoles supportés et leur utilité,
- la représentation des structures de données,
- les aspects utilisés de la microélectronique.

2.1 Normes et Protocoles

Les protocoles utilisés seront souvent référés en rapport à leur niveau dans la norme OSI définie par l'ISO. Cette norme sera une référence sur la complexité d'identification, plus le niveau est haut, plus complexe sera son identification.

2.1.1 Le modèle OSI

L'organisation des paquets est basée sur le modèle OSI (Open System Interconnection). Le principe de ce modèle repose sur l'encapsulation des couches pour permettre à divers équipements réseaux de communiquer.

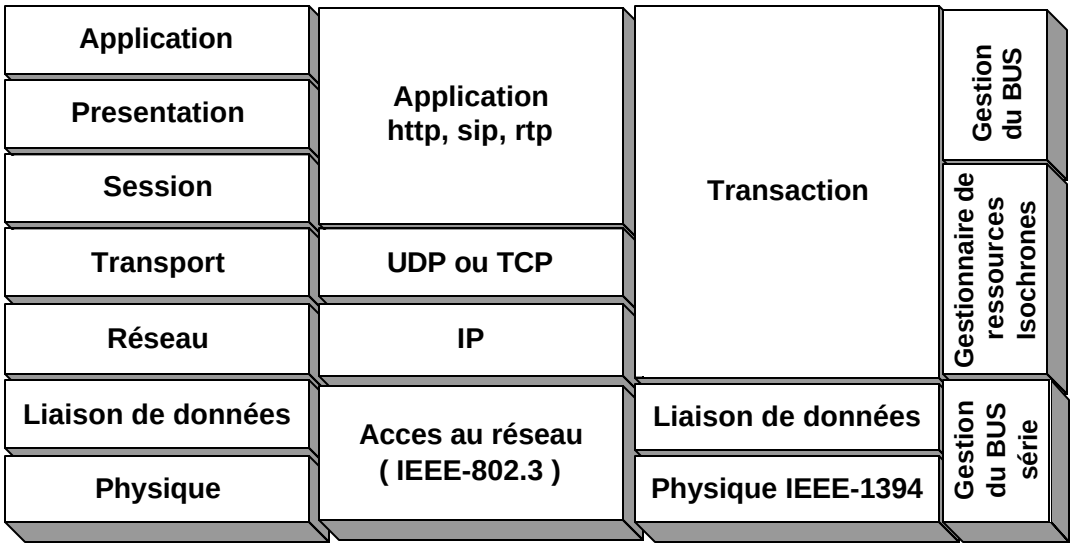

Figure 2.1 : Modèle OSI comparé au modèle Internet et de IEEE1394.

En général, un paquet peut être décomposé en sept couches différentes, comme décrit à gauche de la Figure 2.1. Chaque couche se distingue par son en-tête et sa queue, le reste étant pour elle considérée comme charge utile, une donnée à transporter. Pour plus de détails voir [OMAR98 chap.1]. Le but n'est pas de donner une description exhaustive de chacune des couches, mais de donner une référence par rapport aux interfaces choisies (Figure 2.1) et la complexité d'analyse selon le niveau. Voici un survol des sept couches avec la provenance de l'identification des protocoles de ce niveau.

La couche physique

La couche physique se charge d'interpréter les signaux de l'interface physique. Elle regroupe tout ce qui a trait à la transmission et au contrôle physique des signaux analogiques et numériques du message.

Ce niveau est directement identifiable par sa connexion physique. La mise en œuvre des interfaces physiques ne sera pas discutée dans ce mémoire

car nous supposons que des modules physiques respectant les normes pertinentes nous sont disponibles.

La couche de liaison de données

La couche liaison assure l'échange ordonné des trames et s'occupe du contrôle d'intégrité et des erreurs. Elle corrige les erreurs ou elle demande une retransmission des données.

Ce niveau est généralement traité par les modules d'interface physique. Il spécifie des structures de données (LLC) à respecter que nous traiterons comme des contraintes. Il n'a pas besoin d'algorithme autre que ceux mis en œuvre dans l'interface physique.

La couche réseau

La couche réseau a le mandat de l'acheminement du message au destinataire en utilisant des techniques de routage et d'adressage et s'occupe du contrôle de flux.

Le protocole adopté à ce niveau est annoncé par le type inclus dans la structure MAC (Medium Access Control) Cette structure sera présenté au chapitre 2.2.1. Lorsque le protocole est IP la taille de l'en-tête variable doit être calculée.

La couche transport

La couche transport s'occupe de l'établissement de la communication de bout en bout. Cette couche doit assurer un transfert transparent des fragments des données. Elle corrige les lacunes de la couche réseau tant en ce qui a trait à la qualité de service, la fragmentation, le contrôle de flux des données, la détection d'erreur et la récupération (retransmission) des données.

Il existe deux types de communication, soit orienté connexion (TCP) ou non (UDP). Sous IP, ce protocole est annoncé par un champ. La taille de l'en-tête TCP est variable et doit être calculée.

La couche session

La couche session a pour rôle d'organiser et de synchroniser les processus de la couche présentation. Par exemple, elle s'occupe des services d'établissement de connexion de session entre les applications pour la durée des transactions.

L'identification de protocoles dont il est question dans ce mémoire fait abstraction de ce niveau.

La couche présentation

La couche présentation regroupe les fonctions de présentation des données. Elle traduit le vocabulaire et la syntaxe vers une représentation commune à l'émetteur et au destinataire.

L'identification de protocoles dont il est question dans ce mémoire fait aussi abstraction de ce niveau.

La couche application

La couche application est la seule porte d'entrée dans l'environnement OSI. Elle fournit les services de communication aux utilisateurs. Cette couche est directement liée à l'échange des données avec les applications, les logiciels. C'est l'ensemble des services généralement utilisés par les applications logicielles comme les applications de la vidéo numérique.

L'identification de ce niveau doit être annoncée par un gestionnaire de sessions. Les données sont difficilement identifiables et n'ont pas de signification hors de leurs contextes. On peut envisager le besoin d'un algorithme d'identification avec découverte par apprentissage.

2.2 Les protocoles de base

Dans cette section, on fera un survol d'un ensemble de protocoles d'intérêt pour supporter des communications multimédias. Cette section sert de référence pour comprendre le fonctionnement et l'utilité de chacun de ces protocoles. Le traitement souligne notamment les informations que l'on doit extraire des en-têtes des protocoles supportés afin d'en permettre l'identification.

2.2.1 Ethernet

L'interface physique définie par la norme IEEE 802.3 appelée aussi Ethernet, décode le signal modulé et la fréquence. Elle synchronise ses trames. Elle reconnaît toujours le début de trame. En cas d'erreur, c'est à l'interface de rejeter les paquets erronés. Il y a toujours un début et une fin de trame. L'interface physique, dans ce cas Ethernet, inclut toujours le protocole de liaison de données (LLC) ou IEEE802.2. Les trames complètes sont passées à la couche liaison de données. Cette dernière ne reçoit pas simplement une séquence de bits, mais elle reçoit plutôt des trames complètes et valides. Suite à son intervention, il ne reste que les champs de la couche liaison, définis par la norme IEEE 802.2.

Tableau 2.1 - La Famille d'interfaces Ethernet

Ethernet 10 Mbps		
10 Base-5	coaxial	IEEE 802.3 CSMA/CD
10 Base-2	coaxial	IEEE 802.3
10Base-T	paires torsadées	IEEE 802.3
Ethernet 100 Mbps		
100Base-T4	4 paires torsadées	IEEE 802.3
100Base-TX	2 paires torsadées	IEEE 802.3
100Base-FX	fibre optique	IEEE 802.3
Ethernet 1000 Mbps (1 Gbps)		
1000Base-SX 1000Base-LX	fibre optique	IEEE 802.3z
1000Base-CX	coaxial (25m)	IEEE 802.3z
1000Base-T	N paires torsadées	half-duplex: IEEE 802.3u full-duplex: IEEE 802.3x
Ethernet 10,000 Mpbs (10 Gbps)		
En cours de développement	fibre optique	IEEE 802.3ae

Le nom Ethernet est donné à plusieurs interfaces de communication dont l'usage est largement répandu pour la mise en œuvre des réseaux locaux. Une liste de ses variantes est présentée au Tableau 2.1.

La trame Ethernet originale a évolué vers la norme IEEE802.3. Cette norme inclut toujours le standard (LLC), liaison de données ou IEEE802.2. Ces champs sont utilisés sur toutes les interfaces pour l'initialisation, la déconnexion, l'encapsulation, la synchronisation, l'acquiescement, le contrôle des erreurs ou le contrôle du débit. Le rôle de la couche MAC est de permettre la traduction vers d'autres protocoles.

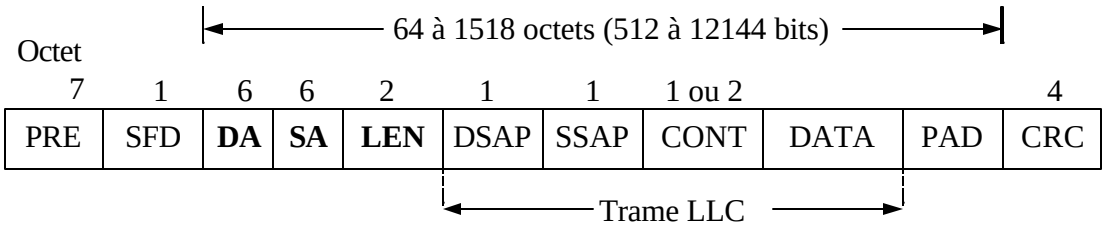

Figure 2.2 -Trame Ethernet MAC avec LLC

Une trame Ethernet est composée des champs originaux (MAC) « Media Access Control » et des champs LLC « Liaison Link Controler » le tout présenté à la Figure 2.2.

A notre niveau, l'identification ne se fera qu'à partir de la couche liaison de données IEEE802.2, en utilisant le champ d'identification principal, appelé « LEN ou TYPE (dans l'Ethernet original)». Voici quelques exemples de valeurs communes du champ « LEN ». Si la valeur en hexadécimal est entre 0000 et 05DC, alors le champ doit être lu comme la longueur de la trame Ethernet de 0 à 1500 en décimal.

Tableau 2.2 - Valeurs du champ type Ethernet

Valeurs (hex)	Nom
0x0800	IP version 4 ; DOD Protocole Internet
0x0805	X.25 niveau 3
0x0806	ARP ; Address Resolution Protocol
0x8035	RARP ; Reverse Address Resolution Protocol
0x8037	IPX ; Novell Netware
0x86DD	IP version 6
0x9000	Loopback (Protocole de Configuration et test)

À ce point, il est plus probable de trouver deux protocoles IP ou ARP. Il serait alors intéressant lors d'un profilage de classer chaque protocoles par sa probabilité d'apparition. Ceci peut permettre de minimiser la liste

des comparaisons successives. Le Tableau 2.2 affiche certaines valeurs possibles.

La section 3.4.7.1 présentera plus de détails sur le débit de cette interface et ses caractéristiques.

2.2.1.1 Protocole IP

Le protocole IP « Internet Protocol » (RFC-791) est un protocole de réseau (niveau 3) du modèle OSI. Un paquet conforme à IP a une taille de 20 octets et plus, selon le nombre d'options inscrites dans le champ IHL et s'il n'y a pas d'options et de remplissage. Une trame IP version 4 est constituée comme suit:

Tableau 2.3 - En-tête de IP version 4

0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31
Version	**IHL**	Type de service	Longueur totale
Identification		Fanion	Fragment Offset
Durée de vie	**Protocole**	Checksum d'en-tête	
Adresse Source			
Adresse Destination			
Options		Remplissage	

Certains champs méritent une explication plus détaillée, parce qu'ils seront utilisés plus tard pour l'identification.

Les champs ***version*** et ***IHL*** sont sur 8 bits, 4 bits chacun. Il faut donc les séparer et interpréter le champ IHL « Internet Header Length » qui est la taille de l'en-tête en mots de 32 bits. Il pointe sur le commencement des données. La valeur minimale pour une en-tête correcte est de 5. Le calcul se fait de la façon suivante (5 x 32) = 160 bits / 8 = 20 octets d'en-tête.

Le champ ***longueur totale***, sur 16 bits, est la taille totale d'un datagramme en octets comprenant l'en-tête IP et les données. Ce champ permet d'avoir une taille de données allant jusqu'à 65,535 octets. Un tel datagramme ne peut pas être transmis directement dans la plupart des systèmes et des réseaux. Chaque système doit être prêt à accepter un datagramme allant jusqu'à 576 octets, fragmenté ou non. Il est recommandé qu'un système n'envoie un datagramme plus grand que 576 octets que s'il a l'assurance que la destination est préparée à les accepter.

Le nombre 576 correspond une taille raisonnable des données à être transmises en plus de l'en-tête d'information. Par exemple, cette taille permet un bloc de données de 512 octets plus 64 octets d'en-tête initiale. La somme des en-têtes maximales d'Internet est de 60 octets. Tandis qu'une en-tête typique Internet est de 20 octets, allouant ainsi une marge pour les en-têtes des protocoles des niveaux supérieurs (tel que RTP).

Le champ ***protocole*** annonce le protocole de niveau réseau encodé dans IP. Une valeur de 6 signifie que c'est TCP tandis qu'une valeur de 17 signifie que c'est UDP.

Tableau 2.4 – Valeurs du champ protocole de IP

Décimal	Nom du Protocole
1	ICMP
3	Gateway-to-Gateway
4	CMCC Gateway Monitoring Message
6	TCP
17	User Datagram
18	Multiplexing

Voici une liste, Tableau 2.4, des protocoles utilisables pour le problème considéré ici, à savoir le transport de la vidéo numérique, sur un réseau

de type Internet. Soulignons que le champ protocole de IP est codé sur 8 bits.

Le champ ***adresse source*** est codé sur 32 bits et identifie de façon unique la machine source. Le champ ***adresse de destination,*** est codé sur 32 bits et identifie de façon unique la machine destination. Les adresses de 32 bits sont séparées en quatre champs (AAA.BBB.CCC.DDD) qui permettent le routage du paquet.

2.2.1.2 Protocole TCP

TCP « Transport Control Protocol » [RFC-793] est un protocole de la couche transport (niveau 4) du modèle OSI. TCP est orienté connexion et est full-duplex. Il permet le transfert des données avec un contrôle et la reprise sur erreur.

La taille de son en-tête est de 20 octets et plus, selon le nombre d'options inscrites dans le champ Décalage des données. Une trame TCP est constituée comme ceci :

Tableau 2.5 - En-tête de TCP

<table>
<tr><td colspan="32">0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31</td></tr>
<tr><td colspan="16">Port Source</td><td colspan="16">Port destination</td></tr>
<tr><td colspan="32">Numéro d'ordre</td></tr>
<tr><td colspan="32">Numéro d'accusé de réception</td></tr>
<tr><td colspan="4">Décalage des données</td><td colspan="3">réservée</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td colspan="16">Fenêtre</td></tr>
<tr><td colspan="16">Somme de contrôle</td><td colspan="16">Pointeur d'urgence</td></tr>
<tr><td colspan="24">Options</td><td colspan="8">Remplissage</td></tr>
<tr><td colspan="32">Données</td></tr>
</table>

Le champ ***port de destination*** est le numéro de port où la source envoie l'information. Ce champ est codé sur 16 bits

Le champ ***port source*** est le numéro de port par lequel la source envoie l'information vers la destination. Le port est codé sur 16 bits.

Le champ ***décalage des données*** codé sur 4 bits permet de repérer le début des données dans le paquet. Il pointe sur le commencement des données. Le décalage est essentiel car le champ d'options est de taille variable. Il donne la taille de l'en-tête en mots de 32 bits, calculé de la façon suivante : le nombre d'octets d'en-tête * 32 / 8.

2.2.1.3 Protocole UDP

UDP « User Datagram Packet » [RFC-768] est un protocole de transport (couche 4) du modèle OSI. Contrairement à TCP, UDP a très peu d'options à gérer et l'en-tête des paquets UDP est beaucoup plus petite, ce qui contribue à rendre son traitement plus rapide. Ces paquets sont appelés des datagrammes et ils sont envoyés en mode non connecté. Le transport est donc non fiable et sans garantie de séquencement des datagrammes. Il n'y a pas de contrôle de reprise sur erreur, mais UDP indique qu'il y a eu erreur.

La taille de son en-tête est fixée à 8 octets. Voici le format de paquet pour UDP dont l'en-tête occupe 64 bits.

Tableau 2.6 - En-tête de UDP

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Port Source	**Port Destination**
Longueur	Somme de contrôle
Données	

Le champ ***longueur***, dans la trame UDP, est donné en octets et elle inclut la taille de l'en-tête. Les champs, ***port source*** et ***port destination,*** ont la même utilité que les ports décrits dans TCP.

2.3 Protocoles de niveau application

Les protocoles suivants utilisent les protocoles d'Internet tel que IP, UDP, TCP. Ils peuvent cependant être utilisés sur une interface Ethernet ou une interface IEEE 1394.

2.3.1 RTP (Real-time Transport Protocol)

Le protocole RTP [RFC1889] permet de transporter des données multimédias par l'Internet. Ce protocole est largement utilisé par la communauté de l'internet: il est utilisé, entre autres, dans QuickTime de Apple, NetMeeting de Microsoft et IP/tv de Cisco [RTPN]. Sur l'Internet, de nos jours, RTP est le protocole de transport multimédias qui reçoit probablement le plus d'attention de la communauté de chercheurs en multimédia.

RTP permet le transport des données en temps réel, incluant l'audio et la vidéo. Il peut être utilisé sur réservation de bande ou de façon interactive. RTP consiste en un ensemble de données et une structure de contrôle nommée RTCP (décrite dans la section suivante). La partie des données de RTP définit un protocole simple incluant la reconstruction de trame, la détection de perte, la sécurité et l'identification du contenu. Le transport de RTP se fait sous le protocole UDP après qu'une session est été acceptée. C'est pourquoi un protocole de gestion de session, tel SIP, doit être utilisé pour annoncer l'arrivé d'une session, ou des paquets, contenant le protocole RTP.

Le port de communication UDP devrait être envoyé ou reçu sur un port pair ≥ 5004 tandis que le port de RTCP devrait être transporté sur le prochain port impair disponible UDP port ≥ 5005. Cependant ces ports ne sont pas réservés et ils peuvent prendre n'importe quelle valeur possible dans 65536 possibilités, ce qui complique l'identification.

La taille de l'en-tête de RTP est fixée à 16 octets. Une trame RTP est constituée comme suit:

Tableau 2.7 - En-tête de RTP

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
V		P	X	CC				M	**PT**							Numéro de séquence															
Horodatage																															
(SSRC) identificateur de synchronisation de la source																															
(CSRC) identificateur de contribution																															

Le champ ***V***ersion, indique le numéro de version de RTP. Il doit être égal à deux pour que nos algorithmes fonctionnent.

Le champ ***PT*** « payload type » de RTP définit le contenu, la charge utile, de RTP. Le Tableau 2.8 de la page suivante présente les valeurs possibles de PT.

L'identification de RTP n'est pas simple. Les données transportées par UDP peuvent avoir n'importe quelle valeur possible. Aucun champ d'identification n'est prévu pour aviser du contenu. Il faut alors se fier à un protocole d'annonce des sessions tel que SIP ou un autre protocole de gestion de session.

Tableau 2.8 - « Payload types » de RTP, codage de l'audio et la vidéo

nom PT	Encodage (A/V)	audio/vidéo (Hz)	échantillonnage (audio)	canaux
0	PCMU	A	8000	1
1	1016	A	8000	1
2	G721	A	8000	1
3	GSM	A	8000	1
4	Libre n/d	A	8000	1
5	DVI4	A	8000	1
6	DVI4	A	8000	1
7	LPC	A	8000	1
8	PCMA	A	8000	1
9	G722	A	8000	1
10	L16	A	44100	2
11	L16	A	44100	1
12	Libre n/d	A	-	
13	Libre n/d	A	-	
14	MPA	A	90000	(voir std.)
15	G728	A	8000	1
16-23	Libre n/d	A		
24	Libre n/d	V		
25	CelB	V	90000	
26	JPEG	V	90000	
27	Libre n/d	V		
28	nv	V	90000	
29	Libre n/d	V		
30	Libre n/d	V		
31	H261	V		
32	MPV	V	90000	
33	MP2T	AV	90000	
34--71	Libre n/d			
72--76	Réservé	N/A	N/A	N/A
77--95	Libre n/d			
96--127	dynamique			

Connaissant ces champs et leurs valeurs possibles, listées au Tableau 2.8, il est ainsi possible d'avoir un indice supplémentaire à savoir si une session est possible ou non. Par exemple, si la valeur de PT est de 34 à 95, ces champs sont réservés ou invalides. Les autres champs de RTP ne seront pas nécessaires à l'identification de RTP.

On peut aussi faire les hypothèses d'identification approximatives de RTP suivantes :

1- Le champ qui spécifie la version vaut 2, (10 en binaire), alors la valeur du premier octet doit être au moins 0x80 pour être une en-tête RTP valide.
2- De plus, les ports source et destination de UDP doivent être de valeurs paires.
3- Le champ PT de RTP est valide.

C'est alors que l'on peut essayer de lire la structure comme du RTP. Cette hypothèse d'identification n'est qu'approximative. En effet, la seule garantie que le contenu du paquet UDP est vraiment du RTP ne peut venir que d'une annonce préalable de session faite par SIP. Cette méthode approximative n'offre aucune garantie de trouver vraiment du RTP. Des données aléatoires au début d'un paquet UDP peuvent être reconnues comme un début de trame RTP. Par contre, cette méthode approximative est utile dans le cas ou l'annonce de session n'a pas été interceptée. La probabilité d'une série de paquets UDP ayant ces caractéristiques est faible, sauf dans le cas ou c'est vraiment du RTP.

2.3.2 RTCP (Real-Time Control Protocol)

RTCP [RFC1889] [RFC1890] donne le support au transport de l'audio ou de la vidéo en temps réel, tel que les conférences multimédias, de groupe de n'importe quelle taille sur l'Internet. Ce support inclut l'identification de source et le support au pont audio/vidéo ainsi que la traduction des paquets de plusieurs destinataires vers un récepteur (multicast-to-unicast). Ce protocole offre aussi l'état de la qualité de service pour les groupes ainsi que la synchronisation des données.

Le protocole de transport initial de RTCP est UDP/IP. La taille de son en-tête est fixée à 12 octets. RTP ne réserve aucune ressource pour

l'assurance de la qualité de service, il se fie à RSVP. La taille de l'en-tête RTCP [RFC-1889, 1890] est de 4 octets avec une longueur variable suivie d'une liste d'items SDES.

Les premiers 32 bits du paquet RTCP, suivis des trames d'extension SDES, transportent les directives à l'utilisation de RTCP donc le contrôle de la vitesse, de la qualité d'encodage et le contrôle de flux de RTP.

La taille de son en-tête est fixée à 12 octets. Une trame RTCP est constituée comme suit:

Tableau 2.9 - En-tête de RTCP

0 1 2 3 4 5 6 7 8 9	1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
V P SC	**PT** longueur
ITEMS SDES ...	
...	

Comme RTP, l'identification de RTCP n'est pas simple. Les données transportées par UDP ne possèdent aucun champ divulguant leur contenu. On sait que UDP peut transporter n'importe quelle valeur. Il faut donc se fier sur un protocole d'annonce des sessions tel que SIP pour déterminer son contenu.

Il est possible de faire certaines hypothèses d'identification approximatives de RTCP. Il suffit de vérifier certains champs tout en posant certaines hypothèses logiques telles que celles-ci :

1- Le champ de version est égal à 2 (10 en binaire).
2- Les ports source et destination de UDP sont de valeurs impaires.
3- Il existe un port RTP valide qui a déjà été lu dont la valeur est inférieure et paire aux ports de RTCP.
4- Le champ PT est valide.

Lorsque toutes ces conditions sont remplies, comme on l'avait fait pour RTP, on peut essayer de lire la structure comme étant possiblement du RTCP.

Voici les valeurs possibles du champ PT de RTCP, ce champ définit les charges utiles, les données transportées dans le paquet.

Tableau 2.10 - Valeur du champ PT de RTCP

200	SR « sender report »
201	RR « receiver report »
202	SDES « source description »
203	BYE « Good Bye »
204	APP « application-defined »

2.4 Les protocoles de gestion de sessions

Les protocoles décrits dans ce chapitre sont utilisés pour identifier ou annoncer certains protocoles de niveau application. Ces protocoles permettent la gestion de sessions. Un exemple de gestion est d'annoncer les sessions et les ports de communication de RTP ou de contrôler le débit de la session.

La Figure 2.3 montre un exemple d'empilement des protocoles. Les protocoles au-dessus de UDP et TCP sont difficiles à identifier. Ils doivent être annoncés par un gestionnaire de session.

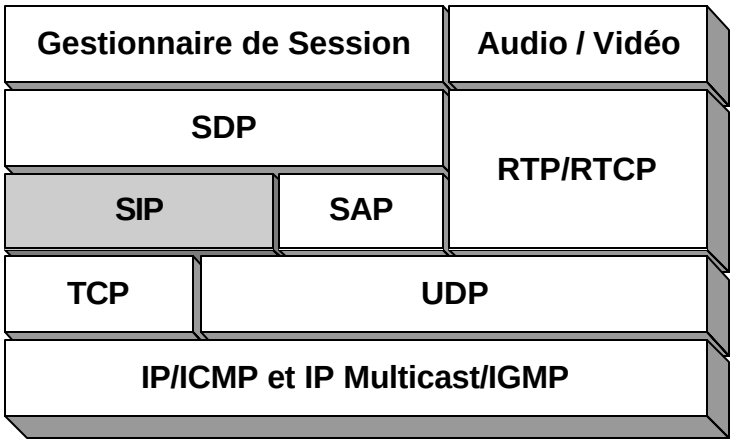

Figure 2.3 - Empilement des protocoles SIP, SDP…

2.4.1 SIP (Session Initialisation Protocol)

SIP est un nouveau protocole de signalisation de l'IETF (Internet Engineering Task Force) [RFC-3261] pour l'établissement en temps réel des appels et des conférences sur les réseaux IP. Chaque session dans une communication peut inclure différents types de données telles que l'audio et la vidéo. SIP gère seulement la partie de signalisation des communications IP. Ceci veut dire que le flot de données est géré par un moyen différent (protocole ou application tel que RTP/RTCP). SIP utilise SDP (Session Description Protocol) pour la description d'information sur l'encodage du média.

2.4.2 SDP (Session Description Protocol)

SDP, présenté dans [RFC-2327], est un protocole pour décrire l'audio et la vidéo dans les sessions multimédias. Les protocoles SIP, MGCP (Media Gateway Control Protocol), SAP (Session Announcement Protocol) et RTSP (Real-Time Streaming Protocol) utilisent tous SDP comme façon de présenter l'information transmise.

2.4.3 RTSP (Real-Time Streaming Protocol)

RTSP, présenté dans [RFC-2326] [RTSP], est un protocole client-serveur de la couche application pour la livraison contrôlée de données multimédias sur les réseaux Internet (IP). Il est créé pour être utilisé avec les protocoles des couches inférieures tels que RTP et RSVP pour donner un service de contrôle complet de type magnétoscope : Comme pause, avance rapide, à reculons et positionnement fixe.

RTSP est codé au niveau de la couche application de l'OSI. Il prend le format d'une adresse http (web). L'appel de son serveur de contrôle se fait comme ceci : « rtsp: //audio.example.com/audio RTSP/1.0 »

2.4.4 H.323

Le protocole H.323 [H323] est le nom donné à un ensemble de protocoles de communication développé par [ITU] « International Telecommunications Union ». Utilisé par Microsoft NetMeeting, pour transmettre de l'audio et de la vidéo sur l'internet, est utilisé dans un ensemble de méthodes de codage/décodage (codec).

La Figure 2.4 montre un empilement des protocoles de la famille H.323. Par exemple H.225 s'occupe des initiations de session tel que SIP et SDP le font.

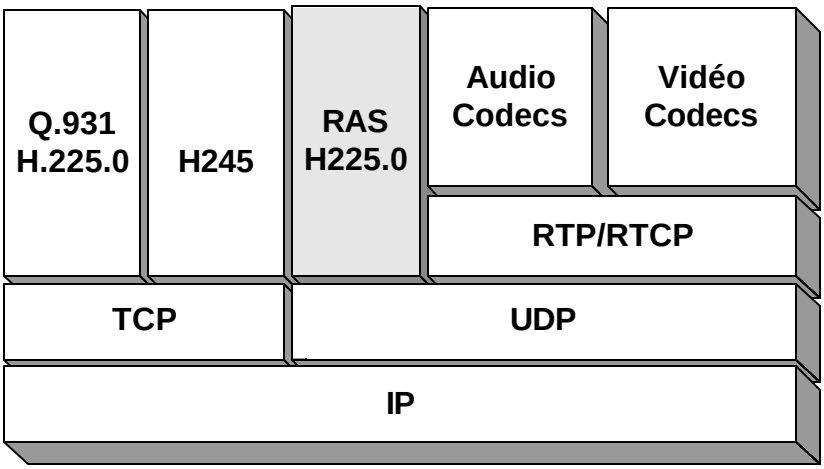

Figure 2.4 - Empilement des protocoles de la famille H.323

2.5 Représentation significative des données

Le fait de connaître ces protocoles ne suffit pas à comprendre comment les interpréter. Dans la façon de lire ces protocoles, il faut tenir compte de la façon de représenter ces données. C'est pourquoi il faut tenir compte de l'architecture utilisée. Chaque plate-forme ou architecture en fait sa propre représentation.

Dans un groupe de bits ou octets, la partie qui contient le plus de poids est la partie la plus significative. À l'intérieur d'un octet, le bit de gauche est le plus significatif. Dans un groupe d'octets, la convention d'ordonnancement des octets détermine la position de l'octet le plus significatif. Deux conventions sont largement utilisées de nos jours, il s'agit des « big-Endian » et « little-Endian ». Il existe aussi le « middle-Endian ».

2.5.1 little-Endian

La convention d'ordonnancement, little-Endian, est utilisée sur les machines Intel 80x86 et DEC. Avec le little-Endian, la partie de gauche de

l'adresse mémoire contient la partie l'octet la moins significative du mot (de 32 bits). Dans l'expression LITTLE_ENDIAN 1234, 1 est la position la moins significative et 4 celle de poids fort. De façon similaire, les Européens écrivent la date dans le format : dd/mm/yy.

2.5.2 big-Endian

La convention d'ordonnancement, big-endian, est utilisée sur les machines Sun (SPARC) et Motorola 680x0. Avec cette notation, la partie de gauche de l'adresse mémoire contient la partie l'octet la plus significative du mot (de 32 bits). Dans l'expression BIG_ENDIAN, 4321, 1 est la position la moins significative et 4 est celle de poinds fort. Dans le même ordre d'idée, les Japonais écrivent la date dans ce format : yy/mm/dd.

2.5.3 middle-Endian

Une autre convention d'ordonnancement a été utilisée sur d'anciens systèmes comme le mini-ordinateur (PDP11). Dans ce cas, les mots contenus en mémoire contiennent une inversion complète des octets et des bits. Avec la notation PDP_ENDIAN 3412, 1 est toujours le champ de poids faible et 4 le champ de poids fort. Un autre exemple de ce type de notation a été adopté par les Américains qui écrivent la date dans le format : mm/dd/yy.

2.6 Notions de microélectronique

L'utilisation de matériel dédié sera peut-être nécessaire pour augmenter les performances et atteindre certains débits de calculs. L'impact de son utilisation sur ces débits devrait être calculé et l'adoption de tels modules d'accélération dépendra des bénéfices qu'ils peuvent apporter. Nous

définirons ici certains termes et concepts nécessaire à la compréhension de cette facette du projet.

2.6.1 SoC (Système sur puce)

Les nouvelles technologies ont porté l'évolution des systèmes embarqués aux systèmes sur des puces. Ce sont des systèmes complets conçus sur une puce au lieu d'être conçus sur un système à périphérique.

Par exemple un PC comporte plusieurs cartes et périphériques tels que: processeur principal (Pentium), carte réseau, bus d'interface, contrôleur de disques, mémoire, etc., alors que le système sur une puce, où SoC, comporte tous ces éléments mais sur une seule puce.

Il est même possible d'y mettre plusieurs processeurs pour faire des tâches spécifiques, des tâches de gestion d'exception par exemple. C'est un gros travail de miniaturisation. La référence [SOC01] traite de ce nouveau type de système sur puce.

2.6.2 FPGA

Les FPGA, (Field Programmable Gate Array), sont des puces configurables. Elles sont constituées d'un grand nombre de portes logiques qui, une fois programmées, simulent le comportement d'un ASIC « Application Specific Integrated Circuit » (à comportement fixe). L'image d'une puce est générée pour être téléchargée dans le FPGA.

Dans notre cas, le FPGA est utilisé comme plate-forme de prototypage. Le cycle de mise à jour n'est prévu que pour des modifications conceptuelles de mise à jour de l'architecture du prototype du convertisseur. Le FPGA ne sera pas modifié lors de l'exécution ou du fonctionnement du convertisseur de protocole.

2.6.3 Mémoires matérielles

Les débits utilisés lors des sessions multimédias de qualité sont grands. Avec un débit d'environ 1 Gbps, une mémoire devrait fonctionner à au moins à 31.25 MHz (1 Gpbs / 32 bits = 31.25 MHz) pour être capable d'effectuer des comparaisons simultanément sur des mots de 32 bits.

La capacité de comparer en parallèle une donnée avec le contenu d'une mémoire pourrait diminuer la latence du traitement d'identification ou de comparaison des paquets.

2.6.3.1 CAM

La signification de CAM est « Content Addressable Memory », donc une mémoire adressable par le contenu.

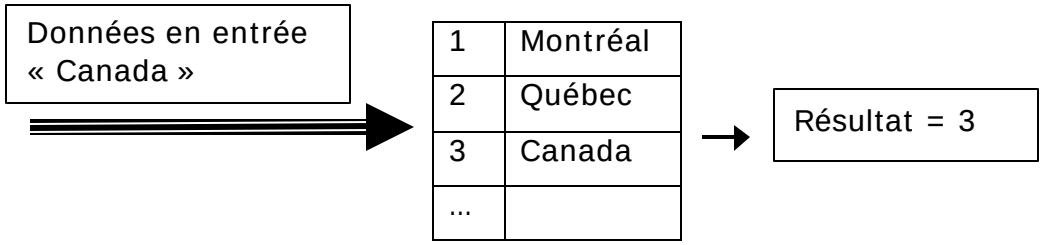

Figure 2.5 - Exemple de fonctionnement d'une CAM

Les CAM effectuent une recherche parallèle dans un tableau de cellules de mémoire, en fonction de la valeur donnée en entrée, voir la Figure 2.5. La CAM cherche à déterminer si l'élément à rechercher se trouve dans le tableau. Un inconvénient des CAM (binaire) concerne la flexibilité des recherches. En effet, les CAM ne permettent pas d'effectuer une recherche sur juste une sous-partie des éléments donnés en entrée. Quand on donne une entrée précise, elle retourne l'adresse de la donnée si elle est présente. Les TCAM « Ternary CAM » remédient à ce problème,

d'où le terme ternaire (CAM ternaire) qui permet de supporter la notion d'entrée indifférente.

2.6.3.2 TCAM

Les CAM et TCAM sont deux mémoires qui, contrairement aux RAM (Random Access Memory), prennent en entrée des données et retournent en sortie la ou les adresses correspondantes. Ce sont des mémoires à contenu adressable. Elles permettent toutes les deux une recherche rapide, car cette dernière se fait en parallèle. Par rapport à la CAM, la TCAM est plus flexible parce qu'elle permet l'utilisation de « don't care » qui spécifient les bits non significatifs dans une donnée. Cela veut dire que pour trouver une donnée dans une TCAM, il n'est pas nécessaire d'en donner que les bits déterminants pour l'identification de la donnée. La TCAM peut produire plusieurs appariement pour une même entrée. Ceci impose d'y inclure un encodeur de priorité.

Chaque case mémoire est composée d'un préfixe, ce dernier est un couplet (*valeur*, *masque*). Les deux ont la même longueur, soit (W) bits. La fusion de ces deux séries de bits facilite la recherche sur des sous parties d'une chaîne de bits.

Tableau 2.11 - Valeurs possibles d'une cellule TCAM

Valeur	Masque	Valeur Ternaire
0	0	Don't Care
0	1	0
1	0	Don't Care
1	1	1

La *valeur* dans le préfixe peut être: 0, 1 ou X (n'importe quelle valeur pour X sera ignorée). Certains auteurs parlent de « trits » [SCA00] et affirment que le X est une valeur (ternaire), mais ils ne font que déguiser

l'utilisation du masque. Le *masque* détermine la longueur significative de la *valeur* dans le couplet. Les bits les plus significatifs du *masque* sont mis à 1, les autres sont mis à 0. Le Tableau 2.11 résume les valeurs possibles d'une TCAM.

L'utilisation de la TCAM n'est pas limitée aux adresses IP. Il est même possible de l'utiliser pour sa particularité de recherche en parallèle comme une CAM, tout en exploitant la possibilité d'ignorer des sous-ensembles des données.

CHAPITRE III

3 MÉTHODES EXISTANTES POUR LE TRAITEMENT DES PAQUETS

Ce chapitre a pour but d'expliquer et de revoir les méthodes existantes pour la classification des paquets. Une revue de littérature a été faite et une synthèse de ces articles est présentée. Nous expliquerons pourquoi ces méthodes sont inadéquates pour notre situation et pour le cadre de cette recherche.

Par la suite, une revue des équipements d'interconnexion des réseaux incluant l'architecture explorée dans le projet sur la conversion de protocoles du GRM sera présentée, ainsi que ses modules principaux seront présentés et expliqués.

Le processus d'identifier les protocoles et de les catégoriser dans un même flot de données se nomme la classification de paquets. Chacune des méthodes de classification des protocoles présentées par la suite opère sur un nombre fini de protocoles. On ne classifie pas les protocoles inconnus. Les mécanismes de découverte par apprentissage ne sont pas abordés dans ces articles, mais ils le seront dans le chapitre 4.3 avec nos méthodes proposées.

En général les méthodes décrites dans cette section se fixent un ensemble de règles permettant la différenciation des protocoles et compilent un graphe ou un arbre permettant l'ordonnancement des tests à effectuer en fonction du niveau (OSI) du protocole. L'intérêt de la compilation est de minimiser le nombre de tests à effectuer sur un paquet et de paralléliser une partie du code.

- **Articles : “Efficient demultiplexing of network packets” [JAY94]**

Cet article constitue un excellent point de départ pour la compréhension du processus de classification de paquet. La première partie de cet article expose la problématique générale de la classification. La deuxième partie présente une méthode inspirée par la reconnaissance linguistique (langage recognition). La section cinq contient une synthèse des travaux concernant CSPF[MOGUL87], BPF[MCCANNE93], MPF et Pathfinder [BAILEY94].

- **Articles : “The CMU/Stanford packet filter” [MOG87]**

Il s'agit d'une des premières implémentations de filtre de paquets. La méthode de classification repose sur un arbre où chaque nœud intermédiaire représente une opération booléenne de type « ET / OU » et chaque nœud final effectue une comparaison.

Le problème de cette méthode provient de l'arbre statique (pas de composition de règles) qui induit de nombreuses redondances dans les tests à effectuer. De plus, la fragmentation et les en-têtes de tailles variables ne sont pas gérées. Néanmoins cet article constitue une référence sur laquelle s'appuie les méthodes décrites par la suite.

- **Articles : “BSD Packet Filter” [MCC93]**

Le BSD packet filter (BPF) constitue une des premières tentatives d'optimisation du filtrage de paquets. Il s'agissait de réduire le nombre de test à effectuer et d'implanter la classification de manière logicielle au niveau le plus bas possible (au niveau d’un système d’exploitation en l'occurrence) pour des raisons de performance.

L'originalité de cette méthode réside dans la programmation basée sur des pseudos registres de type assembleur qui a permis à l'époque un gain de performance remarquable. Concernant l'ordre des tests à effectuer, le BPF n'utilise pas un arbre de décision mais un graphe acyclique qui permet l'élimination des tests redondants. De plus, BPF apporte le support des champs de tailles variables.

Au chapitre des reproches figure l'impossibilité de composer des règles de filtrage. En effet, il est nécessaire de créer un nouveau BPF pour chaque application utilisateur cherchant à utiliser le filtre de paquet. De ce fait, cette solution supporte mal l'augmentation du débit.

- **Articles : "PATHFINDER" [BAI94]**

Pathfinder repose sur un système de graphe acyclique contenant des cellules (pointeur vers une région du paquet). La juxtaposition de plusieurs cellules permet de définir des règles d'identification des protocoles.

L'intérêt de PathFinder est qu'il gère la fragmentation au moyen du traitement différé de paquets, lorsque celui-ci est fragmenté. Il gère les en-têtes de tailles variables grâce aux références relatives. La méthode proposée peut être implémentée aussi bien de manière matérielle que logicielle.

Cet article explique l'intérêt d'utiliser une solution matérielle pour la classification d'un nombre restreint de protocoles (généralement les protocoles les plus utilisés). Le matériel agit comme une cache pour le module logiciel. Ce qui permet de traiter un nombre plus important de protocoles au prix d'une performance moindre.

Enfin, l'article comporte une étude de performance qui montre entre autres que la latence induite par le traitement des protocoles augmente de manière quasi-linéaire avec le niveau du protocole dans le modèle OSI.

- **Articles : “RFC (Recursive Flow Classification)” [GUP99]**

Cet article est plus récent, la montée en charge des algorithmes a été prise en compte dès le départ pour gérer un nombre important de protocoles. D'autre part, la méthode est optimisée pour effectuer des compositions de règles de filtrage. Cette méthode repose sur le découpage en fragments de chaque paquet. Ces fragments seront éventuellement traités en parallèle. Cet article considère également la dualité logicielle/matérielle et se distingue par l'étude de la consommation de mémoire de l'algorithme.

- **Articles : “PTree” [BEC01]**

Il s'agit d'un projet réalisé par des étudiants, inspiré de Pathfinder (pour la génération et l'optimisation de l'arbre des règles) et un projet appelé « net filter » (pour la capture et le filtrage niveau noyau). L'intérêt de ce projet réside dans la disponibilité du travail effectué sur internet. L'implantation logicielle a été achevée, la solution matérielle est en cours de développement.

- **Articles : “BPF+” [BEG99]**

Cet article se concentre sur l'optimisation du problème de classification par le biais d'algorithmes d'optimisation de circulation des flux de données (appelés élimination des prédicats redondants). La particularité de la solution réside dans la modularité de l'architecture proposée. La solution comporte une interface faisant la traduction du langage de description et un optimiseur de code indépendant. De même, un autre

module indépendant est chargé de garantir la validité du code et possiblement la sécurité du système.

- **Articles : “High speed policy-based packet forwarding” [LAK98]**

Il s'agit encore d'un article orienté optimisation qui présente trois algorithmes. Le premier permet la programmation en parallèle des filtres. Le second garantit l'utilisation efficace de la mémoire disponible. Il tient compte du nombre de paquets à traiter (ceux présents dans la file d'attente) et du temps accordé pour le traitement de ces paquets. Le troisième s'applique au cas particulier de règles de filtrage sur deux champs (par exemple dans le cas du trafic multicast ou dans la mise en place de routage à base de politiques).

- **Articles : “Fast and scalable layer four switching” [SRI98]**

Dans la même optique que l'article précédent, cet article propose deux algorithmes d'optimisation. Le premier traite des règles de filtrage basées sur deux champs en construisant des tableaux d'essai (*grid of tries*). Le second algorithme s'intéresse à la réduction des temps de réponses lorsque le nombre de règles de filtrage est élevé et que l'on dispose de beaucoup d'espace de stockage.

- **Articles : “Algorithms for packet classification” [GUP01]**

Cet article présente différents types d’algorithmes de classification. Il discute aussi la classification sur plusieurs champs qui est plus complexe et qui nécessite des algorithmes plus performants. Il décrit quatre catégories d’algorithmes : «recherche de base», «recherche géométrique», «recherche par heuristique» et «recherche accélérée par du matériel dédié».

Les flots sont caractérisés par certaines règles qui sont appliquées sur les paquets. Par exemple, tous les paquets avec les mêmes adresses de source et destination pourront être classés dans un même flot. Cet article a inspiré en partie ce mémoire.

- **Articles : “An architecture for fast and flexible packet classification” [CLA01]**

Cet article sur la classification propose une architecture programmable, comme un co-processeur, pouvant faire la classification à une vitesse de OC48. Certains exemples démontrent l'emploi d’algorithmes utilisés en conjonction avec l’architecture proposée soit pour aiguiller, transférer ou filtrer des paquets allant jusqu’au niveau 7 de l’OSI. Une revue de certaines mesures de complexité est présentée (space, time, power, update, rule).

- **Articles : “Network Processing in Content Inspection Applications” [WEL01]**

Cet article démontre bien la problématique d’aller au-delà des niveaux 4 de l’OSI pour faire une identification avancée faisant l’inspection des données du niveau application. Il propose un langage de haut niveau utilisable par un chip de haute performance pour la recherche et comparaison de caractères. Cet article pourrait inspirer la suite de nos travaux.

- **Articles : “Fast incremental update on Ternary-CAMs for routing lookups and packet classification” [GUP00]**

Cet article explique les TCAM et leur fonctionnement. Il propose deux algorithmes de mise à jour rapides des TCAM pour des délais minimaux même dans les pires cas. Cependant les auteurs ne semblent pas avoir lu l’article suivant [USP] avant de publier leur article.

- **Articles : “Ternary content addressable memory (CAM) having fast insertion and deletion of data values” [USP]**

Ce brevet américain décrit une mémoire TCAM ayant une grande capacité de comparaison en parallèle avec la possibilité d’insertion et suppression rapide de ces éléments. Cette mémoire contient une série de mémoire CAM ayant une zone de données et une zone de masques. Ce brevet diminue la nécessité de [GUPTA00] mais la valide par une implémentation matérielle.

3.1.1 Conclusion de la synthèse

Certaines des méthodes présentées ne sont que partiellement adéquates alors que d’autres le sont moins. Elles se concentrent surtout sur les aspects d’optimisation de cas complexe de la classification. À l’exception des articles faisant des implémentations matérielles, tel que [WEL01] et [USP], qui sont presque directement ré-utilisables, les autres proposent des algorithmes de base visant une implémentation logicielle : une limitation importante lorsqu’il est question de transmission à haut débit.

Notre situation demande plus une identification immédiate et rapide du protocole dans le but de faire la conversion immédiate d’une trame ou d’un paquet.

3.2 Interconnexion des réseaux hétérogènes

Pour le cas des réseaux hétérogènes, les équipements d’interconnexion réseau doivent faire correspondre les services offerts par chacun des réseaux. En d’autres mots, ils doivent faire la correspondance entre les services remplissant des fonctions similaires. Dans le cas où la correspondance serait impossible, des valeurs doivent être produites. Il

est impératif que la présence de tels dispositifs soit transparente aux utilisateurs des réseaux.

Les processeurs de paquets réseaux permettent la manipulation de trames provenant de différentes interfaces et de différents protocoles appartenant à plusieurs couches de l'ISO.

Les fonctionnalités normalement incluses dans ces processeurs sont les opérations de filtrage, les conversions de protocoles, les commutations de paquet, etc. Ces processeurs sont spécialisés par leurs jeux d'instructions dans la reconnaissance, la manipulation et le traitement des paquets et par la présence de co-processeurs matériel qui accélèrent les calculs.

L'augmentation de la vitesse des réseaux tend à créer une saturation lors de la conversion entre les interfaces d'interconnexion. Il est donc devenu nécessaire d'accélérer ces processus de conversion. Ces accélérations peuvent se faire par l'intégration matérielle des algorithmes utilisés dans une version logicielle. Cependant il est difficile de reproduire la flexibilité offerte par les logiciels au niveau matériel.

3.2.1 Les convertisseurs de protocole

L'invention des réseaux hétérogènes a amené un besoin grandissant d'interconnexions rapides entre les réseaux. Il existe trois catégories de convertisseurs de protocole classiques faisant cette interconnexion. Ils opèrent à différents niveaux. Les ponts, les routeurs et les passerelles. [OMA98 chap.2] La Figure 3.1 illustre le rôle de chacun de ces trois types de convertisseurs de protocoles : ils peuvent soit rester au niveau des données physiques ou s'étendre jusqu'à la couche application du modèle de l'OSI.

Les passerelles sont souvent des logiciels qui s'occupent de la conversion des protocoles jusqu'a la couche application. Cette situation augmente pour chaque niveau la complexité du traitement. Les logiciels sont souvent utilisés pour donner une flexibilité de conversion. C'est à cause d'une telle complexité et d'un manque de performance pour le traitement des grands débits, qu'une nouvelle architecture est en développement.

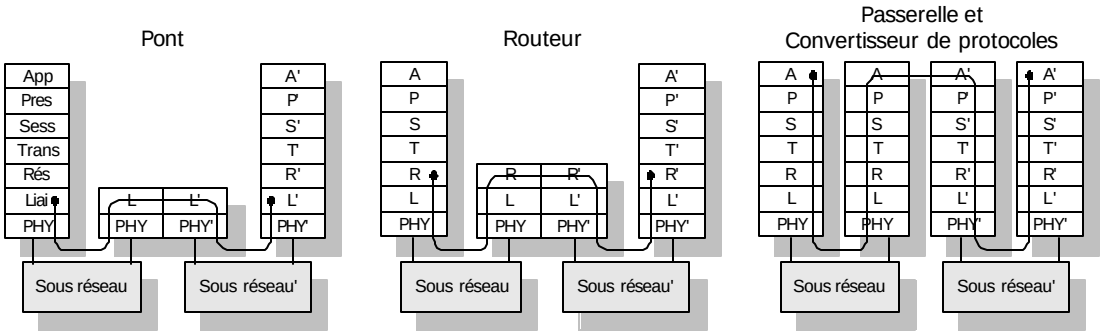

Figure 3.1 - Trois catégories de convertisseurs de protocoles

L'accélération matérielle des parties de l'algorithme ne se limiterait, dans notre cas, qu'à la couche de liaison des données. Or, nous avons montré qu'un convertisseur de protocoles, Figure 3.1, pouvait également opérer sur les couches supérieures du modèle OSI.

3.3 Profilage d'algorithmes

Le profilage consiste à identifier les parties d'une spécification logicielle (les algorithmes) qui consomment un temps de calcul élevé et qui méritent d'être accélérées. Il existe deux types de profilage logiciel. Il y a le profilage dynamique et le profilage statique.

Le profilage statique consiste à analyser le code source. Cette technique ne requiert pas l'utilisation de données d'entrées. Elle consiste à bâtir un graphe de flot de contrôle dont tous les branchements possibles seront extraits. Le chemin le plus long pouvant être exécuté sera trouvé. Des

informations telles que les bornes supérieures et inférieures des boucles sont nécessaires afin de résoudre le graphe. Une fois le chemin le plus long trouvé, les blocs de base sont ordonnancés et leur exécution est estimée à l'aide d'un modèle de l'architecture du microprocesseur ciblé.

Le profilage dynamique, consiste à exécuter plusieurs fois le programme avec différentes données d'entrées. Cette méthode exige que le flux des données entrant couvre tous les branchements du code évalué. Les cycles d'horloge requis pour chacune des exécutions sont comptabilisés et utilisés pour déterminer le temps d'exécution maximal. Une telle analyse peut devenir presque impossible sans l'aide d'outils.

Un profilage dynamique est nécessaire dans le cas ou le jeu d'instructions n'est pas supporté par l'outil utilisé comme Cinderella [CIN]. Ce type de profilage est plus pertinent si le temps de calcul est variable et qu'il dépend des données, par exemple dans le cas de boucles avec variables non bornées.

3.4 Le convertisseur de protocole du GRM

La première étape franchie par le groupe qui étudie la conversion de protocoles au GRM, fut de créer un convertisseur de protocole logiciel entre le protocole IPv4 et le protocole FireWire IEEE-1394. Ce convertisseur est spécifié en logiciel, (en C++).

La deuxième étape fut le profilage du code logiciel pour déterminer les co-processeurs utiles à l'accélération du processus de conversion des protocoles.

La troisième étape avait pour but de ne pas se limiter à la transposition des couches liaison de données. Pour pouvoir accélérer la conversion,

l'équipe a décidé de créer une nouvelle architecture matérielle, nommée convertisseur de protocoles.

L'architecture est développée et une révision dans le but de la rendre simulable est en cours. Il n'existe pas encore de compilateur C/C++ pouvant compiler du code pour cette architecture mais le jeu d'instructions assembleur de la portion programmable est prêt avec un compilateur de langage machine.

3.4.1 Les buts recherchés

Il serait intéressant d'avoir une solution permettant au convertisseur de protocoles de couvrir toutes les couches de protocole avec une méthode générique. Cela revient à réaliser une nouvelle architecture permettant la manipulation de paquets réseaux de tous niveaux, tel qu'une passerelle en matérielle mais facilement reconfigurable.

Le Groupe de Recherche en Microélectronique (GRM) de l'École Polytechnique de Montréal est en cours de création d'une plate-forme pour la conversion de protocoles. Une des premières étapes matérielles a été de cibler la conversion de IEEE 1394 vers IEEE 802.3 (Ethernet) et vice versa. Un tel convertisseur de protocoles pourrait être utilisé comme passerelle dans les réseaux à domicile vers les réseaux externes: dans ce cas un réseau IEEE 802.3 vers un réseau IEEE 1394.

IP version 4 sur IEEE 1394 est proposée dans une recommandation de norme pour l'Internet (Internet draft [RFC2734], [RFC3146]. Elle définit les éléments nécessaires pour permettre à IPv4 de fonctionner sur un réseau 1394. Il existe un intérêt réel et croissant sur le sujet, même les fournisseurs de logiciels comme Microsoft ont adapté leurs nouveaux systèmes d'exploitation pour supporter TCP/IP sur 1394. MS-Windows Me et XP supporte l'interconnexion via une simple interface physique 1394.

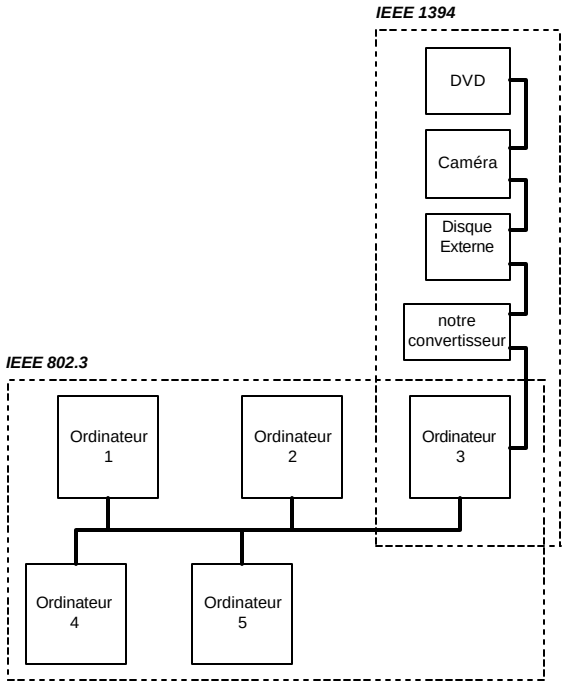

Figure 3.2 - interconnexion possible entre IEEE 802.3 et IEEE 1394

Le schéma de la Figure 3.2 démontre que l'ordinateur 3 peut agir en tant que passerelle entre les deux réseaux. Une action possible d'un usager distant, tel que sur l'ordinateur 4, serait d'envoyer une commande par le réseau IEEE 802.3, reçue et convertie sur l'ordinateur 3 pour être expédiée en format IEEE 1394 et envoyée à la caméra. La caméra pourrait alors capturer une image et la transmettre à l'ordinateur 3 qui la traduirait dans le format IEEE 802.3 pour l'envoyer vers l'ordinateur 4. Certains services d'adressage sont nécessaires tels que définis dans [RFC2855].

3.4.2 Architectures développées

La Figure 3.3 et la Figure 3.4 présentent les diagrammes blocs des spécifications fonctionnelles, en cours de développement par l'équipe du GRM et qui définit l'architecture du convertisseur de protocoles. Nous les

présentons en référence. Il s'agit de la version originale et de la plus récente version créée, à laquelle nous avons participée.

3.4.2.1 Architecture version 1

La première version a été développée avec la collaboration d'un groupe du cours ELE6305 à l'École Polytechnique sous la direction de Jean-Marc Tremblay, étudiant candidat à la maîtrise qui a débuté ce projet.

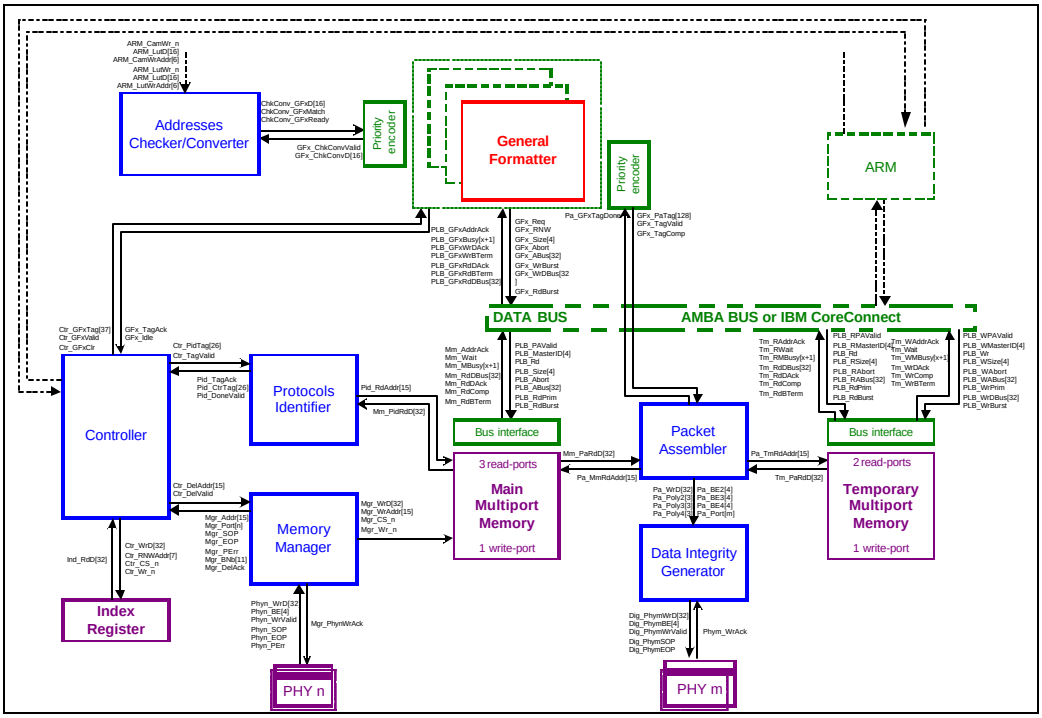

Figure 3.3 - Schéma bloc du convertisseur de protocoles (version 1)

3.4.2.2 Architecture version 3 (en développement)

La version 3 présentement en développement a été élaborée suite aux discussions sur les besoins pour l'identification rapide. Elle évite les copies multiples des données d'une mémoire à l'autre et a un flot de données plus optimal qui diminue la latence induite par la conversion.

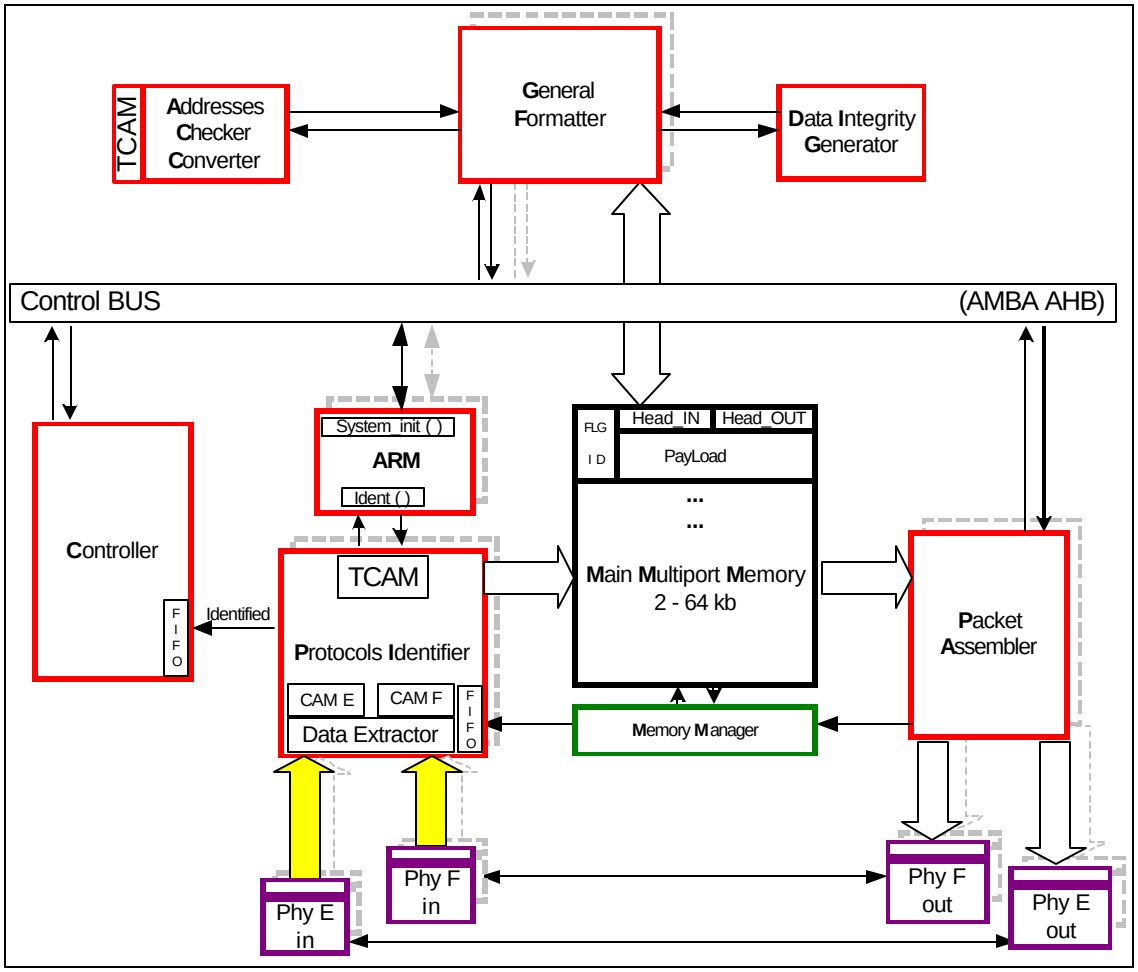

Figure 3.4 - Schéma bloc du convertisseur de protocoles (version 3.0, en développement)

3.4.3 Limitation matérielle

Les mémoires utilisées dans l'architecture sont des mémoires embarquées (sur FPGA). Elles sont de petites tailles, soit entre 100 kilo-octets et 2 méga-octets. Nous ne disposons pas de mémoires CAM ou TCAM directement utilisables dans une version ASIC du convertisseur. Nous avons donc exploré la création d'une mémoire CAM expérimentale sur un FPGA.

3.4.4 Limitation logicielle

Les outils de développement utilisant le processeur ARM sont limités. Un modèle fonctionnel du co-processeur d'identification à été créé sur l'outil de co-simulation logiciel/matériel Seamless CVE de Mentor Graphics. Le code source C, joint en annexe, a été développé avec les compilateurs de ARM, CodeWarrior et Microsoft Visual C. Le tout a été testé sur une plate-forme de prototypage ARM et sur un PC.

3.4.5 L'Engin de Formattage (General Formatter : GF)

Le GF est au cœur du processus de conversion. Suite à son activation par le contrôleur, il exécute le programme de conversion qui est implanté dans la mémoire d'instructions. Chaque GF est en charge d'un paquet. Il s'agit essentiellement pour lui de faire des accès à la mémoire multiport d'entrées pour y chercher les en-têtes des différentes couches à convertir. Les résultats des conversions sont ensuite emmagasinés dans la mémoire multiport temporaire, dans laquelle chaque paquet possède un espace mémoire alloué. Lorsque l'ensemble de la conversion est terminé, le GF avertit le coprocesseur d'assemblage de commencer son traitement. Dans la première version, les accès aux différents modules se font par des bus dédiés.

3.4.6 Le processeur d'identification des protocoles

Sujet de ce mémoire, ce co-processeur permet d'identifier les protocoles entrant de tous les niveaux du modèle OSI. Il permet d'identifier les protocoles qui encapsulent les paquets devant être remaniés avant leur ré-expédition. Cette identification permet de faciliter et même d'accélérer le temps de conversion que prendront les engins de formatage (GF) pour appliquer des transformations appropriées sur le message

source appelé datagramme. Une fois que le protocole entrant est connu, les co-processeurs de conversion sauront s'ils doivent ou non effectuer des conversions.

Ce module a été placé, dans la première architecture du convertisseur, entre le contrôleur et la mémoire multiport. Pour chaque paquet reçu dans la mémoire, le contrôleur transmet au co-processeur d'identification l'adresse du paquet en mémoire à être identifier et en réponse il devra retourner le résultat au contrôleur.

Les nouvelles architectures du convertisseur tiennent compte de l'apprentissage fait. La version 3 tient compte des résultats et des besoins découverts, qui seront présentés en partie dans ce mémoire. Il a semblé normal de commencer par identifier un paquet avant de vouloir faire autre chose sur celui-ci.

3.4.7 Détails des interfaces choisies

Cette section complémente et offre plus de détails sur les interfaces choisies dans le cadre du convertisseur de protocole du GRM. Elle donne les caractéristiques et les chiffres nécessaires aux calculs présentés plus loin.

3.4.7.1 Giga bit Ethernet

Les propriétés du protocole giga bit Ethernet sont les mêmes que le protocole Ethernet décrit à la section 2.2.1. Ce protocole respecte IEEE 802.3 et l'en-tête est sur 15 octets suivi de son CRC et de ses extensions. Le paquet est divisé en champ décrit dans le Tableau 3.1.

Tableau 3.1– Description du paquet Giga-Ethernet

Nom	Taille en bits	Fonction
Début de trame	8 (1 octet)	Toujours 0xAB, bit 0 d'abord
Adresse de destination	48 (6 octets)	Adresse de destination.
Adresse source	48 (6 octets)	Adresse source
Longueur / type	16 (2 octets)	Taille des données ou indicateur du type de paquet
Données	1500 octets max	Données
Remplissage	x	Remplissage si la taille de la trame < que la taille min de trame (pour Gb Ethernet, c'est 64 octets)
Vérificateur de Séquence des Trames	32 (4 octets)	CRC
Extension	y	Information concaténée à la fin de la trame. Nous ne l'utiliserons pas pour l'instant mais rapport au Giga-Ether.

Les trames conformes à ce protocole possèdent ces caractéristiques :

- Taille minimale des trames: 64 octets (DA à PAD inclusivement)
    - Plus 12 octets pour le MAC (PRE, SDF et CRC)
- Taille maximale des trames: 1518 (DA à PAD inclusivement)
    - Plus 12 octets pour le MAC (PRE, SDF et CRC)
- Vitesse de transfert: 1000 Mb/s

La vitesse de transfert, en Mb/s, se calcule avec le nombre de bits seconde de la couche liaison (64 ou 1518 octets) mais aussi avec le nombre de bits seconde de la couche MAC (synchronisation physique) de (12 octets). Le 12 octets ne devrait jamais être oublié lors des calculs de performance d'un réseau.

3.4.7.2 IEEE-1394

Ce protocole est décrit ici pour en éclaircir la compréhension et en faire une introduction. La norme IEEE-1394 est un protocole qui inclut certains niveaux de l'ISO, de la couche physique à la couche transport et la couche application. La Figure 2.1 affichait la comparaison des couches de l'OSI avec le modèle de 1394.

Le nom Firewire est la version de Apple et i.LINK est le nom de la version de Sony de l'interface IEEE-1394. Elle utilise deux types de méthodes de transport : isochrone et asynchrone.

La méthode de transport isochrone est utilisée pour les données sensibles au temps. Elle garantit la largeur de bande et la latence requise pour le transfert à haut débit de données temps réel, notamment des données vidéo (DV) et audio numériques. Lors de sa connexion, un nœud isochrone fait une demande de bande passante à l'Isochronous Resource Manager. Celui-ci assigne alors un numéro de canal (0 à 63) au nœud. La largeur de bande est divisée en unité de largeur de bande par cycle (un cycle dure 125 us).

La méthode asynchrone est utilisée pour le transport d'informations de contrôle. Par exemple, le Bus Master signale à une caméra vidéo qu'elle peut commencer à transmettre ses données sur son canal réservé en lui envoyant un message asynchrone. La communication asynchrone est aussi utilisée pour la connexion d'appareils plus lents (imprimantes, modems, etc) où l'intégrité est importante comme avec le protocole TCP. Pour transmettre de façon asynchrone, un nœud n'a qu' à utiliser le temps de transmission asynchrone disponible dans un cycle (fairness interval). La méthode asynchrone a une priorité plus basse que la

méthode isochrone et n'entre en jeu que lorsque tous les nœuds qui disposent d'un canal isochrone ont terminé d'émettre.

Tableau 3.2 - Description du paquet IEEE1394

Nom	Taille (bits)	Fonction
ID de destination	16	Identificateur (adresse) de destination
Transaction label	6	Nom unique donné à chaque transaction.
Code d'essai	2	Indique si la transaction est une nouvelle tentative après un premier essai manque. Forcer à 01 pour cette implantation (01 = non-supporte).
Code de transaction	4	Identifie le type de paquet. Le code de ce type représente soit isochrone ou asynchrone.
Priorité	4	Fixé à zéro
ID Source	16	Identificateur (adresse) de la source
Code de Réponse	4	Identifie si le paquet est une réponse à une requête précédente.
Réservé	44	Réservé pour expansion future: mettre tout à zéro.
Longueur de données	16	Taille des données
Code de Transaction Étendu	16	Pas utilise dans cette implantation: forcer à 0x0000
En-tête de CRC-1	32	CRC calcule à partir de l'en-tête
Données	2048 octets ou 4096 octets	Bloc de données (max)
CRC-2 des données	32	CRC calcule à partir des données

Le paquet est divisé en champ décrit dans le Tableau 3.2. En plus de ces champs l'interface 1394 possède les caractéristiques suivantes:

- Taille minimale des trames: 24 octets
    - ID à CRC2 inclusivement (excluant les données)

    - Excluant l'en-tête de transport de RTP
- Taille maximale des trames avec en-tête et données :
    - 1394a = 2072 octets
    - 1394b = 4148 octets
- Vitesse de transfert maximale :
    - 1394a = 400Mbps
    - 1394b = 800 Mbps

La conversion du protocole isochrone 1394 vers Ethernet/IP/UDP et vice versa peut être accomplie en associant et insérant des valeurs par défaut sur les champs qui n'ont pas d'équivalence.

Tableau 3.3 - Association entre 1394 et 802.3

Protocole 1394	Pile de Protocole 802.3
Isochronous/Asynchronous Stream	UDP/IP
Asynchronous block	TCP/IP

Chaque protocole de IEEE 1394 peut-être associé (Tableau 3.3) avec un protocole correspondant de la famille IEEE 802.3. Ce qui signifie qu'un transfert de bloc asynchrone de réseau IEEE 1394 ferait appel à un transfert TCP/IP sur les réseaux IEEE 802.3. Dans le même sens qu'un transfert isochrone ferait appel à UDP/IP.

CHAPITRE IV

4 MÉTHODES D'IDENTIFICATION

Ce chapitre est la partie principale de ce mémoire. Il décrit les théories existantes d'identification et celles développées dans le cadre de ce mémoire. Ce chapitre va répondre aux questions relatives à l'identification des protocoles. Nous débutons par une description des méthodes existantes avant de proposer des méthodes adaptées à notre application.

4.1 Algorithme d'identification logicielle

Les méthodes d'identification des protocoles existantes varient d'une application à une autre. Cependant les logiciels d'analyse de paquets, les routeurs, les ponts ou les passerelles utilisent tous des méthodes similaires. Peu de détails sont disponibles sur la manière précise utilisée dans les appareils existants pour l'identification de protocoles. Il tient à chaque équipement réseau de respecter les standards définis. Voici donc comment ces équipements font pour identifier les protocoles de bas niveau.

4.1.1 Méthode de base

Cette section a pour but d'expliquer diverses méthodes d'identification des protocoles des niveaux inférieurs 2 à 4, du modèle OSI, empilés dans une trame Ethernet.

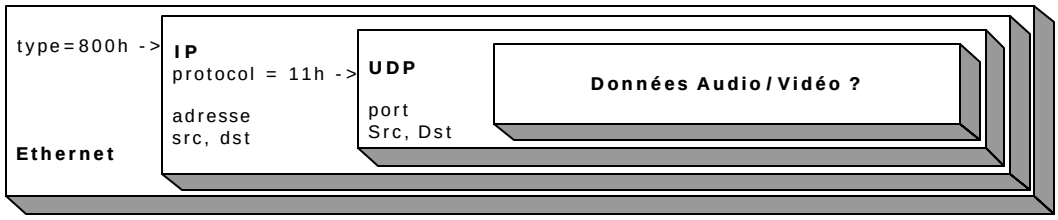

Figure 4.1 - Empilement des protocoles

La Figure 4.1 montre l'imbrication des protocoles des quatre premiers niveaux de l'OSI. Il s'agit d'Ethernet (LLC), d'IPv4 et d'UDP. Ce dernier peut finalement contenir du RTP transportant des données temps réel pour l'audio ou la vidéo. Il peut aussi contenir n'importe quel autre protocole connu ou non, puisque le contenu de UDP n'est pas annoncé.

Ces trois protocoles encapsulés l'un dans l'autre sont facilement identifiables. Ethernet, niveau liaison, est connu par le numéro de l'interface. Ethernet annonce, par son champ « type », le protocole de niveau réseau. Par exemple, si le type est 0x800, le protocole est IP. Le protocole IP annonce par son champ « protocole » le protocole de niveau transport. Le protocole UDP, par exemple, est identifié par la valeur 0x11. UDP transporte les données de niveau application. Ces données sont utilisées par l'usager, par exemple, dans un logiciel de vidéo-conférence.

La Figure 4.1 propose une façon simple pour identifier les trois premiers niveaux de protocoles (Quatre premiers niveaux du modèle OSI) pour le cas de l'empilement Ethernet/IP/UDP.

4.1.1.1 Gestion des en-têtes de longueur variable

Comme expliqué précédemment, la classification se fait à partir de l'analyse des champs contenus dans les en-têtes. Lorsque celles-ci sont de tailles fixes, il suffit de définir une référence absolue, au début de la trame, pour localiser le champ à analyser dans la trame. Cependant,

certains protocoles dont IP et TCP comportent des en-têtes de tailles variables. Il faudra alors calculer la longueur de l'en-tête et utiliser des références relatives. Le problème n'est pas encore complet, car pour ces premiers niveaux il existe aussi le problème de la fragmentation.

4.1.1.2 Gestion de la fragmentation

Certains protocoles (dont IP) utilisent la fragmentation pour prendre en compte l'hétérogénéité des réseaux. Par exemple, si l'on utilise UDP, il est possible que la taille des datagrammes dépasse la capacité des trames Ethernet, ce qui implique la fragmentation du paquet, par conséquent un ré-assemblage à sa réception est nécessaire. Il est possible que certains champs impliqués dans un paquet soient présents, en partie, dans deux fragments différents. Il sera alors nécessaire de différer la classification du premier paquet jusqu'à la réception du second ou de la totalité de la trame.

4.2 Implémentations existantes

Autant les implémentations de haut niveau, telles que les passerelles et les logiciels d'analyse réseau, doivent tenir compte de la fragmentation et des calculs relatifs. Un examen des moyens existants faisant l'analyse du réseau et le procédé d'identification du contenu des trames est nécessaire pour comprendre ce problème d'identification.

Certains logiciels d'analyse de réseau « Paquet sniffer » et analyseur de paquets capturent en premier et font une analyse différée des paquets. Ces outils permettent de faire l'analyse des flots de données et de détecter certains problèmes dans le réseau. Certains de ces logiciels proviennent du domaine public. Ils rendent disponibles leurs codes sources d'analyse. Ce code est complexe mais disponible et il peut-être utilisé pour la définition d'algorithmes d'identification.

Un exemple d'analyseur de réseau est le logiciel « Ethereal » [ETH]. Il permet l'analyse des paquets capturés, d'une interface Ethernet, avec le logiciel winPcap [PCAP] qui est un logiciel d'interception de type « driver » développé par « Netgroup - Politecnico di Torino (École Polytechnique de Torino, Italy) ». Les données capturées sur la carte réseau sont ensuite analysées avec un logiciel tel que Ethereal.

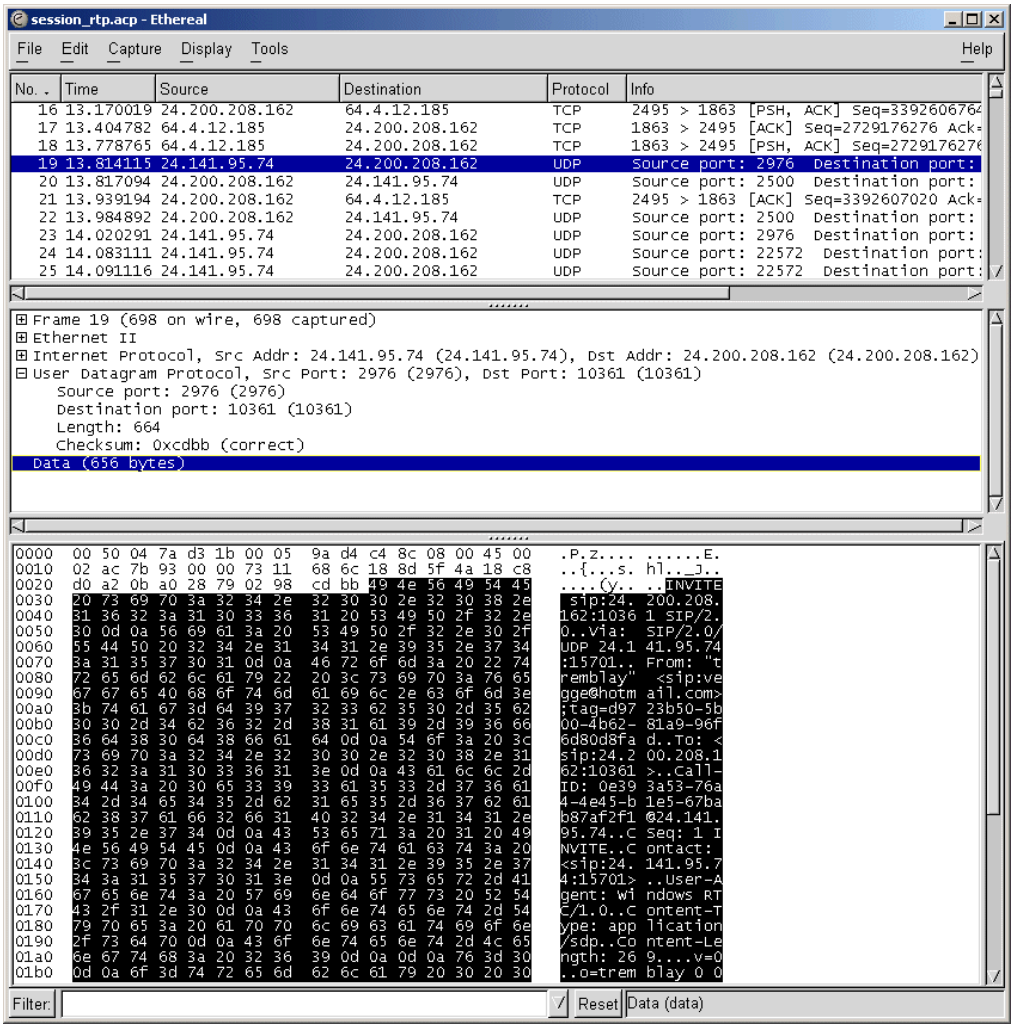

Figure 4.2 – Détail de la trame 19 présenté par Ethereal

La Figure 4.2 est divisée en 3 parties du haut vers le bas. En haut on voit la liste des trames capturées, au milieu l'interprétation des structures de

données, finalement au bas, en noir, la représentation des données, à gauche en hexadécimal et à droite en format ASCII.

La Figure 4.2 montre que les données Ethernet 802.3, (Frame no. 19), Ethernet 802.2, IP et UDP sont facilement identifiés, tandis que le contenu de UDP n'est pas encore décodé. L'analyse de UDP montre le contenu du paquet, les données transportées par UDP. La lecture de la partie de droite, les données ASCII, commencent par « INVITE sip » ce qui indique que le protocole transporté est SIP.

Ce logiciel donne la possibilité de spécifier comment décoder les données. La Figure 4.3 montre ces possibilités et indique que dorénavant le port source UDP 2976 de l'adresse source IP 24.141.95.74 sera décodé comme du SIP pour le reste de la session ou de l'analyse courante.

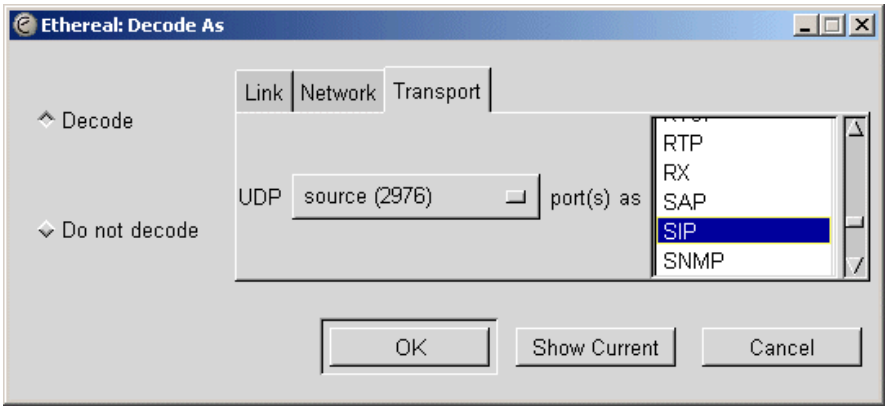

Figure 4.3 - Ethereal exploité pour décoder la trame 19

La Figure 4.4 montre qu'une fois les données décodées, les champs de SIP apparaissent aussitôt et le protocole SDP inclut ici dans SIP est alors identifié et affiché. Il a été détecté par un champ de SIP, non affiché ici mais présent. Ce champ se nomme « Content-Type » et la valeur inscrite est tout simplement « application/SDP ».

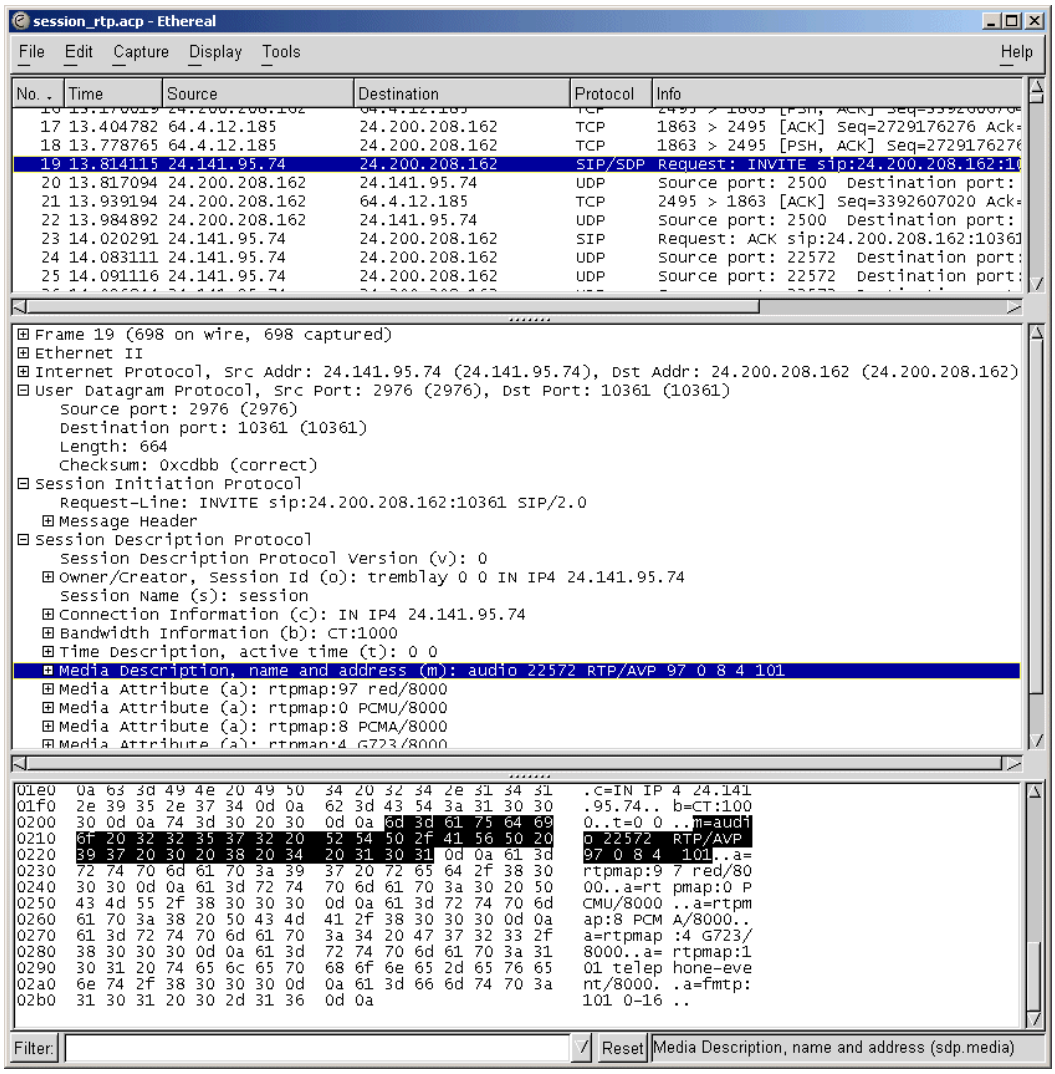

Figure 4.4 -Ethereal traitant la trame 19 de type SIP/SDP

Lorsqu'on étudie la ligne indentée, dans la partie du centre de la Figure 4.4, on peut lire la description du média de SDP. Il indique le média annoncé. Le champ « Media Description » annonce ici une session audio transportée par RTP sur le port UDP 22572 de l'adresse 24.141.94.74.

En continuant le processus, on peut éventuellement décoder complètement chaque session, ce qui nous permettrait de connaître tous les types de paquets existants dans cette session capturée.

4.2.1 Analyse du réseau multimédia

L'analyse de la session nous porte à réfléchir sur les types de paquets dans le réseau lors des sessions multimédias. Les logiciels d'analyses de paquets permettent aussi de faire des calculs simples pour arriver à certaines conclusions. Ils permettent aussi de savoir quel sont les paquets importants à analyser ou leurs probabilités d'apparition lors de la session.

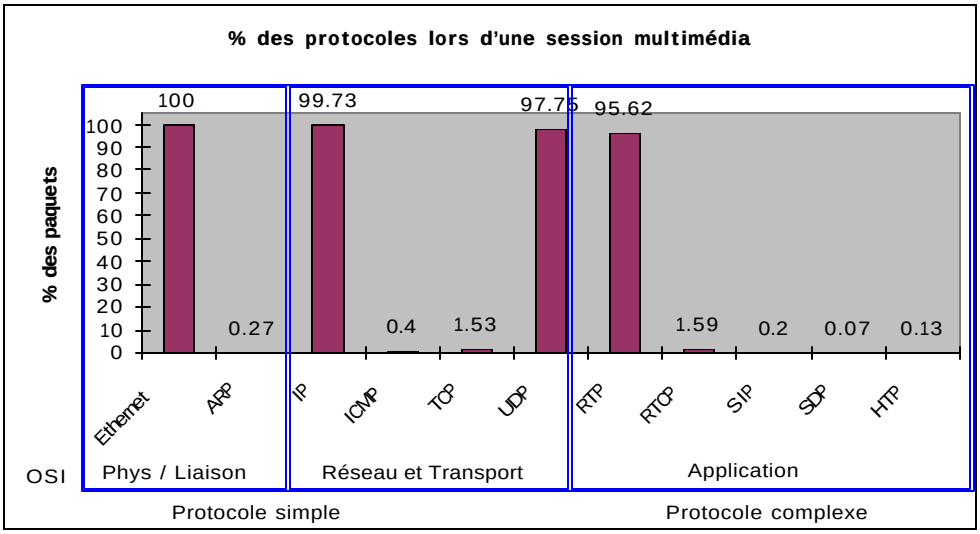

Figure 4.5 - Répartition des paquets lors d'une session multimédia

L'analyse de la session courante donne un exemple représentatif de ces probabilités. L'analyse de la répartition des paquets montre que évidemment 100% des paquets sont de type Ethernet, puisque c'est l'interface principale. 99.73% des paquets sont de type IP, puisque c'est le protocole réseau principal, ensuite le protocole le plus utilisé lors de session multimédia est UDP pour sa simplicité. Ces résultats sont conformes à la discussion du chapitre 2.2.1.3.

Les sessions multimédias sont généralement transportées par le protocole de transport UDP qui occupe 97.75% mais aussi complétées par RTP à 95.62%. Comme nous avons déjà expliqué dans le chapitre 2.3.1, RTCP

(1.59%) contrôle le flux de données de RTP tandis que SIP (0.2%) annonce les sessions RTP.

La probabilité de rencontrer des paquets SIP est une des plus basses, mais ces paquets sont aussi très importants, car ils annoncent les protocoles de type RTP. Ainsi ces pourcentages aident à comprendre que, dans un réseau contenant des données temps réel transportées par RTP, le pourcentage de paquets contenant un protocole de haut niveau tel que RTP est très grand. Par exemple, la Figure 4.5 montre que 97.5% des paquets UDP contiennent les protocoles RTP, RTCP, SIP ou SDP. Il est donc justifié, lors de sessions multimédias, d'identifier le contenu de tous les paquets UDP le plus rapidement possible, parce qu'ils représentent une grande partie de la bande passante du réseau lors de passage de sessions multimédias.

4.3 Méthode logicielle proposée

Notre méthode d'identification propose une analyse avec un mécanisme de découverte par apprentissage. Cela signifie que l'on recherchera, dans tous les nouveaux paquets UDP, des informations nécessaires à créer des signatures servant à identifier par anticipation les sessions et les protocoles de niveau application.

Chaque protocole, contenu dans le paquet UDP, n'est analysé qu'une seule fois par l'algorithme qui crée une signature pour le reconnaître à sa première ou à sa prochaine apparition. Comme nous l'avons fait manuellement au chapitre 4.2, la recherche des mots clés et l'interprétation des informations lues est nécessaire à l'identification des flots de niveau application. Notre algorithme ne fera que des recherches dans le but d'identifier les paquets UDP contenant les protocoles de haut niveau comme : RTP, RTCP, SIP et SIP/SDP.

4.3.1 Les signatures IP

Les signatures doivent être créées à partir d'informations immuables contenues dans un des protocoles ou dans l'empilement des protocoles. Une telle partie est l'adresse source et l'adresse destination de IP, qui identifie la source et la destination, ainsi que les ports UDP correspondants.

La signature IP ainsi qu'une référence au nom du protocole correspondant sera créée et emmagasinée dans une table de référence (appelé CAM_soft). Ces signatures serviront de condition d'identification des paquets dès l'arrivé d'un prochain paquet de même type.

La méthode d'identification utilise cette table de référence contenant des signatures, IP/UDP ou possiblement IP/TCP, des protocoles de niveau application et le nom du protocole identifié. Les signatures emmagasinées contiennent les adresses réseau (IP) source et destination, les ports de transport (UDP ou TCP) source et destination.

Tableau 4.1 - Exemple de matrice d'identification CAM_soft.

No	Addresses IP		Ports UDP ou TCP		Nom
	AS (source)	**AD** (destination)	**PS** (source)	**PD** (destination)	
0	24.200.208.162	24.141.95.74	2500	15701	SIP
1	24.200.208.162	24.141.95.74	60442	22572	RTP
2	...				

Lorsqu'un paquet UDP est reconnu, on regarde dans une table, comme le Tableau 4.1, afin de savoir si ce type de paquet a déjà été analysé. S'il a été analysé, une signature a été ajoutée dans cette table pour permettre de ne plus avoir à en refaire l'analyse. Donc, à chaque nouvelle

identification positive dun protocole connu, une signature est créée et insérée dans la table.

4.3.2 L'algorithme développé

Voici le pseudo code de l'algorithme développé.

Tableau 4.2 - Algorithme proposé

Algorithme développé	**Méthode de base**	**ident_frame()** Placer les pointeurs pour Ethernet 802.2 **Selon** le **TYPE** Ethernet (protocole réseau) Si c'est **IP** Placer les pointeurs de IP Calcul de l'en-tête variable **Selon** le **Protocole** de transport sous IP Si c'est **UDP**
	Méthode par apprentissage	Placer les pointeurs UDP, PayLoad Calcule du PayloadSize **SearchCAM() -** recherche dans la CAM d'une signature avec les mêmes adresses et ports source et destination si rien trouvé alors **checkSIP()** vérifie si le contenu est du SIP si oui alors vérifie s'il contient **checkSDP()** **Insert_signature()** (de SIP de SDP) si rien trouvé alors **checkRTP()** vérifie si le contenu peut-être du RTP ou du RTCP selon certains critères approx. **Add_signature()** (de RTP ou de RTCP)
	Mixte	Si c'est **TCP** Placer les pointeurs TCP et du PayLoad Calcul de l'en-tête variable Calcul du PayloadSize Recherche de protocole dans TCP **checkTCP()** (pas développé) AUTRES protocoles transport annoncés selon le numéro du champ protocole AUTRES protocoles réseau annoncés selon le champs type.

Cet algorithme tient compte de l'identification de toute la trame Ethernet décrite dans la section « méthode de base » du Tableau 4.2. Par la suite la partie « Méthode par apprentissage » sera appelée pour chacun des

paquets UDP non identifié. La section « mixte » identifiera les autres protocoles.

La seule restriction à la *méthode par apprentissage* est qu'avant son commencement la trame doit être complète en mémoire. Ceci est nécessaire pour l'analyse de SIP ou SDP qui font la recherche de champ jusqu'aux derniers octets du paquet UDP.

Voici l'explication des fonctions principales utilisées dans l'algorithme du Tableau 4.2.

- **ident_frame()** est la fonction principale. Elle assume l'identification au complet.
- **SearchCAM()** recherche dans la matrice (CAM_soft) la signature créée à partir des Adresses IP source et destination et des ports UDP source, destination. Cette recherche se fait séquentiellement.
- **checkSIP()** fait une recherche de caractère, par octet, pour vérifier si le contenu est le protocole SIP et quel type de SIP, une requête de session ou une réponse. Si c'est du SIP, alors il vérifie s'il contient du SDP.
- **checkSDP()** fait une recherche de caractère, par octet, pour vérifier si la description du type de media y est présentée. Il en extrait le protocole annoncé et le port source. Dans notre cas le protocole sera RTP et le numéro de port d'arrivé.
- **Insert_signature()** insère une signature de SIP, de SDP ou de RTP au début de la table (CAM_soft) pour y donner une priorité sur les autres. Elle décale tous les éléments de la table vers le bas.
- **checkRTP()** vérifie si le contenu peut-être du RTP ou du RTCP approximatif, selon certains critères décrits au chapitre 2.3.1 de RTP et RTCP.
- **Add_signature()** ajoute une signature dans la CAM_soft de RTP et RTCP approximatif, à la fin de la table pour ne pas y donner de priorité sur les autres.

Les fonctions checkSIP et checkSDP utilisent un algorithme de recherche de caractères, recherche de motifs, naïf (simple), tel que décrit dans [COR94 chap. 34] La section 4.3.4 fait une analyse au niveau temps d'exécution et espace mémoire de chacune des fonctions, **en gras,** de cet algorithme.

4.3.3 Preuve du fonctionnement

La Figure 4.6 montre une capture d'écran lors du fonctionnement de l'algorithme sur un PC. En regardant de plus près, on voit des exemples de trames importantes pour cette session.

Figure 4.6 - Écran d'identification des trames 18 à 23 en débug niveau 0

Encore une fois, la trame #19 indique qu'elle contient un paquet UDP contenant du SIP et du SDP. L'analyse de SDP permet de créer des signatures IP pour une session RTP de l'ordinateur A vers B et de l'ordinateur B vers A. Ces signatures sont ajoutées dans la CAM_soft pour permettre d'anticiper l'arrivé de RTP. Il sera nécessaire d'attendre les premiers échanges de RTP pour ajouter le port de destination et les signatures de RTCP. Les différentes signatures de SIP sont aussi ajoutées. Suivant l'échange des trames 18 à 23. Il est montré à la Figure 4.7 que la session RTP commence à partir de 24 et la majorité des paquets transporte du RTP. Ils sont maintenant tous identifiés par la CAM.

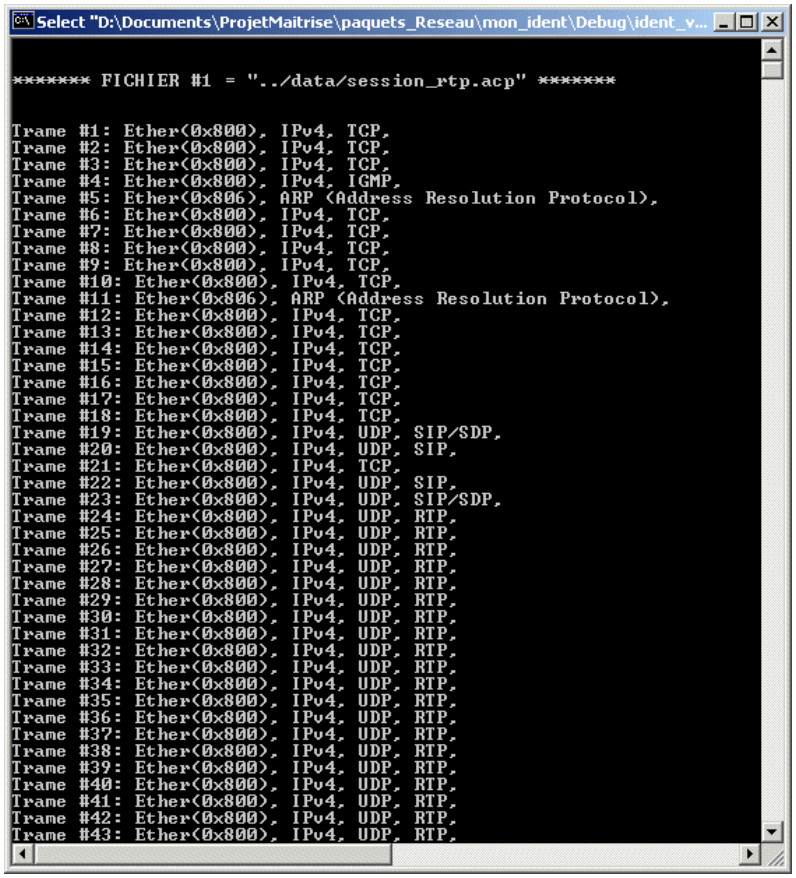

Figure 4.7 - Écran d'identification en débug niveau 7

4.3.4 Statistiques, Complexité et profilage du programme

Lors de l'exécution du programme sur un PC, des statistiques ont été automatiquement accumulées. Celles-ci, montrées dans la capture d'écran de la Figure 4.8, ont servi à comprendre combien de paquets UDP pouvaient être reçus au total et quel pourcentage de ces paquets il était possible d'accélérer en les identifiant avec les résultats des découvertes par apprentissage.

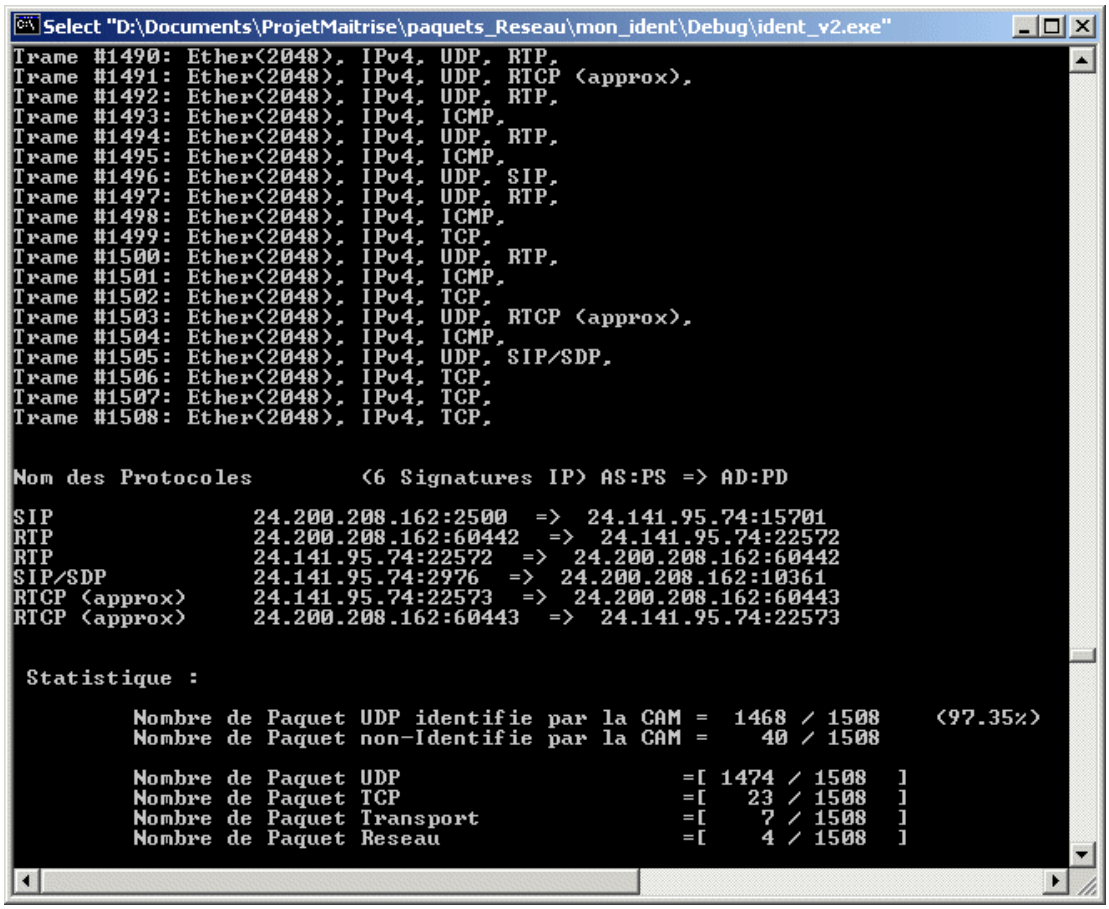

Figure 4.8 - Écran d'identification et statistiques

Ce relevé s'est déroulé sur une période de 30.19 secondes. Durant cette période 1508 trames sont arrivées contenant 1474 paquets UDP, 23 paquets TCP, 7 paquets de niveau transport et 4 paquets de niveau réseau. Des 1474 paquets UDP reçus, seulement 6 ont servi à créer des

signatures IP contenant (SIP/SDP, RTP, SIP ou RTCP). Ces signatures ont servi à identifier (1468) 97.35% des paquets UDP par le processus CAM, (CAM logicielle ou CAM matérielle), sans avoir besoin d'exécuter l'algorithme d'identification développé. Cette méthode a permis de traiter les cas fréquents qui représentent 97.35% des trames totales.

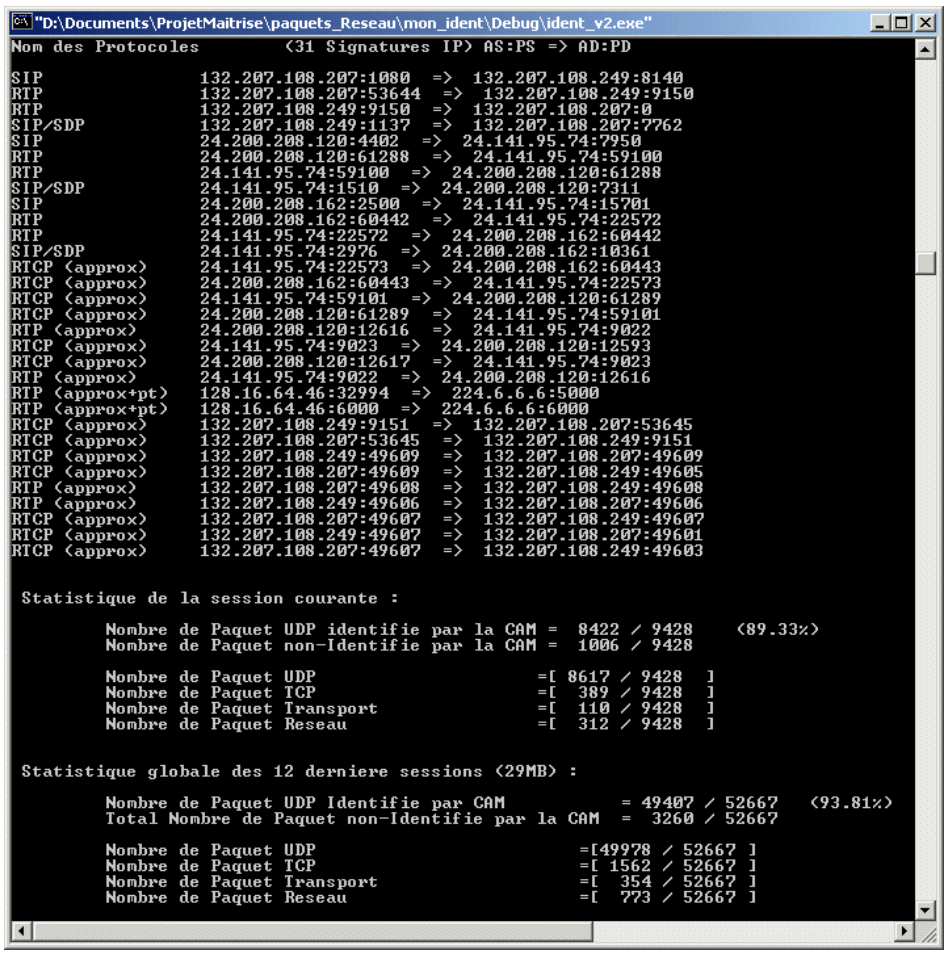

Figure 4.9 - Statistiques globales

L'analyse précédente a été répétée sur un plus grand ensemble de données. Les résultats obtenus sont présentés à la Figure 4.9. Ces statistiques découlent de l'analyse de 14 sessions différentes, qui ne comprennent pas juste des données multimédias. Cet échantillon de données correspond à des pourcentages d'identification par la CAM qui

varient de 0% à 99%. Cet ensemble de données a une taille de 29 méga-octets. Il comprend des données capturées dans le cadre de sessions multimédias et de session de travail normales, TELNET, FTP, XTERM, etc. Ce type de dénombrement exhaustif vise un profilage dynamique représentatif d'un flot de données varié. Il valide le fonctionnement dans des conditions différentes et soutenues. Dans le cas de la Figure 4.9, les statistiques globales d'identification des 52,667 trames traitées montrent que 93.81% de ces trames ont été reconnues via la CAM, qui contenait 31 signatures à la fin du traitement.

La démonstration que notre algorithme fonctionne est faite. Il faut maintenant analyser sa complexité. Puisque le projet vise la réalisation d'un convertisseur de protocoles embarqué, tel que décrit au chapitre 3.4, le projet utilise le processeur ARM-7TDMI. Le code a donc été porté et compilé pour ce processeur, avec le compilateur nécessaire à l'utilisation d'un second outil, Seamless [CVE]. Seamless est un simulateur de processeur qui apporte un modèle exact du jeu d'instructions du processeur ciblé. Il permet la vérification de l'exécution du code assembleur. Il permet aussi la vérification des échanges et des communications entre les composantes dans la dans le cadre d'un co-design, logiciel et matériel, d'un système. Le logiciel Seamless utilise deux sous logiciels, Xray pour la simulation logicielle et ModelSim pour la simulation matérielle.

Tableau 4.3 - Nombre de cycles par niveau de l'OSI

Niveau OSI	Protocole identifié	Nombre de cycles du ARM	Nb. signatures créées
2	Ethernet	82	0
3	IP	99	0
3	ARP	108	0
4	IGMP	87	0
4	ICMP	87	0
4	UDP	193	0
4	TCP	249	0
7	SIP / SDP	82,270 max.	3
7	SIP	10,412 max.	1
7	RTP / RTCP approx	5,698 max.	1

Le temps d'identification de la première apparition des protocoles selon leur niveau de l'OSI est présenté au Tableau 4.3. Les cycles affichés sont pour la couche courante de l'OSI seulement. Elle n'inclut pas la couche précédente. Pour les fonctions de niveau 7, le temps de la fonction en cycles dépend du nombre de signatures existantes dans la CAM_soft.

Tableau 4.4 – Nombre de cycles des fonctions

Nom des fonctions	Trame		Nombre de cycles ARM	Taille
Ident_frame	19	SIP/SDP	**86,990**	**698 octets**
- SearchCAM	19		50	0 signature
- checkSIP	19		81,690	+1 signature
- checkSDP	19		62,528	+2 signatures
- Insert_signature	19		5,250	3 signatures
Ident_frame	20	SIP	**16,994**	**381 octets**
- SearchCAM	20		2,081	3 signatures
- checkSIP	20		7,801	
- Insert_signature	20		7,112	+1 signature
Ident_frame	30	RTP	**6,327**	**214 octets**
- SearchCAM	30		3,249	4 signatures
- checkSIP	30		349	-
- checkRTP	30		1,509	+1 signature
- Add_signature	30		1,220	1 signature

Le Tableau 4.4 affiche chacune des fonctions avec le numéro de trame sur laquelle la fonction s'exécute, le protocole identifié et le temps en cycles pour chacune des sous fonctions clés de l'algorithme d'identification. Ces temps sont pris d'une analyse de l'exécution du code avec le simulateur Seamless. Ce type de dénombrement correspond à un profilage dynamique limité sur la trame indiquée, les données ne sont pas lues de façons automatisées mais choisies et insérées une trame à la fois dans le simulateur dans le but de tester la fonctionnalité de l'algorithme dans notre architecture.

La taille des paquets influence dans certains cas le temps d'identification. Par exemple le Tableau 4.4 regarde trois trames 19 (SIP/SDP), 20 (SIP) et 30 (RTP). La taille des paquets est décroissante et le nombre de cycles d'analyse est aussi décroissant. Il faut tout de même faire attention, la trame 19 est beaucoup plus longue à analyser. Ce cas n'est qu'en partie relié à la taille du paquet. C'est en fait partiellement dû à l'analyse de SDP qui est de loin plus complexe à exécuter que les autres analyses. Dans le cas de RTP ou RTP approximatif (discuté au chapitre 2.3.1), la taille du paquet n'a aucune influence sur le temps d'analyse.

Tableau 4.5 - Instruction plus complexe

Instructions	Nb. Cycles
MOV	1
ADD	1
STR	2
LDR	3
LDRB	3

Il faut aussi regarder les instructions assembleur pour se rendre compte de la complexité du code. Elle peut augmenter selon le choix de certaines instructions. En général le temps de chaque instruction assembleur sur un

processeur de type RISC prend 1 cycle. Cependant, certaines instructions peuvent prendre plus d'un cycle. Le Tableau 4.5 donne quelques exemples du temps d'exécution de certaines instructions.

Tableau 4.6 - assignation d'un pointeur

MAIN 975..976 : ether = (struct IEEE802_2 *) frame ;		8 cycles
00000257C E59F3330	LDR r3,0x28b4	
000002580 E59F2330	LDR r2,0x28b8	
000002584 E5832000	STR r2,[r3,#0]	

Tableau 4.7 - Assignation et calcul de valeur

MAIN 977 : type = (ether->type[0] << 8) + ether->type[1];		22 cycles
000002588 E59F2324	LDR r2,0x28b4	
00000258C E5923000	LDR r3,[r2,#0]	
000002590 E5D3200C	LDRB r2,[r3,#0xc]	
000002594 E1A03402	MOV r3,r2,LSL #8	
000002598 E59F1314	LDR r1,0x28b4	
00000259C E5912000	LDR r2,[r1,#0]	
0000025A0 E5D2100D	LDRB r1,[r2,#0xd]	
0000025A4 E0833001	ADD r3,r3,r1	
0000025A8 E50B3010	STR r3,[r11,#-0x10]	

Le Tableau 4.6 et le Tableau 4.7 affichent quelques lignes de code C traduites en langage machine, à gauche, avec leur représentation assembleur, à droite, suivi du nombre de cycles correspondant à ce groupe d'instructions. Le Tableau 4.6 compte une simple assignation en C qui correspond à 3 instructions assembleur prenant 8 cycles au total.

Par contre, au Tableau 4.7, une assignation de valeur et un calcul prennent jusqu'à 22 cycles, et pourtant la ligne en C correspond à 9 instructions assembleur. Un « MOV », déplacement entre deux registres, est 3 fois plus rapide qu'un déplacement entre deux espaces mémoires.

Dans le but de calculer les cycles et le temps des fonctions, nous avons besoin des caractéristiques de vitesse du processeur ARM7TDMI, [ARM] utilisé dans l'architecture de la plate-forme développée au GRM.

Tableau 4.8 - Caractéristiques du ARM7TDMI

ARM7TDMI – Caractéristique et performances				
Procédé de fabrication	0.35µ	0.25µ	**0.18µ**	0.13µ
Fréquence maximum (MHz)	45	60	**80**	130
Taille du dé (mm^2)	2.14	1.0	**0.5**	0.3

Dans un but informatif, nous affichons aussi la taille de ce processeur. Les calculs du nombre critique d'opérations seront réalisées avec les paramètres suivants :

- Fréquence du système à 80 MHz
- Un cycle dure 12.5ns (1/80MHz)
- Une opération par cycle (processeur de type RISC) sauf pour les instructions d'accès à la mémoire et aux registres.

Tableau 4.9 - Temps en cycles et en nano-secondes

Trame		**Nombre de cycle de ARM**	**Temps nano-secondes (ns) 1 x 10^{-9}**
19	SIP/SDP	82,270	1,028,375 ns
20	SIP	10,412	130,150 ns
30	RTP	5,698	71,225 ns

Par exemple, en reprenant les chiffres du Tableau 4.4 au Tableau 4.9 nous obtenons le temps d'identification et d'analyse de certaines trames en nano-secondes.

4.3.5 Accélération possible de l'algorithme logiciel

Une des règles de l'accélération est de diminuer en temps les fonctions les plus complexe. Dans notre situation ce cas est l'analyse de SIP contenant du SDP. Le pire cas de l'analyse SIP/SDP arrive lorsque la recherche de la chaîne « Content-Type » est précédée de plusieurs lignes qui commencent par une sous-chaîne de celle-ci. Par exemple, lorsque des lignes débutant par « Con… » sont rencontrées. « Content-length », « Contact »… Ces situations causent plusieurs appels d'une fonction récursive de recherche de caractère.

Il est possible d'accélérer la recherche de chaînes de caractères en prenant avantage de l'architecture 32 bits. Il suffit de rechercher quatre caractères à la fois, en les convertissant en entier de 32 bits, au lieu de rechercher un seul caractère de 8 bits. On peut donc prendre quatre caractères de huit bits et les jumeler dans un mot de 32 bits. Cette façon de faire accélère en théorie de 4 fois la recherche de chaînes si les chaînes recherchées sont alignées sur des frontières de 32 bits. Une autre possibilité serait de regarder les algorithmes de recherche de motifs. L'exemple d'un tel algorithme est celui proposé par Knuth-Morris-Pratt décrit dans [COR94 chap. 34]

Une autre accélération possible améliore tous les cas. Pour chaque trame, l'algorithme fait plusieurs branchements conditionnels, des « switch case ». Par exemple 100 valeurs possibles équivalent, en langage machine, à 100 tests conditionnels et successifs si c'est le 100ième qui est recherché. Il est possible de remplacer ces 100 branchements conditionnels par des sauts indirects. La façon de faire est de disposer

d'une mémoire numérotée de zéro à la valeur du dernier test et de faire un saut indirect à la position mémoire correspondant à cette condition. Par exemple si la valeur du type Ethernet, tel que décrite au chapitre 2.2.1, vaut 0x800, alors le saut irait à l'adresse 0x800, où se situerait les instructions nécessaires pour traiter ce cas.

CHAPITRE V

5 JUSTIFICATION DE L'IMPLÉMENTATION MATÉRIELLE

Afin de justifier une implémentation matérielle, on doit entre autre démontrer que la complexité actuelle de l'algorithme est trop grande pour être exécutée sur un seul processeur. Il faut aussi calculer le débit du réseau et le nombre maximal de trames par seconde qu'il sera possible de recevoir. À partir de ces résultats, nous serons en mesure d'évaluer si la performance de l'algorithme logiciel est adéquate ou non pour identifier cette quantité de trames.

5.1 Analyse des fonctions

La fonction d'insertion « Insert_signature » décale vers le bas, dans la CAM_soft, toutes les signatures, pour chaque nouvelle ajoutée. Nous avons remarqué que dans notre cas, le temps de décalage augmente d'environ 1700 cycles à chaque nouvelle insertion. De la trame 19 à la trame 20, on passe de 3 à 4 signatures. Lors de l'analyse de la trame 20, il y a 3 signatures à décaler et une à insérer. 3 x 1700c + 1700c = 6800 cycles. Cet estimé est près de la valeur mesurée de 7112 cycles.

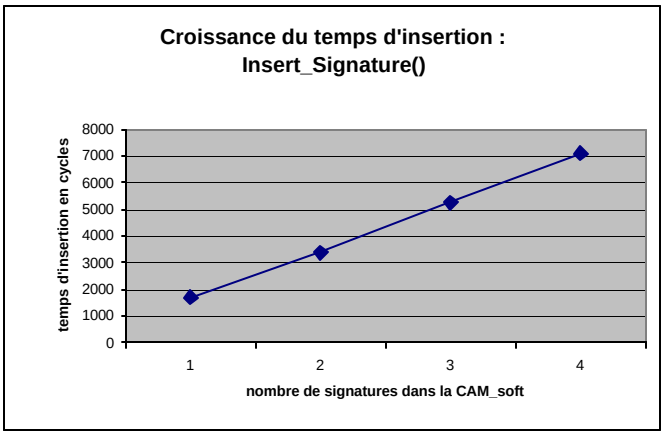

Figure 5.1 - Croissance du temps d'insertion

La Figure 5.1 démontre bien que la croissance en nombre de cycles est linéaire et que la complexité en pratique augmente en O(n) avec la croissance de la taille de la table « CAM_soft ».

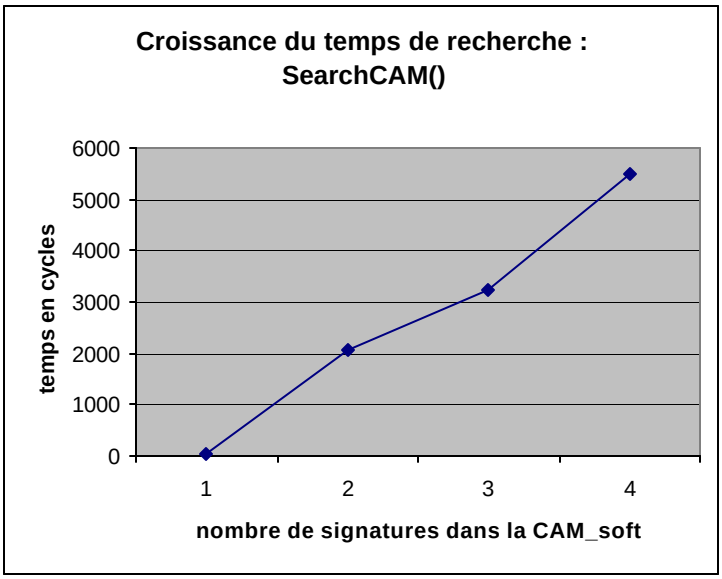

Figure 5.2 - Croissance du temps de recherche

De la même façon, la recherche séquentielle de la fonction « SearchCAM » augmente aussi de façon linéaire O(n) lorsque la taille de la table « CAM_soft » augmente. Voir la Figure 5.2.

Une accélération matérielle possible serait d'améliorer l'insertion et la recherche. La fonction « SearchCAM » recherche dans la CAM_soft (logicielle) mais elle pourrait être remplacé par une recherche dans une CAM en matérielle. La complexité de cette recherche est montrée à la Figure 5.2. Elle est en pratique O(n), mais elle pourrait passer à O(1) avec une CAM matérielle. Chaque recherche de signature pour chaque trame serait donc grandement accélérée, ce qui justifie de considérer l'usage d'une CAM en matériel, telle que décrite au chapitre 2.6.3.

L'analyse des temps de traitement des trames, de taille maximale ou de taille minimale, montre qu'elles prennent presque le même nombre de cycles CPU, sauf dans le cas où le paquet contient du SIP/SDP, tel qu'expliqué dans la section 4.3.5. Ce type de paquets consomme plus de cycles, parce que leur analyse implique des recherches de chaînes de caractères sur toute la taille du paquet UDP. Le temps de traitement dépend donc en partie de la taille des données. La machine décrite dans [WEL01] inspecte le contenu des paquets et cherche des chaînes de caractères à l'aide d'un support matériel. Elle devrait être évaluée comme une solution possible pour le cas le plus complexe. Ce cas n'est toutefois pas traité dans ce mémoire.

5.2 Évaluation du débit réel des interfaces choisies

Un des objectifs de ce projet est la transmission temps réel de vidéo numérique de qualité studio sur des réseaux locaux. Il était donc d'intérêt de choisir des interfaces de communication à haut-débit, environ 360 Mbps (Équation 5.1), capables de transporter ce genre d'information [SMPTE]. C'est pourquoi les interfaces sont déjà choisies dans le projet du convertisseur du GRM.

L'Équation 5.1 indique comment le calcul de 360 Mbps à été obtenu en utilisant la norme discutée dans [SMPTE].

```
= 2288 pixels/ligne x 10 bits/pixel x 625 lignes x 25 images/sec
= 357.5 Mbps
```

Équation 5.1 - Bande passante requise pour le transfert de vidéo numérique non compressé en temps réel.

Il faut ensuite connaître le débit utilisable d'un réseau pour savoir combien de trames à la seconde, au maximum, il est possible de recevoir, ainsi que la quantité d'informations à transmettre. Il est nécessaire d'évaluer ce taux d'arrivée des paquets et la taille possible des trames pour savoir combien de trames devront être identifiées par seconde.

Il existe deux façons de calculer ces nombres. Premièrement, on peut calculer le nombre de trames par seconde avec leurs tailles maximales de trames, pleine de données, ou avec des trames de tailles minimales, sans données. Dans ce cas, on doit s'attendre à une augmentation importante du nombre de trames par seconde. Il ne faut pas oublier qu'un des buts est de déterminer la complexité d'un système d'identification lorsque le débit d'arrivé des trames varie.

Notons que si les trames sont pleines (taille max.), on a plus de temps pour les traiter, puisque le débit d'arrivée des trames diminue. D'autre part, si les trames sont vides (taille min), il y aura certainement un débit d'arrivée des trames beaucoup plus élevé. On serait porté à penser qu'une croissance du nombre de paquets par seconde augmente la complexité de l'identification. Ce n'est cependant pas si évident, simplement parce que la taille des données à analyser est plus petite et donc le temps pour parcourir la trame est plus court. Le chapitre « 4.3.4 Complexité et profilage » discutait ces résultats.

La consommation en temps CPU augmente lorsque les fonctions d'identification sont appelées avec cette fréquence plus élevée. Le but est de déterminer si le pire cas se produit lorsque les paquets qui arrivent sont de taille minimale ou de taille maximale.

Tableau 5.1 - nombre trames/seconde avec la taille maximale d'une trame

Interfaces	Débit max. (Mbps)	Taille **maximale** de la trame (bits)	(E2) NB de trames (brut) à la seconde	Taille standard des en-têtes (en bits)				(E3) Charge utile possible (bits/paquet)	(E4) débit réel LB max (Mbps)
				Physique 802.3	Liaison 802.2	Réseau IP	Transport UDP		
IEEE 802.3	1,000	12240	81,699	96	112	160	64	11,808	965
IEEE 1394 b	800	33,184	24,108	0	160	160	64	32,800	791
IEEE 1394 a	400	16,576	24,131	0	160	160	64	16,192	391

Le Tableau 5.1 résume les résultats des calculs pour les interfaces choisies. Par exemple, en regardant la ligne relative à IEEE-802.3 du Tableau 5.1, le débit brut de cette interface est de 1 giga bits par seconde. La taille maximale de ce type de trames est 1518 octets, plus la taille des bits de synchronisation du médium, soit 12 octets. Converti en bit, ceci donne 12,240 bits par trame brute.

= Débit maximum de l'interface (Mbps) / taille de la trame
= nombre de trames par seconde

Équation 5.2 - Nombre de trames à la seconde

En utilisant l'Équation 5.2, la colonne E2 donne le débit maximal en nombre de trames (taille maximale) par secondes soit 81,699 trames par secondes.

= taille max – tailles des en-têtes physique, liaison, réseau, transport
= charge utile possible à analyser

Équation 5.3 - Charge utile disponible par paquet

L'Équation 5.3, montre la taille réelle, l'espace disponible pour le transport de données dans une trame. Ce nombre est donc le nombre de bits susceptibles d'être analysés c'est-à-dire 11,680 bits ou 1460 octets.

$$\text{LB utilisée} = \frac{\text{Nombre de trames}}{\text{seconde}} \times \frac{\text{Charge utile}}{\text{paquet}}$$

Équation 5.4 - largeur de bande utile disponible

Nous pouvons évaluer la largeur de bande (LB) réelle de l'interface. C'est notamment la partie qui transporte les données qui nous intéresse. Sans les en-têtes connues, elle contient la partie à identifier. L'Équation 5.4 montre qu'en multipliant le nombre de trames brut à la seconde par la charge utile des paquets, on peut transporter 956 Mbps avec Ethernet 1GB.

Nous considérons que les réseaux de type Ethernet qui utilisent des équipements réseaux intermédiaires ne peuvent opérer correctement que jusqu'à un régime maximal équivalent à 80% de la bande passante [OMA98]. Dans le cas des réseaux locaux, sans routage, ni passerelle, il est possible de faire l'hypothèse que tout fonctionnera correctement jusqu'à 100% du débit mentionné. Ainsi, les réseaux de type 1394 qui sont des réseaux fermés, peuvent fonctionner correctement à 100% de leur débit.

Nous avons vu les calculs et les débits réels avec les trames de taille maximale. Regardons maintenant les mêmes résultats avec les trames de tailles minimales.

Tableau 5.2 - nombre trames/seconde avec la taille minimale d'une trame

Interfaces	Débit max. (Mbps)	Taille **minimale** de la trame (bits)	(E2') NB de trame (brut) à la seconde	Taille standard des en-têtes (en bits)				(E3') Charge utile possible (bits/paquet)	(E4') débit réel max (Mbps)
				Physique 802.3	Liaison 802.2	Réseau IP	Transport UDP		
IEEE 802.3	1,000	608	1,644,737	96	112	160	64	176	289
IEEE 1394 b	800	512	1,562,500	0	160	160	64	128	200
IEEE 1394 a	400	512	781,250	0	160	160	64	128	100

Encore une fois pour le IEEE-802.3 du Tableau 5.2. Le débit brut de cette interface est de 1 giga bits par seconde. La taille minimale de chaque paquet est 64 octets plus la taille des bits de synchronisation du médium, soit 12 octets. Convertit en bit, ça nous donne 608 bits par trame brute. En se servant encore de l'Équation 5.2, la colonne E2' donne le nombre de trames (taille minimale) qui peut arriver sur une interface à la seconde, soit 1,644,737 trames par seconde. Ce résultat est évidemment beaucoup plus grand que dans le cas où les trames étaient de tailles maximales.

Il faut cependant remarquer à la colonne E3' que la taille de la charge utile transportable est de petite taille soit de 176 / 8 = 22 octets. Dans ce cas, le débit réel possible chute à 289 Mbps, ce qui est insuffisant pour notre besoin. De plus, les cas complexes d'analyses de SIP/SDP sont maintenant presque impossibles, sauf en utilisant la fragmentation, puisque la taille requise pour transmettre l'information est inadéquate. L'hypothèse d'exploiter des trames de tailles minimales est donc écartée, ces trames ne peuvent pas contenir toute l'information pertinente.

Un problème existe avec la taille de SIP ou SDP; elle est variable. La taille est indiquée dans un champ du corps exprimé en ASCII et qui se nomme « Content-Length ». Un estimé expérimental sera utilisé.

Tableau 5.3 - Taille minimale fonctionnelle

Nom du Protocole	**Taille minimale du paquet seul. (octets)**	**Probabilité d'apparition %**
Ethernet	76	100%
IP	96	99.73%
UDP	104	97.75%
RTP	120	95.62%
RTCP	**116**	**1.59%**
SIP	376	0.20%
SIP/SDP	698	0.07 %

Il serait logique de se demander quelle serait la taille minimale des paquets contenant possiblement toute l'information pertinente. Nous pourrions aussi ajouter la fréquence d'apparition de chaque type de paquet. Le chapitre « 4.2.1 Analyse du réseau multimédia » donne ces probabilités. Le Tableau 5.3 résume ces chiffres avec la taille minimale des trames pouvant contenir des paquets intéressants.

La preuve qu'une mise en oeuvre logicielle serait inadéquate pour faire l'analyse de ces paquets passe par l'estimation du nombre de paquets par seconde qui doit être multiplié par le temps nécessaire à l'identification logicielle de chacun. Le cas le plus fréquent est RTP et le pire cas serait RTCP.

Tableau 5.4 - nombre trames/seconde avec la taille minimale de paquet (RTCP)

Interfaces	Débit max. (Mbps)	Taille **minimale** du paquet (bits)	(E2') NB de trame (brut) à la seconde
IEEE 802.3	1,000	1,024	976,563
IEEE 1394 b	800	1,120	714,286
IEEE 1394 a	400	1,120	357,143

Plaçons la taille minimale d'un paquet à la taille de RTCP (116 octets) plus 12 octets de synchronisation de l'interface : 128 octets * 8 bits = 1,024 bits. Ce n'est pas le cas le plus fréquent, mais c'est la plus petite taille de paquet qui peut contenir toute l'information pertinente.

Le Tableau 4.3 donnait environ 5,698 cycles pour identifier RTCP. Par ailleurs, le Tableau 5.4 dit que sur l'interface 802.3, il est possible de recevoir un maximum de 976,563 paquets RTCP à la seconde (1 milliard de bits seconde / 1024 bits par paquet pour les paquets de taille minimale). Pour pouvoir identifier toutes les trames RTCP, il faudrait pouvoir exécuter 5,564,455,974 cycles à la seconde (environs 5 milliards).

La probabilité d'observer un paquet RTCP n'est que de 1.59% (5% selon [RTP]) et celle de d'observer un paquet RTP est de 95.62%, la taille d'un paquet RTP n'est que de 4 octets de plus que celle d'un paquet RTCP, ce qui ne change pas les résultats de façon significative. Le nombre de cycles total par seconde serait, avec 1056 bits par trame (RTP), 946,970 * 5,698 = 5,395,835,060 de cycles. Encore une fois, le calcul requiert environ 5 milliards de cycles par seconde.

Un processeur ARM utilisant une technologie CMOS 0.18µ (voir Tableau 4.8) a une période de 12.5ns. On rencontre le pire cas quand on rencontre un flot RTCP. Dans ce cas, le temps nécessaire à l'identification

est de : 5,564,455,974 cycles * (12.5 x10^{-9}) secondes par cycles = 69.56 secondes. Ce temps est nécessaire à un logiciel s'exécutant sur le ARM pour identifier ce qui arrive en une seconde sur une interface à 1Gbps,.

En théorie il est donc nécessaire d'accélérer l'identification logicielle par un facteur d'au moins 70, dans le cas ou les paquets seraient tous de types RTCP ou RTP avec une taille minimale. Le pire cas de taille minimale avec RTCP est improbable, car la probabilité d'observer des paquets RTCP est de seulement 1.59%. Le pire cas pour des paquets de taille minimale RTP est aussi improbable, parce que le protocole RTP transporte toujours des données. Puisque les conditions théorique nécessitant un facteur d'accélération de 70 sont improbables, on peut le calcul pour le cas d'une trame de taille moyenne. La bande passante utilisée par RTCP est alors de 5% [RTCP], en supposant que le reste de la bande passante est utilisée pour transporter RTP, qui constituerait alors 95% du trafic.

= Taille moyenne de RTCP * 5% + taille maximum de RTP * 95%
= (1024 bits * 5%) + (4008 bit * 95%)
= 3858.8 bits

Équation 5.5 – La taille pondérée des paquets

À l'aide de l'Équation 5.5 il est possible de calculer la taille moyenne indépendamment du protocole d'arrivé. Nous pouvons faire ceci en pondérant la taille des paquets de chaque protocole avec leurs fréquences d'arrivée. Le nombre de trames moyenne reçues à la seconde, pour une interface 802.3, est égal à 259,148 (1GB / 3858.8 bits par trames). Si on multiplie ce nombre par 5698 cycles (nécessaire à l'identification de RTP ou RTCP) ça demande 1,476,624,857 cycles par secondes pour faire une identification moyenne. De ce nombre de cycles, on peut calculer le temps requis par le processeur ARM : 1,476,624,857 * (12.5 x10^{-9}) secondes par cycles = 18.46 secondes. Cette valeurs est le temps dont le

processeur a besoin pour faire identification des trames qui arrivent a chaque seconde. L'algorithme logiciel est donc 18.46 fois trop lent. Une analyse basée sur une taille moyenne de paquets ne change donc pas nos conclusions générale en ce qui concerne la nécessité d'une accélération matérielle.

CHAPITRE VI

6 IMPLÉMENTATION MATERIELLE/LOGICIELLE

L'architecture co-design proposée se fonde sur la version 3 de l'architecture, présentée au chapitre 3.4.2, développée par le groupe netpro du GRM. Elle est une sous partie importante créée pour cette nouvelle architecture. En modifiant les algorithmes, elle pourrait aussi être utilisée dans d'autres types de systèmes tels que les systèmes d'interception de paquets.

La méthode de fonctionnement de l'architecture du PI, « Protocol Identifier » ou identificateur de protocole matériel, vise à améliorer le temps de traitement du cas moyen. L'architecture tient compte du fait que, lors des sessions multimédias, les paquets de type SIP/SDP viennent avant et annoncent les paquets RTP. Bien que le nombre de paquets RTP soit supérieur au nombre de paquets SIP/SDP, ce dernier vient en premier. Il est identifié en utilisant un algorithme logiciel qui peut mettre à jour un engin de recherche matériel comportant des CAM/TCAM. Cet engin de recherche peut alors identifier en 1 cycle tous les paquets RTP suivants de cette session. Cette technique améliore les performances globales du système en rendant l'identification du cas le plus fréquent (RTP), le plus simple possible.

6.1 Architecture du PI

Le système d'identification exploite une structure matérielle/logicielle conçue suivant une approche co-design. Ce système proposé exploite d'une part les possibilités du matériel, car il définit des modules matériels grâce au langage VHDL, modules qui sont réalisables comme un ensemble de portes logiques, transistors et interconnexions où comme une partie

d'un FPGA. Il exploite aussi les possibilités du logiciel car une partie de la fonctionnalité est exprimé en langage C compilé en vue d'une éxécution sur un processeur ARM. Le processus d'identification des niveaux supérieurs de l'OSI, niveau application, est invoqué par une interruption provoqué par le matériel. L'appel et l'exécution se font au besoin sur le processeur ARM. Lorsque le contenu du paquet, de niveau transport, n'est pas reconnu, il est alors passé au logiciel pour une analyse plus poussée.

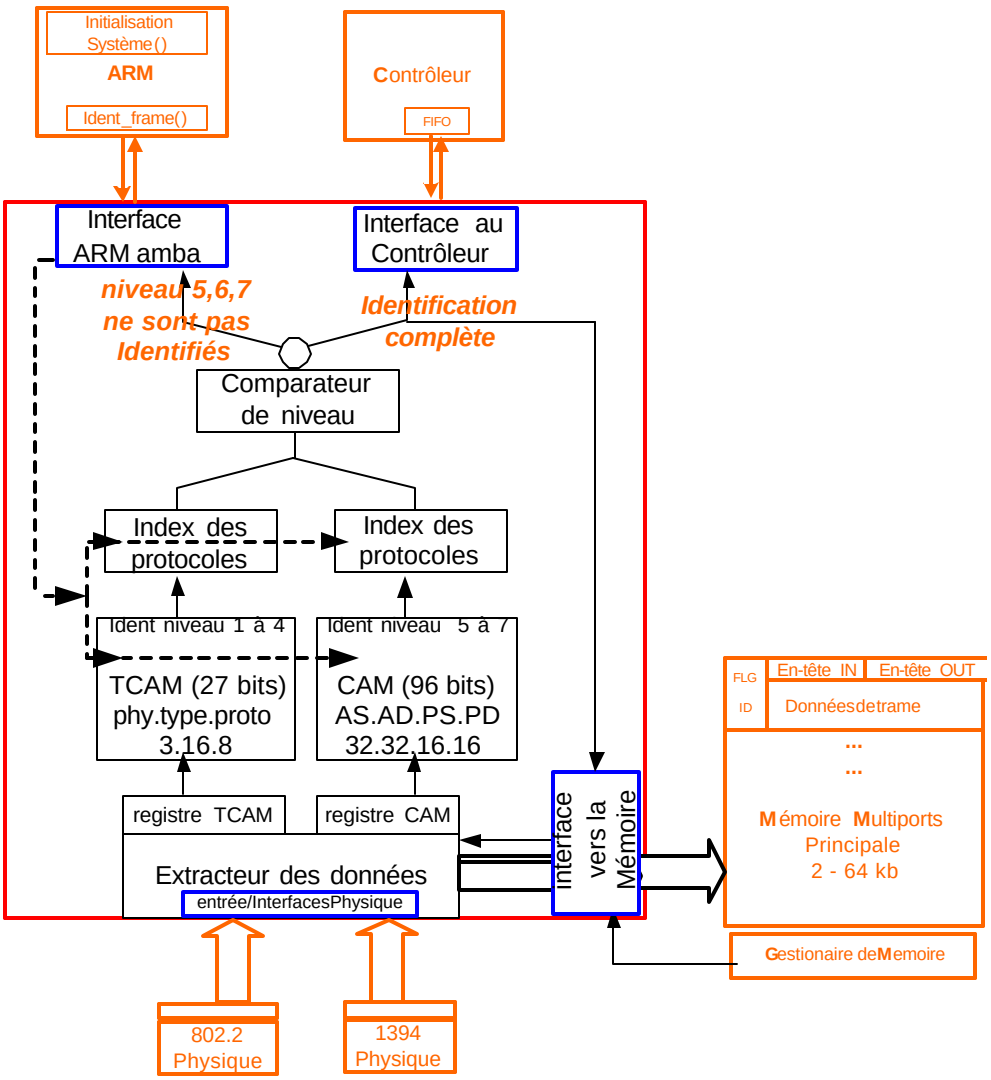

Figure 6.1 - Architecture du PI proposée

La Figure 6.1 présente un diagramme de l'architecture matérielle réalisée. Elle doit être regardée du bas, où se trouvent les interfaces d'entrées, vers le haut, où l'on y trouve le co-processeur d'identification.

Dans un premier temps, les données qui arrivent sont automatiquement transférées à la mémoire principale du convertisseur. Dans un deuxième temps, une certaine partie des données est extraite et stockée dans des registres. Les données extraites sont par la suite envoyées vers l'engin de recherche matérielle soit une CAM ou une TCAM, selon les niveaux (OSI) auxquels correspondent les protocoles présents.

L'extracteur de données reçoit des trames complètes de l'interface physique. De ces trames, des champs sont extraits. La liste des champs à extraire est programmable et contenue dans une table d'instructions. Les instructions sont exécutées par un circuit programmable appelé extracteur de données. Le principal avantage de cette méthode est sa programmabilité. En effet, cette solution correspond à une machine à états programmable.

La sortie des engins de recherche, CAM/TCAM, sera évaluée par le comparateur de niveau pour savoir s'il doit aviser soit le contrôleur, qu'une trame connue est prête pour la conversion, soit le processeur ARM de lancer l'exécution de l'algorithme d'identification par apprentissage. (Tableau 4.2) Dans ce cas les trames devront être en mémoire et complètement accessibles avant que la fonction n'ait débuté. Il est possible de faire l'appel de la fonction lorsque UDP est identifié et même avant que le dernier bit des données soit arrivé. La latence entre le temps de l'appel et la fin de l'arrivée des données peut justifier un appel hâtif de la fonction d'identification.

6.1.1 L'extracteur de données

L'extracteur de données est un circuit programmable qui sert à extraire les données pertinentes de l'en-tête d'une trame. Pour faire cela, une série d'instructions lui sont fournies par l'intermédiaire d'une mémoire d'instructions. Les instructions sont divisées en 5 champs. Les quatre premiers sont ceux qui nous intéressent dans le cadre de ce projet, le cinquième est réservé pour développement futur.

Le premier champ de l'instruction, d'une taille d'un bit, contient la destination de la donnée à extraire. Selon le cas, cette dernière sera envoyée soit à une TCAM soit à une CAM.

Le deuxième champ de l'instruction contient le numéro de la tranche de 32 bits de données. Chaque mot de 32 bits qui entre dans le convertisseur de protocoles est numéroté. Au premier mot d'une trame est assigné le numéro 0. Le champ de l'instruction aura une taille de 4 bits, ce qui correspond à une en-tête de taille maximale de 512 bits (16X32). Au fur et à mesure que l'analyse de protocoles se fera, il est possible que la largeur maximale soit augmentée.

Tableau 6.1 - Structure d'une instruction pour l'extraction de champ

31	30	26	21	16	0
Destination	**# du paquet de 32 bits**	**Position du Premier bit**	**Longueur de l'extrait**	**Réservé**	
1	0011	00000	01111	Réservé	

Le troisième champ contient la position du premier bit à extraire tandis que le quatrième champ contient la longueur de la séquence à extraire. Ces deux champs auront chacun une taille de 5 bits parce que la représentation de 32 bits se fait avec 5 bit ($2^5 = 32$).

Le Tableau 6.1 présente la composition d'une instruction. Pour exécuter cette instruction, la machine à états attendra premièrement que le compteur de mots de 32-bits arrive à 3 (96/32 = 3). De ce mot de 32-bits, il doit extraire les bits 0 à 15 (32 * 3 + 0 = 96 et 32 * 3 + 15 = 111). La position du premier bit dans le mot est donc 00000 et sa longueur est 15 donc 01111.

À chaque mot de 32 bits qui entre, la machine à états vérifie si l'instruction en mémoire est associée à ce mot. Si l'instruction ne correspond pas à ce mot, aucun bit n'est extrait et le pointeur d'instructions ne change pas. Dans le cas contraire, les bits sont extraits et le pointeur d'instructions est incrémenté. Il ne peut donc y avoir qu'une seule série de bits à extraire par mot de 32 bits. Si par un conflit de protocoles, il fallait extraire plus qu'une série de bits, il faudrait changer l'instruction pour extraire tous les 32 bits. Il serait bon de noter que les instructions doivent être placées en ordre croissant de mots à extraire.

6.1.2 Le rôle de la TCAM dans le processus d'identification

La TCAM est utilisée pour faire l'identification des protocoles qui reposent sur trois des 4 premiers niveaux du modèle OSI, soient les couches liaison de données, réseau et transport. Le quatrième, le niveau physique, est implicitement pris en charge par le module d'interface physique.

La TCAM est initialement divisée en 3 parties qui sont l'**interface d'entrée**, le champ **type** et le champ **protocole**. La première, l'interface, détermine le numéro d'interface physique par laquelle arrivent les données. Un deuxième champ, le type, est extrait de l'en-tête Ethernet et est utilisé pour déterminer le protocole réseau. Le troisième, le champ protocole, est un champ de l'en-tête de la trame IPv4. Ce champ indique le protocole de niveau transport.

Tableau 6.2 - Structure de la TCAM et de ses registres

Index TCAM		**Registre d'instructions de la FSM**				**Valeur de la TCAM sur 27 bits Niveau OSI 1 à 4**		
Pos. dans la TCAM	**Nom du protocole**	**1er champ à extraire 16bit**		**2ième champ à extraire 8bits**		**0 ième**	**1er**	**2ième**
						no.PHY	**TYPE**	**PROTO**
						3 bits	16 bits	8 bits
		Pos début	**Pos fin**	**Pos début**	**Pos fin**			
1	Ethernet	-	-	-	-	1		
2	IP v4	96	111	-	-	1	0x800	-
3	UDP	96	111	216	224	1	0x800	17
4	TCP	96	111	216	224	1	0x800	6
5	ARP	96	111	-	-	1	0x806	
6	IP v6	96	111	193	201	1	0x86DD	17
7	FireWire					2		
8	FireWire Asynchronous					2		
9	FireWire Isochronous					2		

Dans ce modèle de base du PI, la TCAM aura une taille de 27 bits de large (bit 0 à bit 26 inclusivement). Un exemple de contenu de la TCAM est montré au Tableau 6.2.

6.1.3 Le rôle de la CAM dans l'identification

La CAM est utilisée afin de compléter l'identification amorcée par la TCAM. Elle a comme tâche de faire l'identification des protocoles de niveau 5 à 7, du modèle OSI. Par exemple, dans le cas où la TCAM identifierait le protocole UDP, le paquet peut transporter du RTP au niveau application. Dans le cas où le protocole a déjà été identifié par apprentissage, la CAM

pourrait vérifier cette hypothèse. Dans le cas contraire, l'algorithme logiciel en fera l'analyse lors de son appel par interruption.

Tableau 6.3 - Structure de la CAM et de ses registres

Index CAM		**CAM (96 bits)**				
Pos dans la CAM	**Nom du protocole**	**Pos**	**Adresses IP source et destination**		**Port source et destination**	
8 bits	8 bits	8 bits	AS 32 bits	AD 32 bits	PS 16 bits	PD 16 bits
1	SIP/SDP	1	132.207.108.217	132.207.108.35	5060	5060
2	RTCP	2	132.207.108.217	132.207.108.35	5005	5005
3	RTP	3	132.207.108.217	132.207.108.35	5004	5004
4	RTSP	4	132.207.108.55	132.207.108.72	554	554
5	SIP	5	132.208.136.58	132.207.108.24	5060	5060
6	RTCP	6	132.208.136.58	132.207.108.24	5005	5005
7	RTP	7	132.208.136.58	132.207.108.24	5004	5004

Comme déjà expliqué dans les chapitres précédents, l'identification de RTP ne peut pas être faite directement comme les protocoles des couches inférieures parce que les adresses et les ports utilisés changent de session en session. Il n'est donc pas possible d'utiliser une valeur statique pour en faire l'identification.

Une session RTP est annoncée par un autre protocole tel que "session initiation protocol" (SIP). Le SIP est envoyé tout d'abord pour initier une session RTP. Il indique les ports source et destination par lesquels s'effectuera la session RTP et RTCP. Avec ces informations en main et l'adresse source et destination de IP, il est possible de mettre à jour la CAM pour identifier le protocole. À la fin de la session, le protocole SIP ou

RTCP est utilisé pour terminer la session. À ce moment, la CAM doit être remise à jour pour ne plus reconnaître ces sessions. La structure de la CAM est montrée au Tableau 6.3.

6.1.4 Index (LUT) des CAM et TCAM

Les sorties de la CAM et de la TCAM sont les positions en mémoire où se trouvent les données recherchées. D'autre part, la CAM et la TCAM ont été créées pour pouvoir insérer d'autres protocoles, au fur et à mesure que la recherche se poursuit. En ajoutant un protocole, la position en mémoire d'un protocole peut être modifiée.

La structure de la TCAM est présentée au Tableau 6.2, Si pour une raison quelconque, il fallait insérer un protocole de niveau transport sous IPv4, il faudrait insérer une nouvelle entrée avant la position 5. Cela causerait un changement de la position mémoire des protocoles ARP, IPv6 et FireWire. Les modules du PI n'auront aucun moyen de savoir que les positions qui correspondaient aux protocoles ARP, IPv6 et FireWire correspondraient à un autre protocole. Ceci introduit donc l'usage d'une LUT.

La LUT a pour but de suivre les changements dans la liste de protocoles à identifier. Elle garde en mémoire l'association de la position mémoire (CAM/TCAM) avec la liste des protocoles identifiables. Si, comme dans l'exemple précédent, un protocole était inséré, la LUT aurait l'information de ce changement. Les circuits du PI ne verraient donc pas de différence.

6.1.5 Partie logicielle

Dans l'architecture du PI, les trames sont premièrement identifiées par la TCAM et ensuite comparées avec les signatures dans la CAM. Dans les cas qui ne sont pas reconnues par la CAM, une interruption est levée au processeur ARM. Il exécute un algorithme d'identification pour les protocoles supportés et met à jour la CAM, lorsque possible.

Le comportement du logiciel, qui roule sur le ARM, peut être décrit de la façon simplifiée suivante :

- Il attend une interruption de l'identificateur de protocoles (protocole non-identifiable en matériel),
- Exécute l'algorithme logiciel d'identification sur les en-têtes et les données « Payload »
- si un protocole application est identifié il :
    - Extrait l'adresse source,
    - Extrait l'adresse destination,
    - Extrait le port source,
    - Extrait le port destination,
    - Ajoute une signature à la CAM pour identifier cette session,
- Retourne à l'étape 1.

6.2 Comparaison des cycles et temps de traitement

Le nombre de cycles pour l'exécution de l'algorithme logiciel avec le ARM a été caractérisé aux chapitres 4.3.4 Complexité et profilage et 5.1 Analyse des fonctions. Refaisons ici le même exercice mais en remplaçant les temps d'exécutions avec le temps d'exécution du PI et du matériel dédié.

Lors du chapitre « 5.1 Analyse des fonctions » la complexité du temps d'insertion et du temps de la recherche dans la CAM était en pratique O(n) et passe maintenant vers une complexité plus simple et constante de O(1). La fonction logiciel « searchCAM » ne recherchait pas dans une CAM mais dans un tableau en mémoire RAM. Le remplacement de ce tableau par une CAM matérielle change la complexité de la recherche et de l'insertion dans le tableau. Par exemple le nombre de cycles pour faire une écriture (ajouter une signature) dans la CAM via le ARM est de 39

cycles. Le nombre de cycles pour une lecture (vérifier l'existence d'une signature tel que searchCAM) dans la CAM via le ARM est de 41 cycles. Cependant avec le PI la vérification, la lecture de signature, se font au niveau du module « Extracteur des données ». Ceci ne prend que 3 cycles parce que les transferts par le bus AMBA ne sont pas gérés.

Tableau 6.4 – Nombre de cycles des fonctions utilisant le PI

Nom des fonctions	**No. Trame**		**Nombre de cycles sur ARM**	**Nombre de cycles PI**	**Accélération Améliorée**
Ident_frame	19	**SIP/SDP**	**86,990**	**81,732**	1.06
- SearchCAM	19		50	3	16.66
- checkSIP	19		81,690	81,690	0.00
+- checkSDP	19		62,528	62,528	0.00
- Insert_signature	19		5,250	39	134.00
Ident_frame	20	**SIP**	**16,994**	**7,843**	2.16
- SearchCAM	20		2,081	3	693.00
- checkSIP	20		7,801	7,801	0.00
- Insert_signature	20		7,112	39	182.00
Ident_frame	30	**RTP**	**6,327**	**1,900**	3.33
- SearchCAM	30		3,249	3	1,083.00
- checkSIP	30		349	349	0.00
- checkRTP	30		1,509	1,509	0.00
- Add_signature	30		1,220	39	31.28

Le Tableau 6.4 compare les performances de l'algorithme logiciel (ARM) versus l'algorithme accéléré matériel (PI). Le rapport d'accélération est indiqué dans la colonne de droite « Accel. » On peut remarquer une accélération aussi marquante que 1083 fois plus rapide, pour la recherche de la mémoire utilisant une CAM, lorsque seulement 4 signatures sont présentes. Plus le nombre de signatures augmente plus l'écart sera grand. La complexité de la recherche logicielle est en pratique O(n), tel que démontré à la « Figure 5.2 ». L'écart entre les deux implémentations de la recherche continuera à grandir parce que le temps de la recherche, avec le PI, reste constant, même avec un nombre croissant de signatures dans la CAM.

6.2.1 Modèle de calcul

Un principe de base dans la conception de systèmes est d'assigner les ressources de façon à accélérer le cas le plus fréquent, au détriment des cas rares [HEN98 chap.1.6]. Une loi fondamentale, la loi de Amdahl, peut être utilisée pour quantifier ce principe.

Le gain en performance qui peut être obtenu en améliorant certaines parties est calculé en utilisant la loi de Amdahl. Elle dit que le gain en performance obtenu à partir de l'utilisation d'un mode d'exécution plus rapide est limité par la fraction de temps pendant laquelle ce mode peut-être utilisé. La loi de Amdahl donne une façon simple de trouver l'accélération résultant d'une certaine amélioration lorsqu'elle dépend de deux facteurs.

1- Fraction $_{\text{améliorée}}$. La fraction du temps qui tire partie de l'amélioration. Elle est toujours plus petite que un.

2- Accélération $_{\text{Améliorée}}$. Le gain de vitesse apporté par le mode amélioré. Le temps du mode originel sur le temps du mode amélioré. Dans notre cas le Tableau 6.4 donne cette accélération améliorée.

Le temps d'exécution de la machine originelle munie du dispositif d'amélioration sera le temps d'utilisation de la machine sans le dispositif plus le temps d'utilisation avec amélioration :

$$Temps\_exécution_{améliorée} = Temps\_exécution_{ancien}\ x\left((1-fraction)+\frac{Fraction_{améliorée}}{Accélération_{améliorée}}\right)$$

Équation 6.1 – Amdahl, Temps d'exécution améliorée

L'accélération globale ou nette, est le rapport des temps d'exécution :

$$Accélération_{globale} = \frac{Temps\_exécution_{ancien}}{Temps\_exécution_{nouveau}} = \frac{1}{(1 - Fraction_{améliorée}) + \frac{Fraction_{améliorée}}{Accélération_{améliorée}}}$$

Équation 6.2 - Amdahl, Accélération globale

Considérons le cas par excellence de notre système : Si 99% des paquets sont accélérés, la fraction $_{améliorée}$ est donc égale à (1 - 0.99). Ce cas se produit si la presque totalité des paquets capturés sont des paquets RTP. Le premier sert comme identificateur pour créer une signature « RTP approximative » et tous les autres sont identifiés avec cette signature. Dans un second exemple, nous considérons l'analyse des sessions multimédias. Selon la Figure 4.8, du la section 4.3, 97.35% des paquets peuvent être accélérés. Ceci donne une fraction $_{améliorée}$ de (1 - 97.35%) et dans le cas de la Figure 4.9, le cas moyen d'analyse de plusieurs sessions, 93.81% des paquets peuvent être accélérés, ce qui conduit à une fraction $_{améliorée}$ de (1 - 93.81%). Voyons maintenant l'impact de ces fraction $_{améliorée}$ dans l'équation de Amdahl. En supposant une accélération très grande, on obtient les limites d'accélération suivantes :

1/(1- 0.99) = 100 fois plus vite que juste logiciel.
1/(1- 0.9735) = 37.74 fois plus vite que juste logiciel.
1/(1- 0.9381) = 16.15 fois plus vite que juste logiciel.

Au début de cette section nous disions qu'il faut rendre rapide le cas le plus fréquent. Dans ce cas ci, c'est la recherche du protocole de transport de haut niveau RTP. Il a une probabilité d'apparition de 95.62%. Dans œ cas, l'accélération globale du système utilisant la recherche améliorée de la CAM pour vérifier tous les protocoles reconnus sous UDP (RTP, SIP, SDP) est donnée par l'expression suivante. Elle est valide lorsque 4

éléments sont présents dans la CAM ou que 97.35% des protocoles de haut niveau sous UDP sont identifiés de façon accélérée. (référence aux chiffres de la Figure 4.8):

$$Accélération_{globale} = \frac{1}{(1-0.9735)+\frac{0.9735}{1083}} = \frac{1}{(0.0265)+8.9889x10^{-4}} = 36.5$$

Équation 6.3 - Calcul de l'accélération nette de notre système

Le résultat de l'équation indique qu'une accélération de 1083 sur la recherche se traduit en une accélération nette du système de 36.5 (facteur de la loi de Amdahl). On se souvient qu'à la fin du chapitre 5 on avait besoin de 18.46 secondes pour identifier chaque seconde de flot de façon logicielle. Avec une accélération de 36.5, on arrive à identifier tous les paquets reçus à chaque seconde, avec en moyenne une demi seconde de traitement. Il est donc possible de fonctionner à la vitesse d'une interface gigabits Ethernet.

6.3 Estimation de la charge résultante ARM/Réseau

Cette section analyse la charge de travail du processeur ARM lorsque le débit du réseau est au maximum. Cette charge est fortement réduite grâce à l'identification des signatures par le module physique PI.

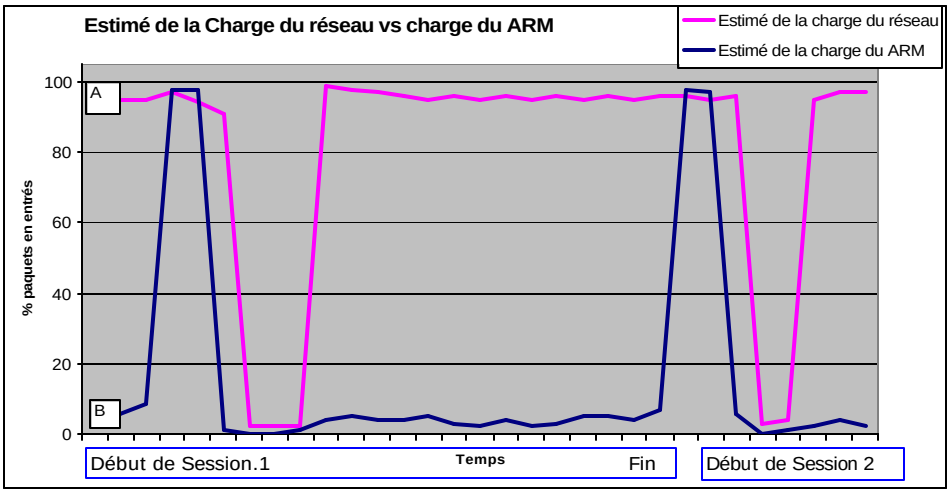

Figure 6.2 - Estimé de charge (Flot réseau vs. ARM)

Considérons la Figure 6.2 où lorsque la charge du réseau est haute, voir ligne A, la charge du processeur ARM, voir la ligne B, suggère que ce dernier aura déjà fini son travail avant la prochaine augmentation de trafic. Cette charge sera principalement constituée de paquets RTP. Ils seront donc tous identifiés par le module PI et sa CAM contenant les signatures appropriées, détecté de l'analyse de SIP. Les deux creux de la ligne A indiquent un début et une fin de session, où le réseau est libre de trafic et le système attend une nouvelle session RTP.

La charge du CPU associée au traitement, ligne B, augmente lorsqu'un paquet SIP/SDP est en cours d'analyse et diminue lorsque tous les paquets font partie de sessions RTP identifiées par des signatures. Nous anticipons que le processeur devrait avoir une charge de travail plus lourde lors des initiations de session avec SIP ou pré-initiations de sessions RTP. Le pourcentage du traitement d'identification effectué par l'algorithme logiciel, sur le CPU, sera maintenant que 2.65% de tous les paquets.

Lorsque aucune signature ne sera créée via l'analyse de SIP, une fonction d'identification pour « RTP approximatif » sera exécuté et créera au besoin une nouvelle signature de RTP approximative.

CHAPITRE VII

7 CONCLUSION

Les contributions scientifiques de ce mémoire se situent dans le domaine de l'identification par champ, de la classification des paquets et de l'identification par apprentissage des protocoles de haut niveau.

Ce mémoire est le premier d'une série dans un projet sur l'architecture d'un convertisseur de piles protocoles. Il avait pour but premier de développer un procédé pour identifier, sans trop de latence, les protocoles désirables pour le convertisseur. Il a aussi comme but d'éclaircir la problématique et de concentrer un ensemble d'informations essentielles à la compréhension des protocoles.

Ce mémoire a débuté par une synthèse des protocoles utilisés dans les sessions multimédias et en a discuté le fonctionnement. Nous avons aussi proposé une revue de littérature sur la classification des protocoles et les méthodes existantes pour l'identification de ces protocoles. Cette étude a confirmé la difficulté de reconnaître certains protocoles, tel que ceux de la famille RTP, pour ensuite proposer une nouvelle façon de les identifier.

Une simulation de la méthode d'identification par apprentissage proposée à été réalisé sous une forme logicielle. Une analyse de complexité de certaines méthodes classiques (purement logicielles) a été réalisée. Par la suite, une réalisation de type co-design exploitant cette approche a été validée à l'aide d'outils spécialisés [CVE]. L'analyse des temps de fonctionnement et d'échange de communication entre les processus logiciel et matériel a été quantifiée et décrite pour créer le module PI, un co-processeur d'identification de protocoles .

La structure proposée pour le PI a permis de séparer la fonctionnalité entre les domaines logiciel et matériel. L'utilisation de la partie matérielle permet d'accélérer l'algorithme, tandis que la partie logicielle s'exécute sur un processeur embarqué inclus dans le PI. Le reste de l'identification se fait via la comparaison de signatures, mises à jour dans une mémoire matérielle de type CAM. Dans nos exemples, on a remarqué que 97.35% de nos trames peuvent être traitées en matériel. Étant donné que le traitement en matériel se fait avec une latence constante, la complexité pratique de l'écriture et de la recherche passe de O(n) à O(1).

Un modèle de calcul basé sur la loi de Amdahl a permis de calculer l'amélioration nette de la performance du système. Bien qu'aucun algorithme complexe de classification n'a été utilisé, l'utilisation seule de l'identification par signature a donné une accélération globale d'un facteur plus grand que 36, alors qu'un facteur de seulement 18 était nécessaire. L'utilisation d'une CAM a servi d'accélérateur de recherche parallèle et le facteur d'accélération amélioré a atteint une valeur de 1083 (Tableau 6.4).

L'utilisation de cette méthode pour l'identification des protocoles de haut niveau permet une diminution de la charge de travail effectuée par le logiciel lors de session multimédia. Ce dernier ne traite alors que 2.65% de la bande passante.

La méthode proposée permet la classification en temps réel ou sans latence de paquets identifiés sur la base de leur contenu et de certains champs d'en-tête.

Les nouvelles avenues d'application de cette technologie peuvent permettre la classification en temps réel des paquets, même si ces derniers arrivent à grand débit (1 Gbps dans ce mémoire). Ce genre d'accélération permettrait notamment la gestion de la qualité de service

ou la re-direction de paquets pour l'écoute électronique. Il serait possible de reconnaître les paquets sur la base de leur contenu et de les classer au niveau réseau selon leur importance, ou encore avec un niveau de priorité indiqué par l'application à laquelle ils sont associés.

Le genre de fonctionnalité présentée dans ce mémoire est très en demande dans les nouvelles familles de processeurs réseau plus intelligents et plus rapides.

7.1 Poursuite de la recherche

La suite du projet devrait explorer les améliorations discutées aux chapitres « 4.3.5 Accélération possible de l'algorithme logiciel ». Ces premières améliorations visent l'accélération de l'algorithme logiciel. Il est nécessaire d'accélérer le cas le plus complexe qui est la recherche de motifs, (chaîne de caractères) pour l'identification de SDP qui est très coûteuse, en nombre de cycles. Des algorithmes sont mentionnés ainsi que des méthodes prenant en compte une architecture à 32 bits qui grouperait 4 caractères de 8 bits pour rechercher 32 bits à la fois.

La section « 5.1 Analyse des fonctions » propose aussi des améliorations possibles au niveau matériel. Les machines de traitement des caractères, au niveau matériel promettent une très bonne accélération des pires cas. L'inspection de paquets UDP avec l'aide de support matériel serait un atout dans l'accélération de la recherche de champs variables, tel que nécessaire avec SDP.

Les signatures utilisées et développées dans ce mémoire utilisent 12 octets. Il serait possible de faire une prospection pour valider que l'algorithme pourrait utiliser seulement une adresse source et un port source, soit une signature de 6 octets au lieu de 12. L'adresse destination

et le port destination sont peut-être superflus pour l’identification par signature. Cette hypothèse reste à vérifier.

RÉFÉRENCES

Bibliographie

[OMA98] O.Cherkaoui, **La téléinformatique**. *McGraw-Hill*, 1998, Montréal, ISBN 2-89461-186-2. 348 pages.

[SOC01] H. Chang, G. Martin. **Surviving the SoC Revolution**. Kluwer Academic Publishers, 2001. ISBN 0-7923-8679-5

[HEN98] J.L. Hennessy, **Architecture des ordinateurs**. *Thomson internationnal.* 1994. France. ISBN 2-84180-022-9. pages 31-35.

[COR94] T. Cormen, C. Leiserson, R.Rivest. **Introduction à l'algorithmique.** *Édition DUNOD 1994. ISBN-2-10-003-3128-7.* 1019 pages.

Articles

[GUP01] Gupta, P.; McKeown, N.; **Algorithms for packet classification.** *IEEE Network*, Volume: 15 Issue: 2 , Mar/Apr 2001 Pages 24 –32

[CLA01] S. Lyer, R.R.Kompella, A. Shelat. PMC-Sierra Inc. **ClassPI: An architecture for fast and flexible packet classification**. *IEEE Network*, Volume: 15 Issue: 2, Mars avril 2001. Pages 33-41

[GUP00] D. Shah, P. Gupta. **Fast incremental updates on Ternary-CAM for routing lookups and packet classification**. HOT Interconnects 8 : un symposium sur l'interconnexion Haute Performance des systèmes et des bus pour les interfaces réseaux. Août 2000. 9 pages. http://www.hoti.org/archive/hoti8papers/018.pdf

[USP] Brevet USA 6,081,440. **Ternary Content Addressable Memory (CAM) having fast insertion and deletion of data values**. http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=/netahtml/srchnum.htm&r=1&f=G&l=50&s1='6,081,440'.WKU.&OS=PN/6,081,440&RS=PN/6,081,440

[WEL01] F. Welfeld, M. Nossik. **Network Processing in Content Inspection Applications**, ISSS'01, Octobre 1-3, 2001,

Montréal, Québec, Canada. 2001. Pages 197-201
http://www.solidum.com/resource_center/pdfs/92002-WP-1.1.1-Content Inspection.pdf

[GUP99] P. Gupta and N. McKeown, **Packet classification on multiple fields.** Proceedings de ACM SIGCOMM, Septembre 1999, pages 147–160.

[MOG87] J. C. Mogul, R. F. Rashid, and M. J. Accetta. **The packet filter : An efficient mechanism for user-level network code**. Proceedings of SOSP. Austin, TX. Novembre 1987. pages 39-51.

[JAY94] M. Jayaram, R. K. Cytron, D. C. Schmidt, and G. Varghese. **Efficient Demultiplexing of Network Packets by Automatic Parsing**. Soumit à ACMSIGPLAN'95, une Conférence sur les langages de programmation, le design et l'Implémentation, ACM, 1994.
http://citeseer.nj.nec.com/cachedpage/56411/2

[MCC93] S. McCanne, V. Jacobson. **The BSD packet Filter : A new architecture for user-level packet capture**. Proceedings de la conférence USENIX hivers 1993. Association USENIX, Janvier 1993. pages 259-269.
http://citeseer.nj.nec.com/mccanne92bsd.html

[BAI94] M. L. Bailey, B. Gopal, P. Sarkar, M. A. Pagels, et L. L. Peterson. **Pathfinder : A pattern-based packet classifier**. Proceedings du 1er Symposium sur les systèmes d'Opération, de Design et d'implémentation. USENIX Association, Novembre 1994.
http://citeseer.nj.nec.com/update/135882

[BEC01] D. Becker, M. Singla, R. Todorovic, Q. Wang. **PTREE : A System for Flexible Efficient Packet Classification**. Rapport CS524, Printemps 2001.
http://www.cs.wustl.edu/~qwang/research/cs535-TBF_DAC.doc

[SRI98] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, **Scalable level 4 switching and fast firewall processing**. Proceedings de ACM SIGCOMM'98, Septembre 1998, pages: 203–214.
http://citeseer.nj.nec.com/article/srinivasan98fast.html

[LAK98] T.V. Lakshman and D. Stiliadis. **High Speed Policy-based Packet Forwarding Using E-cient Multi-dimensional**

Range Matching. Proceedings de ACM SIGCOMM'98, 1998. http://citeseer.nj.nec.com/update/117552

[BEG99] A. Begel, S. McCanne, and S. L. Graham. **BPF+ : Exploiting Global Data-flow Optimization in a Generalized Packet Filter Architecture**. SIGCOMM'99, Août 1999. http://citeseer.nj.nec.com/update/239339

[SCA00] **Packet classification with cyberCam.** Novembre 2000. http://www.sibercore.com/pdf/an_scan001_1.pdf

Références WEB

[CIN96] page de CINDERELLA, http://www.ee.princeton.edu/~yauli/cinderellaa

[RFC1889] **Basic RTP Specs**, spécification du Protocol. http://andrew2.andrew.cmu.edu/rfc/rfc1889.html

[RFC1890] **Bas RTP Profile for Audio and Video Conferences with Minimal Control**. http://andrew2.andrew.cmu.edu/rfc/rfc1890.html

[RTPN] H. Schulzrinne. **RTP News**. http://www.cs.columbia.edu/~hgs/rtp

[RFC1889] **RTP: A Transport Protocol for Real-Time Applications** ftp://ftp.isi.edu/in-notes/rfc1889.txt

[RFC3261] **SIP: Session Initiation Protocol** ftp://ftp.isi.edu/in-notes/rfc3261.txt

[VOIP] **Video Conferencing over IP.** http://www.vide.gatech.edu/cookbook2.0/

[H323] **H.323 Standards**. http://www.openh323.org/standards.html

[RSVP] **ReSerVation Protocol.** http://www.isi.edu/div7/rsvp/rsvp.html

[RSTP] **Real-Time Streaming Protocol (RTSP).** RealNetworks, Netscape Communications et Columbia University. http://www.cs.columbia.edu/~hgs/rtsp/

[RFC2326] **Real Time Streaming Protocol (RTSP)**. H. Schulzrinne, A. Rao, R. Lanphier. Avril 1998. (Statut: Standard proposé)

[RFC2327] **SDP: Session Description Protocol**. M. Handley, V. Jacobson. Avril 1998. (Updated by RFC3266) (Status: PROPOSED STANDARD)

[RFC2734] P. Johansson, **IPv4 over IEEE 1394**. Décembre 1999. ftp://ftp.isi.edu/in-notes/rfc2734.txt

[RFC3146] K. Fujisawa, A. Onoe, **Transmission of IPv6 Packets over IEEE1394 Networks**. Octobre 2001. ftp://ftp.isi.edu/in-notes/rfc3146.txt

[RFC2855] K. Fujisawa, **DHCP for IEEE 1394**. Juin 2000. ftp://ftp.isi.edu/in-notes/rfc2855.txt

[ITU] International Telecommunications Union. http://www.itu.int

[ETH] Gerald Comb, **Ethereal – Analyseur de protocole réseau**. http://www.ethereal.com/

[PCAP] **WinPcap: the Free Packet Capture Architecture for Windows**. http://winpcap.polito.it/

[CVE] Mentor Graphics. **Seamless Hardware/Software Co-Verification**. http://www.mentor.com/seamless/products.html

[SMPTE] SMPTE STANDARD pour la Télévision. **Bit-Parallel Digital Interface Component Video Signal 4:2:2 16´9 Aspect Ratio**. Révision de ANSI/SMPTE 267M-1994, ANSI/SMPTE 267M-1995.

ANNEXES

1. Le code C du programme, le programme d'analyse des protocoles supportant les fichiers de type Wincap, ainsi que le code C du programme pour la simulation dans Seamless [CVE] sont disponibles à cette adresse :

   http://www.grm.polymtl.ca/~lepage/Maitrise/code_C/

2. Le code VHDL d'une CAM matérielle et un exemple de code pour un contrôleur d'interface AMBA, sont également disponibles à cette adresse :

   http://www.grm.polymtl.ca/~lepage/Maitrise/code_VHDL/