

# 2008

UFR Ingénieurs 2000

Vivien Boistuaud

Florence Fraux

Julien Herr

## **ATM – ASYNCHRONOUS TRANSFER MODE SOUS LINUX**

*Ce document a été réalisé par V. Boistuaud, F. Fraux et J. Herr dans le cadre des travaux pratiques d'ATM coordonnés par Catherine Bernard, au sein de l'UFR Ingénieurs 2000 de l'Université de Marne-la-vallée.*

## Sommaire

|                                                               |    |
|---------------------------------------------------------------|----|
| Introduction.....                                             | 3  |
| 1. Présentation générale du TP .....                          | 4  |
| 1. Matériel utilisé.....                                      | 4  |
| 2. Montage initialement utilisé.....                          | 5  |
| 3. Installation des outils nécessaires.....                   | 6  |
| 2. Configuration initiale du matériel ATM.....                | 7  |
| 1. Connexion série au commutateur ATM .....                   | 7  |
| 2. Utiliser le matériel ATM ForeRunner sous Linux .....       | 11 |
| 1. Vérifier que le démon ARP ATM est inactif.....             | 11 |
| 2. Vérifier la présence de la carte ATM .....                 | 11 |
| 3. Charger le pilote de la carte ATM.....                     | 12 |
| 4. Configurer manuellement une commutation sur le Switch..... | 12 |
| 5. Validation de la communication ATM .....                   | 13 |
| 3. Utiliser Ethernet au dessus d'ATM.....                     | 15 |
| 1. Concepts de Classical IP .....                             | 16 |
| 2. Configuration statique avec PVC .....                      | 18 |
| 3. Configuration dynamique avec SVC.....                      | 20 |
| 4. Utilisation d'un serveur ATMARP.....                       | 21 |
| 4. LAN Emulation .....                                        | 24 |
| 1. Principes de la LAN Emulation.....                         | 24 |
| 1. Cas d'un réseau Ethernet simple .....                      | 25 |
| 2. Cas du réseau Ethernet à plusieurs VLAN.....               | 26 |
| 2. Expérimentations d'ELANs.....                              | 27 |
| Conclusion .....                                              | 28 |

## Introduction

Les applications de l'informatique et des communications ont augmenté de façon exponentielle ces vingt dernières années. A cette époque, les technologies de réseaux téléphoniques ne permettaient de transmettre des informations qu'à un faible débit, et les technologies de réseaux informatiques avaient un débit moyen (10Mbps en Ethernet) mais sans aucune notion de qualité de service.

C'est pourquoi, dans les années 1990, un consortium formé par des entreprises européennes, américaines et asiatiques a créé la technologie ATM. Celle-ci a été conçue pour fournir un réseau informatique à haut débit (plusieurs Gigabits actuellement), et qui assure les notions de qualité de service, via un système proche de ceux utilisés en télécommunications : la commutation de cellules.

Dans le cadre de nos études d'ingénieur en informatique et réseaux au sein de l'UFR Ingénieurs 2000 de l'Université de Marne-la-vallée, nous avons pu découvrir la technologie ATM, et expérimenter son fonctionnement au travers d'une séance de travaux pratiques.

Dans un premier temps, ce rapport présente le matériel utilisé pour réaliser le TP, et les conditions dans lesquels il s'est déroulé. Dans un second temps, il précise les moyens utilisés pour configurer le matériel et réaliser nos expérimentations, ainsi que les résultats obtenus et de courtes explications sur ce qui était attendu en théorie lors des manipulations réalisées.

## 1. Présentation générale du TP

Cette section regroupe les informations utiles au lecteur pour comprendre la démarche suivie lors de ce TP, ainsi que le matériel utilisé afin que les manipulations puissent être reproduites. Le matériel indiqué a été fourni par l'UFR aux étudiants dans le cadre de cette séance de TP.

### 1. Matériel utilisé

Pour mettre en œuvre ce TP, nous avons utilisé le matériel suivant :

- 2 ordinateurs équipés de Debian GNU/Linux, distribution Etch (la version la plus stable lors de l'écriture de ce document) ;
- Les logiciels `aread`, `awrite`, `atmarpd`, `atmaddr` installés sur la machine (installation détaillée ci-après) ;
- Une carte ForeRunner LE ATM 155Mbps par ordinateur ;
- 1 commutateur ATM Marconi LE155, 12 ports RJ-45 à 155Mbps sur 3 interfaces ;
- 2 câbles réseaux RJ-45 Ethernet Classe 3 droits.

Le matériel ATM utilisé possède une connectique RJ-45 proche de l'Ethernet. Il est important de veiller à ne pas confondre la carte ForeRunner avec une carte Fast-Ethernet classique dont les ordinateurs de tests sont également dotés.

Dans la section suivante, nous détaillons le montage et la façon dont l'ensemble de ces matériels interagissent ensemble.

## 2. Montage initialement utilisé

A l'origine, le montage que nous avons utilisé dans ce rapport de TP est le suivant :

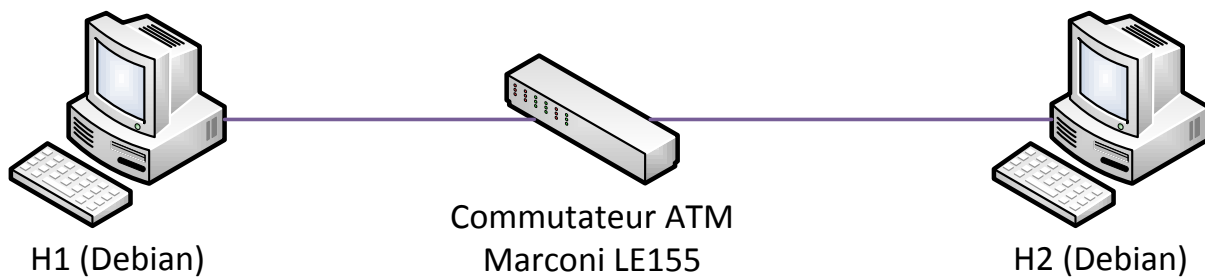


Figure 1 - Configuration du montage principal

Les deux machines sont, comme décrites précédemment, équipées d'interfaces ATM 155Mbps. Celles-ci sont utilisées conjointement à deux câbles RJ-45 droits pour les connecter au commutateur ATM.

Deux choses sont importantes à noter pour la suite de ce TP :

La machine H1 est également reliée au commutateur par un port Série (ttyS0) qui lui permet d'accéder à la console d'administration interne du matériel concerné.

Le commutateur utilisé n'est pas un commutateur cœur de réseau, mais un commutateur périphérique (*edge switch*). Aussi, lors des différentes manipulations et configurations, le *Virtual Path Identifier* (VPI) utilisé pour décrire la table de routage ATM sera toujours à 0 (zéro). Seul le champ VCI (*Virtual Channel Identifier*) aura une valeur variable en fonction du canal de communication.

### 3. Installation des outils nécessaires

Plusieurs outils sont nécessaires à une bonne appréhension de ce TP. Certaines peuvent être installées par le système avancé de gestion de paquets de Debian GNU/Linux.

#### 1. Outils de la suite linux-atm

Le projet Linux-ATM, hébergé par sourceforge, a pour vocation de fournir un ensemble d'outils permettant la gestion de périphériques et de communications ATM sous Linux. Vous trouverez les informations concernant ce projet sur le site internet : <http://linux-atm.sourceforge.net>.

Notez qu'à l'heure actuelle, ces outils sont considérés comme *expérimentaux*. Aussi, il est possible que vous rencontriez des erreurs des logiciels incriminés si vous tentez de réaliser des commandes complexes ou inusuelles.

Pour installer les outils sous Debian (ils sont déjà installés dans les salles administration systèmes de l'UMLV), vous devez exécuter la commande suivante :

```
# apt-get install atm-tools
```

Notez que vous devrez aussi vous procurer les pilotes de la carte ATM que vous utiliserez sur votre machine.

## 2. Configuration initiale du matériel ATM

Afin de prendre contact avec le matériel ATM utilisé dans le cadre de ce TP, nous devons nous familiariser avec les commandes et interfaces de communication qui vont permettre de l'exploiter.

### 1. Connexion série au commutateur ATM

Afin de configurer le commutateur ATM Marconi LE155, il est nécessaire de se connecter par port série, dans la mesure où les interfaces ATM ne sont pas encore configurées.

Pour ce faire, il suffit de connecter le port COM1 de notre machine H1 au port COM1 du Switch ATM. Ensuite, il est nécessaire de lancer l'outil de communication série `minicom` (ou, sous Windows, `hyperterminal` ou tout autre logiciel équivalent).

Ceci se fait via la commande :

```
host1:~# minicom -o
```

Puis, il faut entrer dans le menu interactif de configuration de `minicom`, et entrer les paramètres de communication suivants :

- CTRL-A Z pour entrer dans le menu de configuration
- O dans ce menu, pour configurer le port série.
- Périphérique : `/dev/ttyS0`
- Type de communication : `9600 8N1` (9600 bauds, 8 bits, pas de parité, 1 bit de stop)
- Le contrôle de flux est également désactivé.

Il suffit ensuite de quitter `minicom` et de le relancer pour se connecter au commutateur ATM Marconi.

Une fois connecté, le login à utiliser pour la connexion est « [ami](#) » et aucun mot de passe n'est requis (laisser un mot de passe vide).

Pour réinitialiser la configuration du commutateur, entrez alors la commande suivante :

```
# system cdb init

WARNING: This command will re-initialize the CDB and reboot the switch. It
will
    remove ALL permanent information from the database INCLUDING the
    configuration of all the network interfaces.
Do you wish to continue? (y or n): y
ColdBoot

Serial OK
Testing DRAM...
16 MB DRAM on board
DRAM Test Passed

Switch Control Processor CF-16  May  5 1998
Copyright 1995, FORE Systems, Inc.
Copyright 1992, Intel Corporation

Testing Peripherals...
Timer OK
Clock OK
SRAM OK
SDB OK

-----
Decompressing...

Adding 3025 symbols for standalone.
Attaching network interface lo0... done.

ForeThought Switch Control Software
Copyright (C) 1992-1999  FORE Systems, Inc.
All rights reserved.

Method_Hash_Stats: read 0 writes 435 %-read-collisions: 0 %-write-collisions:
32FEB 13 13:36:01 tAsx:  debug: getting eeprom version
FEB 13 13:36:01 tAsx:  debug: Getting NVRAM
FEB 13 13:36:01 tAsx:  debug: Got the NVRAM correctly
FEB 13 13:36:01 tAsx:  NOTICE: Platform (LE 155)
FEB 13 13:36:01 tAsx:  NOTICE: WED FEB 13 13:35:55 2008

FEB 13 13:36:01 tAsx:  INFO: Trying to restore fs:/hidden/cdb.dat file.....
FEB 13 13:36:01 tAsx:  INFO: Setting Switch Timezone to none
FEB 13 13:36:01 tAsx:  NOTICE: ipft000 initializing to defaults:
FEB 13 13:36:01 tAsx:  debug: Entering asx_open
FEB 13 13:36:01 tAsx:  debug: Initializing HDCOMP
FEB 13 13:36:01 tAsx:  NOTICE: LE 155 Switch Board f20fb19e (0)
FEB 13 13:36:01 tAsx:  debug: fabric_open(): fabric 0 is cold
FEB 13 13:36:01 tAsx:  NOTICE: f_s_m: fabric 0 init completed - progressing to
.FEB 13 13:36:01 tAsx:  NOTICE: link 1CTL up
FEB 13 13:36:01 tAsx:  NOTICE: link 1CTL up
```

```

FEB 13 13:36:01 tAsx:  debug: IL40000 Info: ILMI[--]: sub-system started
succyFEB 13 13:36:01 tAsx:  NOTICE: spvx001 Notification: Upgrading pre FT-6.0
spvx bFEB 13 13:36:01 tAsx:  NOTICE: SPVxC R-GROUP[----]: Configuring SPVC
Redundanc.FEB 13 13:36:01 tAsx:  debug: ROUT000 Info: ftpnni_atmr_pol_init:
default protdFEB 13 13:36:01 tAsx:  NOTICE: SWX_UNI[1CTL(56)]0]
sig_swx_restore_interfaces_nFEB 13 13:36:06 tAsx:  INFO: jalapeno_construct:
autoconfigured jalapeno=1C(0)FEB 13 13:36:06 tAsx:  INFO:
jalapeno_construct: Configured jalapeno=1C(0) "d"FEB 13 13:36:06 tAsx:
INFO: jalapeno_shmem_warm: Jalapeno=1C(0) is cold (JSM)FEB 13 13:36:06 tAsx:
INFO: jalapeno_construct: autoconfigured jalapeno=1B(0)FEB 13 13:36:06 tAsx:
INFO: jalapeno_construct: Configured jalapeno=1B(0) "d"FEB 13 13:36:06 tAsx:
INFO: jalapeno_shmem_warm: Jalapeno=1B(0) is cold (JSM)FEB 13 13:36:06 tAsx:
INFO: jalapeno_construct: autoconfigured jalapeno=1A(0)FEB 13 13:36:06 tAsx:
INFO: jalapeno_construct: Configured jalapeno=1A(0) "d"FEB 13 13:36:06 tAsx:
INFO: jalapeno_shmem_warm: Jalapeno=1A(0) is cold (JSM)
S_ForeThought_6.1.1 FCS (1.50873) (1e155) (ATM SWITCH)

login: FEB 13 13:36:14 tAsx:  debug: netmod_mark_hardware_accessible: for
netm2FEB 13 13:36:14 tAsx:  debug: netmod_mark_nm_connectionReady: for netmod
2
FEB 13 13:36:14 tAsx:  NOTICE: SNMP_TRAP: Specific 5 (asxNetModuleUp) [0] [2]
[]FEB 13 13:36:14 tAsx:  debug: netmod_mark_hardware_accessible: for netmod 1
FEB 13 13:36:14 tAsx:  debug: netmod_mark_nm_connectionReady: for netmod 1
FEB 13 13:36:14 tAsx:  NOTICE: SNMP_TRAP: Specific 5 (asxNetModuleUp) [0] [1]
[]FEB 13 13:36:14 tAsx:  NOTICE: link 1B1 up
FEB 13 13:36:14 tAsx:  NOTICE: SNMP_TRAP: Specific 29 (asxLinkUp) [8] [1B1]
[Xm]FEB 13 13:36:14 tAsx:  NOTICE: link 1B1 up
FEB 13 13:36:14 tAsx:  debug: netmod_mark_hardware_accessible: for netmod 0
FEB 13 13:36:14 tAsx:  debug: netmod_mark_nm_connectionReady: for netmod 0
FEB 13 13:36:14 tAsx:  NOTICE: SNMP_TRAP: Specific 5 (asxNetModuleUp) [0] [0]
[]FEB 13 13:36:14 tAsx:  NOTICE: Hostname is ATM SWITCH
FEB 13 13:36:14 tAsx:  NOTICE: CDBe005: SNMP: pnni_init_all_nodes_from_cdb:
nodtFEB 13 13:36:14 tAsx:  NOTICE: CDBe005: SNMP:
pnni_init_crankback_tries_from_cdtFEB 13 13:36:14 tAsx:  NOTICE: finger_init:
listening on 79
FEB 13 13:36:14 tAsx:  NOTICE: snmp_init_static_route_entries : registered
routsFEB 13 13:36:14 tAsx:  NOTICE: scp_host_init: Start initializing scp 0
FEB 13 13:36:14 tAsx:  NOTICE: scp_host_init: Complete initializing scp
FEB 13 13:36:14 tAsx:  NOTICE: No default route.

FEB 13 13:36:14 tAsx:  NOTICE: syslog facility not found in CDB (err 1),
settinnFEB 13 13:36:14 tAsx:  NOTICE: log_reinit_from_cdb: listening on 6532
FEB 13 13:36:14 tAsx:  NOTICE: No default route.

FEB 13 13:36:14 tAsx:  NOTICE: SNMP_TRAP: Specific 2 (warmStart) [Xmit]
FEB 13 13:36:14 tAsx:  NOTICE: Initialization complete.
FEB 13 13:36:14 tAsx:  NOTICE: sig_swx_set_aalstr_rcvbuf_size: restored
SO_RCVBeFEB 13 13:36:15 tAsx:  NOTICE: Timeout: largest skew so far: 20090ms
skew
FEB 13 13:36:15 tAsx:  NOTICE: SNMP_TRAP: Specific 1092 (asxQ2931Up) [1CTL]
[56]
FEB 13 13:36:58 tAsx:  NOTICE: switch rebooted 1 minute ago due to a request
--

```

Une fois que la ligne « `ATM SWITCH:->` » est affichée, vous pouvez de nouveau saisir le login et le mot de passe de connexion au routeur. Ces informations ne sont pas remises à zéro par la commande `system cdb init`.

## 2. Utiliser le matériel ATM ForeRunner sous Linux

Une fois que nous avons accès à la console d'administration du commutateur, nous devons également nous assurer que les deux hôtes (H1 et H2) seront capables de communiquer par le biais de leurs interfaces ATM.

Sur chaque machine nous allons donc :

- Vérifier que le processus « `atmarpd` » n'est pas lancé ;
- Ajouter le module « `nicstar` » contenant le pilote de la carte ATM ;
- Démarrer deux daemons :
  - « `atmsigd` » pour la signalisation ATM ;
  - « `ilmid` » pour l'auto-configuration.

### 1. Vérifier que le démon ARP ATM est inactif

Pour cela, nous allons tuer les processus correspondants qui tournent potentiellement :

```
# killall atmarpd
```

### 2. Vérifier la présence de la carte ATM

Pour cela, on utilise la commande `lspci` qui donne un résultat proche du suivant (les informations inutiles ont été retirées) :

```
# lspci
00:00.0 Host bridge: Intel Corporation 82845 845 (Brookdale) Chipset Host
Bridge (rev 03)
...
02:07.0 ATM network controller: Integrated Device Technology, Inc.
IDT77201/77211 155Mbps ATM SAR Controller [NICSTAR] (rev 03)
02:08.0 Ethernet controller: D-Link System Inc RTL8139 Ethernet (rev 10)
02:0c.0 Ethernet controller: 3Com Corporation 3c905C-TX/TX-M [Tornado] (rev
78)
...
```

### 3. Charger le pilote de la carte ATM

Le pilote, qui doit être chargé, est utile pour la mise en place de la communication. Selon le type de carte ATM le module n'est pas le même. Dans notre cas, il s'agit de carte ForeRunner 155 et 25, c'est donc le module « nicstar » qui va devoir être chargé. Tout d'abord, il faut vérifier qu'il est présent sur le système à l'aide de la commande `lsmod`, si c'est le cas il peut être chargé avec la commande `modprobe`.

```
# lsmod | grep nicstar
# modprobe nicstar
# lsmod | grep nicstar
nicstar                25340    2
```

### 4. Configurer manuellement une commutation sur le Switch

L'objectif de cette section est de créer manuellement une route dans le commutateur, pour permettre la communication entre H1 et H2 en mode ATM.

La commutation sous ATM est différente de la commutation Ethernet : en Ethernet, on utilise les adresses des machines comme identifiant dans les paquets. En ATM, les adresses ont une longueur de 20 octets. Dans la mesure où un paquet ATM fait 53 octets, si on stockait l'adresse de l'expéditeur et l'adresse du destinataire dans chaque paquet, on pourrait seulement faire transiter 13 octets d'informations, ce qui serait totalement aberrant.

A la place, en ATM, les adresses ne sont utilisées que lorsqu'on souhaite établir dynamiquement une route. La route est alors établie une fois pour toutes : on est en permanence en mode connecté lors d'un transfert de données.

Les tables de routage ATM des commutateurs tiennent alors compte de deux facteurs : le VPI (Virtual Path Identifier) et le VCI (Virtual Chanel Identifier). Ces deux données sont attachées à une interface en particulier.

Les paquets de données ne contiennent alors que ces deux informations : VPI et VCI, soit 28 bits ainsi que 12 autres octets d'en-tête (payload type, cell loss priority...). Le commutateur analyse le VPI et le VCI reçus en fonction de l'interface sur laquelle ils sont reçus : s'ils sont dans la table ATM du commutateur, alors il regarde sur quelle interface, quel VPI et quel VCI il doit retransmettre le paquet. Si les VPI et VCI sont invalides, le paquet est ignoré.

De l'autre côté du commutateur, le périphérique doit connaître les VPI et VCI correspondants à l'interface à laquelle il est connecté pour pouvoir envoyer un paquet à une machine donnée.

Sur notre commutateur, pour créer manuellement une commutation, on doit utiliser la commande suivante :

```
ATM SWITCH:-> connections channel new -ivpi 0 -ivci 57 -ovpi 0 -ovci 57 -  
iatmif 1B1 -oatmif 1A
```

On a alors défini que si on reçoit sur l'interface 1B1 (où est connecté H1) un paquet ATM ayant pour VPI 0 et pour VCI 57, alors on doit le rediriger vers l'interface 1A1 (où est connecté H2) et le marquer avec le VPI 0 et le VCI 57.

*Note : ici, on a défini une communication UNIDIRECTIONNELLE uniquement de H1 vers H2. Pour créer une route inverse, il suffirait d'inverser les noms des ports (1A1 et 1B1 dans la commande).*

## 5. Validation de la communication ATM

Pour vérifier que la communication unidirectionnelle de H1 vers H2 fonctionne correctement, nous avons utilisé les commandes `aread` et `awrite` en précisant que nous souhaitons réaliser de l'ATM natif (option `-a`) comme précisé ci-après.

Du côté de l'émission (H1):

```
# awrite -a 0.57 coucou
6
```

Du côté de la réception (H2):

```
# aread -a 0.57
63 6F 75 63 6F 75
```

Notez qu'il faut exécuter aread avant d'exécuter awrite, si l'on veut recevoir convenablement les informations. De plus, le commutateur -a est indispensable, car nous sommes en ATM natif, et non en IP sur ATM pour le moment.

Si la communication se fait bien entre les deux machines, H2 devrait recevoir le message « coucou » envoyé par H1 (qui sera affiché sous forme hexadécimale).

De plus, on peut faire un test de débit ATM à l'aide de la commande exécutée depuis H1 :

```
pc0b004-5:~/TPATM# ttcp_atm -t -a -s 0.57
ttcp-t: buflen=8192, nbuf=2048, align=16384/0, port=5013 atm -> 0.57
ttcp-t: socket
ttcp-t: 16777216 bytes in 0.998554 real seconds = 16407.725571 KB/sec
(134.412088 Mb/sec)killall atmarpd
```

### 3. Utiliser Ethernet au dessus d'ATM

A l'origine, ATM avait été prévu pour créer des réseaux de bout en bout : depuis le client final, jusqu'au serveur de données. Cependant, avec la percée des solutions Fast-Ethernet et la démocratisation des coûts de ces solutions, ATM s'est surtout imposé au sein des réseaux d'opérateurs, comme solution efficace et fiable pour la backbone du réseau (épine dorsale).

Cependant, de par ses origines, ATM a également été pensé pour pouvoir transporter des paquets Ethernet au dessus d'un réseau ATM de façon transparente. En effet, la philosophie initiale des concepteurs était de pouvoir agréger sur un seul réseau (un seul câble) la transmission de tous types de flux (auto/vidéo/data).

De plus, les principaux problèmes rencontrés avec l'ATM en LAN sont que l'ATM ne permet pas d'envoyer des paquets broadcast, ce qui est nécessaire en LAN, et que l'ATM est orienté connexion, alors que d'autres protocoles avec lesquels il est amené à s'interfacer en LAN (ethernet dans notre cas) sont des protocoles sans connexion.

Deux solutions ont été mises au point pour contourner ces problèmes : la première est appelée Classical IP et permet de connecter à un réseau IP des machines pourvues de cartes ATM, la seconde porte le nom de LAN emulation et permet d'interfacer l'ATM avec l'Ethernet.

Cette section montre la mise en œuvre de la technologie Classical IP, permettant l'échange de paquets Ethernet au dessus d'ATM, et qui peut se configurer de plusieurs façons : de façon statique (PVC) ou de façon dynamique (SVC). Ici, nous allons présenter la configuration nécessaire à la mise en œuvre de ces deux cas de figure.

Dans toutes les captures qui suivent nous prendrons le point de vue de la même machine dont l'adresse IP est 198.168.X.1. Le X correspond au numéro de l'interface ATM.

## 1. Concepts de Classical IP

Classical IP, également appelé CLIP (Classical IP over ATM), est une norme spécifiée au sein de l'IETF. Cette solution fournit un service de niveau 3 au-dessus d'ATM en permettant d'interfacer la couche IP avec la couche AAL-5. La couche IP peut ainsi utiliser directement les services de la couche AAL-5.

Cette solution est également de type client/serveur. Ce modèle émule un sous-réseau IP au-dessus d'un réseau ATM ; le terme employé dans ce cas est *LIS* (Logical IP Subnetwork). Un LIS est une collection de stations ATM et de routeurs IP ATM ; le LIS est une partie d'un sous-réseau IP classique.

Le réseau ATM et les sous-réseaux IP constituant le réseau IP sont interconnectés par des routeurs. Chaque LIS doit posséder un serveur implémentant un protocole de résolution d'adresses ATM/ARP (ATM Address Resolution Protocol). Ce serveur enregistre ainsi les correspondances des adresses IP/ATM des routeurs qui ont établi avec lui une connexion.

L'utilisation de ce protocole n'est pas transparente aux couches supérieures, ce qui permet théoriquement aux applications de bénéficier des qualités de service offertes par la couche ATM. Cependant les qualités de service ATM offertes aux applications ne sont plus disponibles dans le cas de l'interconnexion de deux LIS.

Pour toute opération utilisant ATM/ARP (que ce soit statique ou dynamique), il faut activer le daemon `atmarpd` qui est la passerelle entre le noyau et le réseau :

```
# atmarpd -b -l atmarpd.log
```

L'option `-l` permet de stocker les logs dans le fichier spécifié juste après :

```
atmarpd:ARP: Linux ATM ARP, version 2.4.1
atmarpd:IO: ioctl SIOCGIFADDR: Cannot assign requested address
atmarpd:ARP: itf 0 not found
atmarpd:ARP: itf 0 not found
atmarpd:ARP: itf 0 not found
atmarpd:ARP: itf 0 not found
atmarpd:ITF: no such interface (0)
atmarpd:IO: ioctl ATMARP_SENTRY: Cannot allocate memory
```

## Création d'une interface configurable au niveau IP :

```
# atmarp -c atm0
# ifconfig atm0 192.168.0.1 netmask 255.255.255.0 up
```

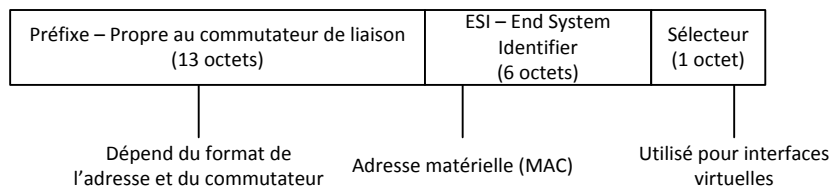
On obtient l'interface suivante :

[illegible]

L'adresse ATM de la carte ATM de l'hôte H1 est obtenue par la commande :

```
# atmaddr -n
47.0005.80FFE1000000F20FB19E.0020485B3EEF.00
```

On remarque, comme cela était expliqué précédemment, que cette adresse a une longueur de 20 octets, et est constituée comme le précise le schéma ci-après :



De plus, le 1<sup>er</sup> octet de l'adresse, appelé AFI, définit le format de celle-ci : 0x47 signifie que le format de cette adresse est de type ICD. Les autres octets du préfixe donnent des informations sur le pays, la zone géographique, le numéro de membre du réseau, le numéro du point d'accès, de la zone et enfin du

commutateur. Ces valeurs sont fonction du commutateur auquel notre machine est connectée, et de sa configuration.

## 2. Configuration statique avec PVC

La technologie Permanent Virtual Channel (PVC) permet de conserver une connexion permanente entre deux unités ATM. Les routes correspondantes au PVC sont fixées manuellement par les administrateurs du commutateur et les machines utilisatrices ayant connaissance d'ATM.

Dans un premier temps, nous avons configuré notre commutateur pour qu'il possède l'adresse IP 192.168.0.3, et qu'il soit possible de joindre cette adresse sur l'interface 1A1 sur le chemin 0.100 et depuis l'interface 1B1 sur le chemin 0.200. Ceci s'est fait à l'aide des commandes :

```
ATM SWITCH:-> interfaces ip modify -ifname qaa0 -ipaddr 192.168.0.3 -mask
255.255.255.0 -state up

ATM SWITCH:-> connections channel new -iatmif 1A1 -oatmif 1B1 -ivpi 0 -ivci 57
-ovpi 0 -ovci 57

ATM SWITCH:-> connections channel new -iatmif 1CTL -oatmif 1A1 -ivpi 0 -ivci
200 -ovpi 0 -ovci 200

ATM SWITCH:-> connections channel new -iatmif 1A1 -oatmif 1CTL -ivpi 0 -ivci
200 -ovpi 0 -ovci 200

ATM SWITCH:-> services atmarp new-pvc -ipaddress 192.168.0.2 -vpi 0 -vci 200

ATM SWITCH:-> connections channel new -iatmif 1B1 -oatmif 1CTL -ivpi 0 -ivci
100 -ovpi 0 -ovci 100

ATM SWITCH:-> connections channel new -iatmif 1CTL -oatmif 1B1 -ivpi 0 -ivci
100 -ovpi 0 -ovci 100

ATM SWITCH:-> services atmarp new-pvc -ipaddress 192.168.0.1 -vpi 0 -vci 100
```

*Remarque : les communications créées sont bidirectionnelles. L'interface qaa0 est une interface IP virtuelle interne au switch. Le port 1CTL est un port interne au commutateur qui permet d'accéder à qaa0.*

Remplissage de la table avec l'adresse de la deuxième machine (192.168.0.2) et celle du switch (192.168.0.3), sur l'hôte H1 :

```
# atmarp -s 192.168.0.2 0.57
# atmarp -s 192.168.0.3 0.100
```

On réalise une opération similaire sur l'hôte H2 pour joindre 192.168.0.1 et 192.168.0.3 (via le chemin 0.200).

Test pour voir s'il est possible de joindre l'autre machine :

```
# ping 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.
64 bytes from 192.168.0.2: icmp_seq=1 ttl=64 time=0.314 ms
64 bytes from 192.168.0.2: icmp_seq=2 ttl=64 time=0.296 ms

--- 192.168.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.296/0.305/0.314/0.009 ms
```

Afin d'aller plus loin, nous avons voulu observer la quantité de paquets échangés lors de l'envoi de messages ICMP ECHO REQUEST de tailles variables. Dans tous les cas, nous avons envoyé 10 paquets exactement (option `-c 10` de la commande ping) mais avons fait varier la taille du paquet échangé (option `-s` de la commande ping). Les résultats sont consignés dans le tableau suivant :

| Taille DATA du paquet ICMP | Nombre de cellules ATM nécessaires |
|----------------------------|------------------------------------|
| <b>4 octets</b>            | 10 cellules                        |
| <b>5 octets</b>            | 20 cellules                        |
| <b>50 octets</b>           | 20 cellules                        |
| <b>100 octets</b>          | 30 cellules                        |
| <b>200 octets</b>          | 50 cellules                        |

Ces résultats s'expliquent car un paquet ATM a une longueur fixe de 53 octets, cependant, le fait d'utiliser ATM, les surcouches AAL5, LLC/SNAP, pour faire

transiter un paquet IPv4 contenant des données au format ICMP ECHO, consomme 49 octets en en-têtes de données :

- En-tête ATM : 5 octets (40 bits)
- Trailer AAL5 (dans la dernière cellule – 8 octets soit 64 bits)
- La LLC (Logical Link Control), 3 octets par paquet
- Le SNAP (SubNetwork Attachment Point), 5 octets par paquet, contenant l'OUI et le PID,
- En-tête IP (20 octets par paquet IP)
- En-tête ICMP (8 octets par paquet IP)

Soit un total de 49 octets d'en-tête.

Aussi, si la taille du paquet ICMP ECHO dépasse les 4 octets, il sera nécessaire d'envoyer deux cellules de plus par paquet IP, ce qui est le cas ici.

On remarque que la technologie PVC est lourde en configuration, tant du côté des clients (une entrée en table ATMARP par machine à contacter) que du côté serveur (établissement des liaisons, adressage IP, routes...). De plus, il est impossible de faire un broadcast sur ce type de réseau.

### 3. Configuration dynamique avec SVC

La technologie Switch Virtual Channel (SVC) permet de créer des connexions dynamiquement quand cela est nécessaire.

Sur le commutateur on peut utiliser la commande `services atmarp new-svc` pour mettre en place un SVC.

Création d'une interface configurable au niveau IP sur l'hôte H1 :

```
# atmarp -c atm1  
# ifconfig atm1 192.168.1.1 netmask 255.255.255.0 up
```

On obtient l'interface suivante :

```
atm0      Lien encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet adr:192.168.1.1  Masque:255.255.255.0
          UP RUNNING  MTU:9180  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:100
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

Remplissage de la table avec l'adresse de la deuxième machine (Hôte H2 ayant pour adresse IP 192.168.1.2) et l'adresse mac de la carte ATM :

```
# atmarp -s 192.168.1.2 47.0005.80FFE1000000F20FB19E.0020485B3EF2.00
```

Test pour voir s'il est possible de joindre l'autre machine :

```
# ping 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=0.314 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=0.296 ms

--- 192.168.1.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.296/0.305/0.314/0.009 ms
Configuration centralisée
avec un serveur ATMARP
```

La configuration est ici plus simple qu'en PVC : il suffit d'ajouter les adresses ATM de chaque machine que l'on souhaite pouvoir contacter sur chaque hôte, et les commutations sont créées automatiquement au niveau du commutateur.

#### 4. Utilisation d'un serveur ATMARP

Après avoir vu comment remplir les tables de commutation manuellement, nous allons maintenant voir la manipulation à effectuer pour que la table ARP se remplisse toute seule. Comme nous l'avons vu précédemment, il faut pour cela définir un serveur ATM/ARP par sous réseau. Dans notre cas, ce dernier se situe sur le commutateur.

Création d'une interface configurable au niveau IP :

```
# atmarp -c atm2
# ifconfig atm2 192.168.2.1 netmask 255.255.255.0 up
```

On obtient l'interface suivante :

```
atm2      Lien encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
00-00

        inet adr:192.168.2.1  Masque:255.255.255.0
        UP RUNNING  MTU:9180  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 lg file transmission:100
        RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

On détermine l'adresse ATM du serveur associé à l'interface qaa2 au moyen de la commande suivante ;

```
service atmarp arpserver show
Interface  State      Arp Server Address
qaa0       enabled   0x47.0005.80.ffe100.0000.f20f.b19e.0020480fb19e.00
qaa1       enabled   0x47.0005.80.ffe100.0000.f20f.b19e.0020480fb19e.01
qaa2       enabled   0x47.0005.80.ffe100.0000.f20f.b19e.0020480fb19e.02
qaa3       enabled   0x47.0005.80.ffe100.0000.f20f.b19e.0020480fb19e.03
```

Remplissage de la table avec l'adresse du switch (192.168.2.3) et l'adresse mac de la carte ATM :

```
# atmarp -s 192.168.2.3 47.0005.80.ffe100.0000.f20f.b19e.0020480fb19e.02
arpsrv
```

Test pour voir s'il est possible de joindre l'autre machine :

```
# ping 192.168.2.2
PING 192.168.2.2 (192.168.2.2) 56(84) bytes of data.
64 bytes from 192.168.2.2: icmp_seq=1 ttl=64 time=0.314 ms
64 bytes from 192.168.2.2: icmp_seq=2 ttl=64 time=0.296 ms

--- 192.168.2.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.296/0.305/0.314/0.009 ms
```

Du côté du commutateur, on peut observer les connexions qui sont créées automatiquement (UBR avec fsig) à l'aide de la commande suivante :

```

ATM SWITCH:-> connections channel show
      Input      Output
AtmIf  VPI  VCI  AtmIf  VPI  VCI  ServCat Protocol Name
1A1    0    5    1CTL  0    49    nrtVBR  fsig    N/A
1A1    0    16   1CTL  0    50    nrtVBR  fsig    N/A
1A1    0    33   1CTL  0    58    UBR     fsig    N/A
1A1    0    35   1B1    0    35    UBR     fsig    N/A
1A1    0    57   1B1    0    57    UBR     pvc    N/A
1A1    0    200  1CTL  0    200   UBR     pvc    N/A
...
1B1    0    35   1A1    0    35    UBR     fsig    N/A
1B1    0    57   1A1    0    57    UBR     pvc    N/A
1B1    0    100  1CTL  0    100   UBR     pvc    N/A
...
1CTL   0    58   1A1    0    33    UBR     fsig    N/A
1CTL   0    100  1B1    0    100   UBR     pvc    N/A
1CTL   0    200  1A1    0    200   UBR     pvc    N/A

```

On observe que les routes suivantes ont été créées :

- Port 1A1, 0.35 => 0.35 Port 1B1
- Port 1B1, 0.35 => 0.35 Port 1A1
- Port 1A1 0.33 => 0.58 Port 1CTL
- Port 1CTL 0.58 => 0.33 Port 1A1

Ces routes coïncident à ce qui était attendu : 1A1 a effectué une recherche ATMARP auprès du serveur (création de la route 0.35/0.58 et vice-versa) puis a envoyé une cellule sur ce chemin à destination de H2. H2 a alors répondu en utilisant le même numéro de chemin (0.35) que celui par lequel il a reçu le paquet.

Ce type de réseau n'est toujours pas capable de supporter le multicast, toutefois la configuration est beaucoup plus simple qu'en PVC car elle ne nécessite que la saisie de deux informations sur chaque machine cliente : l'adresse IP du serveur ATMARP, ainsi que la saisie de l'adresse ATM correspondante.

## 4. LAN Emulation

Il existe une seconde méthode pour la communication Ethernet over ATM, qui est plus sophistiquée que la précédente car elle permet aussi l'émulation des broadcasts Ethernet et la prise en charge des communications Multicast.

De plus, elle est mise en œuvre de façon transparente, ce qui permet d'apporter à la technologie Ethernet les avantages en terme de débit et de fiabilité de la technologie ATM, et ce de façon transparente.

Avec LAN Emulation, il est possible de concevoir un réseau d'entreprise extrêmement étendu mettant en œuvre Ethernet de bout en bout en utilisant ATM comme medium de communication entre les sites, de façon transparente pour les utilisateurs.

### 1. Principes de la LAN Emulation

Comme nous le disions précédemment, l'objectif de la LAN Emulation (ou aussi connu sous l'acronyme LANE) est de permettre au réseau ATM de se comporter comme un réseau LAN, c'est-à-dire de supporter les protocoles d'accès aux média partagés (Ethernet/IEEE 802.3 et IEEE 802.5). En effet, les services disponibles dans une architecture LAN doivent être également disponibles dans une architecture utilisant ATM. Ainsi, il est possible d'interconnecter plusieurs équipements Ethernet de façon transparente pour les utilisateurs.

Le service LAN Emulation doit offrir les caractéristiques suivantes :

- Un service apparemment sans connexion ;
- Un service *multicast* : l'approche adoptée est, dans ce cas, de diffuser les messages à toutes les stations et de laisser chaque station filtrer les messages ;
- Une interface commune avec le(s) driver(s) MAC pour les couches supérieures ;

- La possibilité d'émuler plusieurs LAN sur un même support physique ; l'interconnexion avec des LAN existants.

Le principe du LAN Emulation est d'établir un pont au niveau liaison de la couche MAC vers la couche ATM. Les données sont ensuite échangées entre des stations sur des connexions virtuelles.

Pour cela, l'ensemble des entités sont regroupés en deux sortes : les LEC pour LAN Emulation Client et les LES pour LAN Emulation Serveur. Le LANE a également besoin du BUS (Broadcast and Unknown Server) pour fonctionner (utilisé pour les requêtes broadcast et unicast). Enfin, un LECS (LANE Emulation Configuration Server) est souvent utilisé afin de fournir au LEC l'adresse du LES (un tel service est utile lors de redondance de LES afin de toujours fournir l'adresse d'un LES valide).

## 1. Cas d'un réseau Ethernet simple

Dans le cas d'un réseau Ethernet simple, c'est-à-dire ne comportant qu'un unique VLAN, le réseau ATM doit offrir un LES et un BUS.

Tout équipement Ethernet relié au réseau ATM comporte alors un LEC, qui, lors de son initialisation, établit avec le LES et le BUS une connexion qu'il conservera aussi longtemps qu'il le pourra. Ce LEC est l'unique moyen de communication entre l'équipement Ethernet et le réseau ATM.

L'un des deux problèmes du LEC est donc le suivant : connaissant une adresse MAC, obtenir l'adresse ATM du LEC connaissant cette adresse MAC destination. Il suffit ensuite d'établir une connexion avec cet autre LEC et de lui envoyer le paquet. Le second problème du LEC est de faire parvenir à tous les LEC du réseau les paquets dont l'adresse MAC de destination est l'adresse de broadcast `FF:FF:FF:FF:FF:FF`.

Pour aider les LEC à mener cette tâche à bien, le réseau ATM doit, comme expliqué plus haut, fournir un LES et un BUS. Ceux-ci peuvent faire partie d'un switch ATM ou être des équipements séparés. Lors de son initialisation, chaque

LEC établit une connexion avec le LES et s'enregistre auprès de lui en tant que LEC. Ensuite, lorsqu'un LEC doit faire parvenir un paquet à un autre LEC, il interroge le LES (au moyen d'une requête ARP) en lui fournissant l'adresse MAC de destination. Le LES à son tour interroge les LEC et obtient l'adresse ATM du LEC qui connaît cette adresse MAC, et la renvoie au LEC qui la lui a demandée. Le LEC tient ainsi à jour deux tables : la table ARP, qui à une adresse MAC, associe une adresse ATM, et la table de connexion, qui a une adresse MAC, associe un ou deux VCN (selon que ceux-ci sont utilisés ou non de manière bidirectionnelle).

Enfin, lorsque le LEC doit faire parvenir un paquet broadcast ou de multicast à plusieurs autres LEC, il envoie ce paquet au BUS, qui travaille de concert avec le LES et est donc en mesure de transmettre le paquet à tous les autres LEC. Comme l'AAL 5 ne permettrait pas de savoir à quel paquet Ethernet appartiennent les différentes cellules que le LEC recevra du BUS, le BUS attend d'avoir reçu l'intégralité du paquet dans ses buffers avant de l'envoyer au LEC. Ce fonctionnement permet d'éviter toute confusion entre des paquets broadcast ou multicast en provenance de LEC différents. Un autre rôle du BUS est de transmettre les paquets unicasts lorsque la connexion entre deux LEC n'est pas encore établie, c'est-à-dire le temps que la requête ARP soit exécutée, ce qui permet de gagner du temps. Ainsi, les premiers paquets qui transitent entre deux LEC passent d'abord par le BUS, puis, une fois que la connexion a pu être établie, l'échange entre les deux LEC se fait directement.

Un dernier point à éclaircir est la manière dont les connexions entre le LEC et le LES et le BUS s'établissent. Le LEC a deux moyens de trouver le LES : soit son administrateur lui a donné l'adresse ATM du LES, soit il utilise le LECS. Enfin, lorsqu'il s'enregistre auprès du LES, celui-ci lui indique l'adresse ATM du BUS.

## 2. Cas du réseau Ethernet à plusieurs VLAN

Dans le cas où le réseau Ethernet gère plusieurs VLANs, le réseau ATM qui relie les équipements Ethernet doit en général respecter la topologie vue précédemment. LANE propose donc l'introduction d'une séparation identique à

celle introduite par le VLAN : l'ELAN (Emulated LAN). Chaque VLAN du réseau Ethernet correspond de manière généralement unique à un ELAN du réseau ATM. De même que des paquets Ethernet ne peuvent pas passer d'un VLAN à l'autre, les paquets ATM ne peuvent pas passer d'un ELAN à l'autre. L'explication précédente concernant LEC, LES, et BUS reste valable, à la différence près qu'il y a un LES et un BUS par ELAN, et que chaque équipement Ethernet utilise un LEC par ELAN.

## 2. Expérimentations d'ELANs

Par manque de temps, nous n'avons malheureusement pas pu expérimenter le principe de LAN Emulation.

## Conclusion

Grâce à ce TP, nous avons pu mettre en application les connaissances que nous avons acquises lors des cours d'ATM. Nous avons également pu expérimenter les deux solutions possibles (théoriquement, en pratique qu'une seule malheureusement) pour faire de l'Ethernet sur ATM.

On a pu constater que les deux solutions proposées présentent toutes les deux des avantages similaires : elles permettent la constitution de réseaux virtuels sur le même support physique et l'administration peut être effectuée de manière logique.

Elles présentent également toutes les deux les mêmes inconvénients : des ponts (dans le cas des LAN émulés) ou des routeurs (dans le cas des LIS) sont nécessaires pour permettre l'interconnexion de réseaux virtuels de même nature ; ces équipements risquent de produire des goulots d'étranglement face aux commutateurs ATM beaucoup plus performants en termes de débits ; les problèmes induits par ces goulots d'étranglement risquent de se traduire par des pertes de données et par des délais de transfert importants. Les mécanismes de conversion d'adresses engendrent des échanges de messages entre des entités serveurs et clients, ce qui peut être préjudiciable à certaines applications gourmandes en bande passante.

Les avantages du service LAN Emulation par rapport à Classical IP sont les suivants :

- Cette solution est plus simple ;
- Cette solution est indépendante vis à vis des protocoles de routage ;
- La couche *LAN Emulation* convertit une adresse MAC en une adresse ATM, ce qui rend transparente aux applications l'utilisation de supports ATM, avec cependant l'inconvénient que ces applications ne peuvent pas bénéficier des qualités de service offertes par ATM.

Les avantages de Classical IP par rapport au service LAN Emulation sont les suivants :

- L'encapsulation directe des paquets IP dans les cellules ATM permet d'éviter de passer par les couches protocolaires de niveau MAC, ce qui rend l'utilisation de ces services plus efficaces en termes de débits et de temps de traitements ;
- Classical IP peut de plus soumettre à la couche AAL-5 des tailles maximales de données beaucoup plus grandes que LAN Emulation, ce qui peut améliorer sensiblement les performances de débit entre deux stations ;
- Le trafic de type *broadcast* généré est moins important que dans le cas du service LAN Emulation, puisque la résolution d'adresses MAC n'est pas nécessaire dans ce cas ;
- Il permet enfin aux applications de bénéficier des qualités de service offertes par la couche ATM.