

Version : **2022.02.**

Dernière mise-à-jour : 2022/11/04 05:35

RES402 - Comprendre TCP Version 4

Contenu du Module

- **RES402 - Comprendre TCP Version 4**

- En-tête TCP
- En-tête UDP
- Fragmentation et Ré-encapsulation
- Adressage
- Masques de sous-réseaux
- VLSM
- Ports et sockets
- /etc/services
- Résolution d'adresses Ethernet

En-tête TCP

L'en-tête TCP est codée sur 4 octets soit 32 bits :

1er octet	2ème octet	3ème octet	4 ème octet
Port source		Port destination	
		Numéro de séquence	
		Numéro d'acquittement	
Offset	Flags		Fenêtre
			Checksum
			Pointeur Urgent

1er octet	2ème octet	3ème octet	4 ème octet
Options		Padding	
Données			

Vous noterez que les numéros de ports sont codés sur 16 bits. Cette information nous permet de calculer le nombres de ports maximum en IPv4, soit 2^{16} ports ou 65 535.

L'**Offset** contient la taille de l'en-tête.

Les **Flags** sont :

- URG - Si la valeur est 1 le pointeur urgent est utilisé. Le numéro de séquence et le pointeur urgent indique un octet spécifique.
- ACK - Si la valeur est 1, le paquet est un accusé de réception
- PSH - Si la valeur est 1, les données sont immédiatement présentées à l'application
- RST - Si la valeur est 1, la communication comporte un problème et la connexion est réinitialisée
- SYN - Si la valeur est 1, le paquet est un paquet de synchronisation
- FIN - Si la valeur est 1, le paquet indique la fin de la connexion

La **Fenêtre** est codée sur 16 bits. La Fenêtre est une donnée liée au fonctionnement d'expédition de données appelé le **sliding window** ou la **fenêtre glissante**. Puisque il serait impossible, pour des raisons de performance, d'attendre l'accusé de réception de chaque paquet envoyé, l'expéditeur envoie des paquets par groupe. La taille de cette groupe s'appelle la Fenêtre. Dans le cas d'un problème de réception d'une partie de la Fenêtre, toute la Fenêtre est ré-expédiée.

Le **Checksum** est une façon de calculer si le paquet est complet.

Le **Padding** est un champ pouvant être rempli de valeurs nulles de façon à ce que la taille de l'en-tête soit un multiple de 32

En-tête UDP

L'en-tête UDP est codée sur 4 octets soit 32 bits :

1er octet	2ème octet	3ème octet	4 ème octet
Port source		Port destination	

1er octet	2ème octet	3ème octet	4 ème octet
longueur		Checksum	
Données			

L'en-tête UDP a une longueur de 8 octets.

Fragmentation et Ré-encapsulation

La taille limite d'un paquet TCP, l'en-tête comprise, ne peut pas dépasser **65 535 octets**. Cependant chaque réseau est qualifié par son MTU (Maximum Tranfer Unit). Cette valeur est la taille maximum d'un paquet autorisée. L'unité est en **octets**. Pour un réseau Ethernet sa valeur est de 1 500. Quand un paquet doit être expédié sur un réseau ayant un MTU inférieur à sa propre taille, le paquet doit être **fractionné**. A la sortie du réseau, le paquet est reconstitué. Cette reconstitution s'appelle **ré-encapsulation**.

Adressage

L'adressage IP requiert que chaque périphérique sur le réseau possède une adresse IP unique de 4 octets, soit 32 bits au format XXX.XXX.XXX.XXX De cette façon le nombre total d'adresses est de $2^{32} = 4.3$ Milliards.

Les adresses IP sont divisées en 5 classes, de A à E. Les 4 octets des classes A à C sont divisés en deux, une partie qui s'appelle le **Net ID** qui identifie le réseau et une partie qui s'appelle le **Host ID** qui identifie le hôte :

1er octet	2ème octet	3ème octet	4 ème octet
A Net ID		Host ID	
B	Net ID		Host ID
C		Net ID	Host ID
D	Multicast		
E	Réservé		

L'attribution d'une classe dépend du nombre de hôtes à connecter. Chaque classe est identifiée par un **Class ID** composé de 1 à 3 bits :

Classe	Bits ID Classe	Valeur ID Classe	Bits ID Réseau	Nb. de Réseaux	Bits ID hôtes	Nb. d'adresses	Octet de Départ
A	1	0	7	$2^7=128$	24	$2^{24}=16\ 777\ 216$	1 - 126
B	2	10	14	$2^{14}=16\ 834$	16	$2^{16}=65\ 535$	128 - 191
C	3	110	21	$2^{21}=2\ 097\ 152$	8	$2^8=256$	192 - 223

Dans chaque classe, certaines adresses sont réservées pour un usage privé :

Classe	IP de Départ	IP de Fin
A	10.0.0.0	10.255.255.255
B	172.16.0.0	172.31.255.255
C	192.168.0.0	192.168.255.255

Il existe des adresses particulières ne pouvant pas être utilisées pour identifier un hôte :

Adresse Particulière	Description
169.254.0.0 à 169.254.255.255	Automatic Private IP Addressing de Microsoft
Hôte du réseau courant	Tous les bits du Net ID sont à 0
Adresse de réseau	Tous les bits du Host ID sont à 0
Adresse de diffusion	Tous les bits du Host ID sont à 1

L'adresse de réseau identifie le **segment** du réseau entier tandis que l'adresse de diffusion identifie tous les hôtes sur le segment de réseau.

Afin de mieux comprendre l'adresse de réseau et l'adresse de diffusion, prenons le cas de l'adresse 192.168.10.1 en classe C :

	1er octet	2ème octet	3ème octet	4 ème octet
	Net ID			Host ID
Adresse IP	192	168	10	1
Binaire	11000000	10101000	000001010	00000001
Calcul de l'adresse de réseau				
Binaire	11000000	10101000	000001010	00000000
Adresse réseau	192	168	10	0
Calcul de l'adresse de diffusion				

	1er octet	2ème octet	3ème octet	4 ème octet
	Net ID			Host ID
Binaire	11000000	10101000	000001010	11111111
Adresse de diffusion	192	168	10	255

Masques de sous-réseaux

Tout comme l'adresse IP, le masque de sous-réseau compte 4 octets ou 32 bits. Les masques de sous-réseaux permettent d'identifier le Net ID et le Host ID :

Classe	Masque	Notation CIDR
A	255.0.0.0	/8
B	255.255.0.0	/16
C	255.255.255.0	/24

Le terme **CIDR** veut dire **Classless InterDomain Routing**. Le terme Notation CIDR correspond au nombre de bits d'une valeur de 1 dans le masque de sous-réseau.

Quand un hôte souhaite émettre il procède d'abord à l'identification de sa propre adresse réseau par un calcul AND (ET) appliqué à sa propre adresse et son masque de sous-réseau qui stipule :

- $1 \times 1 = 1$
- $0 \times 1 = 0$
- $1 \times 0 = 0$
- $0 \times 0 = 0$

Prenons le cas de l'adresse IP 192.168.10.1 ayant un masque de 255.255.255.0 :

	1er octet	2ème octet	3ème octet	4 ème octet
Adresse IP	192	168	10	1
Binaire	11000000	10101000	000001010	00000001
Masque de sous-réseau				

	1er octet	2ème octet	3ème octet	4 ème octet
Binaire	11111111	11111111	11111111	00000000
Calcul AND	11000000	10101000	00001010	00000000
Adresse réseau	192	168	10	0

Cet hôte essaie de communiquer avec un hôte ayant une adresse IP de 192.168.10.10. Il procède donc au même calcul en appliquant **son propre masque de sous-réseau** à l'adresse IP de l'hôte destinataire :

	1er octet	2ème octet	3ème octet	4 ème octet
Adresse IP	192	168	10	10
Binaire	11000000	10101000	00001010	00001010
Masque de sous-réseau				
Binaire	11111111	11111111	11111111	00000000
Calcul AND	11000000	10101000	00001010	00000000
Adresse réseau	192	168	10	0

Puisque l'adresse réseau est identique dans les deux cas, l'hôte émetteur présume que l'hôte de destination se trouve sur son réseau et envoie les paquets directement sur le réseau sans s'adresser à sa passerelle par défaut.

L'hôte émetteur essaie maintenant de communiquer avec un hôte ayant une adresse IP de 192.168.2.1. Il procède donc au même calcul en appliquant **son propre masque de sous-réseau** à l'adresse IP de l'hôte destinataire :

	1er octet	2ème octet	3ème octet	4 ème octet
Adresse IP	192	168	2	1
Binaire	11000000	10101000	00000010	00000001
Masque de sous-réseau				
Binaire	11111111	11111111	11111111	00000000
Calcul AND	11000000	10101000	00000010	00000000
Adresse réseau	192	168	2	0

Dans ce cas, l'hôte émetteur constate que le réseau de destination 192.168.2.0 n'est pas identique à son propre réseau 192.168.10.0. Il adresse donc les paquets à la passerelle par défaut.

VLSM

Puisque le stock de réseaux disponibles sous IPv4 est presque épuisé, une solution a du être trouvée pour créer des sous-réseaux en attendant l'introduction de l'IPv6. Cette solution s'appelle le VLSM ou Variable Length Subnet Masks. Le VLSM exprime les masques de sous-réseaux au format CIDR.

Son principe est simple. Afin de créer des réseaux différents à partir d'une adresse réseau d'une classe donnée, il convient de réduire le nombre d'hôtes. De cette façon les bits 'libérés' du Host ID peuvent être utilisés pour identifier les sous-réseaux.

Pour illustrer ceci, prenons l'exemple d'un réseau 192.168.1.0. Sur ce réseau, nous pouvons mettre 2^8-2 soit 254 hôtes entre 192.168.1.1 au 192.168.1.254.

Supposons que nous souhaiterions diviser notre réseau en 2 sous-réseaux. Pour coder 2 sous-réseaux, il faut que l'on libère 2 bits du Host ID. Les deux bits libérés auront les valeurs binaires suivantes :

- 00
- 01
- 10
- 11

Les valeurs binaires du quatrième octet de nos adresses de sous-réseaux seront donc :

- 192.168.1.00XXXXXX
- 192.168.1.01XXXXXX
- 192.168.1.10XXXXXX
- 192.168.1.11XXXXXX

où les XXXXXX représentent les bits que nous réservons pour décrire les hôtes dans chacun des sous-réseaux.

Nous ne pouvons pas utiliser les deux sous-réseaux suivants :

- 192.168.1.00XXXXXX
- 192.168.1.11XXXXXX

car ceux-ci correspondent aux débuts de l'adresse réseau 192.168.1.0 et de l'adresse de diffusion 192.168.1.255.

Nous pouvons utiliser les deux sous-réseaux suivants :

- 192.168.1.01XXXXXX
- 192.168.1.10XXXXXX

Pour le premier sous-réseau l'adresse réseau et l'adresse de diffusion sont :

Sous-réseau #1	192	168	1	01XXXXXX
Calcul de l'adresse de réseau				
Binaire	11000000	10101000	00000001	01 000000
Adresse réseau	192	168	1	64
Calcul de l'adresse de diffusion				
Binaire	11000000	10101000	00000001	01 111111
Adresse de diffusion	192	168	1	127

- L'adresse CIDR du réseau est donc 192.168.1.64/26 car le Net ID est codé sur 24+2 bits.
- Le masque de sous-réseau est donc le 11111111.11111111.11111111.11000000 ou le 255.255.255.192
- Nous pouvons avoir 2^6 -2 soit 62 hôtes.
- La plage valide d'adresses IP est de 192.168.1.65 à 192.168.1.126

Pour le deuxième sous-réseau l'adresse réseau et l'adresse de diffusion sont :

Sous-réseau #2	192	168	1	10XXXXXX
Calcul de l'adresse de réseau				
Binaire	11000000	10101000	00000001	10 000000
Adresse réseau	192	168	1	128
Calcul de l'adresse de diffusion				
Binaire	11000000	10101000	00000001	10 111111
Adresse de diffusion	192	168	1	191

- L'adresse CIDR du réseau est donc 192.168.1.128/26 car le Net ID est codé sur 24+2 bits.
- Le masque de sous-réseau est donc le 11111111.11111111.11111111.11000000 ou le 255.255.255.192
- Nous pouvons avoir 2^6 -2 soit 62 hôtes.
- La plage valide d'adresses IP est de 192.168.1.129 à 192.168.1.190

La valeur qui sépare les sous-réseaux est 64. Cette valeur comporte le nom **incrément**.

Ports et sockets

Afin que les données arrivent aux applications que les attendent, TCP utilise des numéros de ports sur la couche transport. Le numéros de ports sont divisés en trois groupes :

- **Well Known Ports**
 - De 1 à 1023
- **Registered Ports**
 - De 1024 à 49151
- **Dynamic et/ou Private Ports**
 - De 49152 à 65535

Le couple **numéro IP:numéro de port** s'appelle un **socket**.

/etc/services

Les ports les plus utilisés sont détaillés dans le fichier **/etc/services** :

```
trainee@debian8:~$ su -
Password:
root@debian8:~# more /etc/services
# Network services, Internet style
```

```
#  
# Note that it is presently the policy of IANA to assign a single well-known  
# port number for both TCP and UDP; hence, officially ports have two entries  
# even if the protocol doesn't support UDP operations.  
#  
# Updated from http://www.iana.org/assignments/port-numbers and other  
# sources like http://www.freebsd.org/cgi/cvsweb.cgi/src/etc/services .  
# New ports will be added on request if they have been officially assigned  
# by IANA and used in the real-world or are needed by a debian package.  
# If you need a huge list of used numbers please install the nmap package.  
  
tcpmux      1/tcp          # TCP port service multiplexer  
echo        7/tcp  
echo        7/udp  
discard     9/tcp          sink null  
discard     9/udp          sink null  
systat      11/tcp         users  
daytime     13/tcp  
daytime     13/udp  
netstat     15/tcp  
qotd       17/tcp          quote  
msp        18/tcp          # message send protocol  
msp        18/udp  
chargen    19/tcp          ttyst source  
chargen    19/udp          ttyst source  
ftp-data   20/tcp  
ftp        21/tcp  
fsp        21/udp          fspd  
ssh        22/tcp          # SSH Remote Login Protocol  
ssh        22/udp  
telnet     23/tcp  
smtp      25/tcp          mail  
time       37/tcp          timserver  
time       37/udp          timserver
```

```
rlp      39/udp      resource    # resource location
nameserver 42/tcp      name        # IEN 116
--More-- (6%)
```

Notez que les ports sont listés par deux :

- le port TCP
- le port UDP

La liste la plus complète peut être consultée à l'adresse suivante

<https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml>.

Pour connaitre la liste des sockets ouverts sur l'ordinateur, saisissez la commande suivante :

```
root@debian8:~# netstat -an | more
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp        0      0 127.0.0.1:25            0.0.0.0:*
tcp        0      0 127.0.0.1:42370          0.0.0.0:*
tcp        0      0 127.0.0.1:15023          0.0.0.0:*
tcp        0      0 0.0.0.0:111             0.0.0.0:*
tcp        0      0 0.0.0.0:41012            0.0.0.0:*
tcp        0      0 0.0.0.0:22              0.0.0.0:*
tcp        0      0 0.0.0.0:23              0.0.0.0:*
tcp        0      0 127.0.0.1:7127            0.0.0.0:*
tcp        0      0 127.0.0.1:33220           127.0.0.1:50656      ESTABLISHED
tcp        0      0 10.0.2.15:22             10.0.2.2:46432        ESTABLISHED
tcp        0      0 127.0.0.1:50656           127.0.0.1:33220      ESTABLISHED
tcp6       0      0 ::1:25                  ::*:*
tcp6       0      0 :::33476                ::*:*
tcp6       0      0 ::1:15023                ::*:*
tcp6       0      0 :::111                 ::*:*
tcp6       0      0 :::22                  ::*:*
tcp6       0      0 ::1:22                  ::1:39236        ESTABLISHED
```

```

tcp6      0      0 ::1:39236          ::1:22          ESTABLISHED
udp       0      0 0.0.0.0:32899      0.0.0.0:*
udp       0      0 0.0.0.0:995        0.0.0.0:*
udp       0      0 0.0.0.0:52452      0.0.0.0:*
udp       0      0 0.0.0.0:5353       0.0.0.0:*
udp       0      0 0.0.0.0:68         0.0.0.0:*
udp       0      0 127.0.0.1:613       0.0.0.0:*
udp       0      0 0.0.0.0:111        0.0.0.0:*
udp       0      0 0.0.0.0:53110      0.0.0.0:*
udp6      0      0 ::::17599         ::::*
udp6      0      0 ::::995          ::::*
udp6      0      0 ::::5353         ::::*
udp6      0      0 ::::33524        ::::*
udp6      0      0 ::::40492        ::::*
udp6      0      0 ::::111          ::::*

```

Active UNIX domain sockets (servers and established)

Proto	RefCnt	Flags	Type	State	I-Node	Path
unix	2	[ACC]	STREAM	LISTENING	17625	/tmp//.java_pid1791

--More--

Pour connaitre la liste des applications ayant ouvert un port sur l'ordinateur, saisissez la commande suivante :

```

root@debian8:~# netstat -anp | more
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address        State      PID/Program name
tcp     0      0 127.0.0.1:25            0.0.0.0:*
                                         LISTEN      868/exim4
tcp     0      0 127.0.0.1:42370          0.0.0.0:*
                                         LISTEN      1791/Remote Access
tcp     0      0 127.0.0.1:15023          0.0.0.0:*
                                         LISTEN      2449/ssh
tcp     0      0 0.0.0.0:111           0.0.0.0:*
                                         LISTEN      396/rpcbind
tcp     0      0 0.0.0.0:41012          0.0.0.0:*
                                         LISTEN      434/rpc.statd
tcp     0      0 0.0.0.0:22            0.0.0.0:*
                                         LISTEN      471/sshd
tcp     0      0 0.0.0.0:23            0.0.0.0:*
                                         LISTEN      4041/inetd
tcp     0      0 127.0.0.1:7127          0.0.0.0:*
                                         LISTEN      1791/Remote Access
tcp     0      0 127.0.0.1:33220          127.0.0.1:50656
                                         ESTABLISHED 1879/Remote Access

```

tcp	0	0	10.0.2.15:22	10.0.2.2:46432	ESTABLISHED	10584/sshd: trainee
tcp	0	0	127.0.0.1:50656	127.0.0.1:33220	ESTABLISHED	1791/Remote Access
tcp6	0	0	::1:25	:::*	LISTEN	868/exim4
tcp6	0	0	:::33476	:::*	LISTEN	434/rpc.statd
tcp6	0	0	::1:15023	:::*	LISTEN	2449/ssh
tcp6	0	0	:::111	:::*	LISTEN	396/rpcbind
tcp6	0	0	:::22	:::*	LISTEN	471/sshd
tcp6	0	0	::1:22	::1:39236	ESTABLISHED	2415/sshd: trainee
tcp6	0	0	::1:39236	::1:22	ESTABLISHED	2449/ssh
udp	0	0	0.0.0.0:32899	0.0.0.0:*		419/dhclient
udp	0	0	0.0.0.0:995	0.0.0.0:*		396/rpcbind
udp	0	0	0.0.0.0:52452	0.0.0.0:*		482/avahi-daemon: r
udp	0	0	0.0.0.0:5353	0.0.0.0:*		482/avahi-daemon: r
udp	0	0	0.0.0.0:68	0.0.0.0:*		419/dhclient
udp	0	0	127.0.0.1:613	0.0.0.0:*		434/rpc.statd
udp	0	0	0.0.0.0:111	0.0.0.0:*		396/rpcbind
udp	0	0	0.0.0.0:53110	0.0.0.0:*		434/rpc.statd
udp6	0	0	:::17599	:::*		419/dhclient
udp6	0	0	:::995	:::*		396/rpcbind
udp6	0	0	:::5353	:::*		482/avahi-daemon: r
udp6	0	0	:::33524	:::*		482/avahi-daemon: r
udp6	0	0	:::40492	:::*		434/rpc.statd
udp6	0	0	:::111	:::*		396/rpcbind

Active UNIX domain sockets (servers and established)

Proto	RefCnt	Flags	Type	State	I-Node	PID/Program name	Path
unix	2	[ACC]	STREAM	LISTENING	17625	1791/Remote Access	/tmp//.java_pid1791
--More--							

Résolution d'adresses Ethernet

Chaque protocole peut être encapsulé dans une **trame** Ethernet. Lorsque la trame doit être transportée de l'expéditeur au destinataire, ce premier doit connaître l'adresse Ethernet du dernier. L'adresse Ethernet est aussi appelée l'**adresse Physique** ou l'**adresse MAC**.

Pour connaître l'adresse Ethernet du destinataire, l'expéditeur fait appel au protocole **ARP**. Les informations reçues sont stockées dans une table. Pour visualiser ces informations, il convient d'utiliser la commande suivante :

```
root@debian8:~# arp -a
? (10.0.2.2) at 52:54:00:12:35:02 [ether] on eth0
? (10.0.2.3) at 52:54:00:12:35:03 [ether] on eth0
```

Les options de cette commande sont :

```
root@debian8:~# arp --help
Usage:
arp [-vn]  [<HW>] [-i <if>] [-a] [<hostname>]           <-Display ARP cache
arp [-v]      [-i <if>] -d  <host> [pub]                 <-Delete ARP entry
arp [-vnD]  [<HW>] [-i <if>] -f  [<filename>]            <-Add entry from file
arp [-v]  [<HW>] [-i <if>] -s  <host> <hwaddr> [temp]       <-Add entry
arp [-v]  [<HW>] [-i <if>] -Ds <host> <if> [netmask <nmm>] pub    <-'-'

-a                  display (all) hosts in alternative (BSD) style
-s, --set            set a new ARP entry
-d, --delete         delete a specified entry
-v, --verbose        be verbose
-n, --numeric        don't resolve names
-i, --device         specify network interface (e.g. eth0)
-D, --use-device     read <hwaddr> from given device
-A, -p, --protocol   specify protocol family
-f, --file           read new entries from file or from /etc/ethers

<HW>=Use '--H <hw>' to specify hardware address type. Default: ether
List of possible hardware types (which support ARP):
ash (Ash) ether (Ethernet) ax25 (AMPR AX.25)
netrom (AMPR NET/ROM) rose (AMPR ROSE) arcnet (ARCnet)
dlci (Frame Relay DLCI) fddi (Fiber Distributed Data Interface) hippi (HIPPI)
irda (IrLAP) x25 (generic X.25) eui64 (Generic EUI-64)
```

Copyright © 2022 Hugh Norris

From:

<https://www.ittraining.team/> - **www.ittraining.team**



Permanent link:

<https://www.ittraining.team/doku.php?id=elearning:workbooks:reso:4:res402>

Last update: **2022/11/04 05:35**