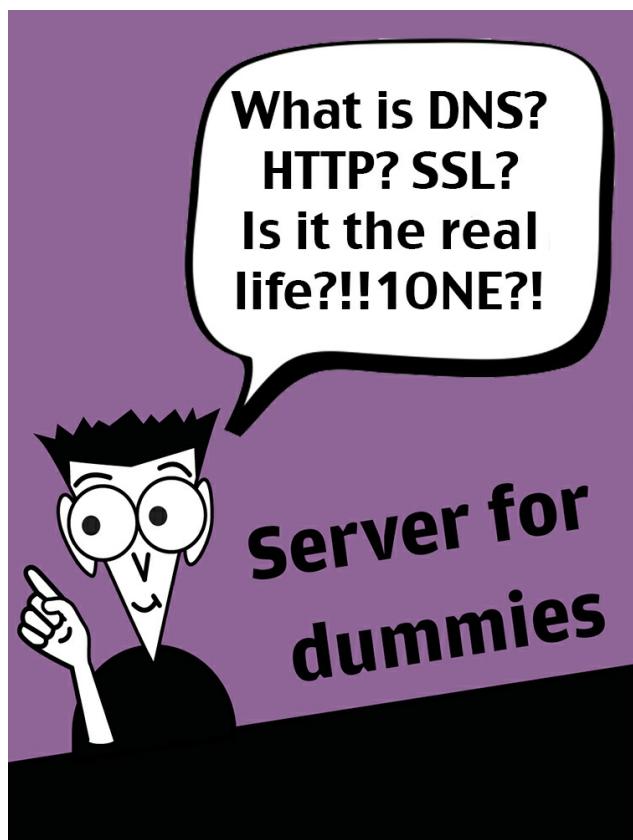


Server for dummies



unicorn approved

[Online](#) | [EPUB](#) | [MOBI](#) | [PDF](#) | [Github](#)

`server for dummies` is an educational project to understand how typical web services work. It's focused in the application layout services.

This documentation is complemented with a script to install and test all services explain in this book. You can find it in the following link:

[Server for dummies on Github](#)

NOTE: This is only a educational example. Don't use in production.

1. Get Started

Service	Transport	Port	Package
DNS	TCP/UDP	53/TCP 53/UDP	bind9
SMTP	TCP	25/TCP 587/TCP (alternative) 465/TCP (SMTP)	exim4
POP3	TCP	110/TCP 995/TCP (encrypted)	dovecot
IMAP	TCP	143/TCP 220/TCP (IMAP3) 993/TCP (IMAPS)	-
LDAP	TCP/UDP	389/TCP 389/UDP	slapd
HTTP	TCP	80/TCP	apache
SSH	TCP	21/TCP	openssh
FTP	TCP	20/TCP (DTP) 21/TCP (IP)	-
IPSEC	-	-	-
TELNET	TCP	23/TCP	-
DHCP	UDP	67/UDP (client) 68/UDP (server)	-

How to use

1) Clone the repo:

```
git clone https://github.com/Kikobeats/server-for-dummies.git
```

2) go to folder `server-for-dummies` and edit `settings.sh` with your options:

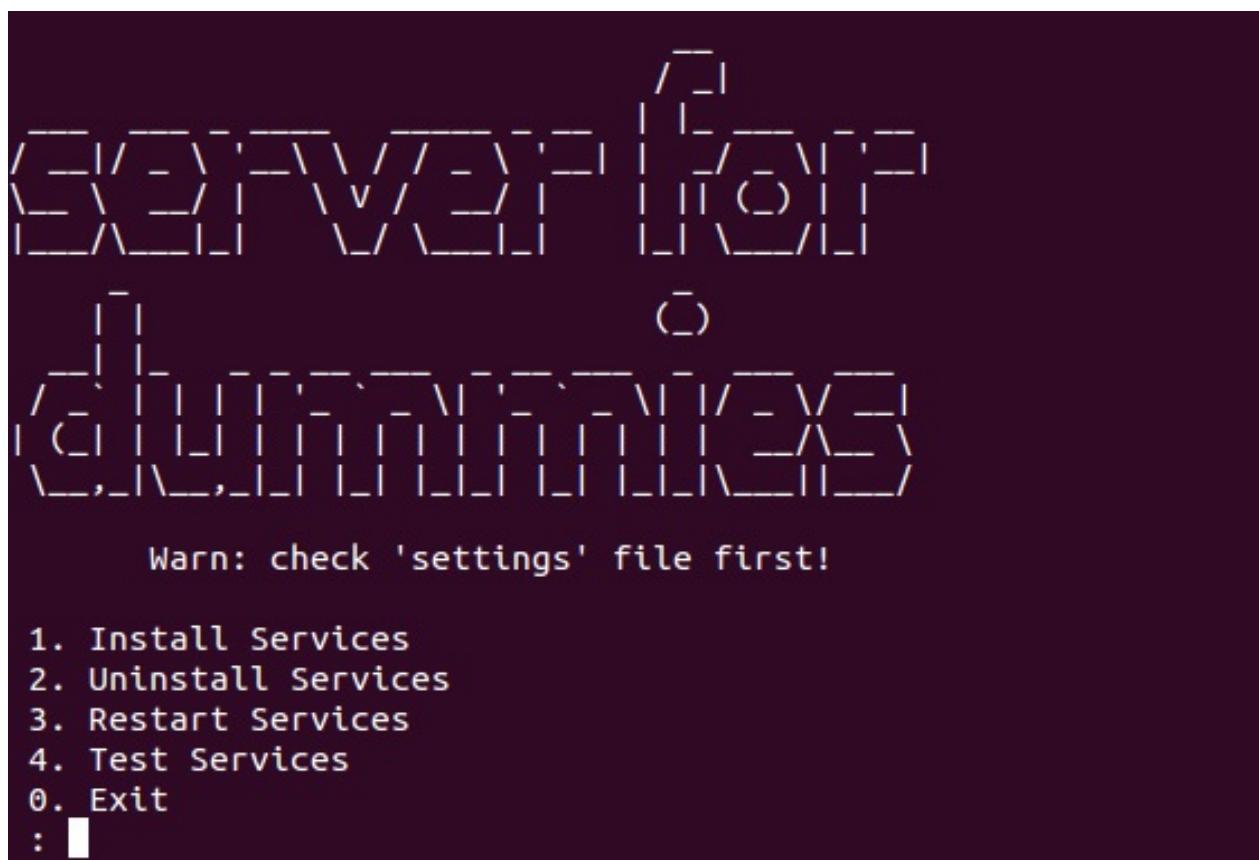
```
## DNS Settings
DNS_NAME="st.um"
PRIMARY_DNS="172.16.22.135"
SECONDARY_DNS="172.16.22.137"
FORWARDERS="155.54.1.10;"

## SMTP Settings
LOCAL_NETWORK="172.16.22.0/24"
RELAY_DOMAINS="$DNS_NAME; um.es"

## OPENSSL Settings
USERNAME="josefranciscoverdugambin"
```

3) Run it with admin account!

sudo sh init.sh



Test services

All services have been tested in VM VMWare under Ubuntu Server 12.04:

- VM 1 – Main client
 - VM 2 – Primary DNS
 - VM 3 – Secondary DNS

For test services connectivity run 'Test services'

Examples

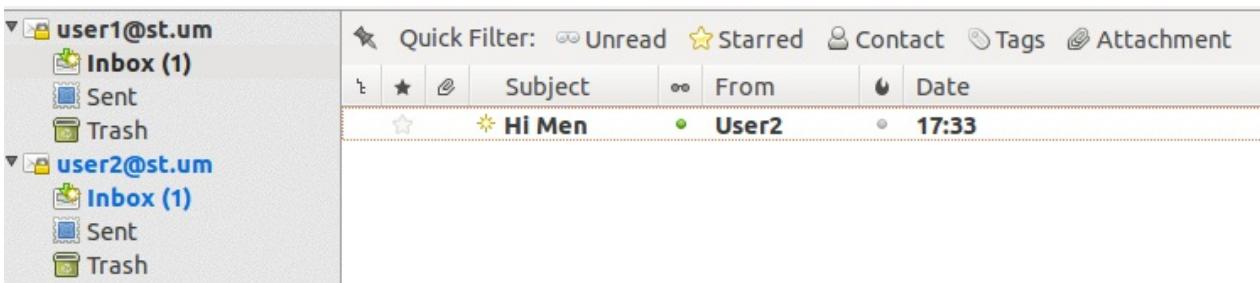
Include:

- 3 example of domains
 - www1.st.um – Basic domain
 - www2.st.um – Domain with HTTP authentication ([authorized users here](#))
 - www3.st.um – Domain with HTTPS (check you are visiting https in the browser!)
- 2 mails accounts examples
 - run `./Services/Mail/account.sh` to create it at first time
- LDAP
 - Sample data for check read and modify operations ([check here](#))

Example SSL



Example Mails



Example SSH

```
user1@st.um's password:  
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.8.0-39-generic i686)
```

```
* Documentation: https://help.ubuntu.com/
```

```
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*copyright.
```

```
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.
```

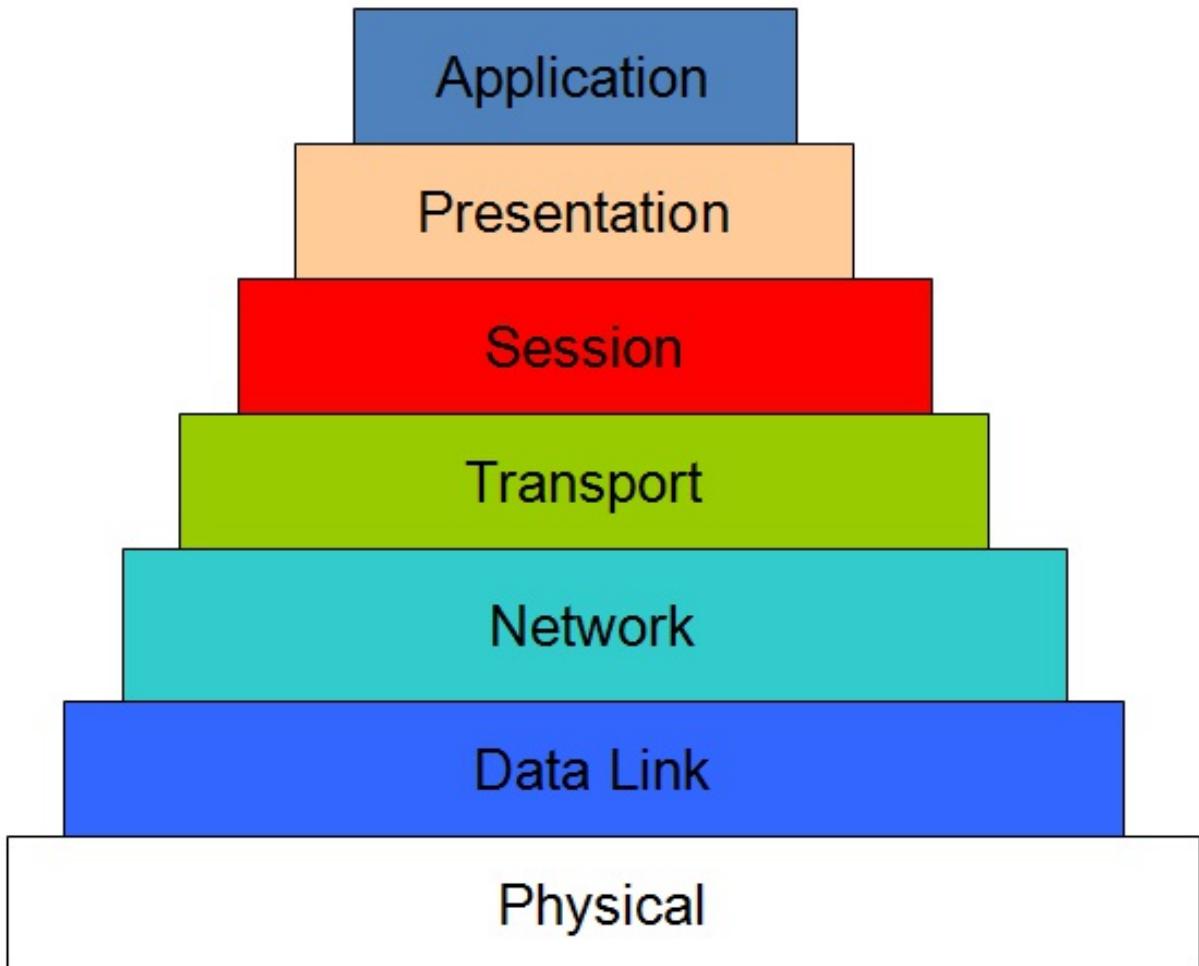
```
$ whoami  
user1  
$ echo "I am a dummie"  
I am a dummie  
$ █
```

Understand

Internet is the best invent in the last 50 years, and maybe someone can say that is the best invention in the world.

First, relax. Internet is not perfect and needs much human value to make it a secure service for daily things, for example, reading your mail, visiting a cats' website, downloading creative commons music and films...

Internet is nested in layers called [OSI model](#). Seven layers compound the OSI model and each layer offers services for different things.

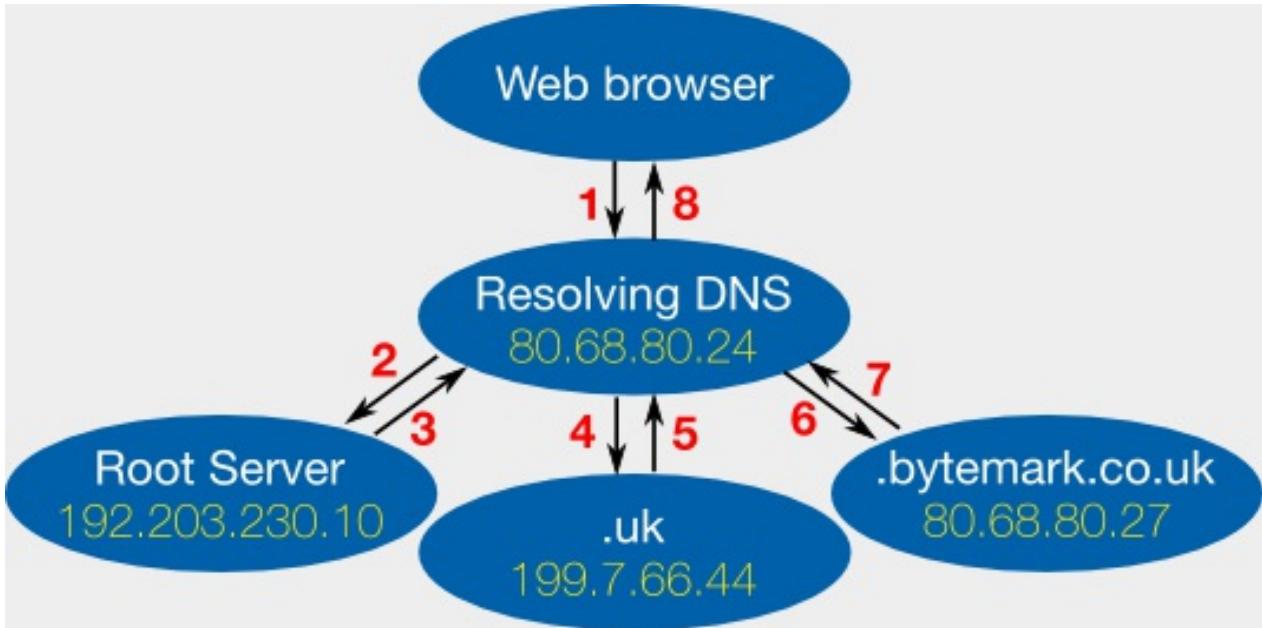


Top-level layers need their lower layer to offer a service. Your browser doesn't load a website if its transfer protocol doesn't work correctly.

Other important networks elements are routers, switches and host and using different protocols (BGP, RIP, OSPF) to communicate with each one and make internet a network of computers that understands how a package travels from EEUU to Spain.

But this is not important here. You'll need to understand how to set up your machine to be accessible for the others and to offer your services for resolving names, having mails accounts under your domain...

The services that you intend to use belong to the application layer. As I said before, this services use services or protocols of lower layers of the OSI model. For example, DNS is an application service, but it needs a transport protocol to resolve name to IP. By the other hand, IP is based on network layer IPv4 (or better, IPv6) to be operative for the application layer.



At the end, all is a stack of abstraction. The purpose is having an application layer available, secure and operative.

Now, is time to take action!

Important files

DNS

- `/etc/resolv.conf`
- `/etc/bind/named.conf.options`
- `/etc/bind/named.conf.local`
- `/etc/bind/db."$DNS_NAME".zone`

SMTP

- `/etc/exim4/update-exim4.conf.conf` # Exim4 settings

POP

- `/etc/dovecot/conf.d/10-mail.conf` # Dovecot settings
- `/etc/dovecot/conf.d/10-auth.conf` # Dovecot settings

LDAP

- `ldapmodify -Y EXTERNAL -H ldapi:/// -f FILE` # Modify LDAP settings
- `ldapadd -Y EXTERNAL -H ldapi:/// -f FILE` # Load database data

HTTP

- `/etc/apache2/sites-available` # Apache virtual hosts
- `/var/www/` # Apache websites data
- `/etc/apache2/groups` # Apache authentication

SSL

- `usr/lib/ssl/openssl.cnf` # Configuration file of SSL

SSH

- `~/.ssh` # Content SSH keys

Most use commands

General

```
netstat -a | more # show ports and services that you are using.
```

DNS

- `dig www.domain.com` # do DNS query
- `host www.domain.com` # know the IP of a name
- `nslookup www.domain.com` # check if DNS is resolve correctly

SMTP

- `telnet xxx.xxx.xxx.xxx 25` # basic query to SMTP service

POP

- `telnet xxx.xxx.xxx.xxx 110` # basic query to POP service

LDAP

- `ldapsearch -x -H ldap://LDAP_IP -b "cn=' ',ou=' ',o=' ',c=' '" FIELD` # Search in the LDAP IP

HTTP

- `curl www.domain.com` # get HTTP source code of a domain

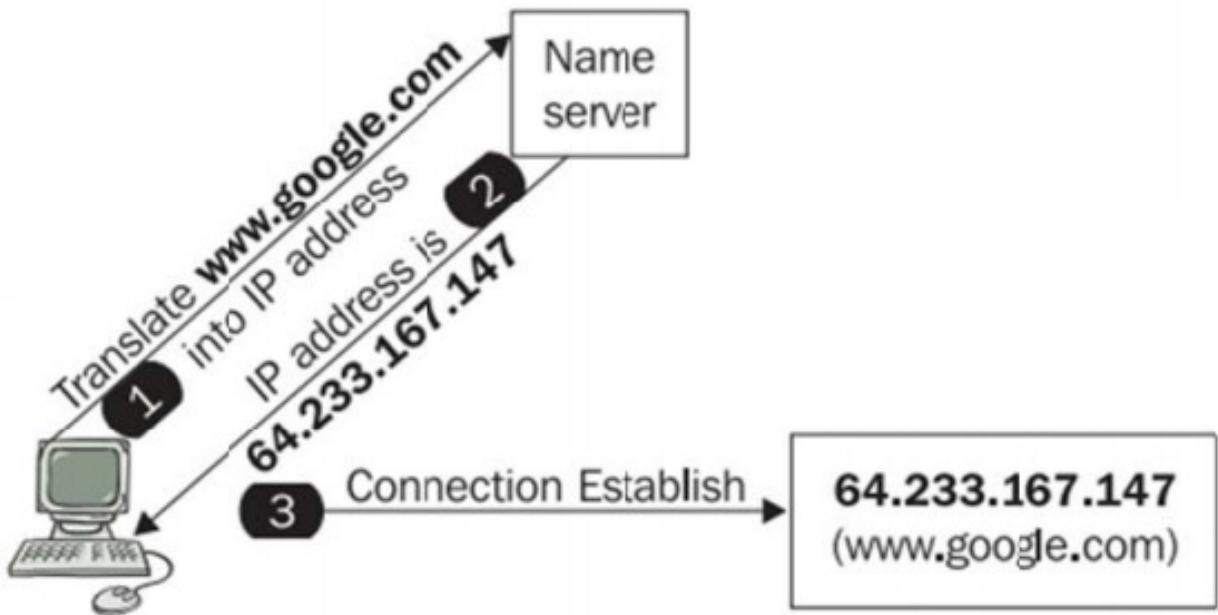
SSL

- `openssl version -d` # report your SSL directory
- `openssl req -x509 -newkey rsa:2048 -keyout cakey.pem -days 3650 -out cacert.pem` # generate CA autosign in the server
- `openssl x509 -in cacert.pem -text` # Check that your server certificate is standard by x509
- `openssl rsa -in cakey.pem -text` # Cehck taht your server certificate is RSA correct
- `openssl req -new -nodes -newkey rsa:1024 -keyout serverkey.pem -out servercsr.pem` # Generate certificate client
- `openssl ca -keyfile cakey.pem -in servercsr.pem -out servercert.pem` # Sign certificate client by the server
- `openssl s_server -cert servercert.pem -key serverkey.pem -www` # Check that your client certificate is valid

DNS

DNS is one of the most important services for internet because is necessary to convert a name of one web in its IP for simply question: Remember a IP is very most difficult than remember a name.

At first for to do it the first computers had a simple file called `host.txt` that works similar to key/value dictionary: One name have one IP translation. And one of this old computers is used to serve the HOSTS.txt updated of the rest of old computers. This is the concept.

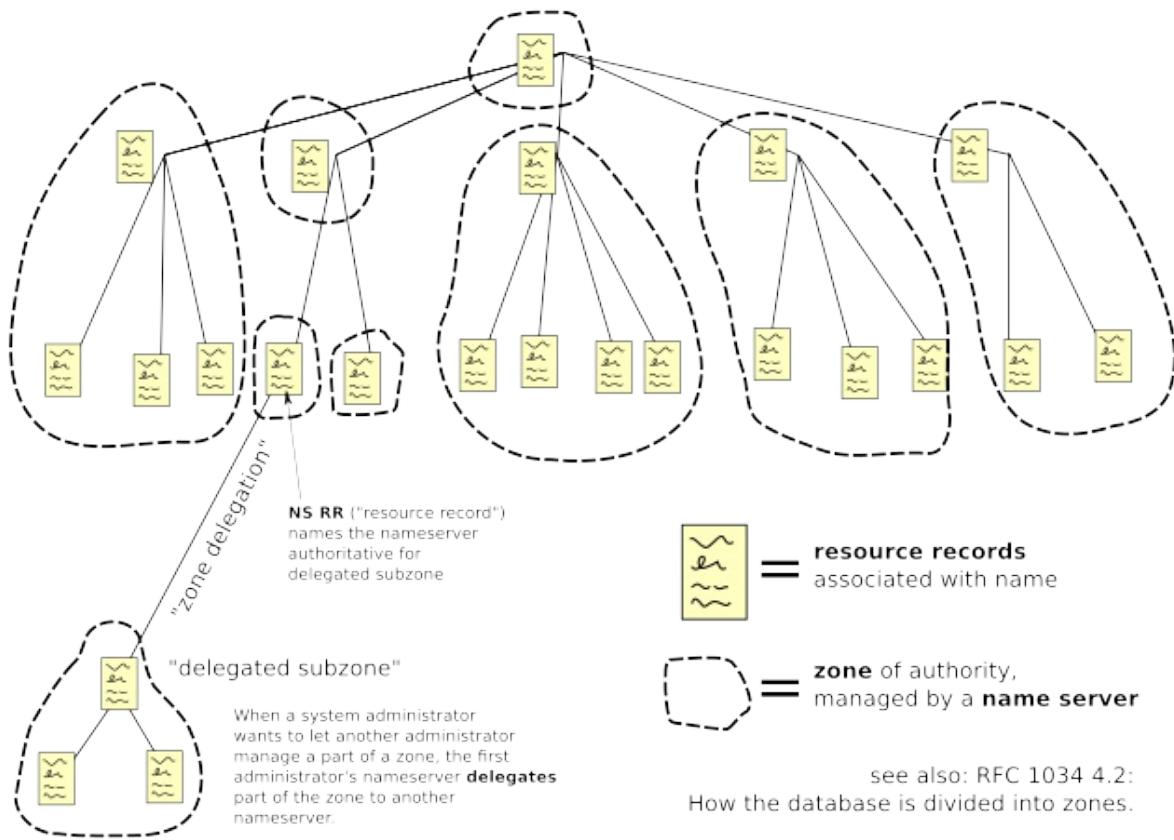


This is simple if you have less than 10 computers in the world. But now, this is a bit complicated with too much computers.

The DNS service is regulated by the [ICANN since 1988](#).

Basically to understand how ICANN regulates it you must know that she designates root nodes to keep all information about how to resolve whatever domain. When you go to website to register a website name, actually you are recording your domain in the ICANN nodes.

Domain Name Space



Each domain have different **records** for the dommain. A record is the basic data component in DNS. Resource records define not only names and IP addresses but domains, servers, zone, and services as well. This list shows you the most common types of resource records:

Type	Purpose
A	Address resource records match an IP address to a host name.
CNAME	<i>Canonical name</i> resource records associate a nickname to a host name.
MX	Mail exchange resource records identify mail servers for the specified domain.
NS	Name server resource records identify servers (other than the SOA server) that contain zone information files.
PTR	Pointer resource records match a host name to a given IP address. This is the opposite of an Address record, which matches an IP address to the supplied host name.
SOA	Start of authority resource records specify which server contains the zone file for a domain.
SRV	Service resource records identify servers that provide special services to the domain.

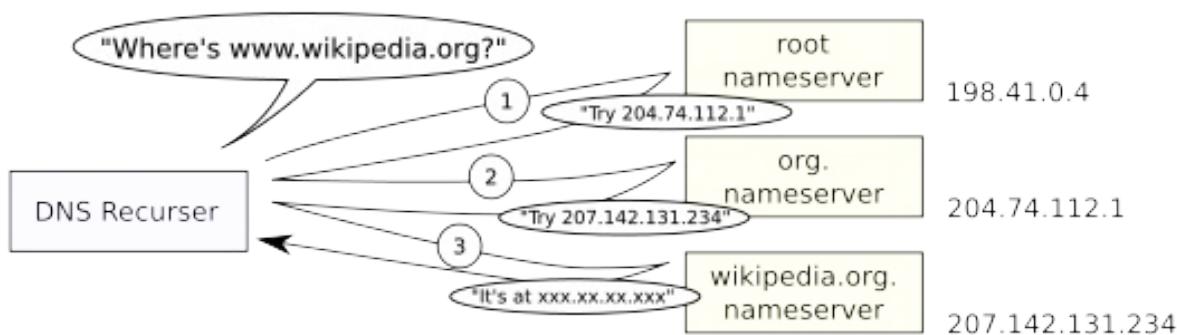
DNS Query

The better way to explain the process is with a example.

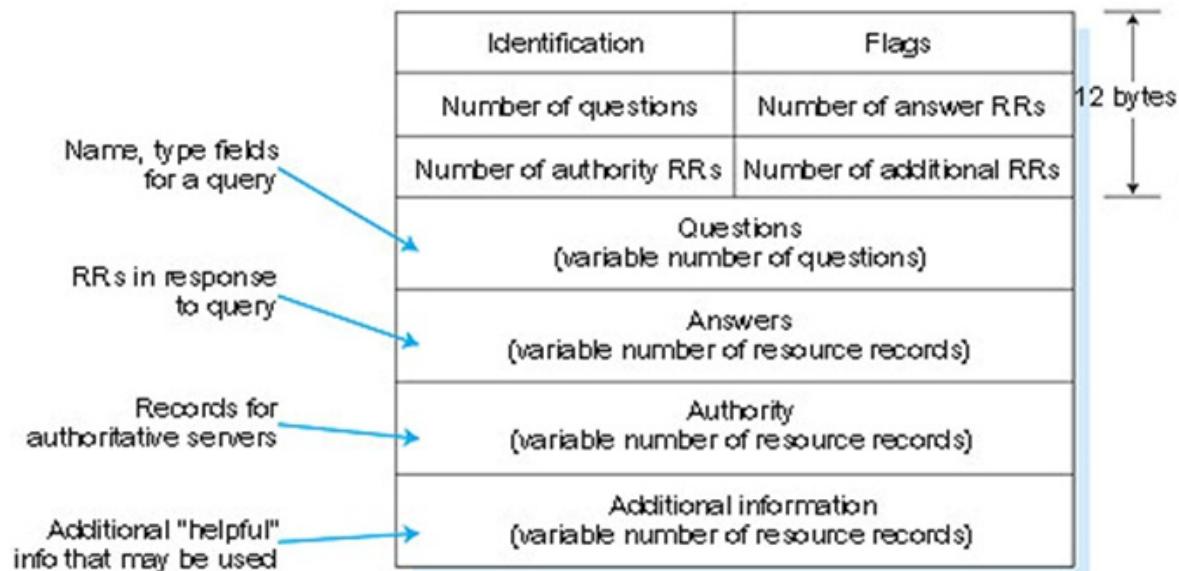
You need to know the resolve of `www.google.com` domain. You do a query of your server and this first search in the local database with `.com` then `google` and then the record `www`. This is known as **iterative query**: you first search a general node, and later descent for the node and search the next element.

If the domain can't be resolve in this DNS, the DNS can be delegate in other server. This is known as **recursive query**: Delegate in other dns server to resolve the query.

The rerve that respond the query is indicate in the **AA (Authoritative Answer)**. Its possible that for each zone have more than one DNS name server. This is known as **Primary** and **Secondary** DNS of the zone and data redundancy is a measure of safety and reliability.



In easy see this in the DNS response message:



Your DNS in the client

You need to say at your client that use your DNS to resolve names. Run the script and select this options:

1. Install Services > client > DNS

Now, you have said in your client that use the IP of your server to resolve names. This is in the file `/etc/resolv.conf` :

```
echo " * Configuring DNS client..."  
echo "# Dynamic resolv.conf(5) file for glibc resolver(3) genera  
ted by resolvconf(8)  
# DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRIT  
TEN  
nameserver $PRIMARY_DNS  
nameserver $SECONDARY_DNS  
#search localdomain  
" > /etc/resolv.conf
```

In my case have two servers for primary and secondary DNS.

Your DNS in the server

Server configuration is also simple, but it requires a few extra steps.

1. Install Services > client > DNS > [Select primary or secondary]

The script do this:

1. Install **bind9**, a application to provide DNS services
2. Edit `/etc/resolv.conf` file and add your own direction
3. Set up **bind9** to define your domain database and your secondary IP if is defined.

```
write_config_local_server(){
echo "zone \"\$DNS_NAME.\\" IN {
    type master;
    file \"/etc/bind/db.\$DNS_NAME.zone\";
    allow-transfer {\$SECONDARY_DNS;};
}; " > /etc/bind/named.conf.local
}
```

1. In the domain database write entries to be resolubles main services that use DNS to access it:

```
write_database(){
echo "\$TTL 604800
@ IN SOA ns1.\$DNS_NAME. root.ns1.\$DNS_NAME. (
    1      ; Serial
    604800 ; Refresh
    86400  ; Retry
    2419200 ; Expire
    604800 ) ; Negative Cache TTL

@           IN      NS      ns1.\$DNS_NAME .
ns1         IN      A       \$PRIMARY_DNS
@           IN      NS      ns2.\$DNS_NAME .

ns2         IN      A       \$SECONDARY_DNS
\$DNS_NAME . IN      A       \$PRIMARY_DNS

smtp        IN      CNAME   \$DNS_NAME .
pop3        IN      CNAME   \$DNS_NAME .
ldap        IN      CNAME   \$DNS_NAME .
www         IN      CNAME   \$DNS_NAME .
www1        IN      CNAME   \$DNS_NAME .
www2        IN      CNAME   \$DNS_NAME .
www3        IN      CNAME   \$DNS_NAME .
" > /etc/bind/db.\$DNS_NAME.zone
}
```

For the secondary server, the configuration is more easy because this use primary server to get information and don't need too more:

```
write_config_local_client(){
echo "zone \"\$DNS_NAME.\" IN {
    type slave;
    file \"/var/cache/bind/db.\$DNS_NAME.zone\";
    masters {\$PRIMARY_DNS;};
}";
" > /etc/bind/named.conf.local
}
```

Check connectivity

You can check connectivity in the client with two commands:

```
josefranciscoverdugambin@ubuntu:~/server-for-dummies$ dig www.st.um

; <>> DiG 9.8.1-P1 <>> www.st.um
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 9712
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 2

;; QUESTION SECTION:
;www.st.um.           IN      A

;; ANSWER SECTION:
www.st.um.          604800   IN      CNAME    st.um.
st.um.              604800   IN      A        172.16.22.135

;; AUTHORITY SECTION:
st.um.              604800   IN      NS       ns2.st.um.
st.um.              604800   IN      NS       ns1.st.um.

;; ADDITIONAL SECTION:
ns1.st.um.          604800   IN      A        172.16.22.135
ns2.st.um.          604800   IN      A        172.16.22.137

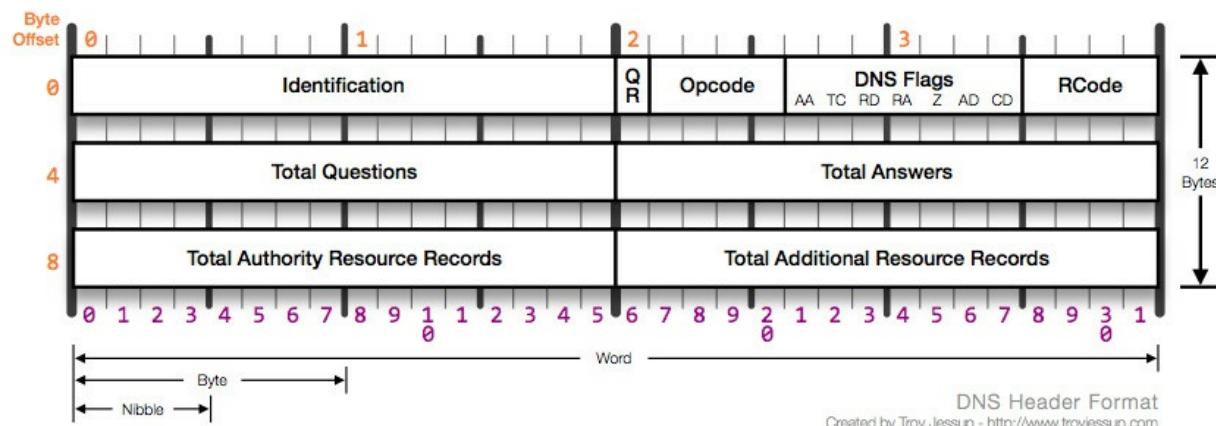
;; Query time: 0 msec
;; SERVER: 172.16.22.135#53(172.16.22.135)
;; WHEN: Sun May 18 19:44:35 2014
```

```
josefranciscoverdugambin@ubuntu:~/server-for-dummies$ nslookup www.st.um
Server: 172.16.22.135
Address: 172.16.22.135#53

www.st.um      canonical name = st.um.
Name:  st.um
Address: 172.16.22.135
```

How DNS works

DNS Header



Internal query

For the example, I run `dig st.um` command and see in the wireshark what happens:

The client do standard query connection at default DNS server (my server). In the header I say that I want to translate `st.um` adress (of type A)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.22.136	172.16.22.135	DNS	65	Standard query A st.um
2	0.000410	172.16.22.135	172.16.22.136	DNS	149	Standard query response A 172.16.22.135

► Frame 1: 65 bytes on wire (520 bits), 65 bytes captured (520 bits)
► Ethernet II, Src: VMware_30:4d:05 (00:50:56:30:4d:05), Dst: VMware_f7:ce:9d (00:0c:29:f7:ce:9d)
► Internet Protocol Version 4, Src: 172.16.22.136 (172.16.22.136), Dst: 172.16.22.135 (172.16.22.135)
► User Datagram Protocol, Src Port: 35522 (35522), Dst Port: domain (53)
▼ Domain Name System (query)
 ► Response In: 21
 Transaction ID: 0x1fef
 ▼ Flags: 0x0100 (Standard query)
 0.... = Response: Message is a query
 .000 0.... = Opcode: Standard query (0)
 0. = Truncated: Message is not truncated
 1 = Recursion desired: Do query recursively
 0.... = Z: reserved (0)
 0 = Non-authenticated data: Unacceptable
 Questions: 1
 Answer RRs: 0
 Authority RRs: 0
 Additional RRs: 0
 ▼ Queries
 ► st.um: type A, class IN
 Name: st.um
 Type: A (Host address)
 Class: IN (0x0001)

The server respond me with same useful information:

- In `Answer` section, server response my query and say 'Hey, the register A of the domain is `172.16.22.135`.
 - In `AA nameserver` section indicate what server responde the query. In this case, my primary (`ns1.st.um`) and secondary (`ns2.st.um`) server.
 - In `Additional Record` section indicate additional information like the IP of the nameservers.
- . In the `answer` entry of the header you can see the IP of my request name, and the adress of the DNS server that serve the information:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.22.136	172.16.22.135	DNS	65	Standard query A st.um
2	0.000410	172.16.22.135	172.16.22.136	DNS	149	Standard query response A 172.16.22.135

► Frame 2: 149 bytes on wire (1192 bits), 149 bytes captured (1192 bits)

- Ethernet II, Src: VMware_f7:ce:9d (00:0c:29:f7:ce:9d), Dst: VMware_30:4d:05 (00:50:56:30:4d:05)
- Internet Protocol Version 4, Src: 172.16.22.135 (172.16.22.135), Dst: 172.16.22.136 (172.16.22.136)
- User Datagram Protocol, Src Port: domain (53), Dst Port: 35522 (35522)
- ▼ Domain Name System (response)
 - [\[Request In: 1\]](#)
 - [Time: 0.000410000 seconds]
 - Transaction ID: 0x1fef
 - ▼ Flags: 0x8580 (Standard query response, No error)
 - 1... = Response: Message is a response
 - .000 0... = Opcode: Standard query (0)
 -1... = Authoritative: Server is an authority for domain
 -0. = Truncated: Message is not truncated
 -1 = Recursion desired: Do query recursively
 - 1.... = Recursion available: Server can do recursive queries
 -0... = Z: reserved (0)
 -0. = Answer authenticated: Answer/authority portion was not authenticated by the server
 -0 = Non-authenticated data: Unacceptable
 - 0000 = Reply code: No error (0)
 - Questions: 1
 - Answer RRs: 1
 - Authority RRs: 2
 - Additional RRs: 2
 - ▼ Queries
 - ▼ st.um: type A, class IN
 - Name: st.um
 - Type: A (Host address)
 - Class: IN (0x0001)
 - ▼ Answers
 - st.um: type A, class IN, addr 172.16.22.135
 - ▼ Authoritative nameservers
 - st.um: type NS, class IN, ns ns2.st.um
 - st.um: type NS, class IN, ns ns1.st.um
 - ▼ Additional records
 - ns1.st.um: type A, class IN, addr 172.16.22.135
 - ns2.st.um: type A, class IN, addr 172.16.22.137

External query

This process is similar to internal query, but now depend of external DNS.

In this case, I do a `ping google.com` and in this process I need first translate `google.com` name into a IP name:

No.	Time	Source	Destination	Protocol	Length	Info
3	0.000105	172.16.190.153	155.54.1.10	DNS	81	Standard query A google.com
5	10.000901	172.16.190.153	216.239.38.10	DNS	81	Standard query A google.com
6	10.081495	216.239.38.10	172.16.190.153	DNS	246	Standard query response A 173.194.34.225 A 173.194.34
11	10.144116	172.16.190.153	155.54.1.10	DNS	98	Standard query PTR 233.34.194.173.in-addr.arpa
12	20.144641	172.16.190.153	216.239.36.10	DNS	98	Standard query PTR 233.34.194.173.in-addr.arpa
13	20.237172	216.239.36.10	172.16.190.153	DNS	125	Standard query response PTR mad01s09-in-f9.le100.net


```
> Frame 5: 81 bytes on wire (648 bits), 81 bytes captured (648 bits)
> Ethernet II, Src: VMware_f7:ce:a7 (00:0c:29:f7:ce:a7), Dst: VMware_fa:67:5e (00:50:56:fa:67:5e)
> Internet Protocol Version 4, Src: 172.16.190.153 (172.16.190.153), Dst: 216.239.38.10 (216.239.38.10)
> User Datagram Protocol, Src Port: 36904 (36904), Dst Port: domain (53)
▼ Domain Name System (query)
  [Response In: 6]
  Transaction ID: 0xc93e
  Flags: 0x0000 (Standard query)
  Questions: 1
  Answer RRs: 0
  Authority RRs: 0
  Additional RRs: 1
  ▼ Queries
    ▼ google.com: type A, class IN
      Name: google.com
      Type: A (Host address)
      Class: IN (0x0001)
  ▼ Additional records
    ▼ <Root>: type OPT
      Name: <Root>
      Type: OPT (EDNS0 option)
      UDP payload size: 4096
      Higher bits in extended RCODE: 0x0
      EDNS0 version: 0
      ▼ Z: 0x8000
        Bit 0 (DO bit): 1 (Accepts DNSSEC security RRs)
        Bits 1-15: 0x0 (reserved)
      Data length: 0
```

The DNS server respond me with a list of available IPs for `google.com` domain:

No.	Time	Source	Destination	Protocol	Length	Info
3	0.000105	172.16.190.153	155.54.1.10	DNS	81	Standard query A google.com
5	10.000901	172.16.190.153	216.239.38.10	DNS	81	Standard query A google.com
6	10.081495	216.239.38.10	172.16.190.153	DNS	246	Standard query response A 173.194.34.225 A 173.194.34.226
11	10.144116	172.16.190.153	155.54.1.10	DNS	98	Standard query PTR 233.34.194.173.in-addr.arpa
12	20.144641	172.16.190.153	216.239.36.10	DNS	98	Standard query PTR 233.34.194.173.in-addr.arpa
13	20.237172	216.239.36.10	172.16.190.153	DNS	125	Standard query response PTR mad01s09-in-f9.le100.net

► Frame 6: 246 bytes on wire (1968 bits), 246 bytes captured (1968 bits)
 ► Ethernet II, Src: VMware_fa:67:5e (00:50:56:fa:67:5e), Dst: VMware_f7:ce:a7 (00:0c:29:f7:ce:a7)
 ► Internet Protocol Version 4, Src: 216.239.38.10 (216.239.38.10), Dst: 172.16.190.153 (172.16.190.153)
 ► User Datagram Protocol, Src Port: domain (53), Dst Port: 36904 (36904)

▼ Domain Name System (response)

- Request In: 51
- [Time: 0.080594000 seconds]
- Transaction ID: 0xc93e
- Flags: 0x8400 (Standard query response, No error)
- Questions: 1
- Answer RRs: 11
- Authority RRs: 0
- Additional RRs: 0

▼ Queries

- google.com: type A, class IN

▼ Answers

- google.com: type A, class IN, addr 173.194.34.225
- google.com: type A, class IN, addr 173.194.34.227
- google.com: type A, class IN, addr 173.194.34.231
- google.com: type A, class IN, addr 173.194.34.230
- google.com: type A, class IN, addr 173.194.34.224
- google.com: type A, class IN, addr 173.194.34.229
- google.com: type A, class IN, addr 173.194.34.238
- google.com: type A, class IN, addr 173.194.34.233
- google.com: type A, class IN, addr 173.194.34.226
- google.com: type A, class IN, addr 173.194.34.228
- google.com: type A, class IN, addr 173.194.34.232

Now, I can continue with my ping!

HTTP

HTTP (short for *HyperText Transfer Protocol*) is the underlying protocol used by the World Wide Web. HTTP defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands. For example, when you enter a URL in your browser, this actually sends an HTTP command to the Web server directing it to fetch and transmit the requested Web page.

All parameter of the protocol are specified in the **HTTP header**

Client Request:

```
GET /index.html HTTP/1.1
Host: www.example.com
```

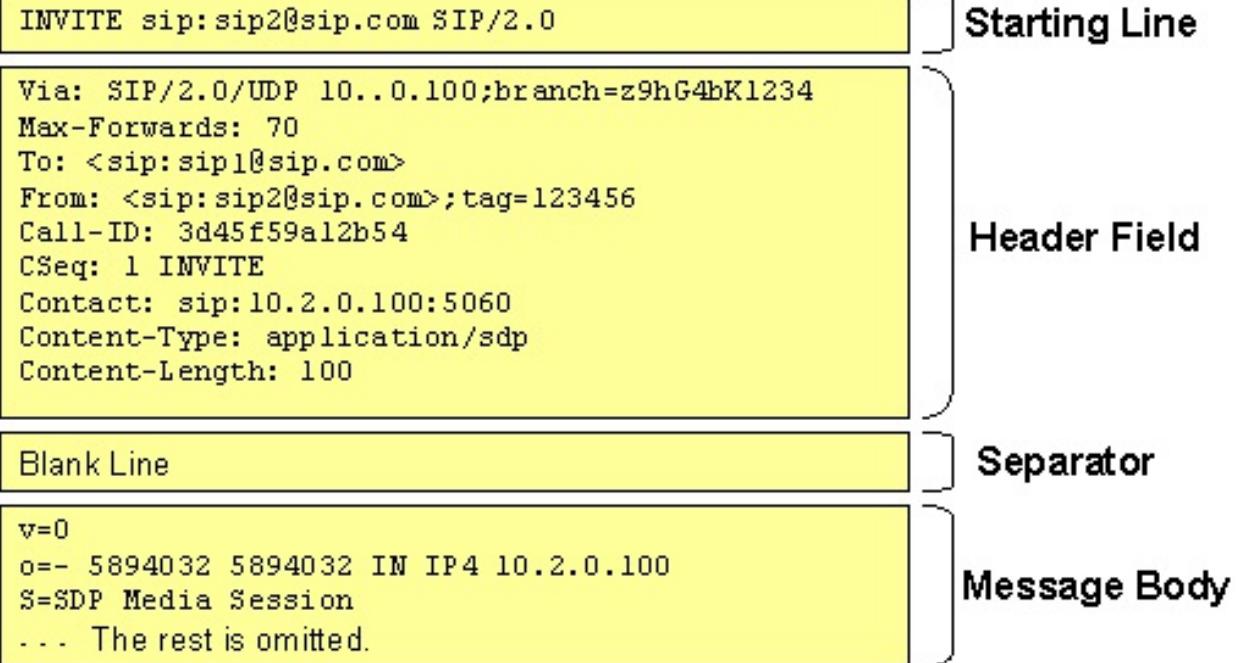
Server Response:

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
ETag: "3f80f-1b6-3e1cb03b"
Content-Type: text/html; charset=UTF-8
Content-Length: 131
Accept-Ranges: bytes
Connection: close

<html>
<head>
  <title>An Example Page</title>
</head>
<body>
  Hello World, this is a very simple HTML document.
</body>
</html>
```

A message in HTTP have 3 parts:

1. First line with *HTTP_Method, Resource ID adn HTTP Version.*
2. Information about the client and the petition.
3. Body request if the client need to transfer data to the server.



The HTTP methods are different for different actions:

Command	Meaning
GET	Return the requested item
HEAD	Request only the header information of an item
OPTIONS	Request communications options of an item
POST	Supply input to a server-side command and return the result
PUT	Store an item on the server
DELETE	Delete an item on the server
TRACE	Trace server communication

And each client request is response with status code:

- **1xx** Informational
- **2xx** Success
- **3xx** Redirection
- **4xx** Client Error
- **5xx** Server Error

Special mention to:

418 I'm a teapot (RFC 2324)

This code was defined in 1998 as one of the traditional IETF April Fools' jokes, in RFC 2324, Hyper Text Coffee Pot Control Protocol, and is not expected to be implemented by actual HTTP servers.

Informational Status Codes	Client Request Incomplete	Server Errors
100 – Continue [The server is ready to receive the rest of the request.]	400 – Bad Request [The server detected a syntax error in the client's request.]	500 – Internal Server Error [A server configuration setting or an external program has caused an error.]
101 – Switching Protocols [Client specifies that the server should use a certain protocol and the server will give this response when it is ready to switch.]	401 – Unauthorized [The request requires user authentication. The server sends the WWW-Authenticate header to indicate the authentication type and realm for the requested resource.]	501 – Not Implemented [The server does not support the functionality required to fulfill the request.]
Client Request Successful	402 – Payment Required [reserved for future]	502 – Bad Gateway [The server encountered an invalid response from an upstream server or proxy.]
200 – OK [Success! This is what you want.]	403 – Forbidden [Access to the requested resource is forbidden. The request should not be repeated by the client.]	503 – Service Unavailable [The service is temporarily unavailable. The server can send a Retry-After header to indicate when the service may become available again.]
201 – Created [Successfully created the URI specified by the client.]	404 – Not Found [The requested document does not exist on the server.]	504 – Gateway Time-Out [The gateway or proxy has timed out.]
202 – Accepted [Accepted for processing but the server has not finished processing it.]	405 – Method Not Allowed [The request method used by the client is unacceptable. The server sends the Allow header stating what methods are acceptable to access the requested resource.]	505 – HTTP Version Not Supported [The version of HTTP used by the client is not supported.]
203 – Non-Authoritative Information [Information in the response header did not originate from this server. Copied from another server.]	406 – Not Acceptable [The requested resource is not available in a format that the client can accept, based on the accept headers received by the server. If the request was not a HEAD request, the server can send Content-Language, Content-Encoding and Content-Type headers to indicate which formats are available.]	
204 – No Content [Request is complete without any information being sent back in the response.]	407 – Proxy Authentication Required [Unauthorized access request to a proxy server. The client must first authenticate itself with the proxy. The server sends the Proxy-Authenticate header indicating the authentication scheme and realm for the requested resource.]	
205 – Reset Content [Client should reset the current document. Ie. A form with existing values.]	408 – Request Time-Out [The client has failed to complete its request within the request timeout period used by the server. However, the client can re-request.]	
206 – Partial Content [Server has fulfilled the partial GET request for the resource. In response to a Range request from the client. Or if someone hits stop.]	409 – Conflict [The client request conflicts with another request. The server can add information about the type of conflict along with the status code.]	
Request Redirected	410 – Gone [The requested resource is permanently gone from the server.]	
300 – Multiple Choices [Requested resource corresponds to a set of documents. Server sends information about each one and a URL to request them from so that the client can choose.]	411 – Length Required [The client must supply a Content-Length header in its request.]	
301 – Moved Permanently [Requested resource does not exist on the server. A Location header is sent to the client to redirect it to the new URL. Client continues to use the new URL in future requests.]	412 – Precondition Failed [When a client sends a request with one or more If... headers, the server uses this code to indicate that one or more of the conditions specified in these headers is FALSE.]	
302 – Moved Temporarily [Requested resource has temporarily moved. A Location header is sent to the client to redirect it to the new URL. Client continues to use the old URL in future requests.]	413 – Request Entity Too Large [The server refuses to process the request because its message body is too large. The server can close connection to stop the client from continuing the request.]	
303 – See Other [The requested resource can be found in a different location indicated by the Location header, and the client should use the GET method to retrieve it.]	414 – Request-URI Too Long [The server refuses to process the request, because the specified URI is too long.]	
304 – Not Modified [Used to respond to the If-Modified-Since request header. Indicates that the requested document has not been modified since the specified date, and the client should use a cached copy.]	415 – Unsupported Media Type [The server refuses to process the request, because it does not support the message body's format.]	
305 – Use Proxy [The client should use a proxy, specified by the Location header, to retrieve the URL.]	416 – Requested range not satisfiable	
307 – Temporary Redirect [The requested resource has been temporarily redirected to a different location. A Location header is sent to redirect the client to the new URL. The client continues to use the old URL in future requests.]	417 – Expectation Failed [The server failed to meet the requirements of the Expect request-header.]	
		Unused status codes
		306- Switch Proxy
		416- Requested range not satisfiable
		506- Redirection failed

HTTP protocol version 1.1 Server Response Codes

<http://www.w3.org/Protocols/rfc2616/rfc2616.html>

Chart created September 5, 2000 by Suso Banderas(suso@suso.org). Most of the summary information was gathered from Appendix A of "Apache Server Administrator's Handbook" by Mohammed J. Kabir.

Your HTTP in the client

If you have a Browser and/or terminal, you don't need additional things.

Your HTTP in the server

We use **apache** and **php**.

apache is used to service files and can be resolve in the client. Also, we can configure it to mount differents domains separated but under the same machine.

For do it, first need define **virtual hosts**. In the example we use 3 virtual hosts:

- virtual host to www1.st.um and basic website.
- virtual host to www2.st.um and basic apache authentication.
- virtual host to www3.st.um to get a secure HTTP based on SSL certificate (HTTPS).

This files are defined in `Services/example/` and with the script we go to copy differents files to differents folders:

- We copy the files of the virtual hosts `Services/example/vhost` to `/etc/apache2/sites-available`
- We copy the files of the websites `Services/example/sites` to `/var/www/`
- We copy the files of the apache authentication methd `Services/example/groups` to `/etc/apache2/groups`

Check connectivity

Run client browser and go to url examples:

- 1) Basic website:

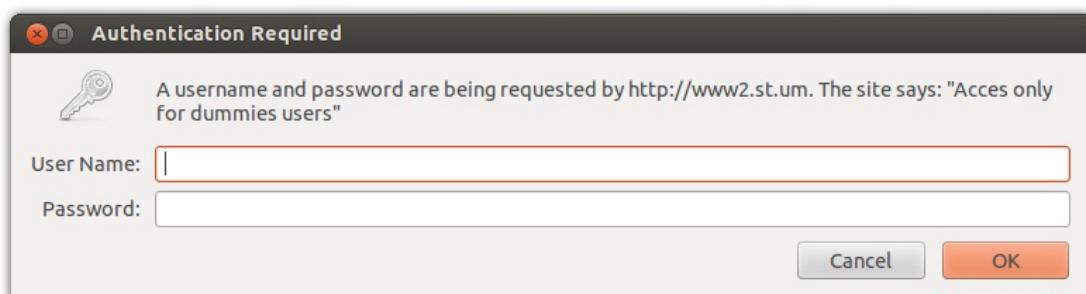


Hello World

- 2) Basic Apache authentication:



Hello World

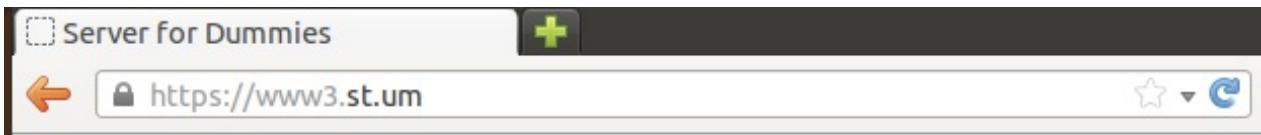


Hello, World!

Basic web with auth

Server for dummies is works.

- 2) Secure HTTP connection:



Hello, World!

Basic web with SSL

Server for dummies is works.

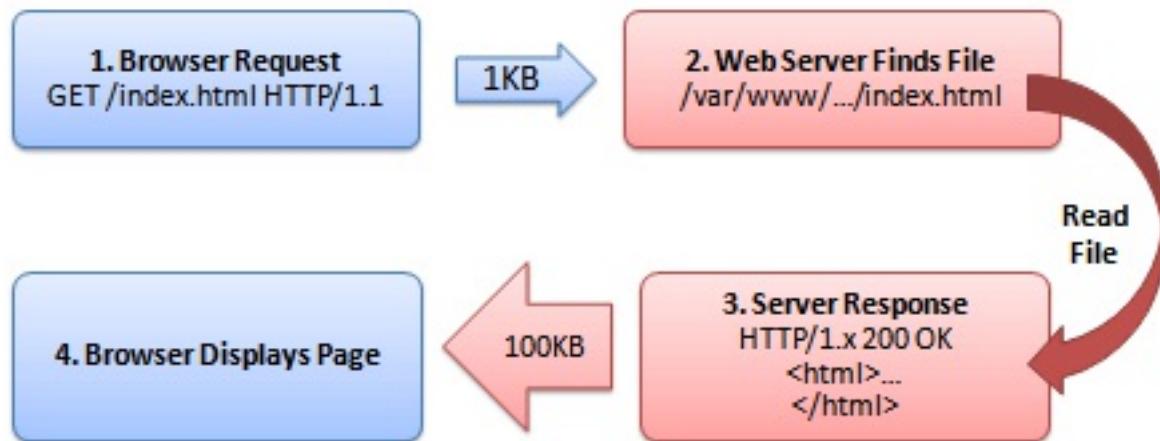
Alternative you can do it with terminal and you get the source code of the website:

```
josefranciscoverdugabin@ubuntu:~/server-for-dummies$ curl www1.st.um
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <title>Test</title>
</head>

<body>
  <h1>Hello World</h1>
</body>
</html>
```

How HTTP works

HTTP Request and Response



No.	Time	Source	Destination	Protocol	Length	Info
10	0.001401	172.16.22.138	172.16.22.139	HTTP	226	GET / HTTP/1.1
12	0.001960	172.16.22.139	172.16.22.138	HTTP	672	HTTP/1.1 200 OK (text/html)

1) HTTP Request (GET Command)

No.	Time	Source	Destination	Protocol	Length	Info
10	0.001401	172.16.22.138	172.16.22.139	HTTP	226	GET / HTTP/1.1
▶ Frame 10: 226 bytes on wire (1808 bits), 226 bytes captured (1808 bits)						
▶ Ethernet II, Src: VMware_30:4d:05 (00:50:56:30:4d:05), Dst: VMware_f7:ce:9d (00:0c:29:f7:ce:9d)						
▶ Internet Protocol Version 4, Src: 172.16.22.138 (172.16.22.138), Dst: 172.16.22.139 (172.16.22.139)						
▶ Transmission Control Protocol, Src Port: 41442 (41442), Dst Port: http (80), Seq: 1, Ack: 1, Len: 160						
▼ Hypertext Transfer Protocol						
▼ GET / HTTP/1.1\r\n						
▼ [Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]						
[Message: GET / HTTP/1.1\r\n]						
[Severity level: Chat]						
[Group: Sequence]						
Request Method: GET						
Request URI: /						
Request Version: HTTP/1.1						
User-Agent: curl/7.22.0 (i686-pc-linux-gnu) libcurl/7.22.0 OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23 librtmp/2.3\r\n						
Host: www1.st.um\r\n						
Accept: */*\r\n						
\r\n						
<u>Full request URI: http://www1.st.um/1</u>						

2) HTTP Response (200 OK code status)

No.	Time	Source	Destination	Protocol	Length	Info
10	0.001401	172.16.22.138	172.16.22.139	HTTP	226	GET / HTTP/1.1
12	0.001960	172.16.22.139	172.16.22.138	HTTP	672	HTTP/1.1 200 OK (text/html)

► Ethernet II, Src: Vmware_f7:ce:9d (00:0c:29:f7:ce:9d), Dst: Vmware_30:4d:05 (00:50:56:30:4d:05)
 ► Internet Protocol Version 4, Src: 172.16.22.139 (172.16.22.139), Dst: 172.16.22.138 (172.16.22.138)
 ► Transmission Control Protocol, Src Port: http (80), Dst Port: 41442 (41442), Seq: 1, Ack: 161, Len: 606

▼ Hypertext Transfer Protocol

▼ HTTP/1.1 200 OK\r\n

 ▼ [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]

 [Message: HTTP/1.1 200 OK\r\n]
 [Severity level: Chat]
 [Group: Sequence]

 Request Version: HTTP/1.1
 Status Code: 200
 Response Phrase: OK
 Date: Wed, 21 May 2014 22:06:49 GMT\r\n
 Server: Apache/2.2.22 (Ubuntu)\r\n
 Last-Modified: Wed, 23 Apr 2014 18:34:45 GMT\r\n
 ETag: "804cc-142-4f7b9fc98baa7"\r\n
 Accept-Ranges: bytes\r\n

▼ Content-Length: 322\r\n

 [Content length: 322]
 Vary: Accept-Encoding\r\n
 Content-Type: text/html\r\n
 X-Pad: avoid browser bug\r\n
 \r\n

▼ Line-based text data: text/html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">\n<html xmlns="http://www.w3.org/1999/xhtml">\n<head>\n    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />\n    <title>Test</title>\n</head>\n\n<body>\n    <h1>Hello World</h1>\n</body>\n</html>\n
```

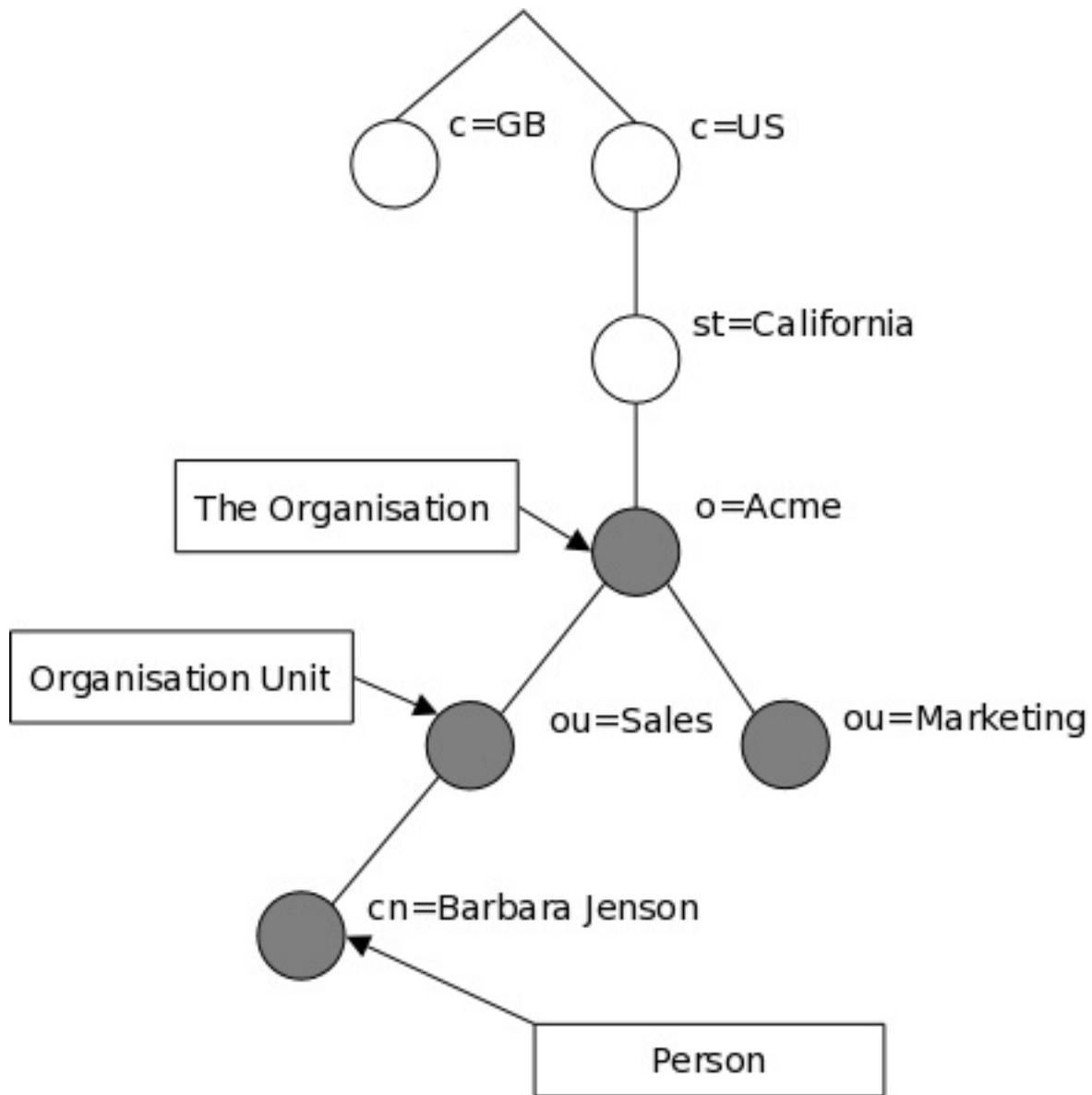
LDAP

Maybe at this moment you don't know what is a LDAP but maybe use LDAP in the real life.

LDAP (*Lightweight Directory Protocol*) concept is similar to use a address book to annotate information about people around you: Adress, email, work mobile,...

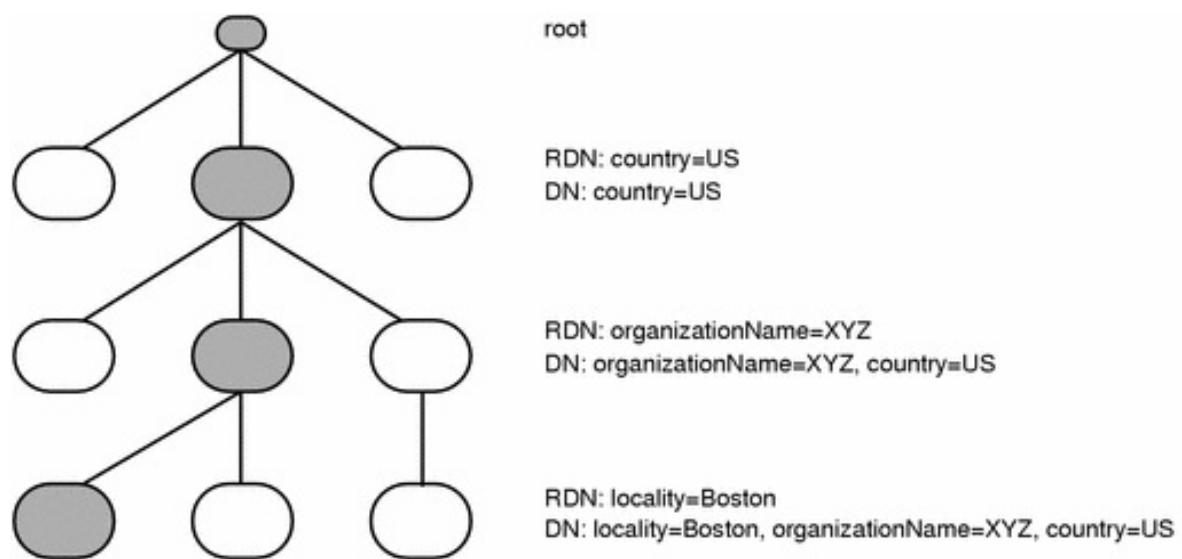
As Internet, LDAP is your adress book. And you can use LDAP for this. Your Adress book online!

LDAP is like database, but is specially designed for queries and put public information about employees of an organization or similar. The tree structure of a LDAP is known as **Directory Information Tree (DIT)**:



LDAP object are referenced by its **Distinguished Name (DN)**. A DN is a sequence of **Relative Distinguished Names (RDN)** connected by commas.

An RDN is an attribute with an associated value in the form attribute=value; normally expressed in a UTF-8 string format.



Each entry is a **ObjectClass** that can contain a define type of attributes.

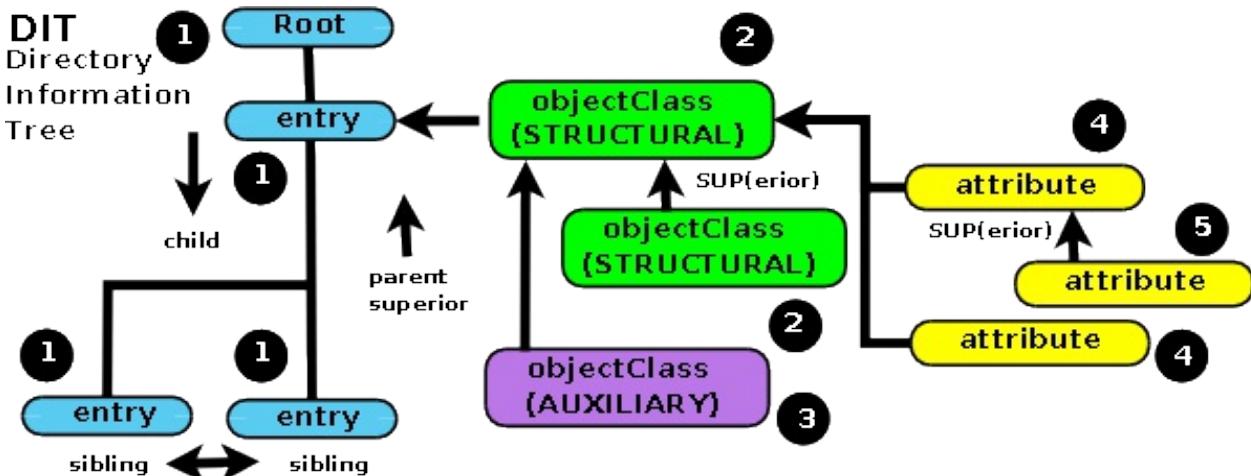
LDAP Structure

LDAP is structured in entry. The first entry of a LDAP is simulated.

Typically LDAP have the form of a tree, but not always is it because can be entries that are *alias* of a other entry.

The entry of a LDAP is limited:

- **STRUCTURAL**: Is the ObjectClass that define the entry
- **AUXILIARY**: When the Objectclass is a parent of a other ObjectClass. The *root ObjectClass* is **top**
- **ABSTRACT**: Optional ObjectClass that the entry can be container.



Commonly Used ObjectClass

Type	Top	Attribute required	Attribute Optional
top		objectClass	
alias		aliasedObjectClass	
person	top		description, seeAlso, telephoneNumber, userPassword
organizationalPerson	person		I, ou, postalAdress, postalCode, st, street, title
inetOrgPerson	organizationalPerson		jpegPhoto, mail, uid, o, userCertificate,...

Commonly Used Attributes

String	Attribute type
DC	domainComponent
CN	commonName
OU	organizationalUnitName
O	organizationName
STREET	streetAddress
L	localityName
ST	stateOrProvinceName
C	countryName
UID	userid

Your LDAP in the client

For search in the LDAP client need to install **ldap-utils** command. Later We can configure your typical mail client (Thunderbild, for example) to search in the LDAP when you need a compose a mail and need to know the destination account.

Your LDAP in the server

In the server need to install **slapd** service configue the database schema and insert sample data. In this case we have the defintions of the data and the information in the `Services/LDAP/*.dif` files and use `Services/LDAP/server.sh` for boostrapping process. This is set up when you choice install LDAP:

```
``` echo " * Installing LDAP..." apt-get -y install slapd ldap-utils  
echo " * Adding LDAP database..."
```

# Load database

```
sudo ldapmodify -Y EXTERNAL -H ldapi:/// -f "$DIR"/loadDBD.ldif;
echo " * Loading LDAP example schema..."
```

## Load database models schema

```
sudo ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/ldap/schema/inetorgperson.ldif; sudo
ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/ldap/schema/cosine.ldif;
echo " * Adding LDAP custom schema ..."
```

## Load own schema for the example

```
sudo ldapadd -Y EXTERNAL -H ldapi:/// -f "$DIR"/stSchema.ldif;
echo " * Configuring LDAP database..."
```

# Load database configuration

```
sudo ldapmodify -Y EXTERNAL -H ldapi:/// -f "$DIR"/confDatabase.ldif;
echo " * Inserting LDAP data..."
```

## Insert fake data for the example

```
sudo ldapmodify -D "cn=admin,o=um,c=es" -W -H ldap:/// -f "$DIR/st.ldif;
```

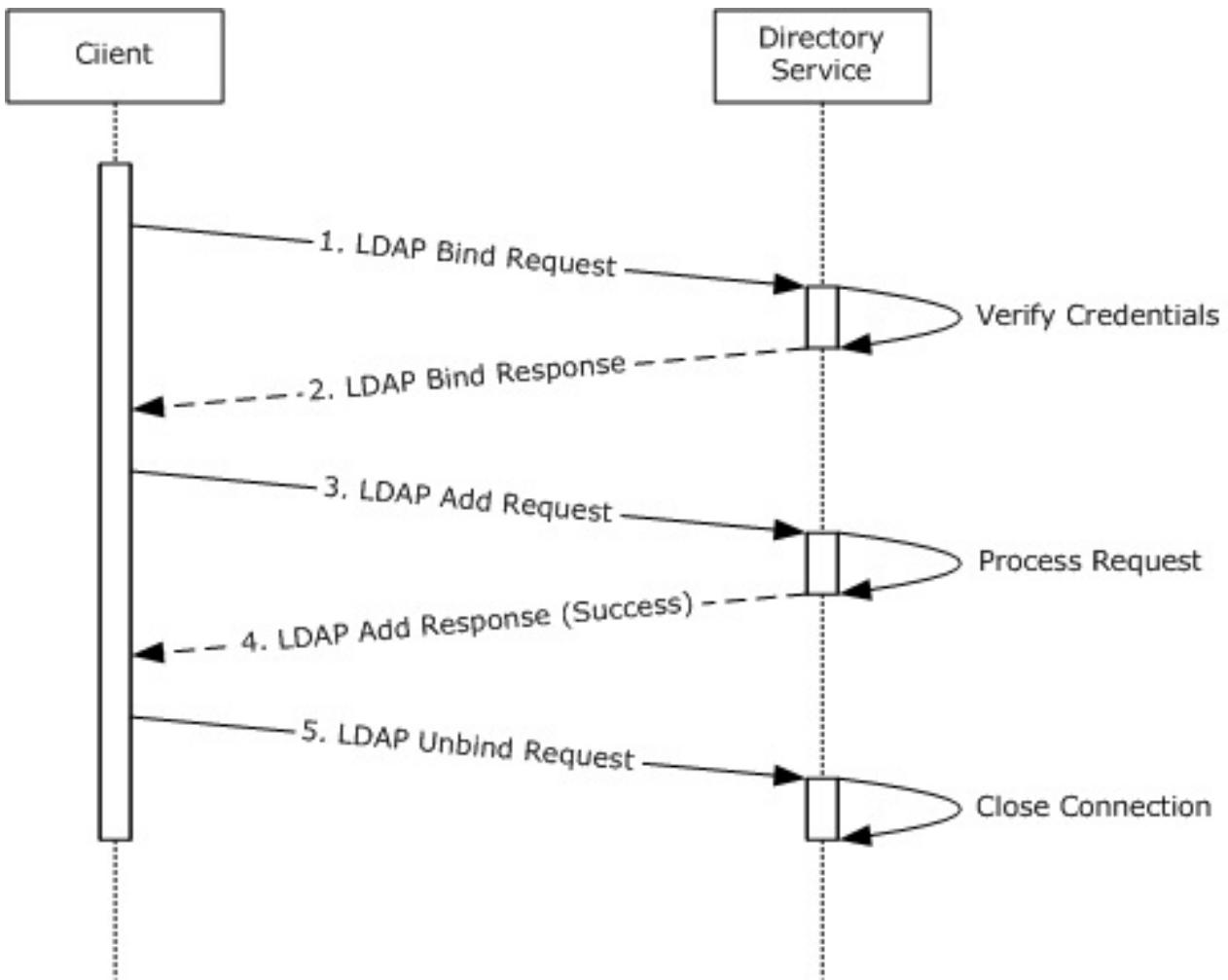
# Check connectivity

```
josefranciscoverdugabin@ubuntu:~/server-for-dummies$ sudo ldapsearch -x -H ldap://ldap.st.um -b "cn=Server Administrator,ou=st,o=um,c=es" mobile
extended LDIF
#
LDAPv3
base <cn=Server Administrator,ou=st,o=um,c=es> with scope subtree
filter: (objectclass=*)
requesting: mobile
#
Server Administrator, st, um, es
dn: cn=Server Administrator,ou=st,o=um,c=es
mobile: 61111111

search result
search: 2
result: 0 Success

numResponses: 2
numEntries: 1
```

# How LDAP works



Resume:

No.	Time	Source	Destination	Protocol	Length	Info
10	0.001651	172.16.22.136	172.16.22.135	LDAP	80	bindRequest(1) "<ROOT>" simple
12	0.002139	172.16.22.135	172.16.22.136	LDAP	80	bindResponse(1) success
14	0.002668	172.16.22.136	172.16.22.135	LDAP	152	searchRequest(2) "cn=Server Administrator,ou=st,o=um,c=es" wholeSubtree
15	0.003105	172.16.22.135	172.16.22.136	LDAP	139	searchResEntry(2) "cn=Server Administrator,ou=st,o=um,c=es"
16	0.003198	172.16.22.135	172.16.22.136	LDAP	80	searchResDone(2) success [1 result]
18	0.003808	172.16.22.136	172.16.22.135	LDAP	73	unbindRequest(3)

## 1) BindRequest (Authentication)

No.	Time	Source	Destination	Protocol	Length	Info
10	0.001651	172.16.22.136	172.16.22.135	LDAP	80	bindRequest(1) "<ROOT>" simple
▶ Frame 10: 80 bytes on wire (640 bits), 80 bytes captured (640 bits)						
▶ Ethernet II, Src: VMware_30:4d:05 (00:50:56:30:4d:05), Dst: VMware_f7:ce:9d (00:0c:29:f7:ce:9d)						
▶ Internet Protocol Version 4, Src: 172.16.22.136 (172.16.22.136), Dst: 172.16.22.135 (172.16.22.135)						
▶ Transmission Control Protocol, Src Port: 56207 (56207), Dst Port: ldap (389), Seq: 1, Ack: 1, Len: 14						
▼ Lightweight Directory Access Protocol						
▼ LDAPMessage bindRequest(1) "<ROOT>" simple						
messageID: 1						
▼ protocolOp: bindRequest (0)						
▼ bindRequest						
version: 3						
name:						
▼ authentication: simple (0)						
simple: <MISSING>						
<a href="#">[Response In: 12]</a>						

## 2) BindResponse

No.	Time	Source	Destination	Protocol	Length	Info
10	0.001651	172.16.22.136	172.16.22.135	LDAP	80	bindRequest(1) "<ROOT>" simple
12	0.002139	172.16.22.135	172.16.22.136	LDAP	80	bindResponse(1) success
▶ Frame 12: 80 bytes on wire (640 bits), 80 bytes captured (640 bits)						
▶ Ethernet II, Src: Vmware_f7:ce:9d (00:0c:29:f7:ce:9d), Dst: Vmware_30:4d:05 (00:50:56:30:4d:05)						
▶ Internet Protocol Version 4, Src: 172.16.22.135 (172.16.22.135), Dst: 172.16.22.136 (172.16.22.136)						
▶ Transmission Control Protocol, Src Port: ldap (389), Dst Port: 56207 (56207), Seq: 1, Ack: 15, Len: 14						
▼ Lightweight Directory Access Protocol						
▼ LDAPMessage bindResponse(1) success						
messageID: 1						
▼ protocolOp: bindResponse (1)						
▼ bindResponse						
resultCode: success (0)						
matchedDN:						
errorMessage:						
<a href="#">[Response To: 10]</a>						
[Time: 0.000488000 seconds]						

### 3) SearchRequest

No.	Time	Source	Destination	Protocol	Length	Info
10	0.001651	172.16.22.136	172.16.22.135	LDAP	80	bindRequest(1) "<ROOT>" simple
12	0.002139	172.16.22.135	172.16.22.136	LDAP	80	bindResponse(1) success
14	0.002668	172.16.22.136	172.16.22.135	LDAP	152	searchRequest(2) "cn=Server Administrator,ou=st,o=um,c=es" wholeSubtree
▶ Frame 14: 152 bytes on wire (1216 bits), 152 bytes captured (1216 bits)						
▶ Ethernet II, Src: Vmware_30:4d:05 (00:50:56:30:4d:05), Dst: Vmware_f7:ce:9d (00:0c:29:f7:ce:9d)						
▶ Internet Protocol Version 4, Src: 172.16.22.136 (172.16.22.136), Dst: 172.16.22.135 (172.16.22.135)						
▶ Transmission Control Protocol, Src Port: 56207 (56207), Dst Port: ldap (389), Seq: 15, Ack: 15, Len: 86						
▼ Lightweight Directory Access Protocol						
▼ LDAPMessage searchRequest(2) "cn=Server Administrator,ou=st,o=um,c=es" wholeSubtree						
messageID: 2						
▼ protocolOp: searchRequest (3)						
▼ searchRequest						
baseObject: cn=Server Administrator,ou=st,o=um,c=es						
scope: wholeSubtree (2)						
derefAliases: neverDerefAliases (0)						
sizeLimit: 0						
timeLimit: 0						
typesOnly: False						
▼ Filter: (objectclass=*)						
▼ filter: present (7)						
present: objectclass						
▼ attributes: 1 item						
AttributeDescription: mobile						
<a href="#">[Response In: 15]</a>						
[Time: 0.000437000 seconds]						

### 4) SearchResponse

No.	Time	Source	Destination	Protocol	Length	Info
10	0.001651	172.16.22.136	172.16.22.135	LDAP	80	bindRequest(1) "<ROOT>" simple
12	0.002139	172.16.22.135	172.16.22.136	LDAP	80	bindResponse(1) success
14	0.002668	172.16.22.136	172.16.22.135	LDAP	152	searchRequest(2) "cn=Server Administrator,ou=st,o=um,c=es" wholeSubtree
15	0.003105	172.16.22.135	172.16.22.136	LDAP	139	searchResEntry(2) "cn=Server Administrator,ou=st,o=um,c=es"
▶ Frame 15: 139 bytes on wire (1112 bits), 139 bytes captured (1112 bits)						
▶ Ethernet II, Src: Vmware_f7:ce:9d (00:0c:29:f7:ce:9d), Dst: Vmware_30:4d:05 (00:50:56:30:4d:05)						
▶ Internet Protocol Version 4, Src: 172.16.22.135 (172.16.22.135), Dst: 172.16.22.136 (172.16.22.136)						
▶ Transmission Control Protocol, Src Port: 56207 (56207), Dst Port: ldap (389), Seq: 15, Ack: 101, Len: 73						
▼ Lightweight Directory Access Protocol						
▼ LDAPMessage searchResEntry(2) "cn=Server Administrator,ou=st,o=um,c=es" [1 result]						
messageID: 2						
▼ protocolOp: searchResEntry (4)						
▼ searchResEntry						
objectName: cn=Server Administrator,ou=st,o=um,c=es						
▼ attributes: 1 item						
▼ PartialAttributeList item mobile						
type: mobile						
▼ vals: 1 item						
6111111111						
<a href="#">[Response To: 14]</a>						
[Time: 0.000437000 seconds]						

### 5) SearchDone

No.	Time	Source	Destination	Protocol	Length	Info
10	0.001651	172.16.22.136	172.16.22.135	LDAP	80	bindRequest(1) "<ROOT>" simple
12	0.002139	172.16.22.135	172.16.22.136	LDAP	80	bindResponse(1) success
14	0.002668	172.16.22.136	172.16.22.135	LDAP	152	searchRequest(2) "cn=Server Administrator,ou=st,o=um,c=es" wholeSubtree
15	0.003105	172.16.22.135	172.16.22.136	LDAP	139	searchResEntry(2) "cn=Server Administrator,ou=st,o=um,c=es"
16	0.003198	172.16.22.135	172.16.22.136	LDAP	80	searchResDone(2) success [1 result]

► Frame 16: 80 bytes on wire (640 bits), 80 bytes captured (640 bits)  
 ► Ethernet II, Src: Vmware\_f7:ce:9d (00:0c:29:f7:ce:9d), Dst: Vmware\_30:4d:05 (00:50:56:30:4d:05)  
 ► Internet Protocol Version 4, Src: 172.16.22.135 (172.16.22.135), Dst: 172.16.22.136 (172.16.22.136)  
 ► Transmission Control Protocol, Src Port: ldap (389), Dst Port: 56207 (56207), Seq: 88, Ack: 101, Len: 14  
 ▼ Lightweight Directory Access Protocol  
 ▼ LDAPMessage searchResDone(2) success [1 result]  
 messageID: 2  
 ▼ protocolOp: searchResDone (5)  
 ▼ searchResDone  
 resultCode: success (0)  
 matchedDN:  
 errorMessage:  
 [Response To: 14]  
 [Time: 0.000530000 seconds]

## 6) unbinRequest

No.	Time	Source	Destination	Protocol	Length	Info
10	0.001651	172.16.22.136	172.16.22.135	LDAP	80	bindRequest(1) "<ROOT>" simple
12	0.002139	172.16.22.135	172.16.22.136	LDAP	80	bindResponse(1) success
14	0.002668	172.16.22.136	172.16.22.135	LDAP	152	searchRequest(2) "cn=Server Administrator,ou=st,o=um,c=es" wholeSubtree
15	0.003105	172.16.22.135	172.16.22.136	LDAP	139	searchResEntry(2) "cn=Server Administrator,ou=st,o=um,c=es"
16	0.003198	172.16.22.135	172.16.22.136	LDAP	80	searchResDone(2) success [1 result]
18	0.003808	172.16.22.136	172.16.22.135	LDAP	73	unbindRequest(3)

► Frame 18: 73 bytes on wire (584 bits), 73 bytes captured (584 bits)  
 ► Ethernet II, Src: Vmware\_30:4d:05 (00:50:56:30:4d:05), Dst: Vmware\_f7:ce:9d (00:0c:29:f7:ce:9d)  
 ► Internet Protocol Version 4, Src: 172.16.22.136 (172.16.22.136), Dst: 172.16.22.135 (172.16.22.135)  
 ► Transmission Control Protocol, Src Port: 56207 (56207), Dst Port: ldap (389), Seq: 101, Ack: 102, Len: 7  
 ▼ Lightweight Directory Access Protocol  
 ▼ LDAPMessage unbindRequest(3)  
 messageID: 3  
 ▼ protocolOp: unbindRequest (2)  
 unbindRequest

# SSL

SSL (Secure Sockets Layer) is the standard security technology for establishing an encrypted link between a web server and a browser. This link ensures that all data passed between the web server and browsers remain private and integral. SSL is an industry standard and is used by millions of websites in the protection of their online transactions with their customers.

SSL does two things:

1. **Authenticates the server to the client.** (*Handshake*)
2. Encrypts your HTTP session.

We go to explain **Handshake** process:

- Client send `clientHello` message specifying encrypted options: SSL version, compress method,... and sending random numbers that they use later.
- The server receive the message and select:
  - What simetric algorythm (**secret key**) use (e.g. **AES, 3DES, RC5**).
  - What asimetric algorythm (**public key**) use (e.g. **RSA, Diffie-Hellman**).
  - What MAC algorythm use (**SHA**).

Why select simetric and asimetric algorythm? Why not use only one?

When client finally send data to the server, they use simetric algorythm, but first, to get it, they need to negotiate and exchange confidence information. This proceed is known as **Pre Master Key**.

There are two ways to do Pre Master Key process:

1. With **Diffie-Helman**: Client and server exchange `SERVER_KEY_EXCHANGE` and `CLIENT_KEY_EXCHANGE` and both parts know the information to operate with simetric algorythm.
2. With **RSA**: The client catch the public key of the Server Certificate (message `CERTIFICATE`) and encrypted the master key. Only the server can desncrypted this with her private key.

And send to the client `serverHello` with this options, distinctive server number and their certificates to check the identify of the server domain.

But, one moment... Why am I going to trust the server? and if not who he claims to be?

Before Pre Master Key process is necessary for the server to authenticate it, for do it is also necessay asimetric algorythm, normally with RSA. And then, thec MAC algorythm is use to verify that data of the session is authenticathed.

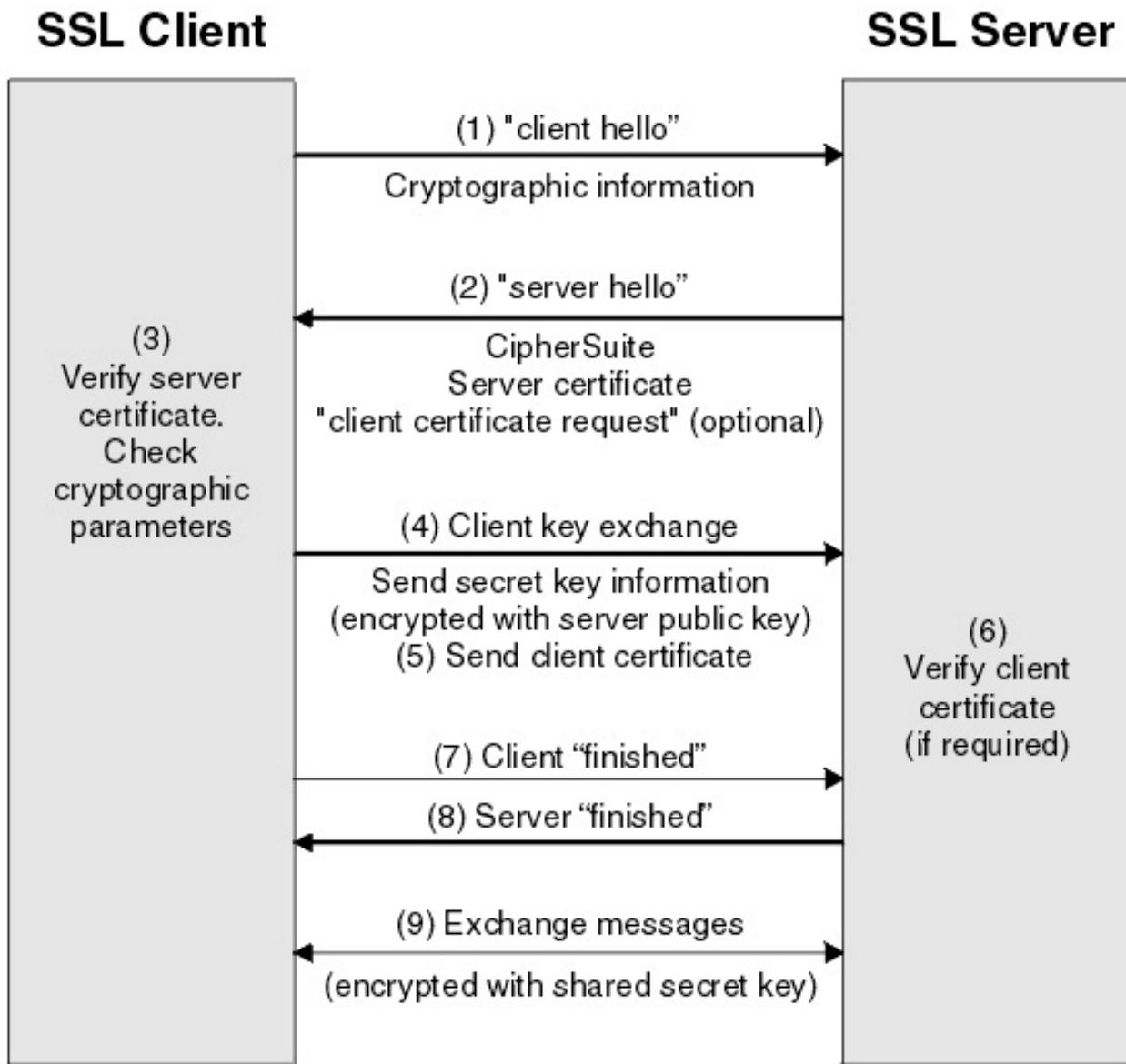
We can resume all the process with two variants of message:

- **TLSv1\_RSA\_WITH\_AES\_128\_CBC\_SHA**
  - **TLSv1**: version of TLS (SSL protocol)
  - **RSA**: For server authentication and pre master key process.
  - **AES**: For simetric algorythm after master key
  - **SHA**: For MAC encryption
- **TLSv1\_DHE\_RSA\_WITH\_AES\_256\_CBC\_SHA**
  - **TLSv1**: version of TLS (SSL protocol)
  - **DHE**: For server authentication based of Diffie-Helman for pre master

key.

- **RSA**: For server authentication.
- **AES**: For simetric algorythm after master key.
- **SHA**: For MAC encryption.

The resume of the process in one image:



The complexities of the SSL protocol remain invisible to your customers. Instead their browsers provide them with a key indicator to let them know they are currently protected by an SSL encrypted session - the lock icon in the lower right-hand corner, clicking on the lock icon displays your SSL Certificate and the details about it. All SSL Certificates are issued to either companies or legally accountable individuals.

## SSL in the client

If the server has done its homework anything is necessary in the client, only have a modern browser and the `*.pem` to load it (see part of SSL in the server).

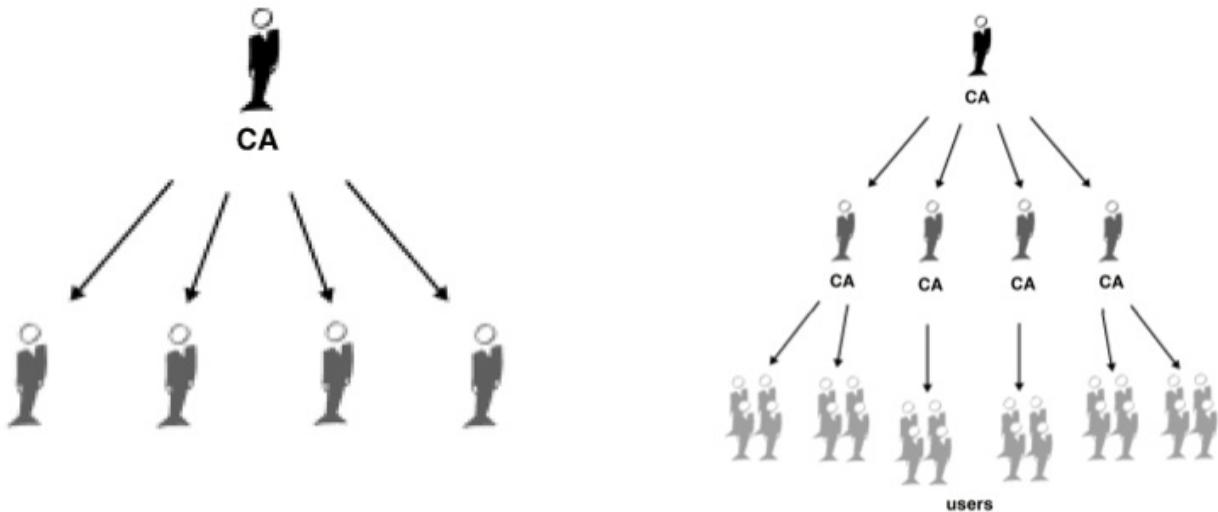
# SSL in the server

For have SSL in HTTP known as HTTPS first you need to have HTTP service.

For have HTTPS you need a certificate to expose that your domain is secure and you are your owner.

Now we are to explain how to generate your certificate for do a illustrative example, but in the practice the SSL certificate is created by **Certification Authority (CA)** that typically are companies of hosting. Remember this.

The CA establishes a chain of trust between he and domains of users:



About what your need in your server we use **openssl** that is typically installed in UNIX systems by default.

The process is divided in two steps: First need to generate the CA autosign for the server and later generate the client certificate and sign it with serve CA.

Typically an SSL Certificate will contain your domain name, your company name, your address, your city, your state and your country. It will also contain the expiration date of the Certificate and details of the Certification Authority responsible for the issuance of the Certificate. When a browser connects to a secure site it will retrieve the site's SSL Certificate and check that it has not expired, it has been issued by a Certification Authority the browser trusts, and that it is being used.

## CA Certificate in the server

first need to know OpenSSL directory. In terminal type:

```
openssl version -d
```

And report you the directory, typically `usr/lib/ssl`. Now we need to edit `openssl.cnf` and update this entries with your values. In my case this are my values:

```
....
[CA_default]
...
dir = ./demoCA
countryName_default = ES
stateOrProvinceName_default =
```

```
0. organizationalName_default = UMU
organizationalUnitName_default = ST
...
```

because my domain is a Spanish domain and the domain for the example is `st@um`.

Now go to adjust options to generate a certificate. in the same file and go to `[ policy_match ]` section and update set `stateOrProvinceName` value to `optional`.

Go to the path of CA\_default and create `demoCA` folder with this structure:

```
cd $HOME
mkdir -p demoCA && cd demoCA
mkdir -p newcerts
mkdir -p certs
touch index.txt
echo "00" > clrnumber
echo "01" > serial
```

Now you are ready to generate the certificate. To do it, run the appropriate command:

```
openssl req -x509 -newkey rsa:2048 -keyout cakey.pem -days 3650
-out
cacert.pem
```

You can check that your certificate is correct with this commands:

```
openssl x509 -in cacert.pem -text
openssl rsa -in cakey.pem -text
```

## CA Certificate in the client

First generate it:

```
openssl req -new -nodes -newkey rsa:1024 -keyout serverkey.pem -
out servercsr.pem
```

And later sign it:

```
openssl ca -keyfile cakey.pem -in servercsr.pem -out servercert.
pem
```

The certificate on `*.pem` is necessary because is the format that the browser can read the certificate.

You can check that certificate is valid running this command:

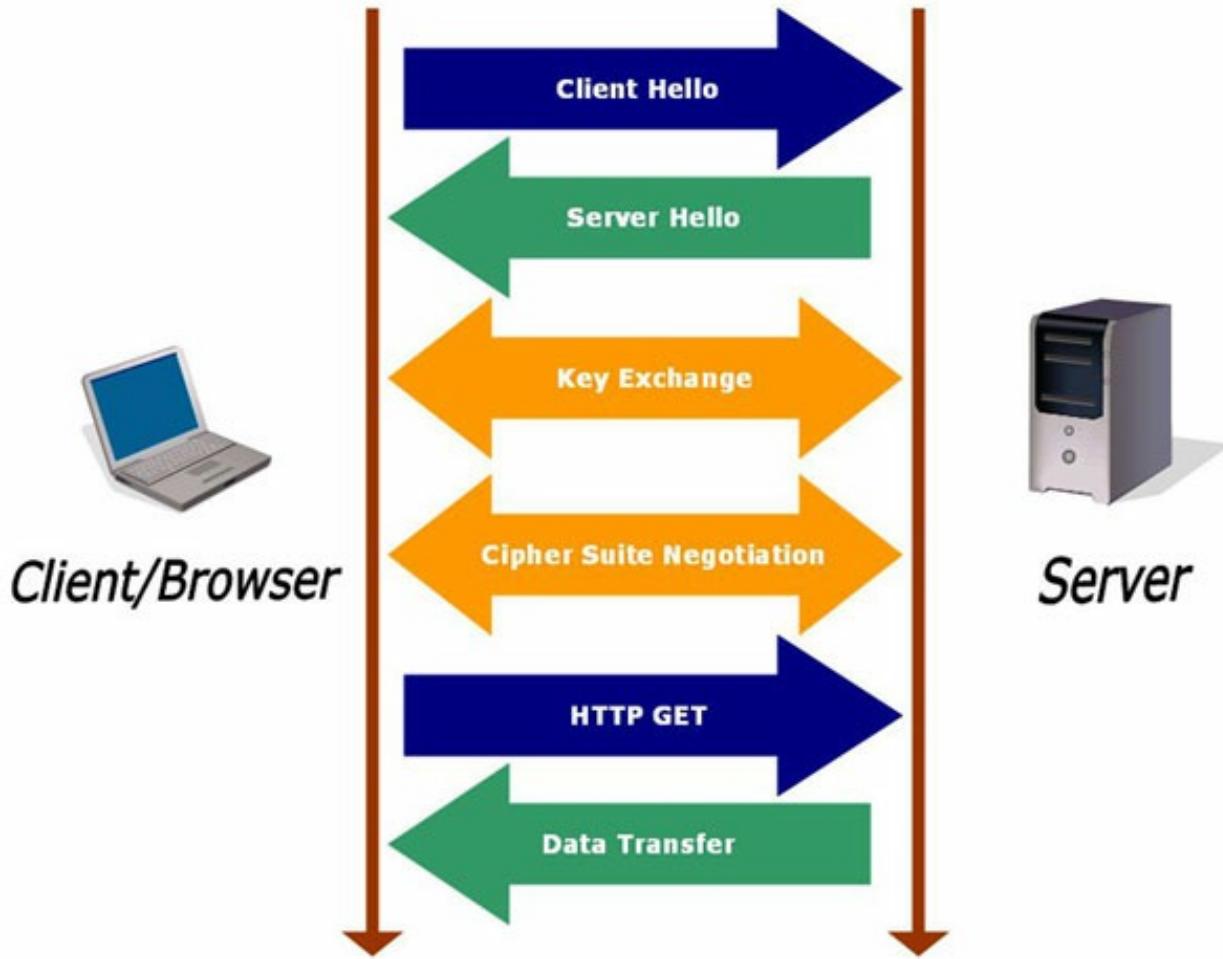
```
openssl s_server -cert servercert.pem -key serverkey.pem -CAfile
clientcert.pem
```

```
josefranciscoverdugambin@ubuntu:~/server-for-dummies/Services/HTTP/example/demoCA$ openssl s_server -cert servercert.pem -key serverkey.pem -CAfile clientcert.pem
Using default temp DH parameters
Using default temp ECDH parameters
ACCEPT
■
```

# Check connectivity

# How SSL Works

Resume of the process:



Resume of messages:

No.	Time	Source	Destination	Protocol	Length	Info
8	0.004607	172.16.22.138	172.16.22.140	TLSv1.2	243	Client Hello
10	0.018600	172.16.22.140	172.16.22.138	TLSv1.2	1342	Server Hello, Certificate, Server Key Exchange, Server Hello Done
12	0.022892	172.16.22.138	172.16.22.140	TLSv1.2	280	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
13	0.025471	172.16.22.140	172.16.22.138	TLSv1.2	364	Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Message
14	0.025796	172.16.22.138	172.16.22.140	TLSv1.2	407	Application Data
15	0.033750	172.16.22.140	172.16.22.138	TLSv1.2	710	Application Data, Application Data, Application Data, Application Data
17	0.072971	172.16.22.138	172.16.22.140	TLSv1.2	407	Application Data
18	0.073772	172.16.22.140	172.16.22.138	TLSv1.2	742	Application Data, Application Data, Application Data, Application Data
20	5.074473	172.16.22.140	172.16.22.138	TLSv1.2	119	Encrypted Alert

- 1) Client send **HELLO** message with distinctive random number and information about what compression and Cipher suites support and request to the server the X509 certificate to trust him.

No.	Time	Source	Destination	Protocol	Length	Info
8	0.004607	172.16.22.138	172.16.22.140	TLSv1.2	243	Client Hello
▶ Frame 8: 243 bytes on wire (1944 bits), 243 bytes captured (1944 bits)						
▶ Ethernet II, Src: VMware_30:4d:05 (00:50:56:30:4d:05), Dst: VMware_f7:ce:9d (00:0c:29:f7:ce:9d)						
▶ Internet Protocol Version 4, Src: 172.16.22.138 (172.16.22.138), Dst: 172.16.22.140 (172.16.22.140)						
▶ Transmission Control Protocol, Src Port: 51860 (51860), Dst Port: https (443), Seq: 1, Ack: 1, Len: 177						
▼ Secure Sockets Layer						
▼ TLSv1.2 Record Layer: Handshake Protocol: Client Hello						
Content Type: Handshake (22)						
Version: TLS 1.0 (0x0301)						
Length: 172						
▼ Handshake Protocol: Client Hello						
Handshake Type: Client Hello (1)						
Length: 168						
Version: TLS 1.2 (0x0303)						
▼ Random						
gmt_unix_time: May 21, 2025 23:45:52.000000000 CEST						
random_bytes: 50c7dd9f8e0971dd6c060ae7bce18d6b9ad44c1b2e333e1f...						
Session ID Length: 0						
Cipher Suites Length: 46						
► Cipher Suites (23 suites)						
Compression Methods Length: 1						
► Compression Methods (1 method)						
Extensions Length: 81						
▼ Extension: server_name						
Type: server_name (0x0000)						
Length: 15						
Data (15 bytes)						
► Extension: renegotiation_info						
► Extension: elliptic_curves						
► Extension: ec_point_formats						
► Extension: SessionTicket TLS						
► Extension: Unknown 13172						
► Extension: status_request						
► Extension: signature_algorithms						

2) The server receive the message and:

1. Read the **HELLO** message of the client and select the options of the session.
2. Send to the client the Certificate to show that you can trust him.
3. do **SERVER\_KEY\_EXCHANGE** as part of the process of Pre Master Key

8 0.004607	172.16.22.138	172.16.22.140	TLSv1.2	243 Client Hello
10 0.010600	172.16.22.140	172.16.22.138	TLSv1.2	1342 Server Hello, Certificate, Server Key Exchange, Server Hello Done
► Frame 10: 1342 bytes on wire (10736 bits), 1342 bytes captured (10736 bits)				
► Ethernet II, Src: Vmware_f7:ce:9d (00:0c:29:f7:ce:9d), Dst: Vmware_30:4d:05 (00:50:56:30:4d:05)				
► Internet Protocol Version 4, Src: 172.16.22.140 (172.16.22.140), Dst: 172.16.22.138 (172.16.22.138)				
► Transmission Control Protocol, Src Port: https (443), Dst Port: 51860 (51860), Seq: 1, Ack: 178, Len: 1276				
▼ Secure Sockets Layer				
▼ TLSv1.2 Record Layer: Handshake Protocol: Server Hello				
Content Type: Handshake (22)				
Version: TLS 1.2 (0x0303)				
Length: 57				
▼ Handshake Protocol: Server Hello				
Handshake Type: Server Hello (2)				
Length: 53				
Version: TLS 1.2 (0x0303)				
► Random				
Session ID Length: 0				
Cipher Suite: TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x0033)				
Compression Method: null (0)				
Extensions Length: 13				
► Extension: server_name				
► Extension: renegotiation_info				
► Extension: SessionTicket TLS				
▼ TLSv1.2 Record Layer: Handshake Protocol: Certificate				
Content Type: Handshake (22)				
Version: TLS 1.2 (0x0303)				
Length: 796				
► Handshake Protocol: Certificate				
▼ TLSv1.2 Record Layer: Handshake Protocol: Server Key Exchange				
Content Type: Handshake (22)				
Version: TLS 1.2 (0x0303)				
Length: 399				
► Handshake Protocol: Server Key Exchange				
▼ TLSv1.2 Record Layer: Handshake Protocol: Server Hello Done				
Content Type: Handshake (22)				
Version: TLS 1.2 (0x0303)				
Length: 4				
► Handshake Protocol: Server Hello Done				

3) Client receive server message and proceed to:

1. Validate the identity of the server.
2. do **CLIENT\_KEY\_ECHANGE** as part of the process of Pre Master Key.

at this moment client can be calculate the **Master Key**.

No.	Time	Source	Destination	Protocol	Length	Info
8 0.004607		172.16.22.138	172.16.22.140	TLSv1.2	243 Client Hello	
10 0.010600		172.16.22.140	172.16.22.138	TLSv1.2	1342 Server Hello, Certificate, Server Key Exchange, Server Hello Done	
12 0.022892		172.16.22.138	172.16.22.140	TLSv1.2	280 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message	
► Frame 12: 280 bytes on wire (2240 bits), 280 bytes captured (2240 bits)						
► Ethernet II, Src: Vmware_30:4d:05 (00:50:56:30:4d:05), Dst: Vmware_f7:ce:9d (00:0c:29:f7:ce:9d)						
► Internet Protocol Version 4, Src: 172.16.22.138 (172.16.22.138), Dst: 172.16.22.140 (172.16.22.140)						
► Transmission Control Protocol, Src Port: https (443), Dst Port: 51860 (51860), Seq: 1, Ack: 178, Len: 214						
▼ Secure Sockets Layer						
▼ TLSv1.2 Record Layer: Handshake Protocol: Client Key Exchange						
Content Type: Handshake (22)						
Version: TLS 1.2 (0x0303)						
Length: 134						
▼ Handshake Protocol: Client Key Exchange						
Handshake Type: Client Key Exchange (16)						
Length: 130						
▼ TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec						
Content Type: Change Cipher Spec (20)						
Version: TLS 1.2 (0x0303)						
Length: 1						
Change Cipher Spec Message						
▼ TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message						
Content Type: Handshake (22)						
Version: TLS 1.2 (0x0303)						
Length: 64						
Handshake Protocol: Encrypted Handshake Message						

4) Server receive client message and can calculate **Master Key**. From here both part have a shared secret

No.	Time	Source	Destination	Protocol	Length	Info
8	0.004607	172.16.22.138	172.16.22.140	TLSv1.2	243	Client Hello
10	0.010600	172.16.22.140	172.16.22.138	TLSv1.2	1342	Server Hello, Certificate, Server Key Exchange, Server Hello Done
12	0.022892	172.16.22.138	172.16.22.140	TLSv1.2	280	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
13	0.025471	172.16.22.140	172.16.22.138	TLSv1.2	364	Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Message

► Frame 13: 364 bytes on wire (2912 bits), 364 bytes captured (2912 bits)

► Ethernet II, Src: Vmware\_f7:ce:9d (00:0c:29:f7:ce:9d), Dst: Vmware\_30:4d:05 (00:50:56:30:4d:05)

► Internet Protocol Version 4, Src: 172.16.22.140 (172.16.22.140), Dst: 172.16.22.138 (172.16.22.138)

► Transmission Control Protocol, Src Port: https (443), Dst Port: 51860 (51860), Seq: 1277, Ack: 392, Len: 298

▼ Secure Sockets Layer

▼ TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message

    Content Type: Handshake (22)

    Version: TLS 1.2 (0x0303)

    Length: 218

    Handshake Protocol: Encrypted Handshake Message

▼ TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec

    Content Type: Change Cipher Spec (20)

    Version: TLS 1.2 (0x0303)

    Length: 1

    Change Cipher Spec Message

▼ TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message

    Content Type: Handshake (22)

    Version: TLS 1.2 (0x0303)

    Length: 64

    Handshake Protocol: Encrypted Handshake Message

5) The information from here can travel encrypted. Is not possible see the data content

No.	Time	Source	Destination	Protocol	Length	Info
8	0.004607	172.16.22.138	172.16.22.140	TLSv1.2	243	Client Hello
10	0.010600	172.16.22.140	172.16.22.138	TLSv1.2	1342	Server Hello, Certificate, Server Key Exchange, Server Hello Done
12	0.022892	172.16.22.138	172.16.22.140	TLSv1.2	280	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
13	0.025471	172.16.22.140	172.16.22.138	TLSv1.2	364	Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Message
14	0.025796	172.16.22.138	172.16.22.140	TLSv1.2	407	Application Data

▶ Frame 14: 407 bytes on wire (3256 bits), 407 bytes captured (3256 bits)

▶ Ethernet II, Src: VMware\_30:4d:05 (00:50:56:30:4d:05), Dst: VMware f7:ce:9d (00:0c:29:f7:ce:9d)

▶ Internet Protocol Version 4, Src: 172.16.22.138 (172.16.22.138), Dst: 172.16.22.140 (172.16.22.140)

▶ Transmission Control Protocol, Src Port: 51860 (51860), Dst Port: https (443), Seq: 392, Ack: 1575, Len: 341

Secure Sockets Layer

  ▼ TLSv1.2 Record Layer: Application Data Protocol: http

    Content Type: Application Data (23)

    Version: TLS 1.2 (0x0303)

    Length: 336

    Encrypted Application Data: d595f3a0873a43cf12271d8ed9f99ee9e1e623b3cf90cd2...

No.	Time	Source	Destination	Protocol	Length	Info
8	0.004607	172.16.22.138	172.16.22.140	TLSv1.2	243	Client Hello
10	0.016600	172.16.22.140	172.16.22.138	TLSv1.2	1342	Server Hello, Certificate, Server Key Exchange, Server Hello Done
12	0.022892	172.16.22.138	172.16.22.140	TLSv1.2	280	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
13	0.025471	172.16.22.140	172.16.22.138	TLSv1.2	364	Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Message
14	0.025796	172.16.22.138	172.16.22.140	TLSv1.2	407	Application Data
15	0.033750	172.16.22.140	172.16.22.138	TLSv1.2	710	Application Data, Application Data, Application Data, Application Data

▶ Frame 15: 710 bytes on wire (5680 bits), 710 bytes captured (5680 bits)

▀ Ethernet II, Src: Vmware\_f7:ce:9d (00:0c:29:f7:ce:9d), Dst: Vmware\_30:4d:05 (00:50:56:30:4d:05)

▀ Internet Protocol Version 4, Src: 172.16.22.140 (172.16.22.140), Dst: 172.16.22.138 (172.16.22.138)

▶ Transmission Control Protocol, Src Port: https (443), Dst Port: 51860 (51860), Seq: 1575, Ack: 733, Len: 644

Secure Sockets Layer

- ▼ TLSv1.2 Record Layer: Application Data Protocol: http
  - Content Type: Application Data (23)
  - Version: TLS 1.2 (0x0303)
  - Length: 320
  - Encrypted Application Data: 0d548663c4033dc2e466a538eabbd052aa6021e55f34239b...
- ▼ TLSv1.2 Record Layer: Application Data Protocol: http
  - Content Type: Application Data (23)
  - Version: TLS 1.2 (0x0303)
  - Length: 48
  - Encrypted Application Data: e707499c51cd6d76678b4b8cc1f3cb1e989fc22b2917005d...
- ▼ TLSv1.2 Record Layer: Application Data Protocol: http
  - Content Type: Application Data (23)
  - Version: TLS 1.2 (0x0303)
  - Length: 208
  - Encrypted Application Data: eded16b95fa77fc13dcefa5a95f7df466929d983fd435298...
- ▼ TLSv1.2 Record Layer: Application Data Protocol: http
  - Content Type: Application Data (23)
  - Version: TLS 1.2 (0x0303)
  - Length: 48
  - Encrypted Application Data: 1719c014dade7280433f285d933753d65f45f46d82246639...

No.	Time	Source	Destination	Protocol	Length	Info
8 0.004607	172.16.22.138	172.16.22.140	TLSv1.2	243	Client Hello	
10 0.010600	172.16.22.140	172.16.22.138	TLSv1.2	1342	Server Hello, Certificate, Server Key Exchange, Server Hello Done	
12 0.022892	172.16.22.138	172.16.22.140	TLSv1.2	280	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message	
13 0.025471	172.16.22.140	172.16.22.138	TLSv1.2	364	Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Message	
14 0.025796	172.16.22.138	172.16.22.140	TLSv1.2	407	Application Data	
15 0.033750	172.16.22.140	172.16.22.138	TLSv1.2	710	Application Data, Application Data, Application Data, Application Data	
17 0.072971	172.16.22.138	172.16.22.140	TLSv1.2	407	Application Data	

► Frame 17: 407 bytes on wire (3256 bits), 407 bytes captured (3256 bits)  
 ► Ethernet II, Src: VMware\_30:4d:05 (00:50:56:30:4d:05), Dst: VMware\_f7:ce:9d (00:0c:29:f7:ce:9d)  
 ► Internet Protocol Version 4, Src: 172.16.22.138 (172.16.22.138), Dst: 172.16.22.140 (172.16.22.140)  
 ► Transmission Control Protocol, Src Port: 51860 (51860), Dst Port: https (443), Seq: 733, Ack: 2219, Len: 341  
 ▾ Secure Sockets Layer  
 ▾ TLSv1.2 Record Layer: Application Data Protocol: http  
 Content Type: Application Data (23)  
 Version: TLS 1.2 (0x0303)  
 Length: 336  
 Encrypted Application Data: 2981eb4f81af17clf450dc33d76f41ddef096d770d4637b...

No.	Time	Source	Destination	Protocol	Length	Info
8 0.004607	172.16.22.138	172.16.22.140	TLSv1.2	243	Client Hello	
10 0.010600	172.16.22.140	172.16.22.138	TLSv1.2	1342	Server Hello, Certificate, Server Key Exchange, Server Hello Done	
12 0.022892	172.16.22.138	172.16.22.140	TLSv1.2	280	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message	
13 0.025471	172.16.22.140	172.16.22.138	TLSv1.2	364	Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Message	
14 0.025796	172.16.22.138	172.16.22.140	TLSv1.2	407	Application Data	
15 0.033750	172.16.22.140	172.16.22.138	TLSv1.2	710	Application Data, Application Data, Application Data, Application Data	
17 0.072971	172.16.22.138	172.16.22.140	TLSv1.2	407	Application Data	
18 0.073772	172.16.22.140	172.16.22.138	TLSv1.2	742	Application Data, Application Data, Application Data, Application Data	

► Frame 18: 742 bytes on wire (5936 bits), 742 bytes captured (5936 bits)  
 ► Ethernet II, Src: VMware\_f7:ce:9d (00:0c:29:f7:ce:9d), Dst: VMware\_30:4d:05 (00:50:56:30:4d:05)  
 ► Internet Protocol Version 4, Src: 172.16.22.140 (172.16.22.140), Dst: 172.16.22.138 (172.16.22.138)  
 ► Transmission Control Protocol, Src Port: https (443), Dst Port: 51860 (51860), Seq: 2219, Ack: 1074, Len: 676  
 ▾ Secure Sockets Layer  
 ▾ TLSv1.2 Record Layer: Application Data Protocol: http  
 Content Type: Application Data (23)  
 Version: TLS 1.2 (0x0303)  
 Length: 304  
 Encrypted Application Data: 3c6d073c01adaaf066eb32e730053b13ca8c9cc6ade631ed...  
 ▾ TLSv1.2 Record Layer: Application Data Protocol: http  
 Content Type: Application Data (23)  
 Version: TLS 1.2 (0x0303)  
 Length: 48  
 Encrypted Application Data: 1d8a8b714925cla85276957cc3b3a64545b33692f41130f3...  
 ▾ TLSv1.2 Record Layer: Application Data Protocol: http  
 Content Type: Application Data (23)  
 Version: TLS 1.2 (0x0303)  
 Length: 256  
 Encrypted Application Data: 361d78df2bdcc31353920cb334db4b29ab494efc816888a0...  
 ▾ TLSv1.2 Record Layer: Application Data Protocol: http  
 Content Type: Application Data (23)  
 Version: TLS 1.2 (0x0303)  
 Length: 48  
 Encrypted Application Data: 673727184c5636cce18035e7b7b03581d0db71909491af4fd...

Wireshark Network Traffic Analyzer						
Filter: ssl						
Expression... Clear Apply						
No.	Time	Source	Destination	Protocol	Length	Info
8 0.004607	172.16.22.138	172.16.22.140	TLSv1.2	243	Client Hello	
10 0.010600	172.16.22.140	172.16.22.138	TLSv1.2	1342	Server Hello, Certificate, Server Key Exchange, Server Hello Done	
12 0.022892	172.16.22.138	172.16.22.140	TLSv1.2	280	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message	
13 0.025471	172.16.22.140	172.16.22.138	TLSv1.2	364	Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Message	
14 0.025796	172.16.22.138	172.16.22.140	TLSv1.2	407	Application Data	
15 0.033750	172.16.22.140	172.16.22.138	TLSv1.2	710	Application Data, Application Data, Application Data, Application Data	
17 0.072971	172.16.22.138	172.16.22.140	TLSv1.2	407	Application Data	
18 0.073772	172.16.22.140	172.16.22.138	TLSv1.2	742	Application Data, Application Data, Application Data, Application Data	
20 5.074473	172.16.22.140	172.16.22.138	TLSv1.2	119	Encrypted Alert	

► Frame 20: 119 bytes on wire (952 bits), 119 bytes captured (952 bits)  
 ► Ethernet II, Src: VMware\_f7:ce:9d (00:0c:29:f7:ce:9d), Dst: VMware\_30:4d:05 (00:50:56:30:4d:05)  
 ► Internet Protocol Version 4, Src: 172.16.22.140 (172.16.22.140), Dst: 172.16.22.138 (172.16.22.138)  
 ► Transmission Control Protocol, Src Port: https (443), Dst Port: 51860 (51860), Seq: 2895, Ack: 1074, Len: 53  
 ▾ Secure Sockets Layer  
 ▾ TLSv1.2 Record Layer: Encrypted Alert  
 Content Type: Alert (21)  
 Version: TLS 1.2 (0x0303)  
 Length: 48  
 Alert Message: Encrypted Alert

# SSH

SSH (*Secure Shell*) is a cryptographic network protocol for secure data communication, remote command-line login, remote command execution, and other secure network services between two networked computers.

It was designed as a replacement for Telnet and other insecure remote shell protocols such as the Berkeley rsh and rexec protocols, which send information, notably passwords, in plaintext, rendering them susceptible to interception and disclosure using packet analysis. The encryption used by SSH is intended to provide confidentiality and integrity of data over an unsecured network, such as the Internet.

# Your SSH in the client

The client needs **openssh-client** that is usually installed on UNIX systems default

Also need to generate a **SSH public key** to identify the computer. SSH keys are a way to identify trusted computers, without involving passwords.

## Check your SSH keys

```
cd ~/.ssh
ls -al
Lists the files in your .ssh directory
```

## Generate a new SSH key

```
ssh-keygen -t rsa -C "your_email@example.com"
Creates a new ssh key, using the provided email as a label
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/you/.ssh/id_rsa):
[Press enter]
Enter passphrase (empty for no passphrase): [Type a passphrase]
Enter same passphrase again: [Type passphrase again]
Your identification has been saved in /Users/you/.ssh/id_rsa.
Your public key has been saved in /Users/you/.ssh/id_rsa.pub.
The key fingerprint is:
01:0f:f4:3b:ca:85:d6:17:a1:7d:f0:68:9d:f0:a2:db your_email@example.com
```

# Your SSH in the server

Only need to install **openssh-server**.

# Check connectivity

```
josefranciscoverdugamin@ubuntu:~$ ssh user1@st.um
The authenticity of host 'st.um (172.16.22.142)' can't be established.
ECDSA key fingerprint is 16:04:38:b4:08:ed:ca:80:2f:59:e0:c3:30:2e:b2:2e.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/josefranciscoverdugamin/.ssh/known_hosts)
.
user1@st.um's password:
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.8.0-39-generic i686)

 * Documentation: https://help.ubuntu.com/

Last login: Fri May 23 16:17:24 2014 from 172.16.22.141
$ ls -l
total 48
drwxr-xr-x 2 user1 user1 4096 may 18 17:22 Desktop
drwxr-xr-x 2 user1 user1 4096 may 18 17:22 Documents
drwxr-xr-x 2 user1 user1 4096 may 18 17:22 Downloads
drwx----- 5 user1 user1 4096 may 18 21:02 Maildir
drwxr-xr-x 2 user1 user1 4096 may 18 17:22 Music
drwxr-xr-x 2 user1 user1 4096 may 18 17:22 Pictures
drwxr-xr-x 2 user1 user1 4096 may 18 17:22 Public
drwxr-xr-x 2 user1 user1 4096 may 18 17:22 Templates
drwxr-xr-x 2 user1 user1 4096 may 18 17:22 Videos
-rw-r--r-- 1 user1 user1 8445 abr 16 2012 examples.desktop
$
```

# How SSH works

Resume of messages:

No.	Time	Source	Destination	Protocol	Length	Info
10	0.006358	172.16.22.142	172.16.22.141	SSHv2	107	Server Protocol: SSH-2.0-OpenSSH_5.9p1 Debian-Subuntu1.4\r\n
12	0.006592	172.16.22.141	172.16.22.142	SSHv2	107	Client Protocol: SSH-2.0-OpenSSH_5.9p1 Debian-Subuntu1.4\r\n
14	0.006867	172.16.22.141	172.16.22.142	SSHv2	1338	Client: Key Exchange Init
16	0.007561	172.16.22.142	172.16.22.141	SSHv2	1050	Server: Key Exchange Init
17	0.008331	172.16.22.141	172.16.22.142	SSHv2	146	Client: Diffie-Hellman Key Exchange Init
18	0.010465	172.16.22.142	172.16.22.141	SSHv2	378	Server: New Keys
20	1.597953	172.16.22.141	172.16.22.142	SSHv2	82	Client: New Keys
22	1.635517	172.16.22.141	172.16.22.142	TCP	114	[TCP segment of a reassembled PDU]
24	1.635864	172.16.22.142	172.16.22.141	TCP	114	[TCP segment of a reassembled PDU]

## 1) Client first message

No.	Time	Source	Destination	Protocol	Length	Info
10	0.006358	172.16.22.142	172.16.22.141	SSHv2	107	Server Protocol: SSH-2.0-OpenSSH_5.9p1 Debian-Subuntu1.4\r\n
▶ Frame 10: 107 bytes on wire (856 bits), 107 bytes captured (856 bits)						
▶ Ethernet II, Src: Vmware_f7:ce:9d (00:0c:29:f7:ce:9d), Dst: Vmware_30:4d:05 (00:50:56:30:4d:05)						
▶ Internet Protocol Version 4, Src: 172.16.22.142 (172.16.22.142), Dst: 172.16.22.141 (172.16.22.141)						
▶ Transmission Control Protocol, Src Port: ssh (22), Dst Port: 35414 (35414), Seq: 1, Ack: 1, Len: 41						
▼ SSH Protocol						
Protocol: SSH-2.0-OpenSSH_5.9p1 Debian-Subuntu1.4\r\n\r\n						

## 2) Server first message

No.	Time	Source	Destination	Protocol	Length	Info
10	0.006358	172.16.22.142	172.16.22.141	SSHv2	107	Server Protocol: SSH-2.0-OpenSSH_5.9p1 Debian-Subuntu1.4\r\n
12	0.006592	172.16.22.141	172.16.22.142	SSHv2	107	Client Protocol: SSH-2.0-OpenSSH_5.9p1 Debian-Subuntu1.4\r\n
▶ Frame 12: 107 bytes on wire (856 bits), 107 bytes captured (856 bits)						
▶ Ethernet II, Src: Vmware_30:4d:05 (00:50:56:30:4d:05), Dst: Vmware_f7:ce:9d (00:0c:29:f7:ce:9d)						
▶ Internet Protocol Version 4, Src: 172.16.22.141 (172.16.22.141), Dst: 172.16.22.142 (172.16.22.142)						
▶ Transmission Control Protocol, Src Port: 35414 (35414), Dst Port: ssh (22), Seq: 1, Ack: 42, Len: 41						
▼ SSH Protocol						
Protocol: SSH-2.0-OpenSSH_5.9p1 Debian-Subuntu1.4\r\n\r\n						

## 3) Client send CLIENT\_KEY\_EXCHANGE

No.	Time	Source	Destination	Protocol	Length	Info
10	0.006358	172.16.22.142	172.16.22.141	SSHv2	107	Server Protocol: SSH-2.0-OpenSSH_5.9p1 Debian-Subuntu1.4\r\n
12	0.006592	172.16.22.141	172.16.22.142	SSHv2	107	Client Protocol: SSH-2.0-OpenSSH_5.9p1 Debian-Subuntu1.4\r\n
14	0.006867	172.16.22.141	172.16.22.142	SSHv2	1338	Client: Key Exchange Init
▶ Frame 14: 1338 bytes on wire (10704 bits), 1338 bytes captured (10704 bits)						
▶ Ethernet II, Src: Vmware_30:4d:05 (00:50:56:30:4d:05), Dst: Vmware_f7:ce:9d (00:0c:29:f7:ce:9d)						
▶ Internet Protocol Version 4, Src: 172.16.22.141 (172.16.22.141), Dst: 172.16.22.142 (172.16.22.142)						
▶ Transmission Control Protocol, Src Port: 35414 (35414), Dst Port: ssh (22), Seq: 42, Ack: 42, Len: 1272						
▼ SSH Protocol						
▼ SSH Version 2 (encryption:aes128-ctr mac:hmac-md5 compression:none)						
Packet Length: 1268						
Padding Length: 8						
▼ Key Exchange						
Msg code: Key Exchange Init (20)						
► Algorithms						
Padding String: 0000000000000000						

## 4) Server send CLIENT\_KEY\_EXCHANGE

No.	Time	Source	Destination	Protocol	Length	Info
10	0.006358	172.16.22.142	172.16.22.141	SSHv2	107	Server Protocol: SSH-2.0-OpenSSH_5.9p1 Debian-Subuntu1.4\r\n
12	0.006592	172.16.22.141	172.16.22.142	SSHv2	107	Client Protocol: SSH-2.0-OpenSSH_5.9p1 Debian-Subuntu1.4\r\n
14	0.006867	172.16.22.141	172.16.22.142	SSHv2	1338	Client: Key Exchange Init
16	0.007561	172.16.22.142	172.16.22.141	SSHv2	1050	Server: Key Exchange Init
▶ Frame 16: 1050 bytes on wire (8400 bits), 1050 bytes captured (8400 bits)						
▶ Ethernet II, Src: Vmware_f7:ce:9d (00:0c:29:f7:ce:9d), Dst: Vmware_30:4d:05 (00:50:56:30:4d:05)						
▶ Internet Protocol Version 4, Src: 172.16.22.142 (172.16.22.142), Dst: 172.16.22.141 (172.16.22.141)						
▶ Transmission Control Protocol, Src Port: ssh (22), Dst Port: 35414 (35414), Seq: 42, Ack: 1314, Len: 984						
▼ SSH Protocol						
▼ SSH Version 2 (encryption:aes128-ctr mac:hmac-md5 compression:none)						
Packet Length: 980						
Padding Length: 9						
▼ Key Exchange						
Msg code: Key Exchange Init (20)						
► Algorithms						
Padding String: 0000000000000000						

## 5) Exchange sucess!

No.	Time	Source	Destination	Protocol	Length	Info
10	0.006358	172.16.22.142	172.16.22.141	SSHv2	107	Server Protocol: SSH-2.0-OpenSSH_5.9p1 Debian-5ubuntu1.4\r
12	0.006592	172.16.22.141	172.16.22.142	SSHv2	107	Client Protocol: SSH-2.0-OpenSSH_5.9p1 Debian-5ubuntu1.4\r
14	0.006867	172.16.22.141	172.16.22.142	SSHv2	1338	Client: Key Exchange Init
16	0.007561	172.16.22.142	172.16.22.141	SSHv2	1050	Server: Key Exchange Init
17	0.008331	172.16.22.141	172.16.22.142	SSHv2	146	Client: Diffie-Hellman Key Exchange Init

► Frame 17: 146 bytes on wire (1168 bits), 146 bytes captured (1168 bits)  
 ► Ethernet II, Src: Vmware\_30:4d:05 (00:50:56:30:4d:05), Dst: Vmware\_f7:ce:9d (00:0c:29:f7:ce:9d)  
 ► Internet Protocol Version 4, Src: 172.16.22.141 (172.16.22.141), Dst: 172.16.22.142 (172.16.22.142)  
 ► Transmission Control Protocol, Src Port: 35414 (35414), Dst Port: ssh (22), Seq: 1314, Ack: 1026, Len: 80  
 ▾ SSH Protocol  
 ▾ SSH Version 2 (encryption:aes128-ctr mac:hmac-md5 compression:none)  
 Packet Length: 76  
 Padding Length: 5  
 ▾ Key Exchange  
 Msg code: Diffie-Hellman Key Exchange Init (30)  
 Payload: 00000041042fc8701232d670d2736e1d85697b9b6ae352c1...  
 Padding String: 0000000000

6)

No.	Time	Source	Destination	Protocol	Length	Info
10	0.006358	172.16.22.142	172.16.22.141	SSHv2	107	Server Protocol: SSH-2.0-OpenSSH_5.9p1 Debian-5ubuntu1.4\r
12	0.006592	172.16.22.141	172.16.22.142	SSHv2	107	Client Protocol: SSH-2.0-OpenSSH_5.9p1 Debian-5ubuntu1.4\r
14	0.006867	172.16.22.141	172.16.22.142	SSHv2	1338	Client: Key Exchange Init
16	0.007561	172.16.22.142	172.16.22.141	SSHv2	1050	Server: Key Exchange Init
17	0.008331	172.16.22.141	172.16.22.142	SSHv2	146	Client: Diffie-Hellman Key Exchange Init
18	0.010465	172.16.22.142	172.16.22.141	SSHv2	378	Server: New Keys

► Frame 18: 378 bytes on wire (3024 bits), 378 bytes captured (3024 bits)  
 ► Ethernet II, Src: Vmware\_f7:ce:9d (00:0c:29:f7:ce:9d), Dst: Vmware\_30:4d:05 (00:50:56:30:4d:05)  
 ► Internet Protocol Version 4, Src: 172.16.22.142 (172.16.22.142), Dst: 172.16.22.141 (172.16.22.141)  
 ► Transmission Control Protocol, Src Port: ssh (22), Dst Port: 35414 (35414), Seq: 1026, Ack: 1394, Len: 312  
 ▾ SSH Protocol  
 ▾ SSH Version 2 (encryption:aes128-ctr mac:hmac-md5 compression:none)  
 Packet Length: 292  
 Padding Length: 9  
 ▾ Key Exchange  
 Msg code: Diffie-Hellman Key Exchange Reply (31)  
 Multi Precision Integer Length: 104  
 DH modulus (P): 0000001365636473612d736861322d6e6973747032353600...  
 Multi Precision Integer Length: 65  
 DH base (G): 04ced594c826dd9f3bc1c57bde189c0e9b9874c7e5aa7aa9...  
 Payload: 000000640000001365636473612d736861322d6e6973747032353600...  
 Padding String: 00000000000000000000000000000000  
 ▾ SSH Version 2 (encryption:aes128-ctr mac:hmac-md5 compression:none)  
 Packet Length: 12  
 Padding Length: 10  
 ▾ Key Exchange  
 Msg code: New Keys (21)  
 Padding String: 00000000000000000000000000000000

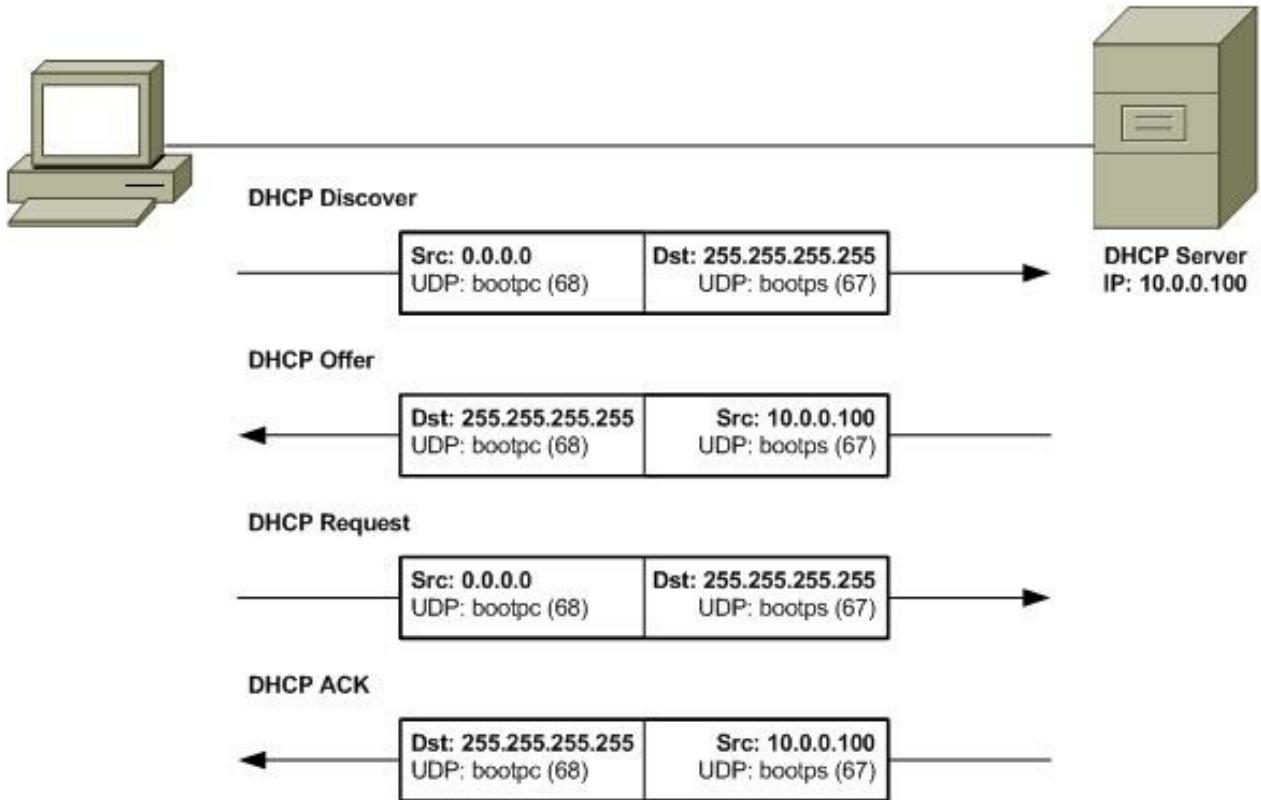
7)

No.	Time	Source	Destination	Protocol	Length	Info
10	0.006358	172.16.22.142	172.16.22.141	SSHv2	107	Server Protocol: SSH-2.0-OpenSSH_5.9p1 Debian-5ubuntu1.4\r
12	0.006592	172.16.22.141	172.16.22.142	SSHv2	107	Client Protocol: SSH-2.0-OpenSSH_5.9p1 Debian-5ubuntu1.4\r
14	0.006867	172.16.22.141	172.16.22.142	SSHv2	1338	Client: Key Exchange Init
16	0.007561	172.16.22.142	172.16.22.141	SSHv2	1050	Server: Key Exchange Init
17	0.008331	172.16.22.141	172.16.22.142	SSHv2	146	Client: Diffie-Hellman Key Exchange Init
18	0.010465	172.16.22.142	172.16.22.141	SSHv2	378	Server: New Keys
20	1.597953	172.16.22.141	172.16.22.142	SSHv2	82	Client: New Keys

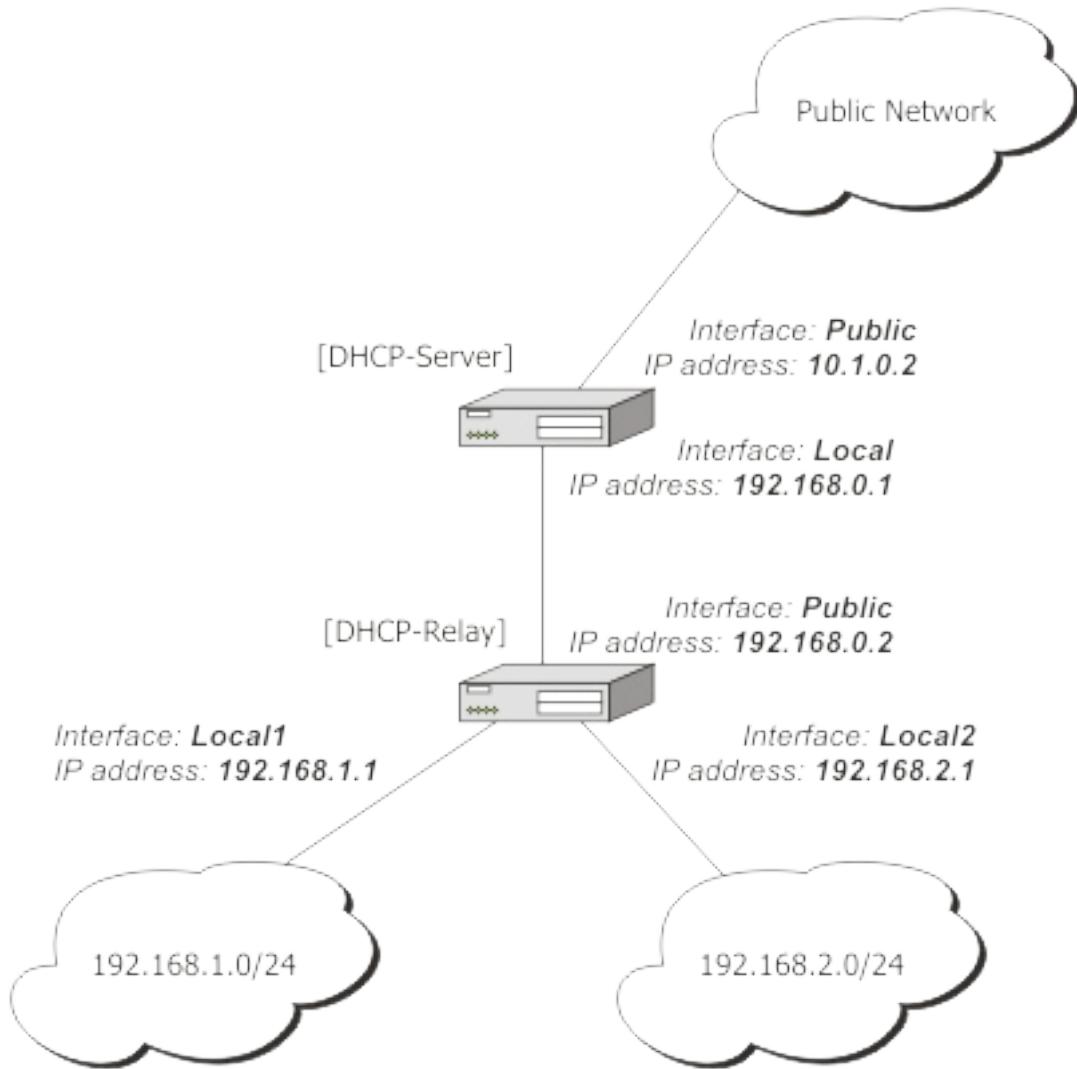
► Frame 20: 82 bytes on wire (656 bits), 82 bytes captured (656 bits)  
 ► Ethernet II, Src: Vmware\_30:4d:05 (00:50:56:30:4d:05), Dst: Vmware\_f7:ce:9d (00:0c:29:f7:ce:9d)  
 ► Internet Protocol Version 4, Src: 172.16.22.141 (172.16.22.141), Dst: 172.16.22.142 (172.16.22.142)  
 ► Transmission Control Protocol, Src Port: 35414 (35414), Dst Port: ssh (22), Seq: 1394, Ack: 1338, Len: 16  
 ▾ SSH Protocol  
 ▾ SSH Version 2 (encryption:aes128-ctr mac:hmac-md5 compression:none)  
 Packet Length: 12  
 Padding Length: 10  
 ▾ Key Exchange  
 Msg code: New Keys (21)  
 Padding String: 00000000000000000000000000000000

# DHCP

The Dynamic Host Configuration Protocol (*DHCP*) is a standardized networking protocol used on Internet Protocol (IP) networks for dynamically distributing network configuration parameters, such as IP addresses for interfaces and services. With DHCP, computers request IP addresses and networking parameters automatically from a DHCP server, reducing the need for a network administrator or a user to configure these settings manually.



Normally is necessary that DHCP is in the same network, but is possible to set this in external network using **DHCP relay** like proxy:



The different message for DHCP protocol are:

Message Type	Description
DHCPDiscover	The first time a DHCP client computer attempts to log on to the network, it requests IP address information from a DHCP server by broadcasting a DHCPDiscover packet. The source IP address in the packet is 0.0.0.0 because the client does not yet have an IP address. The message is either 342 or 576 bytes long—older versions of Windows use a longer message frame.
DHCPOffer	Each DHCP server that receives the client DHCPDiscover packet responds with a DHCPOffer packet containing an unleased IP address and additional TCP/IP configuration information, such as the subnet mask and default gateway. More than one DHCP server can respond with a DHCPOffer packet. The client will accept the first DHCPOffer packet it receives. The message is 342 bytes long.
DHCPRequest	When a DHCP client receives a DHCPOffer packet, it responds by broadcasting a DHCPRequest packet that contains the offered IP address, and shows acceptance of the offered IP address. The

message is either 342 or 576 bytes long, depending on the length of the corresponding DHCPDiscover message.

The selected DHCP server acknowledges the client DHCPRequest for the IP address by sending a DHCPAck packet. At this time the DHCPAcknowledge server also forwards any optional configuration parameters. Upon (DHCPAck) receipt of the DHCPAck, the client can participate on the TCP/IP network and complete its system startup. The message is 342 bytes long.

#### DHCPNak

If the IP address cannot be used by the client because it is no longer valid or is now used by another computer, the DHCP server responds with a DHCPNak packet, and the client must begin the lease process again. Whenever a DHCP server receives a request for an IP address that is invalid according to the scopes that it is configured with, it sends a DHCPNak message to the client.

#### DHCPDecline

If the DHCP client determines the offered configuration parameters are invalid, it sends a DHCPDecline packet to the server, and the client must begin the lease process again.

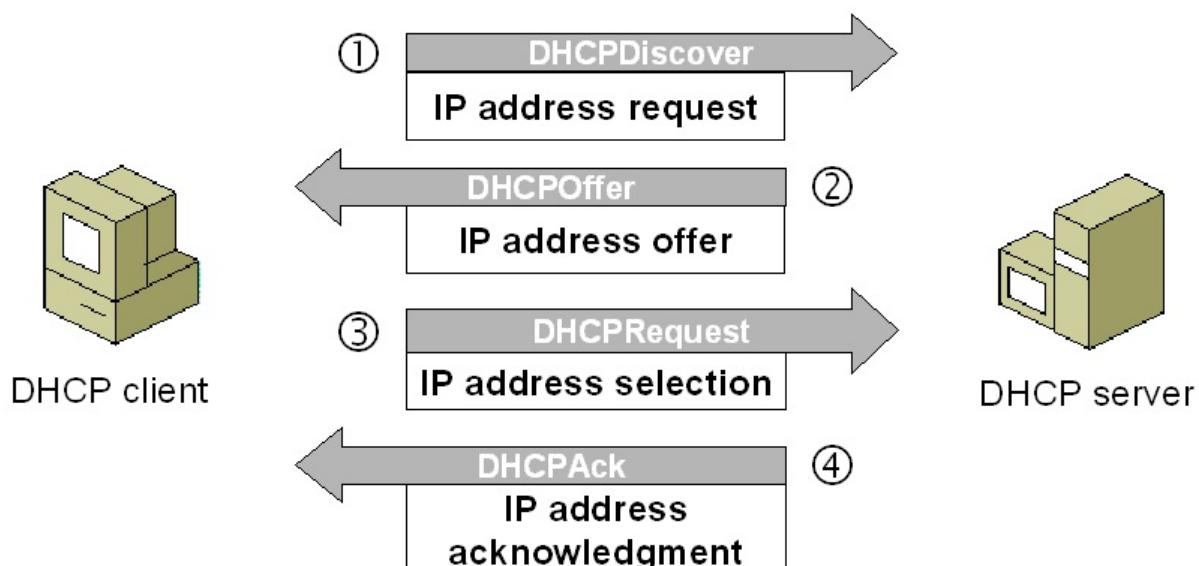
#### DHCPRelease

A DHCP client sends a DHCPRelease packet to the server to release the IP address and cancel any remaining lease.

#### DHCPIinform

DHCPIinform is a new DHCP message type, defined in RFC 2131, used by computers on the network to request and obtain information from a DHCP server for use in their local configuration. When this message type is used, the sender is already externally configured for its IP address on the network, which may or may not have been obtained using DHCP. This message type is not currently supported by the DHCP service provided in earlier versions of Windows NT Server and may not be recognized by third-party implementations of DHCP software.

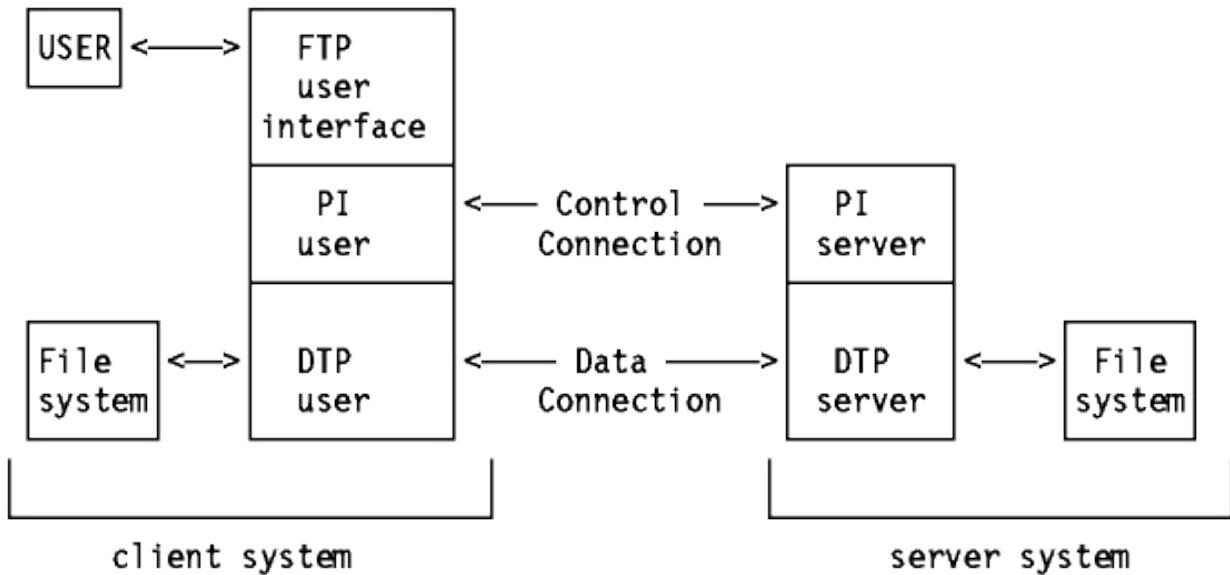
Simply process in one image:



# FTP

The **File Transfer Protocol (FTP)** is a standard network protocol used to transfer computer files from one host to another host over a TCP-based network, such as the Internet.

The FTP protocol is divided in two ways: **Protocol interpreter** and **data transfer process**.



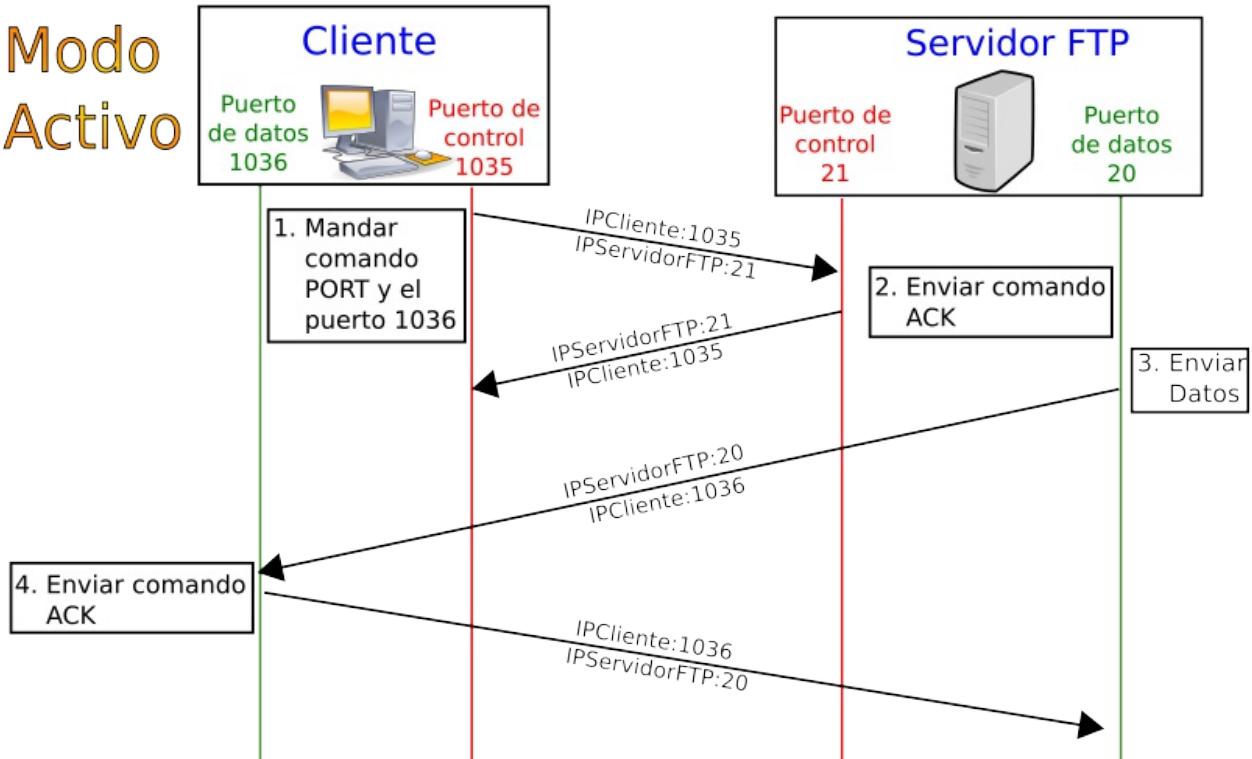
In the client, the ports for PI and DTP are not established. Can be any free port. In the server is typical use 21 and 20 port respectively. Aslo, FTP can work in two modes:

## FTP in Active mode

In active mode, the client have to open a port to communicate with the server.

- Client init connection with server PI and send command **PORT** to indicate the port that server need to init a TCP connection.
- Server create TCP connection and delegate in DTP to send the data.

# Modo Activo



This mode has two inconvenients:

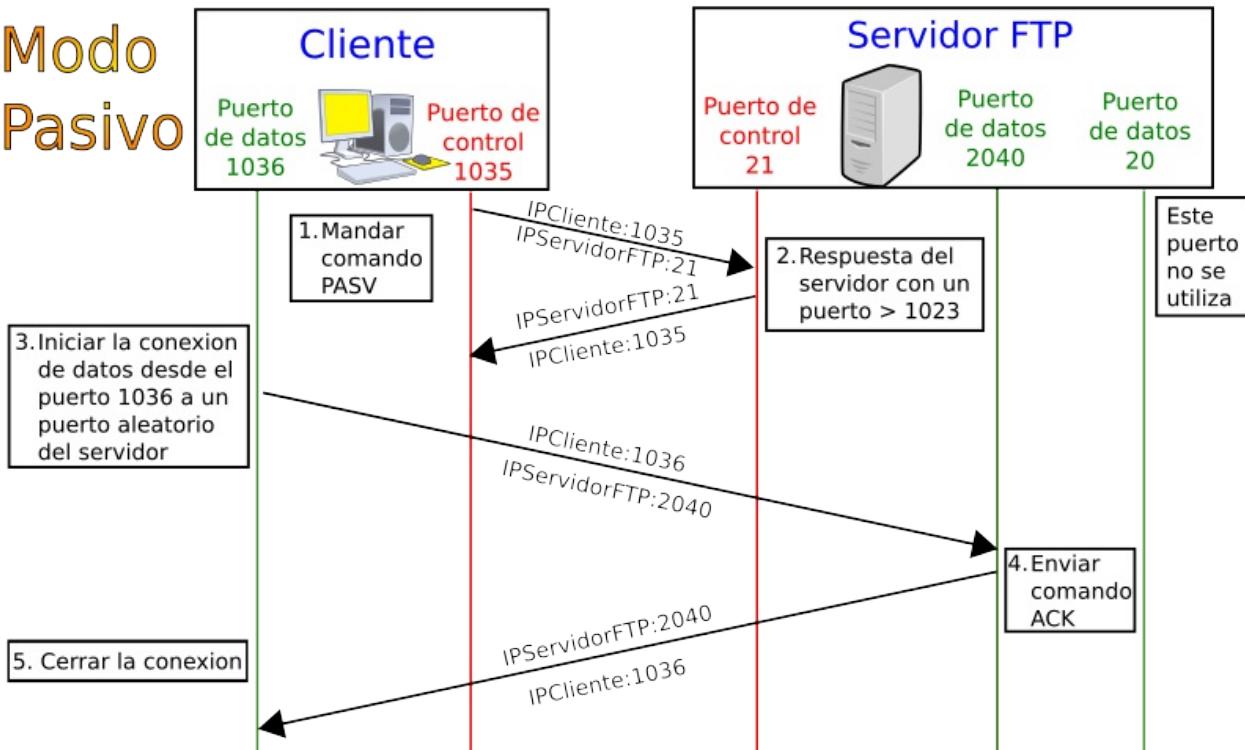
- The client connection is not secure. Client have to accept all traffic in the port of the data, and this is very dangerous.
- Don't work fine if client have a firewall.

## FTP in Passive mode

The idea is fix problem in active mode. In this case, the client negotiate the TCP connection with the server and say him that open a port for transfer data.

- The process is similar to Active mode but in the case client send the command **PASV** to indicate passive mode and the server response with the number of port that can be use. (>1024).
- The client init the TCP connection with the server port.

# Modo Pasivo



## Access control commands

### Command Description

USER	Character string allowing the user to be identified. User identification is necessary to establish communication over the data channel.
PASS	Character string specifying the user's password. This command must immediately follow the <i>USER</i> command. It fails to the client to hide the display of this command for security reasons.
ACCT	Character string representing the user's account. The command is generally required during the response accepting the password, if the response is 230 this stage is necessary, if the response is 332, it is.
CWD	<i>Change Working Directory</i> : this command enables the current directory to be changed. This command requires the directory's access path to be fulfilled as an argument.
CDUP	<i>Change to Parent Directory</i> : this command allows you to go back to the parent directory. This command was introduced to solve problems of naming the parent directory according to the standard (generally "..").
SMNT	<i>Structure Mount</i> :
REIN	<i>Reinitialize</i> :
QUIT	Command enabling the current session to be terminated. The server waits to finish any transfer in progress if the need arises, then supplies a response before closing the connection.

## Transfer parameter commands

### Command Description

PORT	Character string allowing the port number used to be specified.
PASV	Command making it possible to indicate to the DTP server to stand by for a connection on a specific port chosen randomly from among the available ports. The response to this command is the IP address of the machine and port.
TYPE	This command enables the type of format in which the data will be sent to be specified.
STRU	Telnet character specifying the file structure (F for <i>File</i> , R for <i>Record</i> , P for <i>Page</i> ).
MODE	Telnet character specifying data transfer method (S for <i>Stream</i> , B for <i>Block</i> , C for <i>Compressed</i> ).

# Índice

Introduction	1
Started	1
How to use	1
Test services	1
Examples	1
Understand	1
Files	1
Commands	1
DNS	11
Client	11
Server	11
Connectivity	11
How works	18
HTTP	23
Client	23
Server	23
Connectivity	23
How works	23
LDAP	32
Client	32
Server	36
Connectivity	36
How works	43
Mail	43
Client	43
Server	43
Connectivity	43
How works	43
SSL	43
Client	43
Server	49
Connectivity	49
How works	53
SSH	53
Client	53
Server	53
Connectivity	53
How works	53

DHCP	64
FTP	67