Lab Practice 5: NAT – Operation and Trace Analysis

Preliminary Reading: Kurose 7th edition, section 4.3.4 (pages 286–288)
Video: https://www.youtube.com/watch?v=hBloGXo2DlY

Pre-lab Work:

- Read and understand sections **1. Introduction** and **2. Network Address Translation**
- Solve **Exercise 1**

1. Introduction

In this lab practice we will explain how the **NAT (Network Address Translation)** mechanism works.

NAT was developed in response to the proliferation of small home and office networks with Internet connections. When we subscribe to a basic **ISP service**, we are provided with an Internet connection with a certain bandwidth (depending on the chosen plan) and a **single public IP address** to identify ourselves on the Internet.

This setup is sufficient if we only want to connect a **single computer** to the Internet. However, in the common case of having a **small local area network (LAN)** and wanting multiple computers within this network to access the Internet simultaneously, the ISP's standard service is not enough. More specifically, having only **one IP address** (or more generally, having fewer IP addresses than computers) creates a problem: not all computers on the network will be able to connect to the Internet simultaneously, as they lack an IP address to identify themselves.

One solution to this problem would be to **purchase a small range of IP addresses** for home or office use. However, this option is significantly more expensive than a single IP address and also requires managing a router, which is beyond the technical knowledge of most Internet users.

A **better option** is to continue using a **single IP address** and implement a mechanism that allows us to **share that address** among all the computers on the home or office network. This mechanism is known as **Network Address Translation (NAT)**.

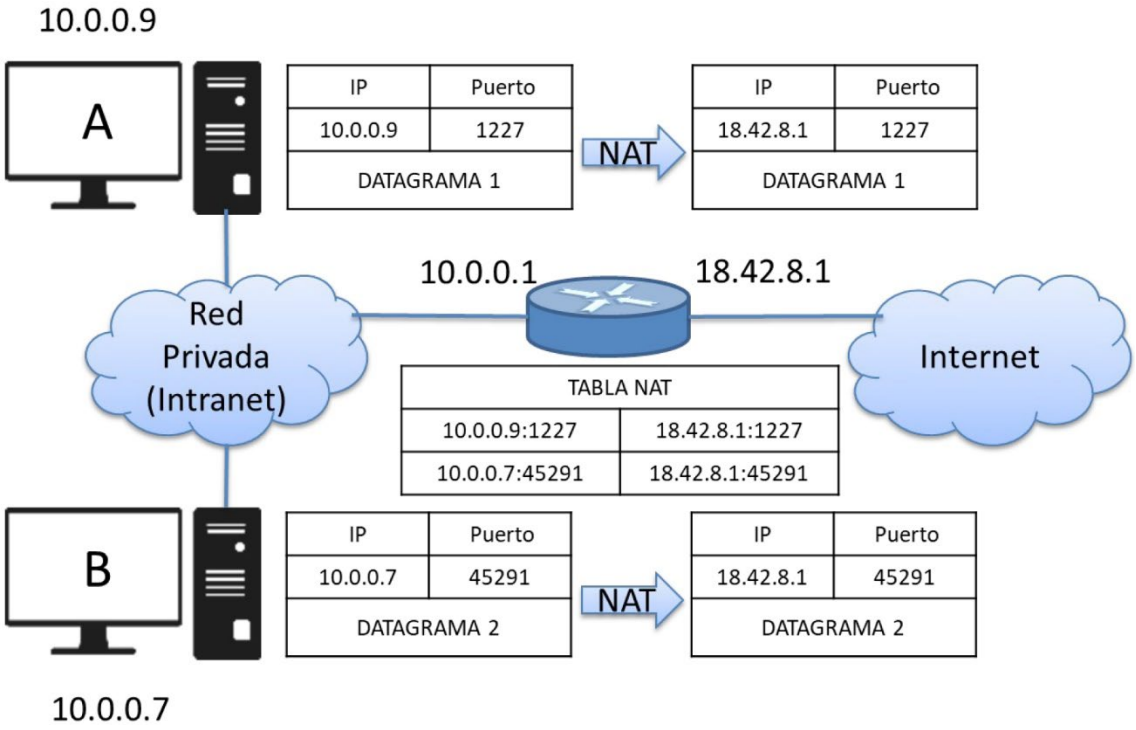
2. Network Address Translation

The **NAT mechanism** is typically used alongside **private IP addresses**, which are commonly found in home or office local networks (also known as **intranets**) that use NAT to access the Internet. However, the NAT mechanism itself is **independent of the type of IP addresses** used within the intranet.

In summary (**for more details, refer to section 4.3.4 of Kurose**), NAT works as follows:

As illustrated in the diagram below, when a computer from the **local network** (also known as a **private network**) accesses a **server on the Internet**, it sends the corresponding **datagram** to the **NAT device** (also known as a **NAT router**, although its functions extend beyond those of a typical router). The **NAT router** serves as the **gateway** for the local network.

This **NAT device modifies the source IP address** of the outgoing datagram— which is typically a **private IP address**—replacing it with the **public IP address** assigned by the **ISP**. Additionally, if necessary (**if the same port is already in use by another computer on the internal network**), the **NAT device also modifies the source port** of the **TCP segment** or **UDP datagram**, assigning it a new port number. This ensures that the **server's response** can be correctly forwarded to the host that initiated the request.



During the translation process, the **NAT device** stores an **entry in a table** (**NAT table** or **translation table**) that maps the **initial source port** to the **new source port** for each datagram passing through it. This allows the NAT device to correctly route incoming responses from the **Internet**.

When a **response** arrives, the NAT device **checks the destination port** of the response (**which corresponds to the modified source port of the original request**) against the entries in its **NAT table**. By finding the matching entry, it determines which **host in the local network** should receive the **server's response**.

During the **forwarding process** back into the **local network**, the **NAT device modifies the destination IP address and destination port** so that they match the **original values** from the initial request. The figure above illustrates an example of a **translation table**.

Exercise 1. The goal is to manually configure the **public network interface** of a **NAT device** with the following values:

Public IP Address:	158.42.180.1
Subnet Mask:	255.255.255.0
Gateway:	158.42.181.250
Primary DNS Server:	158.42.249.8
Secundary DNS Server:	158.42.1.8

After entering these values, the **NAT device** reports that the **gateway IP address is incorrect**. However, upon verifying the values, both the **gateway IP address** and the **public IP address assigned to the NAT** appear to be correct.

1. Because we are using a subnet mask of /24, which accepts IPs in the range 158.42.180.0 to 158.42.180.255, and the gateway address, 158.42.181.250, falls outside that range (so the gateway and public IP are in different subnets).

1. Why does the **NAT device** indicate that the **gateway IP address is incorrect**?
2. Which parameter should be changed to make the configuration correct? Suggest a value for this parameter that would ensure a correct setup.
3. Are the **DNS servers** on the same network as the **configured interface**? If they are not, would the configuration still work correctly? Justify your answer.

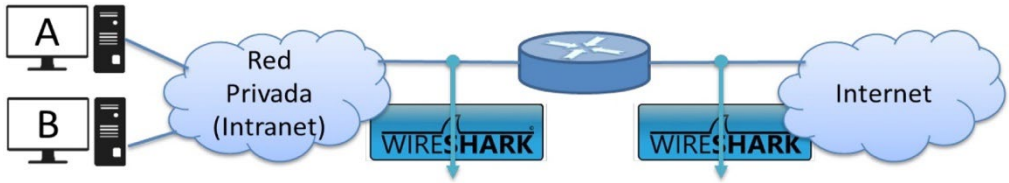
Set the subnet mask to 255.255.255.254 (or the gateway to 158.42.181.250)

No, they are outside the network. However, they will work correctly if they are reachable via routing (through the gateway), because NAT devices can resent DNS requests to external DNS servers.

3. Traffic Analysis

In this section, we will analyze the packets that pass through a **NAT device**. Instead of capturing the packets ourselves using **Wireshark**, we will use **pre-recorded captures** contained in the files: HTTP_LAN_1, HTTP_LAN_2, FTP_LAN_1, SSH_LAN_1, and their respective versions from the ISP's side. These files can be found on **PoliformaT**.

The reason for using pre-captured traffic is that, although we can easily capture the traffic generated by our computer in the **local network**, it is more complex to capture the traffic **leaving the NAT router**, since we normally don't have access to that network. The figure below shows the scenario we'll be working with.



Ejercicio	Captura Intranet	Captura Internet
Ejercicios 2, 3 y 4	HTTP_LAN_1	HTTP_ISP_1
Ejercicio 5	HTTP_LAN_2	HTTP_ISP_2
Ejercicio 6	FTP_LAN_1	FTP_ISP_1
Ejercicio 7	SSH_LAN_1	SSH_ISP_1

3.1 HTTP Traffic Analysis

Exercise 2. The file HTTP_LAN_1 contains a traffic capture generated within the **LAN (intranet)**. Open the file with **Wireshark** and answer the following questions:

1. How many **TCP connections** take place? 2 (one from 52467 and another from 39137)
2. Locate the datagram containing the first web request GET / HTTP/1.1. Note the **IP addresses** and **TCP port numbers** used. Naturally, source and destination will correspond to the **client** and **web server**, respectively.
   Source: 192.168.1.181:52467 Destination: 158.42.4.23:80
3. Locate the datagram containing the "HTTP 200 OK" response sent by the web server. Do the IP addresses and port numbers match those of the client and server found in the previous request? Yes, they match
4. Regarding the **second TCP connection**, do the same client and server participate? Note the **IP addresses** and **TCP port numbers** used.
   Source: 192.168.1.181:39137 Destination: 173.194.34.197:80

Exercise 3. Now we will focus on the **HTTP messages exchanged** between the **NAT router** and the **web server**, which is located in the **external network**. For this, open the file HTTP_ISP_1, which shows the traffic generated on the **external side of the NAT router** (Internet) as a result of the traffic from the **intranet** (previous capture). **Note:** The timestamps of the two captures are not synchronized.

Locate the outgoing message GET / HTTP/1.1. This corresponds to the NAT router **forwarding** the datagram originally generated in the intranet (from Exercise 2).

1. When is this message transmitted? Note the **IP addresses** and **TCP port numbers** used. Do the source and destination correspond to the **client** and **web server**, respectively? Source: 158.42.180.22:52467 Destination: 158.42.4.23:80Destination corresponds to the web server, but the origin is the IP of the NAT router
2. What is the **public IP address** of the NAT router? 158.42.180.22
3. Compared to the corresponding datagram in the intranet, has any field in the **HTTP message** changed? What about in the **IP datagram header**? For any fields that have been modified, explain the reason.
   The source address has been replaced by the public address of the NAT router.

Now locate the incoming message "HTTP 200 OK" sent by the web server.

4. When is the message received? Based on the IP addresses in the datagram, who is the intended **recipient** of the server's response? Justify your answer.
   The intended recipient is the public address of the NAT router, 158.42.180.22

Exercise 4. Using the information observed in the previous two exercises, complete the **NAT translation table** of the router, which records the mappings between **IP addresses** and **port numbers** on both sides of the router for the two TCP connections.

Connection	Entry from LAN		Exit to ISP network	
	Source IP	Source Port	Source IP	Source IP (port?)
1	192.168.18.181	52467	158.42.180.22	52467
2	192.168.18.181	39137	158.42.180.22	39137

Exercise 5. In the previous exercise, we observed that the NAT router kept the **same source port numbers** in the outgoing segments. This policy is just as valid as any other, as long as care is taken to **assign a new port number** if the one to be used is already in the translation table. That is exactly what we'll check in this exercise. We'll force the NAT router to **modify the source port number** because the required one is already in use. To do this, the following **Java program** was executed on **two computers** in the private network, and the traffic was captured with **Wireshark** for later analysis:

```
import java.net.*;
import java.io.*;

class ClienteTCP {
 public static void main(String args[]) throws Exception {
  String mi_IP = "192.168.1.2";
  InetAddress DirIP = InetAddress.getByName(mi_IP);
  Socket s = new Socket("www.redes.upv.es",80, DirIP, 40000);
  PrintWriter esc= new PrintWriter(s.getOutputStream(), true);
  esc.println("GET / HTTP/1.1");
  esc.println("Host: www.redes.upv.es");
  esc.println();
  while(true);
 }
}
```

The variable `mi_IP` contains the **IP address** of the computer where the program is executed. It opens a **TCP connection** with the `www.redes.upv.es` web server and keeps it open, since the server is configured to use **persistent connections**. Thus, when the same program is executed on a **second computer**, the NAT router sees that the **desired port** is already in use and is forced to use a **new available port**.

Open the file `HTTP_LAN_2`, which contains the traffic generated by one of the **LAN computers** running the program above.

1. What are the **IP addresses** and **port numbers** used by the client and server?

Now open the file `HTTP_ISP_2`. This file contains the traffic generated on the **public network (ISP)** by the two computers that executed the above program in the private network.

2. What **source port number** does the NAT router assign to the **outgoing segments** to the public network for each of the two connections?
3. What **headers** did the NAT router have to modify?
4. How can you determine which of the two TCP connections corresponds to the one captured in `HTTP_LAN_2`? *Hint: check the **IP packet identifier** field.*

Based on the information provided, complete the **NAT translation table** (you'll notice one of the cells lacks information):

Connection	Entry from LAN		Exit to ISP network	
	Source IP	Source Port	Source IP	Source Port
1		40000		
2				

3.2 The Application Layer and NAT

Based on the previous exercises, reflect on the layers at which NAT operates.

i) What fields does it modify?
ii) At which layers of the protocol stack?

You will have concluded that, so far, we've seen how NAT modifies fields at the network layer (source IP address) and, if necessary, the transport layer (source port number). This task is relatively simple for a NAT router, as it only has to access fixed positions in the IP datagram or TCP/UDP segment headers, and assign the public IP and a new port number if the original is already in use.

However, in some cases, additional problems arise that must be solved. Imagine an application-layer protocol that needs to send an IP address and/or port number identifying a client located behind a private IP. The NAT router, as we've described so far, cannot solve this issue alone. Thus, beyond the simple substitution of source IP addresses in IP headers or source ports in TCP/UDP segments, sometimes the NAT router must—if capable—analyse whether application-layer messages are exchanging IP addresses or ports and modify them accordingly.

For this to be possible, the NAT router needs the required knowledge to analyse and modify application-layer fields. It must be familiar with the full specification of protocols that might include such information in their messages, locate it, and modify it. This task is not as straightforward.

Let's explore a simple case using the **FTP (File Transfer Protocol)** to understand how this problem is handled. Other protocols, such as those used in video conferencing or real-time video streaming, face similar challenges. There are many techniques to address this. With FTP, we'll examine one of them.

3.2.1 FTP Traffic Analysis

FTP is a classic Internet service that allows files to be transmitted and received. It originated in the 1970s and, although used less frequently today, it still appears in common workflows in web environments. Despite its shortcomings—especially in security—and the existence of improved successors, it remains a good example for analysing NAT-related issues.

FTP uses a client-server architecture. A client A connects to a server B. The client can browse files shared by the server, download or upload files. It uses the TCP transport protocol and establishes **two connections** between client and server:

- A **control connection**, which remains open for the duration of the FTP session and is used to send commands (e.g., list directory contents, create a folder, upload or download a file).
- A **data connection**, which is opened and closed for each file transfer operation (either by the client or server depending on who is sending data).

For the server:

- The control connection uses port **21**.
- The data connection typically uses port **20**.

For the client:

- It uses **arbitrary ports** assigned by the operating system when creating a connection, as we've seen in previous socket programming exercises.

FTP supports two modes: **active mode** and **passive mode**.
For this practice, we focus on **active mode**.

In active mode:

- The client opens the control connection (to server port 21) from a random port.
- The **server** then opens the **data connection** from its port 20 to a port **specified by the client**.

How does the client specify this port?
It sends the **PORT** command via the control connection to server port 21, including its own IP and the port it opened to receive the data connection from server port 20.
Note that this **PORT command** is an application-layer message that contains an IP and port.

The figure below (referenced in the original document) illustrates this setup. Observe how the server opens the data connection from its port 20 to the client's IP and port as indicated by the PORT command.

Once the transfer is complete, this connection is closed. For a new transfer, the client must again send a PORT command with a new IP and port

.

The key idea here is that **it's no longer sufficient to simply change the source IP or source port in headers**—now, the NAT router must **analyse the application layer**, look for PORT commands, and if found, **rewrite** the IP and port values accordingly.

The next exercise helps us understand how NAT handles this, and which extra measures are needed. Refer to the FTP Annex for help with interpreting the captures and completing this exercise.

Exercise 6. Load the capture files FTP_LAN_1 and FTP_ISP_1 in Wireshark.

1. What are the IP addresses of the client and the server?
2. Pick either of the two files. Identify the **control** and **data** connections, and specify which ports the client and server use in each. For the data connection, was it established using active or passive mode?

   Tip: You can use Wireshark's display filter to view only one of the connections, e.g., filter by port number with tcp.port == PORT_NUMBER.

3. In the table below, indicate the order of the TCP segments related to: Connection setup (SYN), Data transfer (FTP Data), Connection teardown (FIN) for both control and data connections. *Ignore acknowledgment packets.*

Order	Action	Connection Type	*TCP Flag*
1	Connection start	Control	SYN
2			
3			
4			
5			

4. Look at the datagram generated in the private network at time 4.307125 and compare it with its corresponding one on the external network.
    a. What changes at the **application layer**?
    b. What do values **176** and **106** represent, and what decimal value do they correspond to?
5. Finally, think about this scenario:

    What if FTP encrypted the exchange of messages?

    Would the NAT scheme we analysed still work?
    (If needed, complete the next exercise before answering this question.)

3.3 Incoming Connections: Servers Within the Intranet

If we stick to the above procedure, when it comes to accessing a server inside the intranet **from the outside**, there will be **no entry** in the translation table. Therefore, the NAT router won't know what transformations to perform. So, we need to **"teach"** the NAT router what to do with incoming requests directed at internal servers. This requires two steps:

a) You must configure the NAT device to **accept requests targeting the server's port**. Moreover, when such a request arrives, the NAT device needs to know **which computer in the intranet** to forward the request to. This is known as **port forwarding**. This configuration must be done **before** the server can be accessed from the outside.

b) Since, in many cases, **intranet IP addresses are dynamically assigned** using a DHCP server built into the NAT router, you must ensure that the computer acting as a server always receives the **same IP address**. If not, when a request arrives for a specific port, the router will forward it to the IP address it has configured, but the server may no longer have that address.

Exercise 7. Regarding **port forwarding**, let's analyze packets arriving at a NAT device **from the outside** that are directed to an **SSH server** running on a computer inside the intranet. For this, use the capture files `SSH_LAN_1` and `SSH_ISP_1`.

1. What are the IP addresses of the SSH client and server, and which port numbers are used by each?
2. What differences do you observe between the datagrams and segments captured **inside** and **outside** the intranet?
   Are the contents of the **SSH messages** being modified, as in the FTP case?

4. Optional Exercises: Do Some Research

Once you've understood the basic operation of address translation, we invite you to dig a bit deeper into **different types of NAT** and **mechanisms for solving the issues** we've encountered in this lab:

1. Search for information (or ask your favourite AI!) about the different types of NAT:
   - **Full Cone NAT**
   - **Restricted Cone NAT**
   - **Port Restricted Cone NAT**
   - **Symmetric NAT**

   Do you understand the differences?

2. What is a **STUN server** (*Session Traversal Utilities for NAT*)?
   What problems does it solve?
3. What is a **TURN server** (*Traversal Using Relays around NAT*)?
   What problems does it solve?

5. Optional Exercises (II): Investigate a Bit More

Due to the **exhaustion of IPv4 address space**, some ISPs (those who acquired fewer IP addresses) have implemented **CGNAT** (*Carrier-Grade NAT*). This means that **multiple clients share the same public IP address** on the Internet! After all these months of coursework, this may sound like heresy. How can different clients have the same IP address?

In reality, it's just applying the same NAT principles **within the ISP's own infrastructure**. It's as if the ISP's internal network were **another private network**. When a packet leaves the ISP toward the Internet, it is assigned a public IP that is **shared by multiple clients**, but each connection is distinguished using the **port numbers**, as we've seen in this lab.

Do some research about CGNAT. You'll find that there is a reserved IP range, which can be considered **another set of private addresses**: `100.64.0.0/10` (**Carrier-Grade NAT Range**), defined in 2012. This address range is used **within the ISP's own network**, which, thanks to its `/10` prefix, can serve a huge number of clients.

Also keep in mind that **multiple ISPs** may use this same address space in their internal networks. When traffic exits toward the public Internet, the CGNAT IP is replaced with a **public IP**, and the mapping is logged so that return traffic can be properly routed to the original client.

For example, a home user might have a private address like `192.168.1.10`. When leaving their home router and entering the ISP's infrastructure, their gateway assigns them an address like `100.64.12.34` (CGNAT space). Then, when accessing the public Internet, the ISP translates this to a **public IP** (e.g., `203.0.113.5`) and a **port** (e.g., `60001`). If multiple clients are assigned the same public IP, **port numbers are used to differentiate them**.

Device	IP before NAT	IP after NAT
Home PC	192.168.1.10	192.168.1.10:54321 → 100.64.12.34:54321
ISP Router	100.64.12.34:54321	203.0.113.5:60001

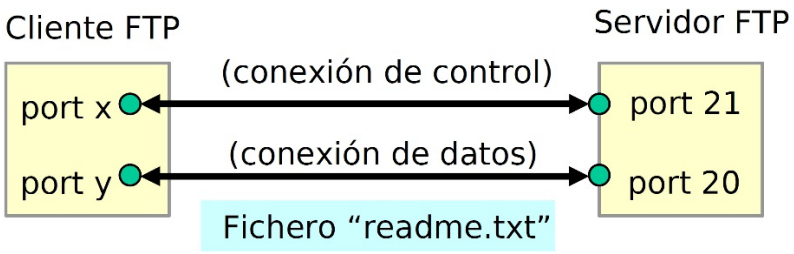
1. Look up information (or ask your favourite AI) about **CGNAT**.
2. Which **ISPs in Spain** use CGNAT?
3. What are the **advantages and disadvantages** of CGNAT?
4. Would you choose an ISP that uses CGNAT? Why or why not?
5. If **multiple users share the same public IP**, could it lead to **false assumptions** about what you're doing online? For instance, could someone wrongly conclude that *you* did something online, when it was actually your **neighbor** sharing the same IP?

FTP APPENDIX

FTP begins with a TCP connection to port 21 on the server, known as the control connection (this occurs when the user runs command `ftp server` or enters the URL `ftp://server` in a browser). The control connection lasts for the entire session, and it is used by both client and server to exchange commands and control codes, respectively.

When a file is to be transferred (using the `RETR`, `STOR` commands) or a directory listing is requested (`LIST`), the FTP protocol will open a new connection known as the data connection, through which the transfer will take place. This connection lasts only as long as needed to transfer the data, and then it is closed. If another transfer is needed later, FTP will open a new data connection.

FTP offers two mechanisms for establishing the data connection: **active mode** and **passive mode**. Each FTP client can use either mode, although it is generally recommended to use passive mode when the client is behind a firewall or NAT, and the FTP server is outside of that network.



Active Mode

In this mode, contrary to the typical client-server model, it is the **server** that opens the data connection to the port previously indicated by the client.

Thus, the client sends the `PORT n1,n2,n3,n4,n5,n6` command of the protocol, indicating to the server that it is waiting for the server to establish a data connection to IP address `n1.n2.n3.n4` (its own IP) and port `n5n6`, with `n5` and `n6` being the high and low bytes, respectively, of the port number (therefore, the port number can be calculated as `n5*256 + n6`; for example, to specify port 1025, `n5 = 4` and `n6 = 1`).

Upon receiving this information, the server initiates the connection from its port 20 to the client's port as indicated in the `PORT` command. Then, the client specifies the data transfer operation it wants to perform (`LIST`, `RETR`, or `STOR`), and the server (or the client, since transfers can occur in both directions) sends the requested data over the newly established data connection.

Passive Mode

The active mode solution is problematic for clients behind a firewall or NAT, as these devices typically block connections initiated from outside the network. In such cases, it is necessary to use a mechanism where the client initiates the data connection (just as it does with the control connection).

In this mode, the server plays its usual passive role in establishing connections. For the client to initiate the data connection, the server must inform it which port it will be listening on (usually port 20). Once the connection is established, the requested file will be transferred through it.

FTP Protocol Commands

Command	**Description**
`USER Username`	User credential for the server
`PASS Password`	User credential for the server
`CWD DirectoryName`	Change remote directory
`LIST [FileList]`	List information about files or directories
`MKD DirectoryName`	Create a remote directory
`PWD`	Display the current remote directory name
`PORT n1,n2,n3,n4,n5,n6`	Client IP address (n1.n2.n3.n4) and port number (n5*256+n6) for data connection
`PASV`	Use passive mode
`RETR FileName`	Request a file from the server
`STOR FileName`	Upload a file to the server
`SYST`	Return the server's operating system type
`TYPE Type`	Set transfer type (A = ASCII, I = binary)
`QUIT`	Disconnect (terminate the control connection)