salesians SARRIÀ

Sòcols i serveis

Protocols i comunicacions

Josep Gutiérrez

Departament d'informàtica
Salesians de Sarrià

Comunicació per xarxa

- Dos ordinadors que estan connectats en xarxa poden compartir informació transmetent-la a través de la xarxa. Per fer-ho, un requisit obligatori és que ambdós ordinadors es posin d'acord en la forma de transmissió. Aquesta forma de transmissió s'anomena **Protocol**.
- Una forma d'aconseguir que dos programes es transmetin dades és la programació de **sockets**. Un **socket** no es res més que un "canal de comunicació" entre dos programes que corren sobre ordinadors diferents o fins i tot en el mateix ordinador.
- Des del punt de vista de la programació, un socket és un fitxer que s'obre d'una manera especial i en el que es pot escriure i llegir dades
- Existeixen bàsicament dos tipus de "canals de comunicació" o sockets, els orientats a connexió i els no orientats a connexió.

Tipus de Sockets

Socket orientat a la connexió

TCP/IP

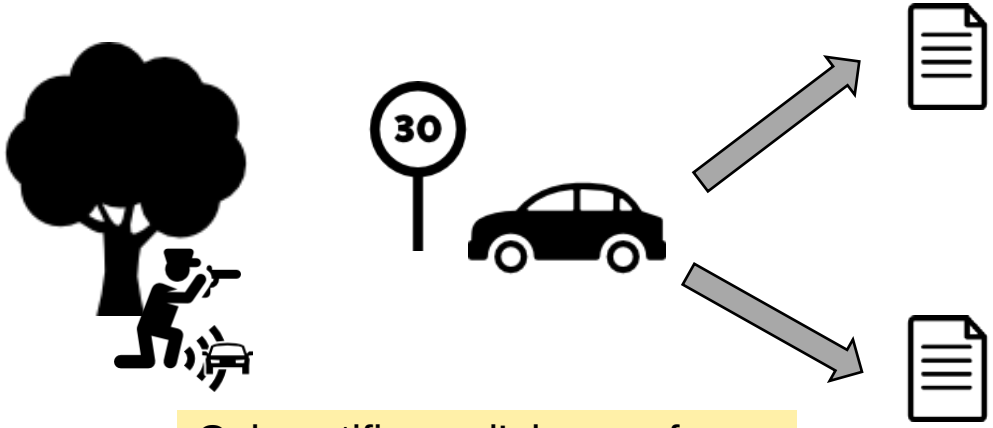
Ambdós programes s'han de connectar entre ells amb un socket i fins que no estigui establerta correctament la connexió, cap dels dos pot transmetre dades. Garanteix que totes les dades van d'un programa a l'altre correctament. S'utilitza quan la informació a transmetre és important i no es pot perdre cap dada.

Socket no orientat a la connexió

UDP

No es necessari que els programes es connectin. Qualsevol d'ells pot transmetre dades en qualsevol moment, independentment de que l'altre programa estigui "escoltant" o no. Garanteix que les dades que arriben són correctes, però no que arribin totes. S'utilitza quan és molt important que el programa no es quedi bloquejat i no importa que es perdin dades.

Tipus de Sockets



Cal notificar d'alguna forma legal al conductor que ha comés una infracció

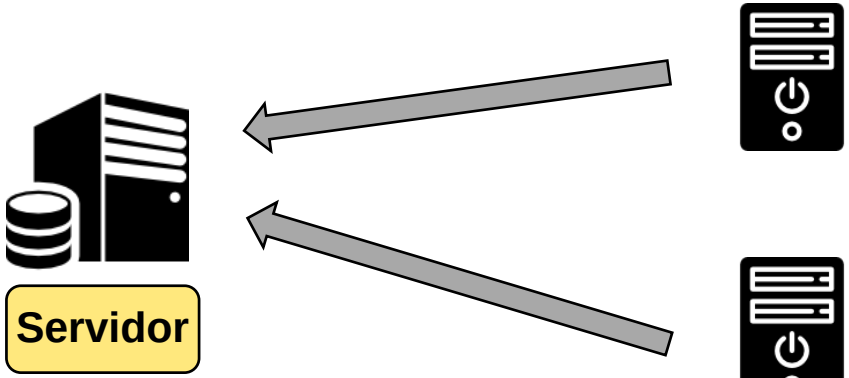
Es pot notificar la multa amb una carta certificada amb justificant de recepció

TCP/IP

Es pot notificar la multa amb un anunci al Diari Oficial de la Generalitat DOG

UDP

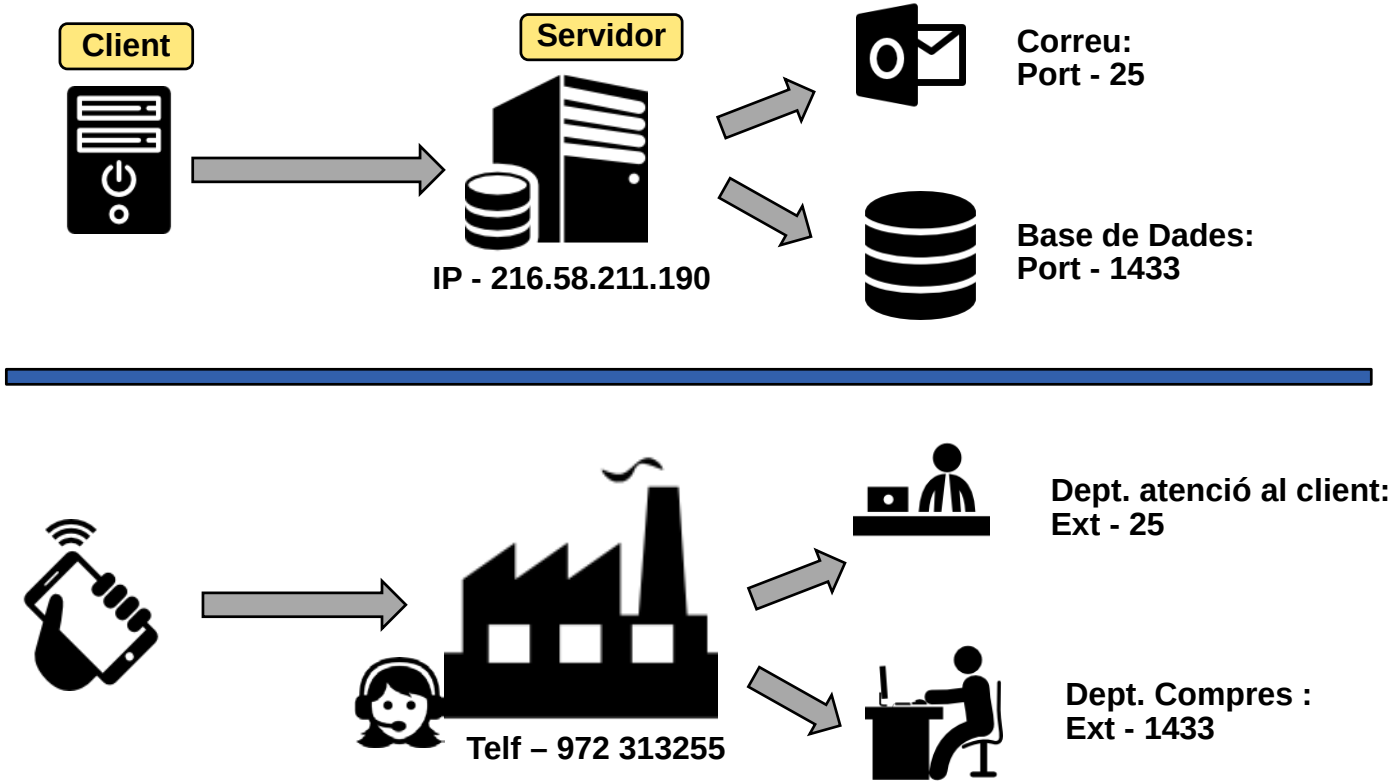
Client-Servidor



Equips que donen el primer pas. Quan arrenca, o quan es necessita, intenta connectar-se per sol·licitar informació o un servei.

Equip que ha d'estar arrancat i en espera de que algun altre vulgui connectar-se a ell. Mai dona "el primer pas" en la connexió. Proporciona serveis i dades a la resta d'equips.

IP Adress i Port



IP Adress i IPEndPoint

- La classe **IPAddress** proporciona una direcció del protocol d'internet (IP):
- Cal remarcar que la IP no és un string i que cal parsejar una IP introduïda en una casella de text utilitzant el mètode **IPAdress.Parse**

```
IPAddress address = IPAddress.Parse(txtIP.text);
```

- Un altre forma d'indicar un punt de connexió és a partir de la classe **IPEndPoint.**
- El seu constructor rep una **IPAdress** i un int32 que representa el port.

```
IPEndPoint endpoint = new IPEndPoint(address, port);
```

La classe Dns

- La classe **Dns** proporciona funcionalitats de resolució de noms de domini.

GetHostAddresses	Retorna la IP del host especificat.
GetHostByAddresses	Crea un **IPHostEntry** d'una IP determinada
GetHostByName	Obté la informació DNS per al Host especificat.
GetHostEntry	Resol una IP o un Host name en una **IPHostEntry**
GetHostName	Obté el host name de l'equip local.
Resolve	Resol el host name o la IP d'una **IPHostEntry**

- La classe **IPHostEntry** associa un nom d'amfitrió del sistema de noms de domini (DNS) amb una matriu d'àlies i una matriu d'adreces IP concordants.

```
IPHostEntry hostInfo = Dns.GetHostEntry("www.starwars.com");
```

Informació de xarxa

- El Namespace **System.Net.NetworkInformation** proporciona accés a dades de trànsit de la xarxa, informació d'adreça de xarxa i notificació de canvis d'adreça per a l'equip local.
- Aquest Namespace presenta algunes classes molt útils:

NetworkChange	Permet que les aplicacions rebin notificacions quan la IP d'una interfície de xarxa canvia.
NetworkInterface	Proporciona configuració i informació estadística per a una interfície de xarxa.
Ping	Permet que una aplicació determini si un equip remot està accessible a través de la xarxa.

Verificació de xarxa

- Si cal comprovar si hi ha una connexió de xarxa disponible, es pot utilitzar el mètode **GetIsNetworkAvailable** com es mostra a continuació.

```
Boolean xarxaDisponible;
xarxaDisponible = NetworkInterface.GetIsNetworkAvailable();
```

- Si volem controlar els canvis en l'estat de connexió a la xarxa, podem utilitzar l'esdeveniment **NetworkAvailabilityChanged** i verificar si **IsAvailable** retorna **true** (tenim connexió) o **false**.

```
tenimXarxa = NetworkInterface.GetIsNetworkAvailable();

void NetworkChange_NetworkAvailabilityChanged
     (object sender, NetworkAvailabilityEventArgs e)
{
    tenimXarxa = e.IsAvailable;
}
```

Ping

- **Ping** és una utilitat que permet fer una primera comprovació de si un equip està en xarxa i està accesible des de la màquina que executa el ping.
- Cal tenir en comptes que algunes configuracions de xarxa i/o Firewalls poden afectar al funcionament de ping. A part d'això **ping** només pot comprovar que l'equip és accessible però no pot comprovar si un determinat servei de nivell superior (per exemple un gestor de base de dades o un servidor de correu) està disponible o no.

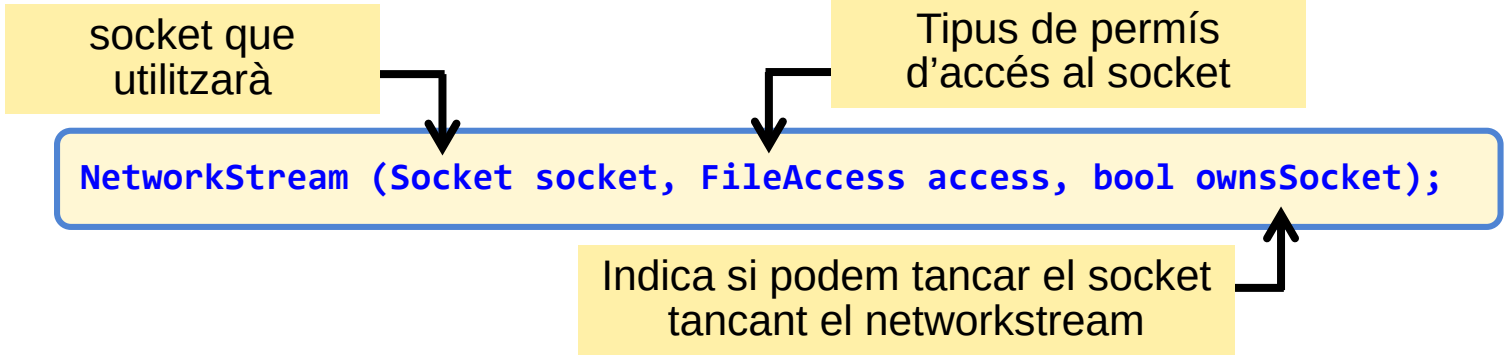
```
Ping myPing = new Ping();
bool responPing = false;
PingReply reply = myPing.Send("192.168.1.3", 1000);
if (reply != null)
 {
  Console.WriteLine("Status :  " + reply.Status +
        " \n Time : " + reply.RoundtripTime.ToString() +
        " \n Address : " + reply.Address);
pingable = reply.Status == IPStatus.Success;
 }
```

Protocols de xarxa i dades

- La majoria de protocols de xarxa utilitzen **networkstreams** per enviar dades a través de la xarxa.
- Un **networkstream** és un tipus especial de stream o flux de dades que presenta mètodes i propietats optimitzats per a la seva transmissió per xarxa.
- Un **stream** és un objecte que representa una seqüència de bytes. Si tenim en compte que qualsevol tipus de dada es pot emmagatzemar com una seqüència de bytes, podem entendre que utilitzant **streams** podem fer abstracció de la lectura i escriptura de dades
- Un concepte important en qualsevol protocol és la **finestra de recepció**. La mida de la finestra de recepció és la quantitat de dades rebudes (en bytes) que poden cabre en la memòria intermèdia de recepció durant la connexió.

NetworkStream

- La classe **NetworkStream** proporciona mètodes per enviar i rebre dades en format **Stream** utilitzant **Sockets**.
- Per defecte, tancar el **NetworkStream** no tanca el **Socket** proporcionat. Si cal que **NetworkStream** pugui tancar el **Socket** proporcionat, s'ha d'especificar que el valor de la propietat **ownsSocket** és **true**.
- Per crear un **NetworkStream** cal proporcionar un socket connectat i opcionalment especificar el permís de **FileAccess** que **NetworkStream** té sobre el **Socket** proporcionat.
- El seu constructor més complert inclou:



NetworkStream

- Així per instanciar i crear un **NetworkStream** caldria fer:

```
mySocket.Connect(myIpEndPoint);
NetworkStream ns;

ns= new NetworkStream(mySocket, FileAccess.ReadWrite, true);
```

- També es pot crear un **NetworkStream** a partir del mètode **GetStream** de la classe **Socket,** que retorna precisament el **NetworkStream** que farà servir per a llegir i escriure dades.

```
TcpClient tcpClient = new TcpClient();
NetworkStream netStream = tcpClient.GetStream();
```

- La classe **NetworkStream** presenta suport per a tasques asíncrones i per tant presenta mètodes amb el sufix **Async** per indicar que realitzen la seva funció de forma asíncrona.

Classe NetworkStream

- Les propietats més rellevants són:

CanRead	Obté un valor que indica si NetworkStream admet la lectura.
CanWrite	Obté un valor que indica si NetworkStream admet l'escriptura.
DataAvailable	Obté un valor que indica si hi ha dades disponibles per a la seva lectura.
Socket	Obté el Socket associat.

- Els mètodes més rellevants són:

Close	Tanca el stream actual i allibera tots els recursos
Close(int32)	Tanca NetworkStream després d'esperar el temps especificat per permetre que s'enviïn les dades.
CopyTo(Stream)	Llegeix els bytes del stream actual i els escriu en un altre stream de destinació.
Read	Llegeix dades de NetworkStream
Write	Escriu dades en NetworkStream

Exemple NetworkStream

```
if(ns.CanRead)
{
  byte[] buffer = new byte[1024]; //estableixo el buffer
  StringBuilder missatge = new StringBuilder();
  int numberOfBytesRead = 0;
  // gestiono el fet que el missatge pot ser més gran que el buffer
  do{
     numberOfBytesRead = ns.Read(myReadBuffer, 0, buffer.Length);
     missatge.AppendFormat("{0}",
     Encoding.ASCII.GetString(buffer, 0, numberOfBytesRead));

     }
   while(ns.DataAvailable); //vaig llegint mentre no s'acabi el contingut
   Console.WriteLine("Has rebut: " + missatge);
 }
 else
 {
  Console.WriteLine("Ho sento. No puc llegir aquest stream.");
 }
```