Servicios de Internet

Area de Ingeniería Telemática
http://www.tlm.unavarra.es

Arquitectura de Redes, Sistemas y Servicios
3º Ingeniería de Telecomunicación

Temario

1. Introducción
2. Arquitecturas, protocolos y estándares
3. Conmutación de paquetes
4. Conmutación de circuitos
5. Tecnologías
6. Control de acceso al medio en redes de área local
7. Servicios de Internet

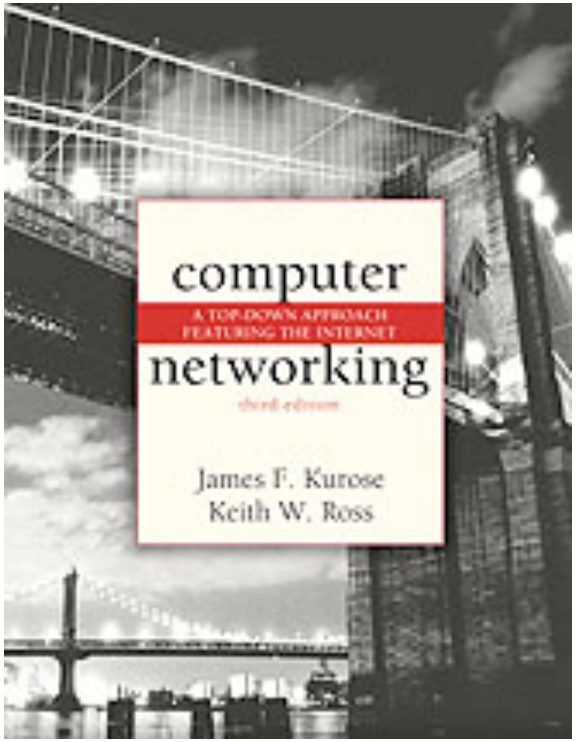
Temario

1. Introducción
2. Arquitecturas, protocolos y estándares
3. Conmutación de paquetes
4. Conmutación de circuitos
5. Tecnologías
6. Control de acceso al medio en redes de área local

- Servicios de Internet
  - La Web
  - DNS
  - E-Mail.
  - FTP. Telnet
  - Otros
  - **Desarrollo de clientes y servidores**

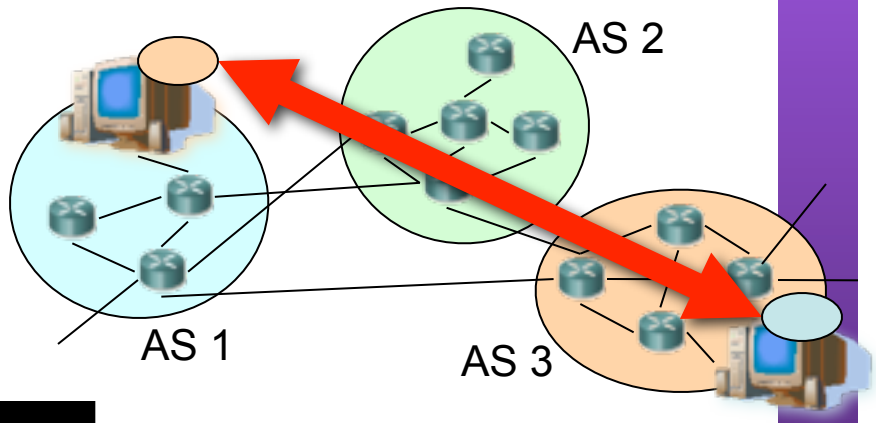
Material

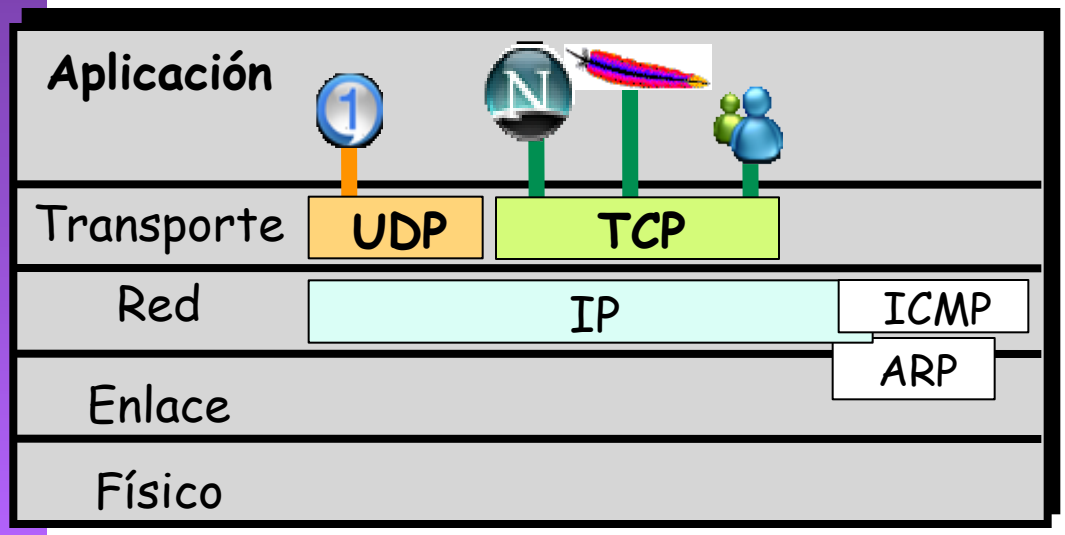
Del Capitulo 2 de
Kurose & Ross,
"**Computer Networking a top-down approach featuring the Internet**"
Addison Wesley



Las aplicaciones

- Son software
- Diferentes máquinas y Sistemas Operativos
- Quienes se comunican son **procesos**

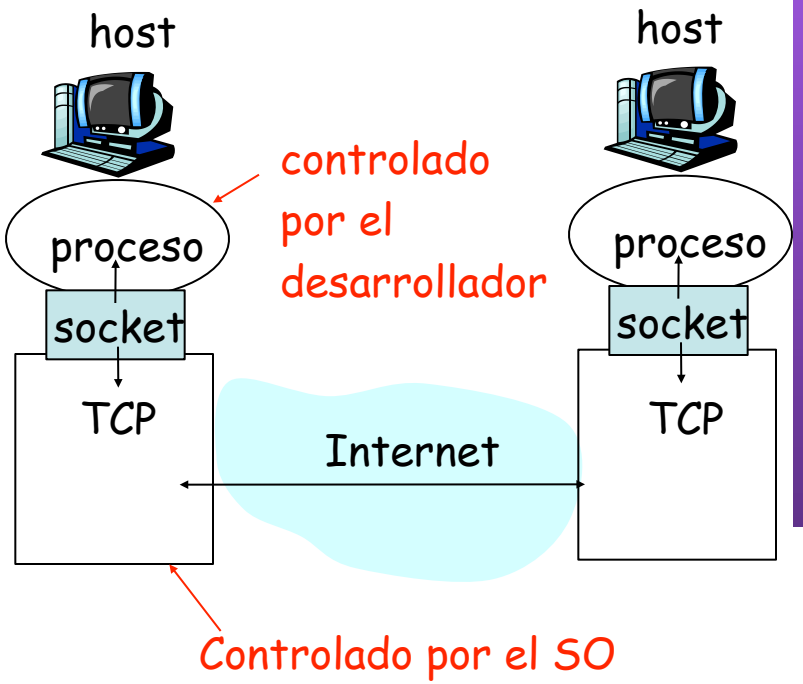




- Intercambian mensajes
- Emplean **Protocolos de nivel de aplicación**

Sockets

- Los procesos envían y reciben mensajes a través de un ***socket***
- Delega en el nivel de transporte para que haga llegar los mensajes al otro socket
- Acceso a través de un **API**
- Puede escoger el protocolo de transporte
- Puede configurar algunos parámetros del mismo
- No controla cómo se comporta



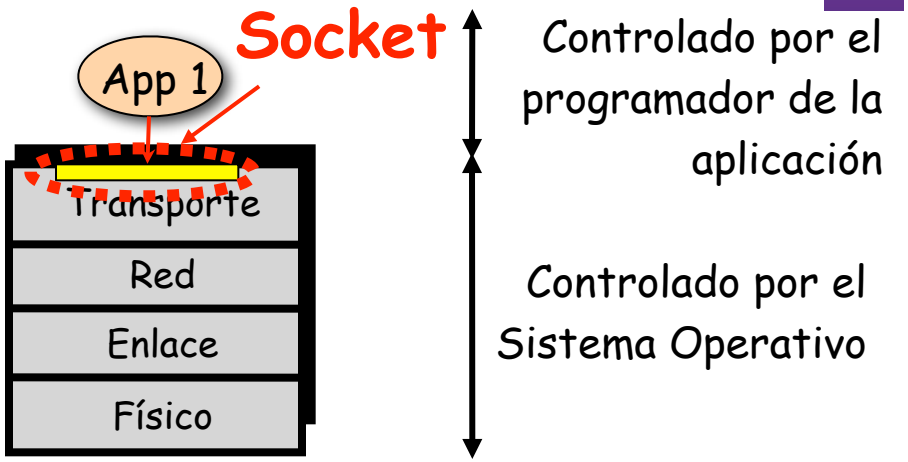
Programación con Sockets

API de Sockets

- Introducida en el UNIX BSD4.2 en 1983
- Centrada en el paradigma cliente/servidor
- Ofrece dos tipos de servicios de transporte:
  - STREAM: flujo de datos fiable orientado a conexión
  - DGRAM: datagramas

Socket

- Creado por la aplicación
- Controlado por el S.O.
- A través suya la aplicación envía y recibe mensajes



Que podemos hacer con sockets?

- Un programa puede construir uno o mas sockets
- Puedo construir sockets de tipo STREAM (TCP) o DGRAM (UDP)
- En los sockets UDP puedo..
  - elegir en que puerto quiero estar (el socket)
  - enviar mensajes individuales
  - recibir los mensajes que me envian otros
  - las dos cosas a la vez?
- En los sockets TCP puedo...
  - elegir en que puerto quiero estar
  - establecer conexiones con otros sockets
  - esperar y aceptar conexiones de otros sockets
  - enviar datos por las conexiones
  - recibir datos por las conexiones
- Extras
  - resolver nombres
  - manejar direcciones...

API

socket()

SOCK_STREAM
SOCK_DGRAM

bind()

sendto()

recvfrom()

bind()

connect()

listen()accept()

read() recv()
write() send()

gethostbyname()

...

Ejemplo con UDP

Servidor UDP escuchando en un puerto e imprime lo que recibe

```
void escucha_para_siempre_en_el_puerto(int puerto) {
    int s, err;
    struct sockaddr_in midireccion;

    s=socket(PF_INET,SOCK_DGRAM,0);

    midireccion.sin_family=AF_INET;
    midireccion.sin_port=htons(puerto);
    midireccion.sin_addr.s_addr=INADDR_ANY;
    err=bind(s, (struct sockaddr *)&midireccion, sizeof(midireccion));
    if (err!=0) {
        printf("no puedo coger el puerto\n");
        exit(-1);
    }

    while (1) {
        struct sockaddr_in origendir;   int origendirlen=sizeof(origendir);
        char buf[5000];                 int recibidos;

        recibidos=recvfrom(s,buf,5000,0,(struct sockaddr *)&origendir, &origendirlen);

        printf("Mensaje recibido de direccion %s puerto %u\n",
                    inet_ntoa(origendir.sin_addr),ntohs(origendir.sin_port));
        buf[recibidos]=0;    printf("[%s]\n",buf);
    }
}
```

Ejemplo con UDP

El mismo ejemplo en otro lenguaje

```
#!/usr/bin/env python

from socket import *
from sys import argv,exit



def escucha_para_siempre_en_el_puerto(puerto):
    s=socket(AF_INET,SOCK_DGRAM)
    try:
        s.bind(('',puerto))
    except:
        print('no puedo coger el puerto')
        exit(-1)

    while (True):
        (recibido,origen)=s.recvfrom(5000)
        print('Mensaje recibido de direccion %s puerto %u' % (origen))
        print('[%s]' % recibido)



escucha_para_siempre_en_el_puerto(int(argv[1]))
```

Ejemplo cliente UDP

Usa un socket UDP para enviar un mensaje simple

Envia este mensaje al servidor server.unavarra.es al puerto 6000

```
$ envia “hola en UDP” server.unavarra.es 6000
enviando a server.unavarra.es puerto 6000
```

```
int main(int argc, char *argv[]) {
    int puerto;
    char *mensaje=argv[1];
    char *servidor=argv[2];
    sscanf(argv[3],"%u",&puerto); // lee el puerto

    printf("enviando a %s puerto %u\n",servidor,puerto);

    envia_mensaje_a_servidor_y_puerto(mensaje,servidor,puerto);

}
```

```
envia_mensaje_a_servidor_y_puerto(“hola en UDP”,”server.unavarra.es”,6000);
```

Ejemplo cliente UDP

Usa un socket UDP para enviar un mensaje simple

```
$ envia “hola en UDP” server.unavarra.es 6000
enviando a server.unavarra.es puerto 6000
ip=130.206.169.123
```

```
void envia_mensaje_a_servidor_y_puerto(char *mensaje, char *nombre,int puerto) {
    int s;
    struct sockaddr_in direccion;
    struct hostent *dnsresult;

    s=socket(PF_INET,SOCK_DGRAM,0);

    dnsresult=gethostbyname(nombre);
    printf("ip=%s\n",inet_ntoa(*(struct in_addr *)(dnsresult->h_addr)));

    direccion.sin_family=AF_INET;
    direccion.sin_port=htons(puerto);
    direccion.sin_addr=*(struct in_addr *)(dnsresult->h_addr);

    sendto(s,mensaje,strlen(mensaje),0,(struct sockaddr *)&direccion, sizeof(direccion));

}
```

Ejemplo con TCP

Construyendo un cliente de web

```
$ getweb http://www.google.com
servidor: www.google.com
puerto: 80
path: /

recibido 598 bytes
```

```python
#!/usr/bin/env python

from socket import *
from sys import argv,exit


url=argv[1]
url=url.split('//')
url=url[1].split('/',1)
servidor=url[0]
if len(url)>1 :
    path=url[1]
else:
    path='/'

print('servidor: %s\npuerto: %u\npath: %s\n'%(servidor,80,path))

pagina=consigue_pagina_web(servidor,80,path)

print('recibido %u bytes' % len(pagina))
```

Ejemplo con TCP

Construyendo un cliente de web

```
def consigue_pagina_web(servidor,puerto,path):
    s=socket(AF_INET,SOCK_STREAM)
    ip=gethostbyname(servidor)
    dir=(ip,puerto)
    try:
        s.connect(dir)
    except:
        return None

    peticion='GET '+path+' HTTP/1.0\n\n'
    s.send(peticion)

    pagina=''
    fin=False
    while  not fin :
        trozo=s.recv(5000)
        if (not trozo==''):
            pagina+=trozo
        else:
            fin=True

    return pagina
```

Conclusiones

- El API de sockets nos da las herramientas para comunicar programas usando los servicios de los protocolos de transporte
- Construir servicios de Internet es organizar la comunicación entre las partes de una aplicación con estas herramientas
- And so it begins...