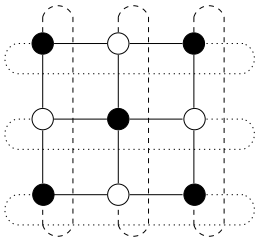

E52. Xarxes Informàtiques

Pràctica 7. El nivell de transport en Internet: Protocols TCP i UDP

Descripció de l'equip i del programari

1. Dos ordinadors de tipus PC amb S.O. Linux.
2. Adaptador de xarxa *Ethernet* amb connector RJ-45 (estàndards 10BaseT/100BaseT).
3. Commutador 10BaseT.
4. Monitor de xarxa i analitzador de protocols *Ethereal*.
5. Compil·lador *gcc* de C.

1 Introducció

Com vàrem veure en la pràctica anterior, el conjunt de protocols TCP/IP consta d'una sèrie de normes pel nivell de xarxa i el nivell de trasport que tenen com a característica comuna l'ús d'IP (*Internet Protocol*) per a l'intercanvi de *datagrames* entre estacions connectades a xarxes de qualsevol naturalesa.

En aquesta pràctica s'analitzaran els protocols del nivell de transport d'Internet, TCP (*Transmission Control Protocol*) i UDP (*User Datagram Protocol*), capturant amb aquest propòsit tràfic d'ambdós tipus mitjançant l'aplicació *Ethereal*. Ací convé recordar que la diferència fonamental entre el mode de funcionament d'aquests protocols és que mentre que TCP ofereix un servei orientat a la connexió i fiable, UDP és tot el contrari: el servei és sense connexió i a més a més no fiable. Al bloc de dades enviat pel protocol TCP (capçalera TCP més dades del nivell d'aplicació) se l'anomena *segment*. Mancant un nom millor, anomenarem al bloc de dades enviat pel protocol UDP igual que al del nivell de xarxa, és a dir, *datagrama*.

Per tal de generar el tràfic necessari per ser analitzat posteriorment, es procedirà a comunicar dos processos de màquines diferents mitjançant el pas de missatges. Aquest és el mecanisme de comunicació entre processos més emprat en l'actualitat. El seu avantatge més destacable és que pot ser emprat en qualsevol tipus d'arquitectura, ja es tracte de monoprocessadors, multiprocessadors amb memòria compartida, multicomputadors o sistemes distribuïts.

Els processos comunicants es generaran a partir d'uns programes senzills escrits en C que usen la interfície de programació d'aplicacions (API) de *sockets*, basada en el pas de missatges. No forma part d'aquesta pràctica l'estudi detallat d'aquesta interfície (que es postposa a pràctiques posteriors), sinò que es fa ús de l'API de *sockets* amb la finalitat de disposar d'un millor control sobre l'enviament o recepció de missatges entre els processos.

2 Desenvolupament de la pràctica

Per realitzar aquesta pràctica, i depenent de la disponibilitat d'equips, es formaran grups de treball de mode que cada grup dispose de dos ordinadors. Per tal d'evitar la confusió amb altre tràfic durant les captures, és molt convenient que estabiliu en *Ethereal* filtres el més precisos possible, que incloguen els ports TCP o UDP implicats en l'intercanvi de paquets.

2.1 Comunicació entre processos mitjançant el pas de missatges

El tràfic TCP es generarà mitjançant un parell de programes molt senzills, escrits en C, que contenen el codi corresponent a un *client* i un *servidor*. Sent TCP un protocol orientat a la connexió, cal que l'intercanvi d'informació vaja precedit i seguit, respectivament, de les fases d'establiment i alliberament de la connexió. Esquemàticament, l'evolució temporal de les accions que porten a terme el client i el servidor es mostra a continuació:

Client	**Servidor**
Fase 1: Establiment de la connexió	
	S1: A l'espera d'una connexió...
C1: Es sol·licita la connexió	
	S2: Es rep la sol·licitud de connexió
Fase 2: Intercanvi de dades	
	S2: A l'espera d'un número...
C2: Es llig un número	
C3: S'envia el número al servidor	
	S3: Es rep el número del client
	S4: S'escriu el número
Fase 3: Alliberament de la connexió	
C4: S'allibera la connexió	
	S5: S'allibera la connexió

El codi en C de `cliente` es mostra a continuació:

```
#include "cliente_servidor.h"
int main()
{
  int                sock, numero;
  struct sockaddr_in servidor;
  struct hostent     *hp;

  hp = gethostbyname(nombre_dns_de_servidor);
  if (!hp) {
    fprintf(stderr, "gethostbyname: %s\n", hstrerror(h_errno));
    exit(1);
  }
  servidor.sin_family = AF_INET;
  servidor.sin_port = htons(puerto_en_servidor);
  memcpy(&servidor.sin_addr, hp->h_addr, hp->h_length);

  sock = socket(AF_INET, SOCK_STREAM, 0);
```

```
  if (sock < 0) {
    perror("socket");
    exit(1);
  }

  /* Fase 1: Establecimiento de la conexión */
  if (connect(sock, (struct sockaddr *) &servidor,
              sizeof(struct sockaddr_in)) < 0) {
    perror("connect");
    exit(1);
  }

  /* Fase 2: Intercambio de datos */
  fprintf(stdout, "Dame el número: ");
  fflush(stdout);
  fscanf(stdin, "%d", &numero);
  if (send(sock, &numero, sizeof(int), 0) < 0) {
    perror("send");
    exit(1);
  }

  /* Fase 3: Liberación de la conexión */
  if (close(sock) < 0) {
    perror("close");
    exit(1);
  }

  exit(0);
}
```

El codi en C de `servidor` es mostra a continuació:

```
#include "cliente_servidor.h"
int main()
{
  int                sock, nuevo_sock, numero;
  socklen_t          longitud;
  struct sockaddr_in yo, cliente;

  yo.sin_family = AF_INET;
  yo.sin_port = htons(puerto_en_servidor);
  yo.sin_addr.s_addr = INADDR_ANY;

  sock = socket(AF_INET, SOCK_STREAM, 0);
  if (sock < 0) {
    perror("socket");
    exit(1);
  }

  if (bind(sock, (struct sockaddr *) &yo,
           sizeof(struct sockaddr_in)) < 0) {
    perror("bind");
    exit(1);
```

```
  }

  if (listen(sock, 5) < 0) {
    perror("listen");
    exit(1);
  }

  /* Fase 1: Establecimiento de la conexión */
  longitud = sizeof(struct sockaddr_in);
  nuevo_sock = accept(sock, (struct sockaddr *) &cliente, &longitud);
  if (nuevo_sock < 0) {
    perror("accept");
    exit(1);
  }

  /* Fase 2: Intercambio de datos */
  if (recv(nuevo_sock, &numero, sizeof(int), 0) < 0) {
    perror("recv");
    exit(1);
  }
  fprintf(stdout, "El número es: %d\n", numero);

  /* Fase 3: Liberación de la conexión */
  if (close(nuevo_sock) < 0) {
    perror("close");
    exit(1);
  }

  close(sock);
  exit(0);
}
```

En últim lloc, tant el codi del client com el del servidor inclouen el fitxer de capçalera `cliente_servidor.h`, el codi del qual es mostra a continuació:

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

#define nombre_dns_de_servidor "labsop01.act.uji.es"
#define puerto_en_servidor 16385
```

Ací és important fer notar que, depenent de l'ordinador on s'execute el procés corresponent al servidor, pot ser necessari modificar la línia del fitxer `cliente_servidor.h` on figura el nom DNS del servidor.

Podeu descarregar els codis d'aquestes rutines des de la plana web de l'assignatura.

2.2 Establiment de la connexió en TCP

En primer lloc s'analitzaran els segments que s'intercanvien durant l'establiment d'una connexió sobre TCP. Per fer-ho cal efectuar les passes següents:

1. Descarregar el codi del client sobre un dels ordinadors disponibles, al qual anomenarem `labsopC`, i el codi del servidor sobre l'altre, al qual anomenarem `labsopS`.
2. Modificar el fitxer `cliente_servidor.h` per tal que el nom DNS del servidor siga correcte en ambdós ordinadors (com a mínim en el client).
3. Generar el fitxer executable corresponent a cadascun dels ordinadors (per exemple, `cc -o servidor servidor.c`).
4. Configurar l'analitzador de tràfic *Ethereal* per tal que capture únicament el tràfic TCP entre `labsopC` i `labsopS` (en el port adequat), i iniciar la captura.
5. Llançar a execució el procés servidor en `labsopS`.
6. Llançar a execució el procés client en `labsopC`.
7. Detenir la captura abans d'introduir el número sol·licitat pel client i analitzar el tràfic generat entre el client i el servidor.

A continuació s'ofereixen algunes preguntes que vos ajudaran en aquesta anàlisi:

- Quants segments TCP s'han intercanviat entre el client i el servidor?
- Quina informació figura en els camps de port d'origen i port de destí del primer segment que circula del client al servidor? Per què?
- Quina informació figura en els camps de número de seqüència i de reconeixement dels segments intercanviats durant l'establiment de la connexió? Per què?
- Relacioneu l'intercanvi d'aquests segments amb el diagrama (o màquina) d'estats corresponent a l'establiment d'una conexió TCP vist en clase de teoria.
- Quina altra informació podeu identificar com a part de la capçalera TCP?

2.3 Intercanvi de dades en TCP

En aquest apartat es pretèn reflexionar sobre l'eficiència de l'intercanvi de dades sobre una connexió TCP. Per fer-ho es repetiran les passes de l'apartat anterior, però incloent també i considerant únicament els segments corresponents a l'intercanvi de dades (la fase 2).

Amb el tràfic capturat durant la fase 2 a la vista, contesteu a continuació les qüestions següents:

- Quants segments TCP s'intercanvien entre el client i el servidor per enviar una única dada (el número intercanviat)?
- Quina és l'eficiència d'aquest protocol (quocient entre octets de dades «útils» i octets de dades enviats)?

- Quina informació figura en els camps de número de seqüència i de reconeixement dels segments intercanviats durant la transmissió de dades? Per què?

Modifiqueu el codi del servidor per tal que en rebre el número, li sume 1 i el retorne al client. Modifiqueu també el codi del client per tal que mostre el número retornat en l'eixida estàndard. Captureu el tràfic corresponent a l'intercanvi de dades dels nous programes i contesteu les preguntes següents:

- Quants segments TCP s'intercanvien ara entre el client i el servidor durant l'intercanvi de dades?
- És eficient?

2.4 Alliberament de la connexió en TCP

L'objectiu d'aquest apartat sobre TCP és analitzar l'etapa final d'aquest tipus de connexions, és a dir, estudiar el tràfic que es genera quan s'allibera la connexió. Per fer-ho serà suficient amb repetir la captura dels paquets intercanviats, però tenint cura d'aconseguir capturar també el tràfic pertanyent a l'alliberament de la connexió. Una volta disponible aquest tràfic en la finestra danàlisi de l'aplicació, és convenient que responguеu les preguntes següents:

- Quants segments de TCP s'han intercanviat entre el client i el servidor per efectuar la desconnexió?
- Quina informació figura en els camps de número de seqüència i de reconeixement dels segments intercanviats durant l'alliberament de la connexió? Per què?
- Relacioneu l'intercanvi d'aquests segments amb el diagrama d'estats corresponent l'alliberament de connexió TCP vist en la classe de teoria.

2.5 Tràfic sobre UDP

El tràfic sobre UDP es generarà també mitjançant un parell de programes en C molt senzills als quals, degut a la naturalesa sense connexió d'aquest protocol, anomarem *emissor* i *receptor*. En aquest cas l'emissor llegirà el número de la seua entrada estàndard i l'enviarà al receptor, que el visualitzarà en la seua eixida estàndard.

Els codis d'aquestes rutines també poden descarregar-se des de la plana web de l'assignatura. Caldrà que modifiqueu el fitxer `emisor_receptor.h` per establir-hi el nom real del receptor.

Per analitzar el tràfic UDP procedirem a configurar adequadament els filtres de l'aplicació *Ethereal* per capturar aquest tipus de tràfic, i a executar `emisor` i `receptor` en màquines diferents. A la vista dels resultats (*datagrames* capturats), contesteu les preguntes següents:

- Quina informació figura en els camps de port d'origen i port de destí del primer datagrama que circula des de l'emisor cap al receptor? Per què?
- Quin valor figura en el camp de longitud de la capçalera del datagrama UDP? A què correspon aquest valor?
- S'ha utilitzat el camp de comprovació d'error?

- Quants datagrames UDP s'intercanvien entre l'emissor i el receptor per enviar una única dada?
- Quin protocol és més eficient, TCP o UDP?