Retele de calculatoare
notiţe de curs

Radu-Lucian Lupşa

8 noiembrie 2004

Cuprins

Capitolul 1

Introducere

1.1 Ce este o reţea de calculatoare?

Termenul *retea de calculatoare* are cel putin doua utilizari distincte:

1. mai multe calculatoare, impreuna cu un sistem (hard+soft) de comunicatii
2. un *sistem de calcul*, bazat pe o retea in sensul 1, dar comportandu-se ca un sistem unitar (de exemplu, prezinta aceleasi conturi de utilizatori pe toate calculatoarele)

1.2 Problemele abordate în curs

1. Realizarea infrastructurii pentru comunicaţii;
2. Aplicaţii (de comunicaţie) in reţea:
   - aplicaţii existente,
   - principiile realizării aplicaţiilor în reţea
3. Elemente de sisteme de operare in retea.

1.3 Problemele infrastructurii reţelei

1. codarea fizică a informaţiei;
2. detectarea şi corectarea erorilor de transmisie;

3. controlul fluxului (asigurarea faptului că emiţătorul nu trimite mai repede decât poate receptorul să primească);

4. controlul coliziunilor şi adresarea, pe medii de tip magistrală (adică în care există mai multe entităţi prevăzute cu emiţătoare şi receptoare care partajează acelaşi mediu);

5. dirijarea comunicaţiei în reţea (astfel încât două calculatoare să poată comunica, chiar dacă nu există legătură directă între ele, ci doar legături prin intermediari);

6. livrarea sigură (mesajele să nu se piardă, să nu ajungă în multiplu exemplar, şi să ajungă în ordinea în care au fost emise);

7. securitatea:

   - confidenţialitate (un mesaj să poată fi recepţionat doar de către destinatarul autorizat),
   - autentificarea (receptorul să poată verifica identitatea emiţătorului,
   - integritatea comunicaţiei (nimeni să nu poată modifica mesajele,
   - non-repudiabilitatea (destinatarul să poată dovedi originea unui mesaj, altfel spus, nici măcar destinatarul să nu poată falsifica identitatea autorului mesajului)

1.4 Tipuri de comunicaţie

Dupa nr. de receptori:

- punct la punct (unicast): un emiţător şi un receptor;
- difuziune (broadcast sau multicast): un emiţător şi mai mulţi receptori.

Nota: *broadcast* este o comunicaţie în care un mesaj este primit de toată lumea; *multicast* desemnează o comunicaţie în care doar o parte din calculatoarele din reţea primesc mesajul.

Dupa existenţa unei conexiuni:

- comunicatie prin conexiune;
- comunicatie prin datagrame.

Deziderate optionale:

- livrare sigura (un mesaj sa fie livrat exact o data)
- transmisie fara erori
- pastratea ordinii relative a mesajelor
- debit minim garantat
- timp maxim de livrare garantat
- confidentialitate
- autentificare

1.5 Arhitectura sistemului de comunicaţii

Capitolul 2

Programarea în reţea — introducere

2.1 Interfaţa de programare `socket` BSD

socket = capat de cale de comunicatie
Presupuneri:

- pe fiecare calculator pot rula mai multe procese, şi fiecare proces poate avea mai multe căi de comunicaţie deschise, prin urmare pe un calculator trebuie să poată exista la un moment dat mai multe socket-uri active;
- o comunicaţie poate fi prin conexiune sau datagrame
- pentru a identifica partenerul de comunicaţie, iniţiatorul unei comunicaţii trebuie să furnizeze o *adresă*; formatul acestei adrese poate să depindă de tipul reţelei (IP, IPX, etc).

API-ul este conceput să fie independent de tipul exact al reţelei (poate funcţiona pe reţele IP, IPX, local pe unix, etc.)

2.1.1 Creare

`socket(proto_family, type, protocol)`
`type`: desemnează tipul de servicii dorite:

`SOCK_STREAM`: conexiune; punct la punct; flux de date bidirecţional la nivel de octet; livrare sigura, cu pastrarea ordinii, transmisie fara erori.

`SOCK_DGRAM:` datagrame punct la punct sau difuziune transmisie fara erori; livrarea nu e sigura si nici ordinea garantata.

`SOCK_RAW:` acces direct la infrastructura; ex. in implementarea comenzii `ping`.

`proto_family` identifică tipul de reţea cu care se lucrează (IP, IPX, etc). Se pune `PF_INET` pentru conexiuni pe protocol Internet (IP).

`protocol` identifică protocolul particular de utilizat. Este intenţionat a fi folosit daca ar exista mai multe protocoale distincte pentru un tip de retea dat si pentru un tip de serviciu dat. Implicit se va pune 0.

Funcţia returnează identificatorul de socket, pe care aplicaţia îl va furniza în apelurile ulterioare.

2.1.2 Utilizare socket stream

Un socket stream poate să fie în una din două stări: neconectat sau conectat. Un socket neconectat poate avea o adresă fixată sau nu.

Orice socket care participă la o operaţie de comunicare trebuie să aibă o adresă; dacă adresa nu i se fixează explicit prin `bind()`, atunci sistemul îi va da o adresă aleatoare.

Conectarea, pe client:

`connect(sock_id, addr, addr_len)` conectează socket-ul local `sock_id` cu socket-ul identificat prin adresa `addr`. Adresa `addr` trebuie să corespundă unui socket de tip stream neconectat

Conectarea, pe server:

`bind(sock_id, addr, addr_len)` fixează adresa socket-ului `sock_id`.

`listen(sock_id, dim_coada)` fixează dimensiunea cozii de aşteptare pentru conexiunile către adresa socket-ului

`accept(sock_id, addr, addr_len)` aşteaptă o cerere de conexiune şi returnează *un nou socket* conectat la socket-ul care a cerut conexiunea. Socket-ul original poate fi folosit ulterior pentru a aştepta noi conexiuni. Parametrul `addr` returnează adresa socket-ului

Comunicaţia propriu-zisă:

`send(sock_id, buf, count, flags)` trimite `count` octeţi

`recv(sock_id, buf, count, flags)` recepţionează cel mult `count` octeţi. Dacă nu există nici un octet sosit, funcţia aşteaptă sosirea a cel puţin un octet (este blocantă). Apoi citeşte minimul între numărul de octeţi sosiţi şi numărul de octeţi ceruţi (`count`). Returnează numărul de octeţi citiţi. Returnează 0 dacă emiţătorul a închis conexiunea.

Închiderea conexiunii:

`close(sock_id)` distruge socket-ul şi închide complet conexiunea ataşată (dacă există)

`shutdown(sock_id, how)` închide comunicaţia, posibil doar într-unul singur sens

Structura unui program:

Clientul:
```
#include <sys/socket.h>
#include <netinet/in.h>
...
  struct sockaddr_in adr;

  sd=socket(PF_INET, SOCK_STREAM, 0);
  memset(&adr, 0, sizeof(adr));
  adr.sin_family = AF_INET;
  adr.sin_port = htons(remote_port);
  inet_addr(remote_ip, &adr.sin_addr);
  if(-1==connect(sd, (struct sockaddr*)&adr,
        sizeof(adr)) )
  {
perror("onnect()"); exit(1);
  }
  ...
  send(sd, ...); recv(sd, ...)
  ...
  close(sd);
```

Serverul:
```
#include <sys/socket.h>
#include <netinet/in.h>
...
  struct sockaddr_in adr;

  sd=socket(PF_INET, SOCK_STREAM, 0);
  memset(&adr, 0, sizeof(adr));
```

```
  adr.sin_family = AF_INET;
  adr.sin_port = htons(remote_port);
  adr.sin_addr = INADDR_ANY;
  if(-1==bind(sd, (struct sockaddr*)&adr,
        sizeof(adr)) )
  {
perror("onnect()"); exit(1);
  }
  listen(sd, 5);
  ...
  struct sockaddr_in client_adr;
  socklen_t client_adr_size;
  sd_c = accept(sd, (struct sockaddr*)&client_adr,
        &client_adr_size)
  ...
  send(sd_c, ...); recv(sd_c, ...)
  ...
  close(sd_c);
  ...
  close(sd);
```

2.1.3 Utilizare socket pentru datagrame

Probabil cel puţin unul din capete trebuie să fixeze adresa socket-ului, la o valoare cunoscută de partener; pentru aceasta foloseşte apelul `bind()`. Celălalt capăt primeşte automat o adresă la prima trimitere de datagramă.

Funcţii:

`bind(sock_id, addr, addr_len)` fixează adresa socket-ului;

`sendto(sock_id, buf, count, flags, addr, addr_len)` trimite o datagramă la adresa specificată;

`recvfrom(sock_id, buf, count, flags, addr, addr_len)` recepţionează o datagramă, şi pune în `addr` adresa de provenienţă.

2.2 Adresarea in internet

Adresa IP:

- identifică unic maşina

- 32 biţi; de obicei scrisă ca 4 numere zecimale separate prin puncte (ex. 193.226.40.130)
- în principiu, fiecare interfaţă de reţea (ex. placă de reţea, linie seriala sau modem folosit pentru o conexiune in reţea) are o adresa IP
- adresa 127.0.0.1 corespunde unei intereţe virtuale intr-o retea (virtuală) în care calculatorul este singur. Altfel spus, 127.0.0.1 desemnează intotdeauna maşina locala.
- unele adrese sunt folosite pentru broadcast

Numarul portului:

- serveşte pentru a diferenţia intre socket-ii de pe aceeaşi maşina
- 16 biţi (1–65535)
- porturile TCP (`SOCK_STREAM`) sunt independente de porturile UDP (`SOCK_DGRAM`)

2.3 Transmiterea mesajelor

Exemplu: clientul anunţă scoaterea dintr-o magazie a unei anumite cantităţi de marfă. Se trimite un mesaj conţinând denumirea şi cantitatea.

2.3.1 Mesaje binare

Exemplu:

```
struct Cerere {
  char denumire[30];
  int cantitate;
};
Cerere c;
...
send(sd, &c, sizeof(c), 0);
```

Probleme:

1. lungimea (numărul de octeţi) pe care se reprezintă un `int` este dependentă de platformă

2. ordinea octeţilor este dependentă de platformă (little endian vs. big endian)

3. alinierea e dependentă de platformă (se poate ca adresa unui întreg să trebuiască să fie multiplu de 2, 4 sau 8 octeţi)

Soluţie:

1. se folosesc `typedef`-urile `uint16_t`, `uint32_t` definite în `netinet/in.h` ; lungimea reprezentării lor e independentă de platformă;

2. se convertesc la o ordine a octeţilor independentă de platformă: funcţiile `htons()`, `htonl()`, `ntohs()`, `ntohl()`;

3. se transmite individual fiecare membru al unei structuri.

Exemplu:

```
struct Cerere {
  char denumire[30];
  unit32_t cantitate;
};
Cerere c;
...
send(sd, c.denumire, 30, 0);
c.cantitate=htonl(c.cantitate);
send(sd, &c.cantitate);
```

2.3.2 Mesaje text

2.4 Asigurarea concurenţei

Capitolul 3

Nivelul fizic

3.1 Transmisia prin fir

3.1.1 Marimi electrice

Reamintim din fizica de liceu următoarele mărimi electrice:

Sarcina electrică (Q) măsoară cantitatea de electricitate. Este analoagă cantităţii de fluid. Putem măsura cantitatea de electricitate (sarcina electrică) ce trece printr-un conductor într-o perioadă de timp. Se măsoară în *coulombi* (C).

Intensitatea curentului (I) (numită pe scurt *curentul*) măsoară sarcina electrică ce trece printr-un conductor împărţită la timpul în care trece. Este analoagă debitului de fluid printr-o conductă. Intensitatea curentului ce trece printr-un conductor se măsoară întrerupând conductorul şi intercalând un instrument de măsură numit *ampermatru*. Ampermetrul trebuie să conducă foarte bine curentul (să aibă impedanţa — vezi mai jos — cât mai mică) pentru a perturba cât mai puţin circuitul în care măsoară. Se măsoară în *amperi* (A).

Tensiunea electrică (U) între două conductoare măsoară cât de puternic ar fi impinsă o sarcină electrică aflată în primul conductor către cel de-al doilea. Este analoagă diferenţei de presiune între două conducte de fluid. Pentru a măsura tensiunea electrică între două conductoare, montăm un instrument numit *voltmetru* cu o bornă conectată la unul din conductoare şi cu cealaltă bornă legată la celălalt conductor. Voltmetrul trebuie să fie cât mai aproape de un izolator — să conducă cât mai puţin curent de la o bornă la alta (aidcă să aibă impedanţa cât mai mare). Tensiunea se măsoară în *volţi* (V).

Puterea şi energia electrică (P) Un element de circuit la bornele căruia se aplică o tensiune U şi prin care trece un curent de intensitate I de acelaşi sens cu tensiunea primeşte o putere $P = UI$. Dacă intensitatea are sens invers tensiunii, elementul livrează puterea $P = UI$ circuitului. Puterea se măsoară în watti (W).

Energia electrică (E) Energia măsoară produsul dintre puterea primită sau cedată de un element de circuit şi timpul cât are loc fenomenul. Unitatea de măsură în Sistemul Internaţional este joule-ul (J), însă frecvent se foloseşte kilowattora (kWh) (atenţie, simbolul kW/h , citit kilowatt pe oră, nu este corect, deşi este folosit curent); un kilowattoră este energia primită de un consumator care primeşte o putere de un kilowatt timp de o oră; în consecinţă 1kWh = 3600000J.

Elemente de circuit şi mărimi caracteristice (*Notă: fiecare element poate fi construit în mod deliberat, însă comportamentul poate să apară şi acolo unde nu este dorit*):

rezistorul si rezistenţa (R) Un rezistor este un element de circuit

condensatorul şi capacitatea (C) .

bobina şi inductanţa (L) .

impedanţa (Z) şi conductanţa (G)

surse şi consumatori

3.1.2 Analiza Fourier

Considerăm un circuit electronic, care are o intrare şi o ieşire (fig. **??**). În particular, o pereche de fire folosite pentru transmisie poate fi considerată un astfel de circuit, capetele dinspre emiţător constituind intrarea, iar cele dinspre receptor, ieşirea.

Tensiunea de la ieşire depinde de tensiunea de la intrare, însă în general depinde de tot istoricul ei. Altfel spus, comportamentul circuitului poate fi descris de o funcţie care primeşte ca argument funcţia tensiune-timp de la intrare si întoarce funcţia tensiune-timp de la iaşire (o funcţie definită pe un spatţiu de funcţii cu valori tot într-un spaţiu de funcţii). Scriem $U_e = T(U_i)$, unde $U_i(t)$ este valoarea tensiunii de intrare la momentul t, iar $U_e(t)$ este tensiunea de ieşire la momentul t.

Multe circuite electronice au un comportament *liniar*, adică dacă $U_i(t)$ poate fi scris ca $U_i(t) = U_{i1}(t) + U_{i2}(t) + \ldots + U_{in}(t)$, atunci $U_e(t) = U_{e1}(t) + U_{e2}(t) + \ldots + U_{en}(t)$, unde $U_{e1} = T(U_{i1})$, $U_{e2} = T(U_{i2})$, etc.

Este în general mult mai uşor de analizat comportamentul unui circuit electronic dacă i se prezintă la intrare un semnal sinusoidal (adică de forma $U(t) = U_0 \sin \omega t + \phi$) decât pentru un semnal oarecare. Mai mult, ieşirea în acest caz este de regulă tot un semnal sinusoidal.

Orice semnal se poate scrie ca o sumă de semnale sinusoidale. *(Nota: scrierea unei funcţii ca sumă de sinusoide este aşa-numita* transformata Fourier. *Condiţiile de existenţă depind de cadrul în care se scrie — funcţie in L^2, distribuţie, etc. Prezentarea de faţă are doar scop de suport intuitiv; lipsesc detaliile matematice şi în consecinţă cele scrise aici nu pot fi folosite pentru a sprijini în mod fiabil raţionamente).*

3.1.3 Proprietăţi ale liniei

Un conductor real prezintă o rezistenţa nenulă şi o impedanţă nenulă; se comportă ca şi când ar consta dintr-un rezistor legat în serie cu o bobină, sau, şi mai bine, multe rezistoare mici înseriate între care sunt intercalate bobine mici. Două conductoare alăturate separate de un izolator se comportă ca un condensator — prezintă o capacitate nenulă între ele. De asemenea, izolatorul nu e perfect şi prezintă o rezistenţă finită.

De aceea, o pereche de conductoare se prezintă ca şi când ar fi formate prin conectarea unul după altul a infinit de multe celule elementare ca în figura . În consecinţă, comportamentul perechii de conductoare va fi descris de ecuaţia

(3.1)

3.1.4 Codarea în banda de bază

3.1.5 Transmisia modulată

3.2 Transmisia prin unde radio

3.2.1 Propagarea undelor

3.2.2 Unde nedirijate şi unde dirijate

3.3 Transmisia prin fibră optică

3.3.1 Propagarea semnalului

Capitolul 4

Nivelul legăturii de date