ANEXA A

ALGORITMI DE REȚEA

Pentru reprezentarea grafică a unei rețele de calculatoare se folosește **metoda grafurilor**.

Nodurile grafului simbolizează calculatoarele sau, mai general, echipamentele terminale de date (DTE), respectiv echipamentele de comunicație de date (DCE).

Legăturile fizice existente în rețea sunt reprezentate în graf prin **săgeți** cu o singură orientare în cazul transmisiei simplex sau cu dublu sens pentru comunicațiile duplex.

Descrierea funcționării rețelei este realizată prin **diagrame cu număr finit de stări,** cărora li se asociază algoritmi de procesare specifici. Aceștia se implementează fie ca procese în cadrul sistemului de operare, fie ca programe software independente activate odată cu sistemul de operare. Se pot utiliza diferite limbaje de programare pentru scrierea acestora.

Prin intermediul algoritmilor de rețea, se realizează controlul traficului de pachete la nivelul nodurilor, al congestiilor și eventual al jetonului de acordare a permisiunii de transmisie.

Apar aspecte distincte în funcționarea unei rețele centralizate, respectiv a uneia descentralizate (Fig. A.1 a; b).

Graful (a) de **rețea centralizată** (Fig.A-1a) cuprinde un nod central (M - *master*) care supervizează activitatea întregii rețele, în sensul că orice mesaj se transmite prin intermediul acestui nod, chiar dacă transmisia se face între două noduri subordonate (*slave*), de exemplu S_k și S_j. Nodul M constituie astfel un controler de rețea. Defectarea acestuia afectează funcționarea întregii rețele.

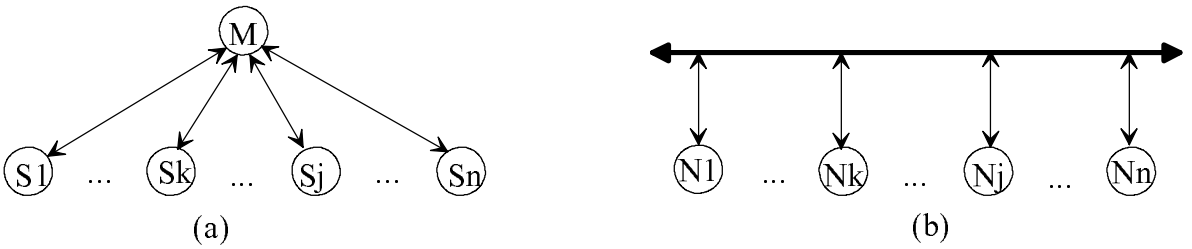

Fig. A.1 Grafuri de rețea: (a) centralizată; (b) descentralizată.

De exemplu, în nodul M poate fi amplasat un *hub* care să controleze accesul la mediu al stațiilor din nodurile *slave* printr-o topologie logică de tip 'stea' (*star*).

În graful asociat unei **rețele descentralizate** (Fig. A-1b), toate nodurile au drepturi egale în rețea, fiecare nod fiind responsabil de controlul traficului de pachete la nivelul său. Într-o rețea cu topologie fizică de tip 'magistrală' (*bus*), accesul la mediu se poate face fie prin metoda CSMA/CD, fie prin cea cu jeton, *token-passing*. Defectarea unui nod nu le afectează pe celelalte.

A.1 Algoritmul nodului *master*

Nodul central M (*master*) adresează în ordine cererea de transmisie fiecărui nod din rețea (S_j). Dacă acesta are un mesaj de transmis, se realizează transmisia lui din nodul S_j către M și stocarea sa în memorie. Expedierea lui către nodul-destinație se va face atunci când se va stabili comunicația cu nodul respectiv (S_k). **Algoritmul nodului *master*** se repetă pe toată durata funcționării rețelei, deservind periodic toate stațiile din rețea.

În figura A.2, se prezintă o schemă logică pentru algoritmul nodului *master*.

O variantă de implementare a acestui algoritm în limbaj C este prezentată în continuare:

```
repeat
      k:=1
      repeat
            if(mesaj pentru nodul Sk) then
                  trimite mesajul nodului Sk;
            endif
            adreseaza cererea de transmisie nodului Sk;
            if(mesaj de transmis din nodul Sk) then
                  acorda permisiunea de transmisie nodului Sk;
                  accepta mesajul din nodul Sk;
            endif
```

```
            k:=k+1
        until k:=n+1 (*n - numarul total de noduri slave din retea *)
    until se intrerupe functionarea.
```

În cazul apariției unui defect la nivelul nodului central, toate comunicațiile dintre nodurile rețelei sunt afectate. Acest dezavantaj nu apare în rețelele descentralizate, care au astfel o mai mare siguranță și flexibilitate în funcționare.

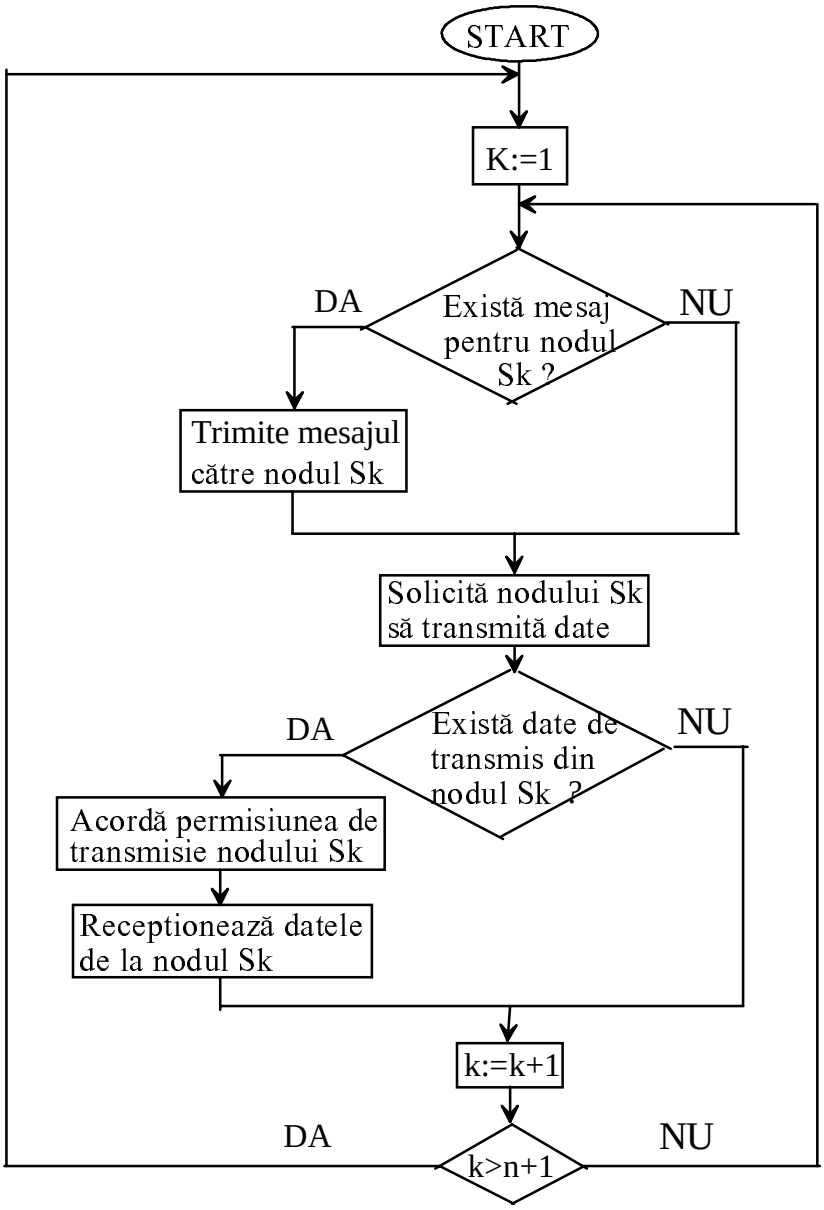

Fig. A.2 Algoritmul nodului MASTER

A.2 Algoritmul unui nod *slave*

Un nod *slave* subordonat celui coordonator central, *master*, se găsește în **starea de așteptare** atâta timp cât nu i se trimit pachete și nu este solicitat să transmită.

Nodul devine **activ** în momentul în care se recepționează un cadru dinspre nodul *master*. Se decide dacă acesta reprezintă un mesaj destinat nodului *slave* sau este un cadru de control de tip 'solicitare de transmisie'. Urmează în funcție de caz, fie recepționarea datelor, fie trimiterea către nodul *master* a cererii pentru acordarea permisiunii de transmisie, dacă există date de transmis și dacă modul de acces la mediu se bazează pe utilizarea 'jetonului de transmisie'.

În figura A.3, se prezintă un exemplu de algoritm pentru nodul *slave*. Algoritmul este rulat ciclic până la întreruperea funcționării nodului respectiv. Eventuala blocare a nodului într-una din stări se poate soluționa prin resetarea acestuia și reinițializarea algoritmului.

Un model de scriere a acestui algoritm în limbaj C este prezentat mai jos.

```
repeat
      asteapta mesaj dinspre nodul master;
      if(mesaj destinat nodului slave) then
            accepta mesajul transmis de nodul master;
      elseif(solicitare de transmisie) then
            if(exista date de transmis) then
                  cere permisiunea de transmisie;
                  asteapta acordarea permisiunii de transmisie;
                  transmite datele catre nodul master;
            endif
      endif
until se intrerupe functionarea nodului.
```

A.3 Algoritmul unui nod dintr-o rețea descentralizată

Într-o rețea descentralizată (Fig. A.1b), toate nodurile au aceeași prioritate.

Modul de ocupare a canalului de comunicații se bazează fie pe metoda de ascultare a canalului pentru a identifica momentul în care acesta este liber pentru transmisie, cu riscul apariției unor coliziuni între pachetele transmise simultan pe aceeași cale, fie pe metoda 'jetonului' pentru evitarea blocajelor de trafic.

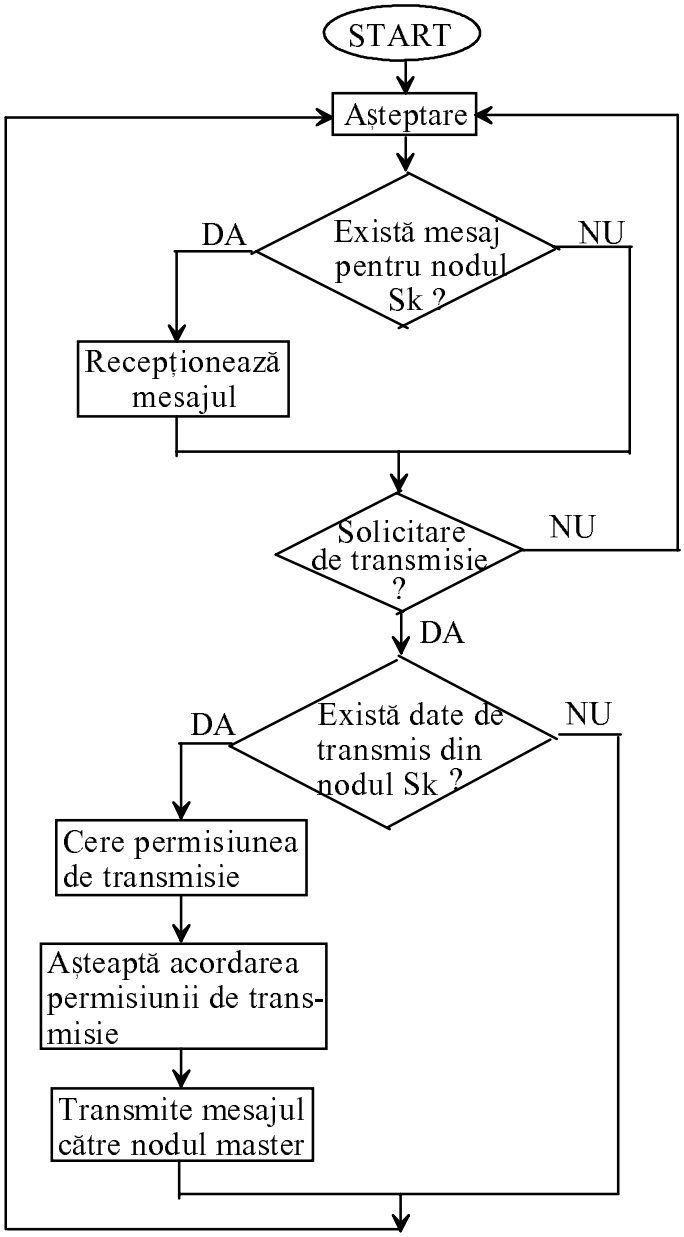

Fig. A.3 Algoritmul nodului SLAVE

Rețelele Ethernet sunt LAN-uri descentralizate CSMA/CD.

Algoritmul pentru un nod N_k dintr-o rețea descentralizată, cu 'jeton de transmisie', este prezentat în figura A.4. Acesta este rulat ciclic până la întreruperea funcționării nodului.

Implementarea algoritmului în limbaj C se poate face ca în următorul model:

```
repeat
      asteapta primirea unui cadru;
      if(mesaj pentru nodul Nk) then
```

```
            receptioneaza mesajul;
      elseif(jeton) then
            if(exista mesaj de transmis din nodul Nk) then
                  transmite mesajul;
            endif
            transmite jetonul nodului urmator;
      endif
until se intrerupe functionarea nodului.
```

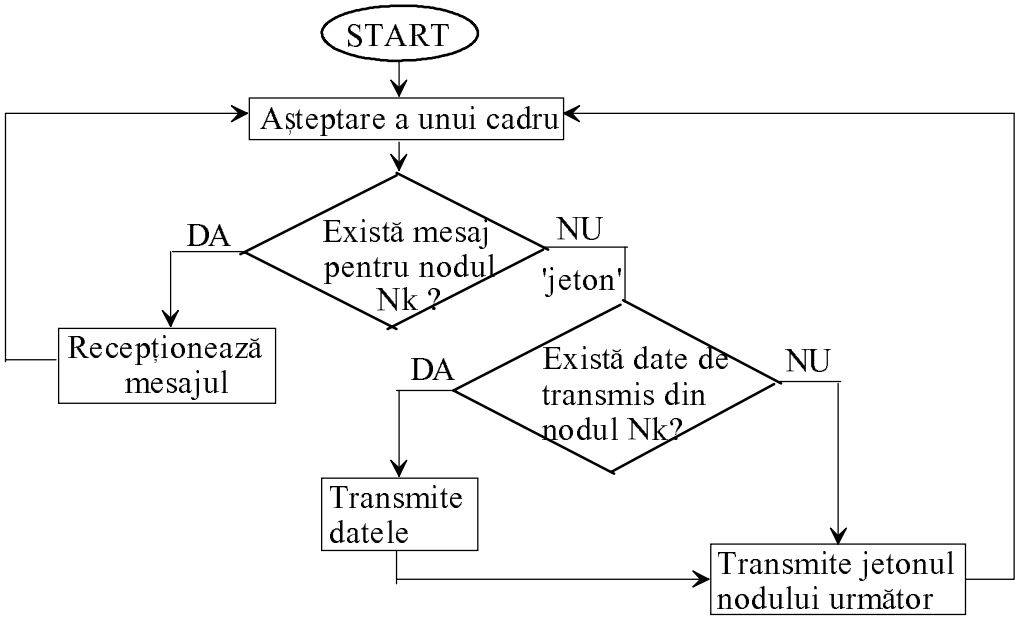

Fig.A.4 Algoritmul unui nod Nk
dintr-o rețea descentralizată

A.4 Algoritmul CSMA/CD

Tehnica CSMA/CD presupune că nodurile rețelei sunt într-o permanentă concurență pentru obținerea dreptului de utilizare a mediului fizic de transmisie. Nodul care ocupă rețeaua la un anumit moment transmite un singur pachet după care este obligat să elibereze canalul. În timp ce unul din noduri transmite, toate celelalte sunt în așteptare. Dacă mai multe noduri încearcă să transmită simultan, atunci apare o coliziune între pachete.

În figura A.5 sunt reprezentate grafic două situații specifice, într-o topologie fizică de tip ' magistrală' (*bus*).

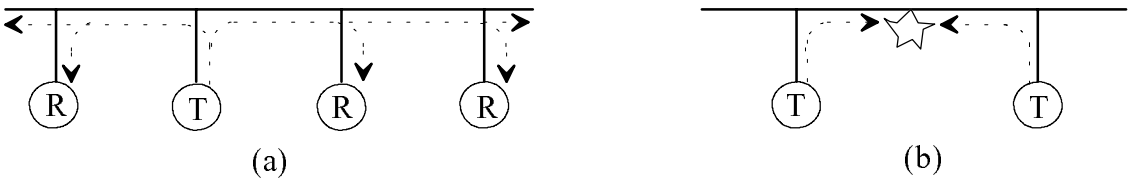

Fig.A.5 Transmisie în rețea 'bus': (a) broadcast; (b) cu coliziune.

Nodurile notate T sunt în starea de transmisie, iar cele notate R recepționează date.

În primul caz (Fig.A-5a), un singur nod transmite către toate celelalte noduri ale rețelei (*broadcast*).

În cazul (b), se observă apariția unei coliziuni cauzată de transmisii simultane efectuate de două noduri din rețea. Semnalele transmise interferă ceea ce conduce la distorsionarea lor și imposibilitatea recuperării datelor, adică pierderea mesajelor.

În această situație, fiecare nod își oprește procesul de transmisie și trece într-o stare de inactivitate (*back-off*), cu durata stabilită pe baza unui generator de secvență aleatoare (*jamming*). Este important ca duratele acestei stări, pentru nodurile implicate în coliziune să aibă durate diferite astfel încât nodurile să nu încerce retransmisia simultan, ceea ce ar conduce la apariția unei noi coliziuni și eventual propagarea fenomenului la infinit. Durata acestei stări poate fi stabilită proporțional cu adresa nodului ceea ce elimină riscul repetării coliziunii, dar introduce o anumită ierarhie de priorități în rețea, întrucât întotdeauna nodul cu adresă de valoare mai mică va ocupa mai repede canalul de comunicație și va transmite datele înaintea unui nod cu o adresă de valoare mai mare.

Schema de aplicare a metodei CSMA/CD rulează în fiecare nod la nivelul legăturii de date, pe baza diagramei de stări prezentate în figura A.6.

La punerea în funcțiune sau după resetarea nodului, acesta se află în **starea inițială** de testare sau "ascultare" a mediului de transmisie.

Dacă are de transmis date, atunci nodul trece în **starea de așteptare** până când se eliberează canalul și începe **transmisia**. Aceasta poate să decurgă normal, caz în care după finalizarea transmisiei nodul revine în starea inițială, sau este posibil să apară o coliziune ceea ce determină oprirea procesului de transmisie și obligarea nodului la **inactivitate** pentru o anumită perioadă de timp, după care intră din nou în starea de așteptare pentru a retransmite cadrul.

Din starea inițială, este posibil ca un cadru transmis să fie destinat nodului, adică în câmpul adresei de destinație să apară chiar adresa nodului respectiv (MID - *"My IDentifier"*). Alte noduri ignoră cadrul respectiv dar nodul căruia îi este adresat îl **acceptă** și începe recepția. Dacă aceasta decurge normal, la final nodul va reveni în starea inițială.

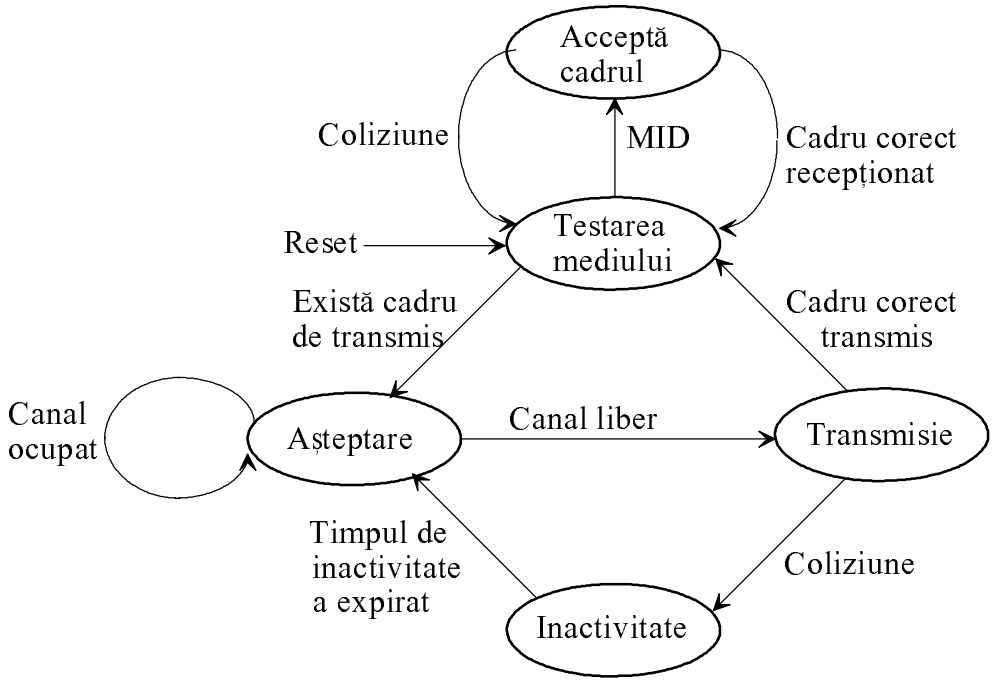

Fig.A.6 Algoritmul CSMA/CD

Dacă apar coliziuni pe durata recepției, nu se confirmă recepția cadrului (NAK - *Not Acknowledge*) și nodul trece în starea de ascultare a canalului. Nodul-sursă va retransmite cadrul respectiv.

A.5 ALGORITMUL TOKEN-BUS

Evitarea apariției coliziunilor într-o rețea locală este posibilă prin utilizarea 'jetonului de acordare a permisiunii de transmisie'.

Inițial rețeaua trebuie configurată pentru a stabili ordinea în care jetonul este transmis de la o stație la alta.

În arhitectura token-bus, terminalele sunt conectate logic în inel ceea ce presupune că fiecare nod din rețea cunoaște propria adresă (MID) și adresa următorului nod (NID - *Next IDentifier*) căruia îi va pasa jetonul. Acest proces de învățare la nivelul fiecărui nod a NID se realizează prin algoritmul de 'votare' (*polling*) care se va încheia în momentul în care se închide inelul logic și toate adresele nodurilor sunt incluse în inelul logic. În figura A.7, este prezentată spre exemplificare o rețea token-bus. Stația 5 nu este în funcțiune și, prin urmare, nu este inclusă în inelul logic.

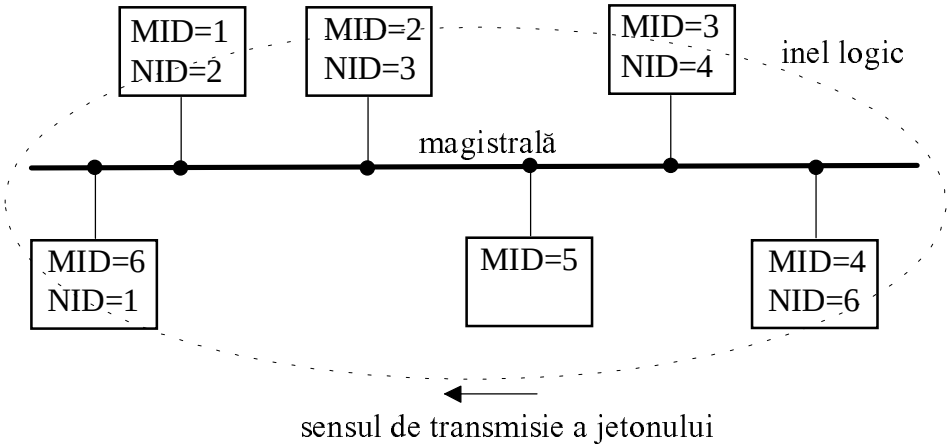

Fig.A.7 Exemplu de rețea Token-Bus

Evident, dacă un nou terminal se conectează la rețea, este necesară reconfigurarea acesteia pentru includerea noului nod în lista de adrese. Aceasta presupune întreruperea procesului normal de funcționare printr-un procedeu de bruiaj (*jamming*) care determină pierderea jetonului și lansează algoritmul de regăsire a acestuia (*token-recovery*).

Inițial toate nodurile se găsesc în starea de "somn", adică sunt inactive pe o durată de timp proporțională cu adresa sau ID-ul fiecăreia. În această stare se ajunge și prin lansarea în rețea a semnalului de bruiaj de către un terminal care dorește să se conecteze la inelul logic. Evident, nodul cu adresa minimă va fi primul activ și va începe procesul de 'votare'. Acesta trimite jetonul pe magistrala de date și așteaptă răspuns din partea următoarei stații, în ordine crescătoare a adreselor. Noua stație va ieși din starea de inactivitate la primirea jetonului și va continua procesul de 'votare'. Stațiile care au determinat NID-ul, intră în procesul normal de funcționare, conform algoritmului token-bus (Fig.A.8). Dacă în rețea este activă o singură stație, nu se poate forma inelul logic și acel terminal rămâne în starea de așteptare până la intrarea în funcțiune a altor stații.

A.6 ALGORITMUL TOKEN-RING

Spre deosebire de rețeaua de tip token-bus, rețeaua token-ring utilizează un inel fizic, ceea ce presupune că transmiterea unui pachet între două noduri neadiacente se face prin intermediul altor noduri.

Coliziunile sunt evitate prin folosirea jetonului de transmisie dar într-o rețea token-ring acesta poate fi 'liber' sau 'ocupat'. Evident, o stație poate transmite date doar dacă primește jetonul liber.

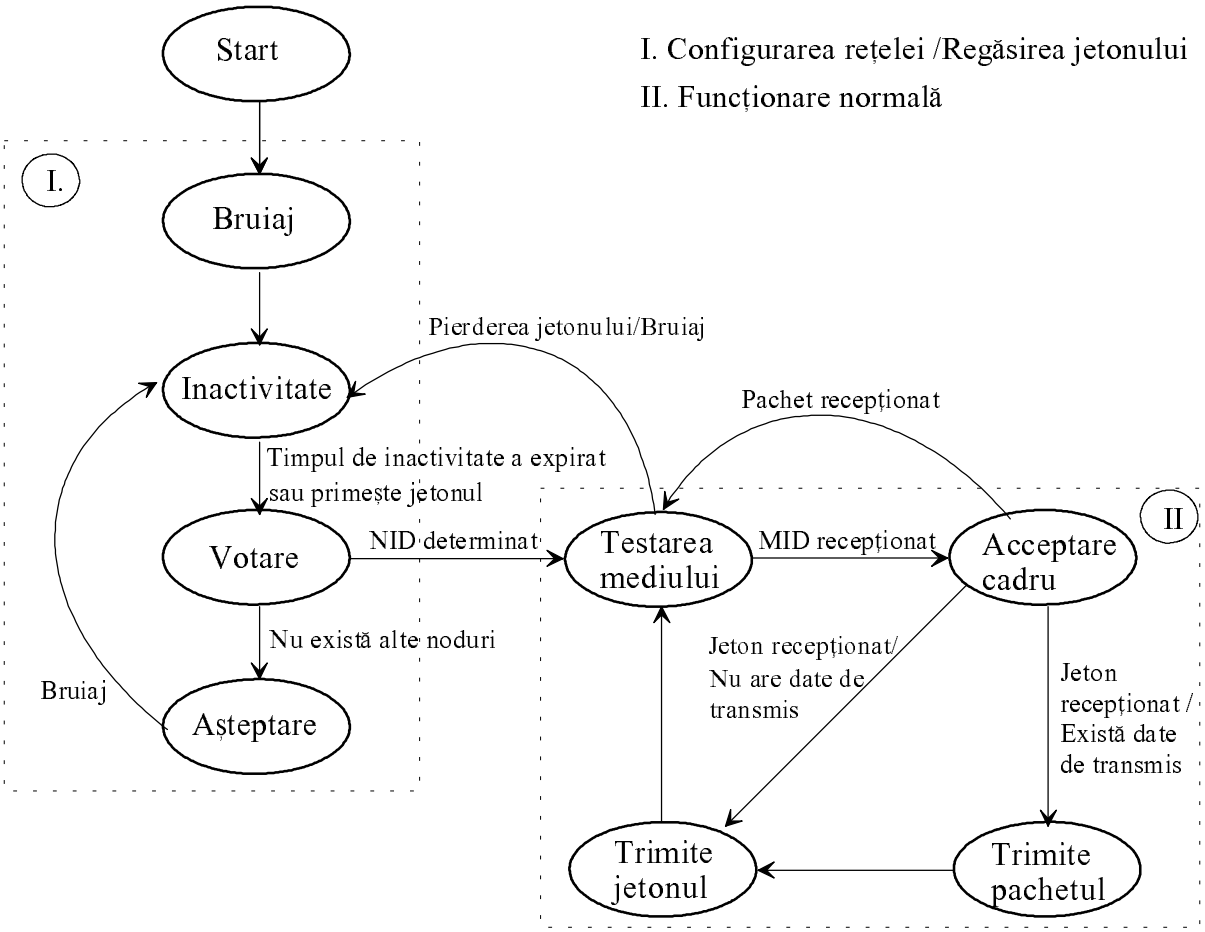

Fig. A.8 Algoritmul Token-Bus

Jetonul este 'ocupat' atunci când precede un pachet de date, urmând a fi eliberat de nodul care a expediat pachetul.

Jetonul liber circulă între stații, pe inelul fizic, în ordinea conectării acestora la inel.

În figura A.9, este prezentată ca exemplu o rețea token-ring cu 4 stații.

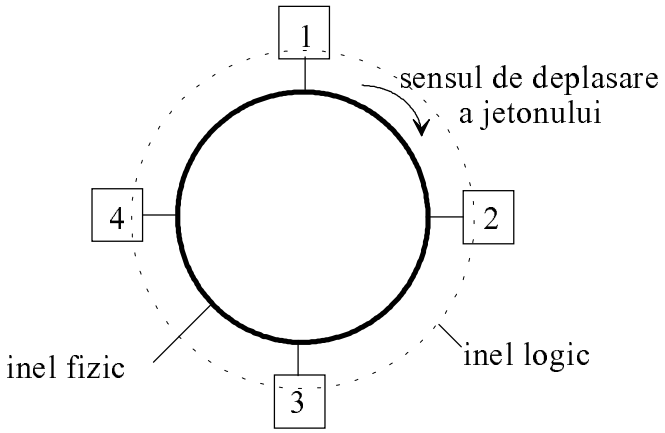

Fig.A.9 Exemplu de rețea în inel

Inițial jetonul circulă liber până când una din stații are mesaj de transmis și devine activă. Să presupunem că stația 1 deține jetonul și trimite un pachet către stația 3. Forma jetonului se modifică pentru a indica starea de 'ocupat' a acestuia pe durata transportării pachetului. Cadrul transmis de stația 1, conținând jetonul 'ocupat' și datele, ajunge la stația 2. Presupunem că și stația 2 are de transmis date. Întrucât jetonul este ocupat, aceasta va rămâne în așteptare și va realiza doar transferul cadrului de la stația 1 către stația 3. Stația 3 își recunoaște adresa (ID) și preia datele dar nu eliberează jetonul ci îl transmite mai departe ca 'ocupat'pe inel împreună cu confirmarea de primire corectă a pachetului (ACK) către stația 1 prin intermediul nodului 4. Stația 1 recepționează pachetul și eliberează jetonul. Jetonul 'liber' ajunge la stația 2 care va putea transmite. Pe durata transmisiei efectuate de un nod, toate celelalte noduri sunt pasive, adică realizează doar transferul pachetului către următorul nod (*forwarding*).

Algoritmul Token-Ring este prezentat în figura A.10.

Apar modificări față de acest algoritm în situația transformării rețelei token-ring în rețea star-ring.

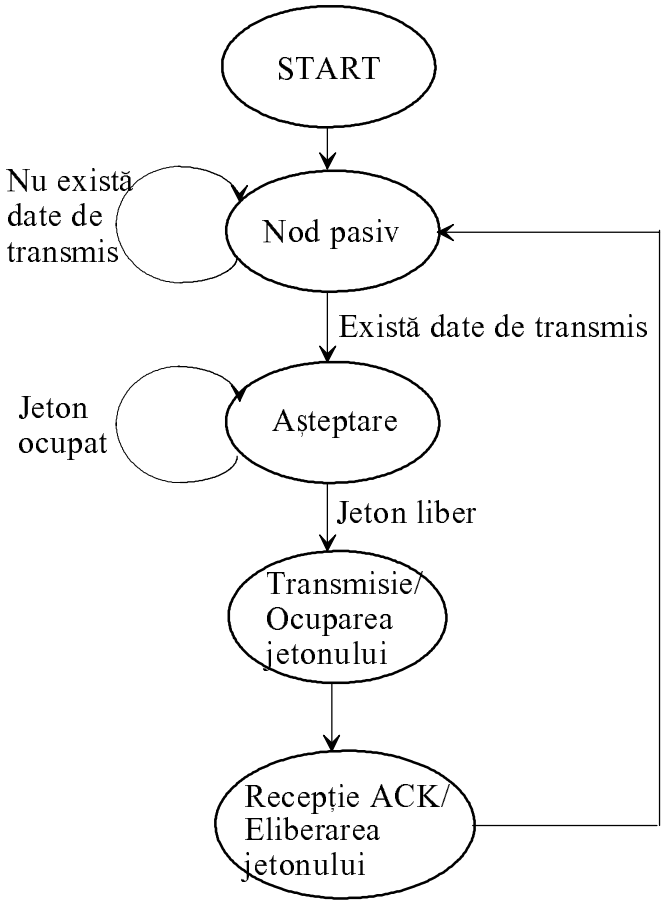

Fig. A.10 Algoritmul Token-Ring

APLICAȚIE

Scrieți în limbaj C următorii algoritmi de rețea:

a. CSMA/CD.

b. "de votare" (*polling*).

c. Token-Bus cu funcționare normală.

d. Token-Bus pentru configurarea rețelei.

e. Token-Ring.