Articole

ALGORITMI DE SIMULARE PENTRU SISTEME DE AŞTEPTARE BAZATE PE ÎMPĂRŢIREA CLIENŢILOR ÎN CLASE DE PRIORITĂŢI

dr. Ion Florea

Facultatea de Matematică-Informatică
Universitatea „Transilvania" Braşov
E-mail: i.florea@info.unitbv.ro

Rezumat.. De obicei, sistemele de aşteptare sunt studiate analitic, adică, pentru anumite repartiţii ale timpilor dintre două sosiri consecutive şi timpilor de servire, precum şi anumite politici de servire ale clienţilor, sunt deduse formule matematice ale factorilor de eficienţă ai sistemului. Există numeroase situaţii când astfel de formule nu există sau sunt extrem de complicate. În aceste situaţii, se recomandă utilizarea unor algoritmi de simulare cu ajutorul calculatorului. Lucrarea prezintă algoritmi de simulare pentru sisteme de aşteptare, în care servirea clienţilor se bazează pe împărţirea lor în clase de priorităţi. De asemenea, se demonstrează că algoritmii prezentaţi au un comportament polinomial. Comparând rezultatele simulării cu cele analitice, obţinute în anumite cazuri particulare, se verifică validitatea algoritmilor prezentaţi.

Cuvinte cheie: Sistem de aşteptare, simulare pe calculator, algoritm, priorităţi.

1. Introducere

Modelele în care disciplina de servire se stabileşte după criterii care nu iau în considerare ordinea intrării clienţilor în sistem, se numesc modele cu prioritate. În astfel de sisteme, clienţii sunt împărţiţi în clase de priorităţi. Dacă notăm cu *m* numărul de total de clase, atunci clienţii clasei *i* au prioritate la servire faţă de clienţii clasei j dacă $i<j$. De asemenea, în cadrul aceleiaşi clase, clienţii sunt serviţi în ordinea FIFO ("primul sosit-primul servit"). Clienţii pot sosi în sistem după aceeaşi repartiţie sau după repartiţii ale timpului între sosiri diferite.

Distingem:

- **sisteme de tip "cap de linie"(HOL – head of line)**, în care, dacă un client sosit are o prioritate superioară clientului servit, clientul sosit aşteaptă până la terminarea servirii curente pentru a fi la rândul lui servit;

- **sisteme cu evacuare** sau **modele cu prioritate absolută**, în care un client sosit de clasă superioară clientului servit va genera „evacuarea" în coadă a clientului servit şi servirea clientului nou sosit; în ceea ce priveşte timpul de servire al clientului evacuat, el poate fi vechiul timp, adică servirea se reia de la început, sau poate fi timpul rămas, adică diferenţa dintre timpul total de servire şi timpul efectuat.

În [10], este prezentat un algoritm de simulare pentru sisteme de aşteptare de tip HOL, cu creştere constantă a ceasului simulării, iar în [2] este prezentat un algoritm de simulare pentru sisteme cu disciplina FIFO, bazat pe un ceas al simulării cu creştere variabilă. Această lucrare prezintă doi algoritmi de simulare, care se bazează pe un ceas al simulării cu creştere variabilă, care corespunde celor două tipuri de disciplină de servire a clienţilor, bazate pe priorităţi.

2. Algoritm de simulare pentru sisteme de aşteptare HOL

2.1 Descrierea entităţilor modelului de simulare şi a mecanismului simulării

Entitatea **coadă** este caracterizată de atributele:

- nc=(nc(1),...,nc(m)), în care nc(i) (i=1,..,m), reprezintă numărul de clienţi din clasa i aflaţi în coadă;

- vectorul bidimensional Ts=(Ts(i,j), i=1,..,m, j=1,..,nc(i)), nc(i) ≥1; Ts(i,j) conţine timpul de servire al celui de-al j-lea client din clasa i, poziţia în listă fiind dată de ordinea sosirii (disciplina FIFO în cadrul clasei).

- • variabila cs conţine indicele de clasă cea mai înaltă, al clienţilor aflaţi în coadă, şi este dată de:

$$cs = \begin{cases} \min\{i/nc(i) > 0, i = 1,..,m\}, \\ daca\ (\exists)\ i \text{ a.i } nc(i) > 0 \\ m+1,\ altfel \end{cases} \tag{2.1}$$

Entitatea **server** este descrisă de:

- Busy ne indică dacă server-ul (stația de lucru) este liber sau ocupat și are valorile

$$Busy = \begin{cases} 0, \text{ daca server - ul este liber} \\ 1, \text{ daca server - ul este ocupat} \end{cases}$$

- *Ctime* reprezintă timpul eveniment al terminării servirii unui client, adică ceasul stației de lucru. Dacă server-ul este liber, și nu există clienți ce urmează să fie serviți (Busy=0), atunci $Ctime=\infty$.

- variabila *Tsc* conține durata serviciului clientului în curs de servire.

Observația 1. Simularea bazată pe evenimente discrete folosește regula „evenimentului următor" („timpului minim"). În cazul nostru, sunt posibile două evenimente viitoare: sosirea unui client sau terminarea unei serviri. Deci, timpul următorului eveniment este dat de $min\{Atime, Ctime\}$.

Entitatea de generare a valorilor aleatoare va genera timpii între sosiri, notat cu Intariv, timpul de servire, notat cu Stime și clasa clientului sosit, notată cu cp. *Atime*, este timpul eveniment al următoarei sosiri; inițial Atime este 0 și, după fiecare generare, valoarea lui Intariv este adăugată lui Atime. Timpii de servire a clienților de clasă i sunt păstrați în linia i a matricii Ts. Dacă j este numărul clienților de clasă cp aflați în coadă, atunci $Ts(cp,j+1)$ primește valoarea.

Dacă $Atime \leq Ctime$, atunci următorul eveniment este o sosire. Prelucrarea următoarei sosiri constă în: dacă stația este liberă, se actualizează timpul de lenevire al stației și se servește imediat clientul sosit, acțiune care constă în faptul că variabila *Ctime* primește valoarea timpului de servire al clientului sosit; dacă stația este ocupată, se actualizează timpii totali de așteptare în coadă ai clienților, corespunzători claselor de clienți, și se introduce în coadă clientul sosit, ultimul în cadrul clasei sale.

Dacă $Atime > Ctime$, evenimentul următor este terminarea unei serviri. Prelucrarea acesteia constă în: actualizarea timpilor totali de așteptare în coadă ai clienților, actualizarea timpilor totali de servire ai clienților împărțiți pe clase de priorități, incrementarea numărului total de servicii și determinarea valorii variabilei *cs*, în funcție de care se execută una din următoarele acțiuni:

- se preia un nou client pentru a fi servit (timpul de servire al acestui client va fi evident $Ts(cs,1)$). După "selectarea" clientului de prioritate *cs*, acesta este scos din coadă, ceea ce este echivalent cu „translatarea" spre stânga a celorlalți eventuali clienți de clasă *cs* (al doilea client devine primul ș.a.m.d.), datorită faptului că sunt serviți după regula FIFO clienții din aceeași clasă.
- stația intră în lenevire.

Observația 2. Simularea se derulează până când numărul de sosiri generate atinge o valoare dată, notată cu *Tnra*. Dacă definim ca ciclu al simulării prelucrarea unei sosiri sau a unei serviri, practic, simularea se desfășoară prin repetarea acestor cicluri. Dacă facem ipoteza că fluxul sosirilor este mai mic decât cel al servirilor, atunci numărul servirilor nu va depăși valoarea *Tnra*. De asemenea, dacă numărul de sosiri simulate are o valoare suficient de mare, atunci aproape toți clienții vor fi până la urmă serviți. Deci, putem spune că numărul de cicluri ale simulării nu depășește valoarea *2Tnra*, pentru sistemele cu disciplina de servire *primul-sosit-primul-servit* sau *head of line*, și *3Tnra*, în cazul sistemelor cu evacuarea clientului servit, deoarece orice evacuare este consecința unei sosiri.

La sfârșitul simulării, entitatea de determinare a factorilor de eficiență, calculează:

- timpul mediu de așteptare în coadă al clienților din clasa k, MTw(k), exprimat prin: MTw(k)=Tw(k)/Nrsc(k) în care Tw(k), respectiv Nrsc(k), reprezintă timpul total de așteptare în coadă al clienților, respectiv numărul clienților serviți din clasa k, k=1,..,m;

- numărul mediu de clienți din clasa k, aflați în coadă, MLq(k), conform formulei MLq(k)=Tw(k)/Ltime;

- lungimea medie a cozii Mqueue, exprimată prin

$$MQueue = \sum_{k=1}^{m} MLq(k);$$

- timpul mediu de servire al clienților din clasa k, dat de relația: MTs(k)=TTsc(k)/Nrsc(k) TTsc(k) fiind timpul total de servire al clienților din clasa k;

- coeficientul de lenevire al stației de lucru, Clen: Clen←Tlen/Ltime, în care Tlen reprezintă timpul de lenevire, exprimat ca fiind diferența dintre Ltime și timpul total de lucru al stației, Tts iar 1-Clen reprezintă intensitatea de trafic a sistemului.

2.2 Descrierea algoritmului de simulare și studiul complexității

În continuare este prezentat algoritmul pseudocod, de simulare al acestui tip de sistem.

Procedure SistAstMclaseOstațieHOL(m,Tnra);

Citește: Parametrii de generare ai intervalului între două sosiri consecutive, ai timpului de servire, ai clasei de priorități;

```
for i=1,m do
        nc(i)←0;Tw(i)←0;
        Tsc(i) ←0;Nrsc(i)←0
endfor;
busy←0;Ltime←0; Ctime←∞; nrs←0;
Gen(IntAriv,cp,Stime);
Atime←IntAriv; Nra←1;
  While Nra≤Tnra do
      If Atime≤Ctime then {Aevent}
       If busy=0 then {Stația este liberă}
         Tlen←Tlen+Atime-Ltime;
         busy←1;Tsc←Stime;
        Ctime←Atime+Stime;
               ks←cp;
                      else
{Actualizarea timpilor de  așteptare  în coadă}
               for i=1,m do
                Tw(i)←Tw(i)+nc(i)*(Atime-Ltime)
                endfor;
              nc(cp)←nc(cp)+1;
{Se introduce în coadă clientul sosit}
                Ts(cp,nc(cp))←Stime;
           endif;
         Gen(IntAriv,cp,Stime);
         Ltime← Atime;  Atime←Atime+IntAriv;
         Nra←Nra+1
          else{Cevent}
           for i=1,m do
{Actualizarea timpilor de așteptare în coadă}
              Tw(i)←Tw(i)+ nc(i)*(Ctime-Ltime)
```

```
            endfor;
            Ltime←Ctime;Nrsc(cs)←Nrsc(cs)+1;
            Tts(cs)←Tts(cs)+Tsc;
      {Selectarea unui nou client pentru servire}
                {cs=min{i/nc(i)>0;i=1,..,m}}
                cs←1;
              while (cs<m) and (nc(cs)=0) do
                        cs←cs+1
              endwhile;
                if ( cs=m) and (nc(cs)=0) then {coada este vidă}
                          busy←0;  ctime←∞;
                    else
      {există clienți în coadă}
                        ctime←ctime+Ts(cs,1);       Tsc←Ts(cs,1);
                        for k=1,nc(cs)-1 do
                              Ts(cs,k)←Ts(cs,k+1)
                        endfor;
                        nc(cs)←nc(cs)-1;
              endif;
      endif
   endwhile
      Mqueue←0;Tts←0;Nrs←0;
      for i=1,m
         MTw(i)←Tw(i)/Nrsc(i);
         Mtsc(i)←Ttsc(i)/Nrsc(i);
         MLq(i)←Tw(i)/Ltime;
          Mqueue←Mqueue+ MLq(i);
         Nrs←Nrs+Nrsc(i);  Tts←Tts+Ttsc(i);
      Endfor;
      Clen←Tlen/Ltime;  MTs←Tts/Nrs;
       Scrie(MTw(i),Mtsc(i),MLq(i),i=1..m), Clen, Mqueue, MTs
end
```

Teorema 1. Algoritmul prezentat are un comportament polinomial, dacă metodele de generare ale timpilor între sosiri, duratei de servire şi clasei de priorități au un comportament polinomial.

Demonstrație. Pentru a arăta că algoritmul are un comportament polinomial, vom estima numărul maxim de comparații. Pentru aceasta, vom folosi unele rezultate fundamentale ale teoriei algoritmilor, privind numărul de comparații ale structurilor de bază [1]. În afara secvenței care descrie derularea ciclurilor simulării, se execută $3(m+1)$ comparații. Testarea condiției de continuare (sau nu) a simulării, va genera $2Tnra+1$ comparații (**observația 2**). Pentru fiecare ciclu al simulării, avem o comparație pentru a determina tipul evenimentului următor. Ramura *{aevent}* va fi executată de *Tnra* ori corespunzător celor *Tnra* sosiri. Pe această ramură, avem: o comparație pentru a testa dacă stația este în lenevire; dacă stația nu este în lenevire, se mai adaugă $m+1$ comparații pentru actualizarea timpilor de aşteptare, la care se adaugă numărul de comparații ale metodei de generare, pe care îl notăm cu *cmg*.

Pe ramura *{Cevent}* (care se execută de cel mult Tnra ori, conform observației 2) vom avea:

- $m+1$ comparații pentru actualizarea timpilor de aşteptare în coadă;
- cel mult $2(m+1)$ comparații pentru determinarea valorii variabilei *cs*;

- o comparaţie pentru a testa dacă coada este vidă sau nu, la care se adaugă $nc(cs)+1$ comparaţii pe ramura în care coada este nevidă, adică, cel mult $1+Tnra$ comparaţii, deoarece $nc(cs)\leq Tnra$ (numărul clienţilor unei clase nu poate depăşi numărul total de sosiri).

Deci, folosind **observaţia 1**, numărul total de comparaţii va fi mai mic decât

$$3(m+1)+1+Tnra+1+2Tnra+Tnra(1+m+1+cmg+m+1+2(m+1)+1+Tnra).$$

Cum numărul de clase m are o valoare mică, în raport cu numărul evenimentelor simulate $Tnra$, care trebuie să fie suficient de mare pentru a avea o situaţie relevantă şi dacă presupunem că numărul de comparaţii ale metodei de generare este neglijabil, rezultă că complexitatea este $O(Tnra^2)$.

3. Modelul cu evacuarea clientului servit

3.1 Entităţile modelului

Entitatea **coadă** este caracterizată de atributele nc şi Ts amintite în cazul *head of line*, la care se adaugă vectorul bidimensional Tc, $Tc=(Tc(i,j),\ i=1,..,m,\ j=1,..,nc(i))$, $nc(i)\geq 1$; $Tc(i,j)$ are valoarea 0, respectiv o valoare întreagă mai mare decât 0, după cum al j-lea client din clasa i nu este evacuat sau este evacuat; în momentul sosirii, componenta corespunzătoare a vectorului este iniţializată cu 0 şi la fiecare evacuare este incrementată.

Entitatea **server** este descrisă de variabilele *Ctime*, *busy*, *Tsc*, care au aceeaşi semnificaţie cu cea prezentată pentru modelul anterior, şi variabila *tcs* care reprezintă tipul clientului în curs de servire.

3.2. Mecanismul simulării şi descrierea algoritmului

Acest algoritm este diferit faţă de cel prezentat anterior, pe ramura *{aevent}*. Diferenţele apar atunci când are loc o sosire şi staţia este ocupată. În cazul sosirii unui client de prioritate mai tare decât cea a clientului servit, sunt actualizate valorile vectorilor care dau timpii totali de servire şi ale celor ce dau numărul de servicii, clientul în curs de servire este evacuat, este introdus în coadă pe prima poziţie în cadrul clasei sale, cu timpul de servire rămas până la completarea serviciului, adică diferenţa dinte *Ctime* şi *Atime*, incrementându-se *tc*-ul său, urmând să fie servit noul client sosit.

La sfârşitul simulării, se determină **factorii de eficienţă**:

- timpul mediu de aşteptare în coadă al clienţilor din clasa k neevacuaţi, MTw(k), exprimat prin: MTw(k)=Tw(k)/Nrsc(k) în care Tw(k), respectiv Nrsc(k) reprezintă timpul total aşteptare al clienţilor, respectiv numărul clienţilor serviţi neevacuaţi din clasa k, k=1,..,m;

- timpul mediu de aşteptare în coadă al clienţilor din clasa k evacuaţi, MTwe(k), exprimat prin: MTwe(k)=Twe(k)/Nrsce(k) în care Twe(k), respectiv Nrsce(k), reprezintă timpul total de aşteptare al clienţilor, respectiv numărul clienţilor serviţi, evacuaţi din clasa k, k=1,..,m;

- numărul mediu de clienţi din clasa k aflaţi în coadă, MLq(k), conform formulei MLq(k)=(Tw(k)+ Twe(k))/Ltime;

- lungimea medie a cozii Mqueue, exprimată prin

$$MQueue=\sum_{k=1}^{m}MLq(k);$$

- timpul mediu de servire al clienţilor din clasa k neevacuaţi, dat de relaţia: MTsc(k)=TTsc(k)/Nrsc(k);

TTsc(k), respectiv Nrsc(k), fiind timpul total de servire, respectiv numărul clienţilor neevacuaţi serviţi din clasa k;

- timpul mediu de servire al clienţilor din clasa k evacuaţi, dat de relaţia: MTsce(k)=TTsce(k)/Nrsce(k);

TTsce(k), respectiv Nrsce(k), fiind timpul total de servire, respectiv numărul clienţilor evacuaţi serviţi din clasa k;

- timpul mediu de servire al stației MTs, exprimat prin MTs←TTs/Nrs, în care Tts este timpul total de servire, iar Nrs reprezintă numărul total de servicii;

- coeficientul de lenevire al stației de lucru, Clen: Clen←Tlen/Ltime; în care, Tlen reprezintă timpul total de lenevire, iar 1-Clen reprezintă intensitatea de trafic a sistemului sau coeficientul de utilizare a stației.

În continuare, voi prezenta procedura pseudocod, care descrie algoritmul de simulare pentru acest model.

Procedure SistAstPrioritEvacOstat(m,Tnra);

Citește: *Parametrii de generare ai intervalului între două sosiri consecutive, ai timpului de servire, ai clasei de priorități;*

```
for i=1,..,m do
  Tw(i)←0;  Twe(i)←0; nc(i)←0;  Nrsc(i)←0 ;
  TTsc(i)←0 ; Nrsce(i)←0;
   TTsce(i)←0 ;
 endfor;
 busy←0;Ltime←0; Ctime←∞;
Gen(IntAriv, cp, Stime);
Atime←IntAriv;Nra←1;
 While Nra≤Tnra do
   If Atime≤Ctime then {Aevent}
     if busy=0 then
   {Actualiz. Timp total lenevire}
         Tlen←Tlen+(Atime-Ltime);
         Ctime←Atime+Stime; Tsc←Stime; cs←cp;Tcs←0
         {Clientul sosit este servit imediat}
                 else
{Actualizarea timpului de aşteptare în coadă}
         for i=1,m do
           for j=1,nc(i) do
            if tc(i,j)=0 then
            Tw(i)←Tw(i)+Atime-Ltime
                 else
             Twe(i)←Twe(i)+Atime-Ltime
          endif
        endfor;
       endfor;
       if cs≤cp  then
{Clientul sosit este introdus în coadă}
         nc(cp)← nc(cp)+1;
          Ts(nc,nc(cp))←Stime;
         Tc(nc,nc(cp))←0
              else
         {are loc o evacuare}
          If Tcs=0 then
              Nrsc(cs)←Nrsc(cs)+1;
            Ttsc(cs)←Ttsc(cs)+Tsc-(Ctime-Atime)
                 Else
           Nrsce(cs)←Nrsce(cs)+1;
          Ttsce(cs)←Ttsce(cs)+Tsc-(Ctime-Atime)
```

```
        Endif;
        {Clientul evacuat este (re)introdus în coadă pe prima poziţie}
        nc(cs)←nc(cs)+1;
        for i=nc(cs), 2 downto
          Ts(cs,i)←Ts(cs,i-1); Tc(cs,i)←Tc(cs,i-1);
        Endfor;
        Tc(cs,1)←tcs+1;
        Ts(cs,1)←Ctime-Atime{Timpul rămas de servire};
        cs←cp; Ctime←Atime+Stime; tcs←0; Tsc←Stime
    endif;
   Gen(IntAriv,cp,Stime);
   Ltime←Atime; Atime←Atime+IntAriv; Nra←Nra+1;
        else {Cevent}
{Actualizarea timpului de aşteptare în coadă}
   for i=1,m do
      for j=1,nc(i) do
         if tc(i,j)=0 then Tw(i)←Tw(i)+Ctime-Ltime
                   else Twe(i)←Twe(i)+Ctime-Ltime
         endif
     endfor;
   endfor;
{Actualizarea timpilor totali de servire şi a nr. de servicii}
   If tcs=0 then
     Nrsc(cs)←Nrsc(cs)+1; Ttsc(cs)←Ttsc(cs)+Tsc
             Else
        Nrsce(cs)←Nrsce(cs)+1;        Ttsce(cs)←Ttsce(cs)+Tsc
   Endif;
        cs←1;
         while (cs<m) and (nc(cs)=0) do
                  cs←cs+1
         endwhile;
         Ltime←Ctime;
         If (cs=m) and (nc(cs)=0) then
        {staţia intră în lenevire}
                 busy←0; Ctime←∞;
                   Else
        {staţia va servi primul client de clasă cea mai înaltă cs}
                 Ctime←Ctime+Ts(cs,1); Tsc← Ts(cs,1); Tcs← Tc(cs,1);
         {"translatare" spre stânga=scoterea din coadă a clientului servit}
                 for j=1,nc(cs)-1 do
                    Ts(cs,j)←Ts(cs,j+1); Tc(cs,j)←Tc(cs,j+1)
                 Endfor;
                   nc(cs)←nc(cs)-1;
             Endif;
        Endif
   Endwhile;
nrs←0;Tts←0; MQueue←0;
```

for i=1,m

Tw(i)←Tw(i)/Nrsc(i);MTwe(i)←Twe(i)/NrSce(i); Mtsc(i)←Ttsc(i)/Nrsc(i); Mtsce(i)←Ttsce(i)/Nrsce(i);

MLq(i)←(Tw(i)+Twe(i))/Ltime; MQueue←MQueue+Mqueue(i); nrs←nrs+nrsc(i)+ nrsce(i);

Tts←TTs+Ttsc(i)+Ttsce(i)

Endfor;

Mts←Tts/nrs;Clen←Tlen/Ltime;

Scrie(Tw(i),Mtwe(i),MLq(i),Mtsc(i),Mtsce(i), i=1..m), Clen,Mts,MQueue)

End.

Teorema 2. Dacă fluxul intrărilor este mai mic decât cel al servirilor şi presupunem că metoda de generare a intervalului de timp între sosiri, duratei timpului de servire şi clasei au un comportament polinomial, atunci, algoritmul de simulare pentru un sistem de aşteptare cu evacuare cu o singură staţie de servire are un comportament polinomial.

Demonstraţie. În afara ciclurilor simulării, se execută $3(m+1)$ comparaţii. Testarea condiţiei de continuare a simulării va genera $Tnra+1$ comparaţii. Deoarece avem $Tnra$ sosiri şi aproape toţi clienţii vor fi serviţi, deoarece fluxul intrărilor este mai mic decât cel al ieşirilor şi orice evacuare este consecinţa unei eventuale sosiri, rezultă că numărul comparaţiilor necesare pentru a determina tipul următorului eveniment va fi de cel mult $3Tnra+1$.

Pe ramura *{aevent}*, care se execută de Tnra ori avem: o comparaţie pentru a testa dacă staţia este ocupată; dacă este ocupată, vom avea în plus: $m(2Tnra+1)+1$ comparaţii pentru actualizarea timpilor ($nc(i)\leq Tnra$, $i=1,..,m$) de aşteptare în coadă; o comparaţie pentru a testa dacă clientul sosit va produce o evacuare sau nu; dacă produce o evacuare vom mai avea o comparaţie pentru a testa tipul clientului evacuat, când se actualizează timpii totali de servire şi numărul total de servicii; cel mult $Tnra$ comparaţii pentru a introduce în coadă clientul în curs de servire ($nc(cs)\leq Tnra$); numărul de comparaţii ale metodei de generare, notat cu *cmg*.

Pe ramura {*Cevent*}, care se execută de cel mult $2Tnra$ ori, vom avea: cel mult $m(2Tnra+1)+1$ comparaţii, pentru actualizarea timpilor totali de aşteptare în coadă; o comparaţie pentru actualizarea timpilor totali de servire şi a numărului total de servicii; cel mult $2(m+1)$ comparaţii pentru selectarea clasei clientului care urmează să fie servit; *o* comparaţie pentru a testa dacă există clienţi în coadă, la care se mai adaugă cel mult $Tnra$ comparaţii pentru scoaterea din coadă a clientului care urmează să fie servit.

Rezultă că numărul maxim de comparaţii va fi:

$$3(m+1)+4Tnra+1+Tnra(4+m(2Tnra+1)+Tnra+ Cmg)+2Tnra((m(2Tnra+1)+1)+1+2(m+1)+1+ +Tnra)$$

Deoarece m este o constantă care are o valoare mică în raport cu $Tnra$ care trebuie să fie suficient de mare pentru ca rezultatele simulării să se stabilizeze, iar *cmg* îl presupunem că are o valoare mică, va rezulta că complexitatea algoritmului (numărul maxim de comparaţii) va fi $O((6m+3)Tnra^2)$.

4. Validarea modelelor şi consideraţii practice

4.1 Modelul Head of Line

i) Considerăm un model cu două clase de priorităţi, în care presupunem că proporţia clienţilor de clasă 1 este K, iar a celor de clasă 2 este 1-K ([5], [6]).

Numărul mediu de clienţi în coadă de clasă $1(LQ_1)$, respective de clasă $2\ (LQ_2)$, sunt:

$$LQ_1 = \frac{K\rho^2}{1-K\rho} \tag{4.1}$$

$$LQ_1 = \frac{(1-K)\rho^2}{(1-\rho)(1-K\rho)} \tag{4.2}$$

Timpul mediu de aşteptare în sistem al clienţilor de clasă $1\ (WT_1)$, respectiv al clienţilor de clasă $2\ (WT_2)$, are expresiile:

$$WT_1 = \frac{\lambda}{\mu(\mu - K\lambda)} \quad (4.3)$$

$$WT_2 = \frac{\lambda}{(\mu - \lambda)(\mu - K\lambda)} \quad (4.4)$$

De asemenea, media timpului de servire este $1/\mu$, iar intensitatea de trafic este λ/μ.

Dacă se consideră modelul în care intervalul între sosiri, respectiv durata serviciului, sunt exponențiale negative de parametri *0.5*, respectiv *1*, numărul claselor de priorități este *2* și numărul sosirilor simulate este *10000*, obținem, **prin simulare**, valorile factorilor de eficiență din tabelul 1. **Analitic** (folosind formulele (4.1), (4.2), (4.3), (4.4)) obținem valorile din tabelul 2.

Tabelul 1.

Factorul de eficiență	Valoarea
MTW[1]	0.6563
MTW[2]	1.3337
MTS[1]	1.0039
MTS[2]	0.9904
Intensitatea de trafic	0.5100
MLQ[1]	0.1635
MLQ[1]	0.3478

Tabelul 2.

Factorul de eficiență	Valoarea
MTW[1]	0.6643
MTW[2]	1.3285
MTS[1]	1.0000
MTS[2]	1.0000
Intensitatea de trafic	0.5000
MLQ[1]	0.1643
MLQ[1]	0.3357

Observăm că rezultatele obținute prin simulare sunt foarte apropiate de cele obținute analitic.

ii) Modelul în care intervalul între sosiri, respectiv durata serviciului sunt exponențiale, negative, de parametri 0.5, respectiv 1, numărul claselor de priorități este 1 și numărul sosirilor simulate este 10000. Rezultatele obținute prin simulare sunt prezentate în tabelul 3.

Tabelul 3.

Factorul de eficiență	Valoarea
MTW[1]	0.99216
MTS[1]	1.00326
Intensitatea de trafic	0.49928
MLQ	0.49875

Observația 3. Deoarece avem o singură clasă de priorități și în cadrul clasei se respectă ordinea sosirilor, modelul este, de fapt ,unul cu disciplina *primul-sosit primul-servit*. Rezultatele simulării concordă cu această observație, fiind aproximativ egale celor obținute analitic în cazul *primul-sosit primul-servit* ([2]).

4.2. Modele cu evacuarea clientului servit

i) Vom considera modelul următor, cu două clase de priorități: Dacă λ este intensitatea fluxului intrărilor, μ este intensitatea fluxului servirilor, K este proporția clienților de clasă 1, 1-K este proporția clienților de clasă 2, $\rho1=K\lambda/\mu$, $\rho2=(1-K)\lambda/\mu$, atunci, mediile timpilor totali de așteptare în coadă a clienților de clasă 1, respectiv clasă 2, notate cu WT1, respectiv WT2, se exprimă prin:

$$WT_1 = \frac{\rho_1}{\mu(1 - \rho_1)} \quad (4.5)$$

$$WT_2 = \frac{1}{\mu(1-\rho_1-\rho_2)}\left(\rho_1+\rho_2+\frac{\rho_1}{1-\rho_1}\right) \quad (4.6)$$

Prin **simulare**, dacă intervalul de timp între sosiri, respectiv durata serviciilor sunt exponenţiale, negative, de parametri 1, respectiv 2, avem două clase de priorităţi şi numărul evenimentelor simulate este 5000, obţinem valorile din tabelul 4.

Tabelul 4.

Factorul de eficienţă	Valoarea
MTW[1]	0.28124
MTW[2]	1.57819
MTS[1]	1.02153
MTSe[1]	0.00000
MTS[2]	0.81120
MTSe[2]	0.77288
MLQ	0.49757
Intensitatea de trafic	0.50248

Analitic, pe baza formulelor (4.5), (4.6) obţinem valorile din tabelul 5.

Tabelul 5.

Factorul de eficienţă	Valoarea
MTW[1]	0.27492
MTW[2]	1.54979

Observăm că mediile timpilor de aşteptare în coadă, determinate prin simulare, sunt apropiate de cele determinate analitic în cazul considerat. De asemenea, timpul mediu de servire al clienţilor de clasă 1 este 0, ceea ce concordă cu faptul că clienţii de clasă 1 nu pot fi evacuaţi.

ii) Parametrii modelului sunt aceeaşi ca şi în cazul anterior, dar numărul claselor de priorităţi este 1. Rezultatele obţinute prin simulare, în acest caz, sunt prezentate în tabelul 6.

Tabelul 6.

Factorul de eficienţă	Valoarea
MTW[1]	0.95864
MTS[1]	0.99970
MTSe[1]	0.00000
Intensitatea de trafic	0.49582
MLQ	0.49757

Observaţia 4. Deoarece, în acest caz, nu este posibilă evacuarea clientului servit, acesta va fi echivalent cu un model HOL, care coincide cu un model FIFO. Rezultatele execuţiei programului confirmă acest lucru.

Bibliografie

1. **CORRMEN, T.H. C.E., LEIRSON, R.L., RIVEST**: Introductions to Algorithms, MIT Press Cambridge, 1992.
2. **FLOREA, I.**: One Algorithmic Approach of First-Come-First-Served Queuing Systems. În: Analele Universităţii Bucureşti, anul X – 2000.
3. **KARIAN, Z.A, E. J. DUDEWICZ**: Modern Statistical Systems and GPSS Simulation. În: Computer Science Press Inc, 1990.
4. **KLEINROCK ,L.**: Queuing Systems, Volume I: Theory, John Wiley & Sons, New York, 1975.
5. **KLEINROCK, L.**: Queuing Systems, Volume II: Computer Applications, John Wiley & Sons, 1976.
6. **LEE, A.M.**: Teoria aşteptării cu aplicaţii, Editura Tehnică, Bucureşti, 1976.
7. **MIHOC, GH., G. CIUCU, A. MUJA:** Modele matematice ale aşteptării, Editura Academiei, Bucureşti, 1973.
8. **TANNER, M.**: Practical Queuing Analysis, McGraw-Hill Book Company, 1995.
9. **VĂDUVA, I.**: Modele de simulare cu calculatorul, Editura Tehnică, Bucureşti, 1977.
10. **VĂDUVA, I., M. STOICA, I. ODĂGESCU**: Simularea proceselor economice, Editura Tehnică, Bucureşti, 1983.