

TOMOIU CRISTINA-ANDREEA

Calculatoare română, grupa 3.3b

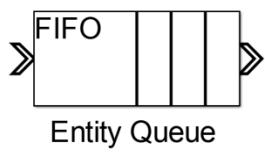
PORTOFOLIU PRACTIC MODELARE ȘI SIMULARE

Lucrarea 1

BLOCURI NOI FOLOSITE

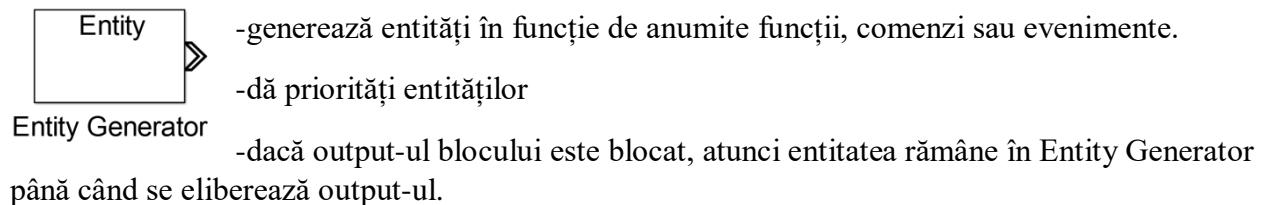
În acest laborator au fost prezentate câteva blocuri din biblioteca SimEvents și Scope-ul. O să explic ce face fiecare dintre cele pe care le-am folosit.

1. ENTITY QUEUE

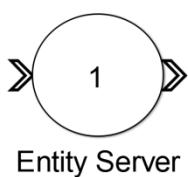


- memorează entități într-o coadă, unde pot fi sortate în funcție de prioritate sau momentul de timp în care au intrat în coadă
- stiva începe să se golească doar atunci când blocul de downstream (cel la care este legat cu săgeata dreaptă) poate primi entități.

2. ENTITY GENERATOR



3. ENTITY SERVER



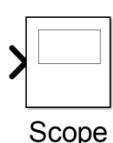
- gestionează o singură entitate la un anumit moment dat
- dacă output-ul blocului este blocat, atunci entitatea rămâne în Entity Server până când se eliberează output-ul.

4. ENTITY TERMINATOR



- se asemănă cu blocul Terminator, doar că este pentru entități
- poate fi privit ca un ground, ceea ce intră în interiorul său este distrus.

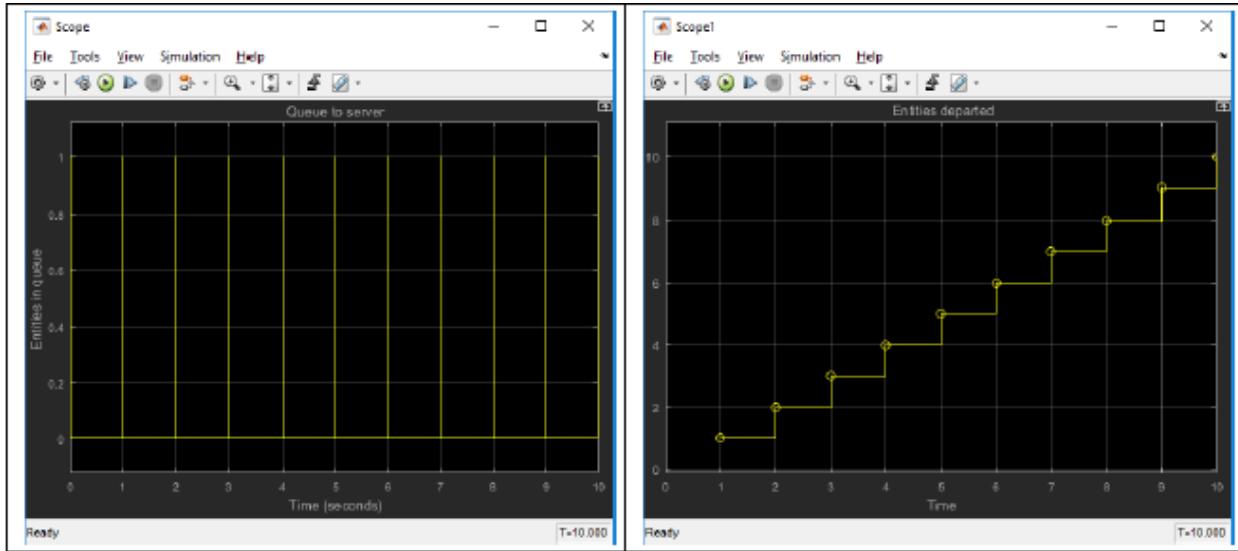
5. SCOPE



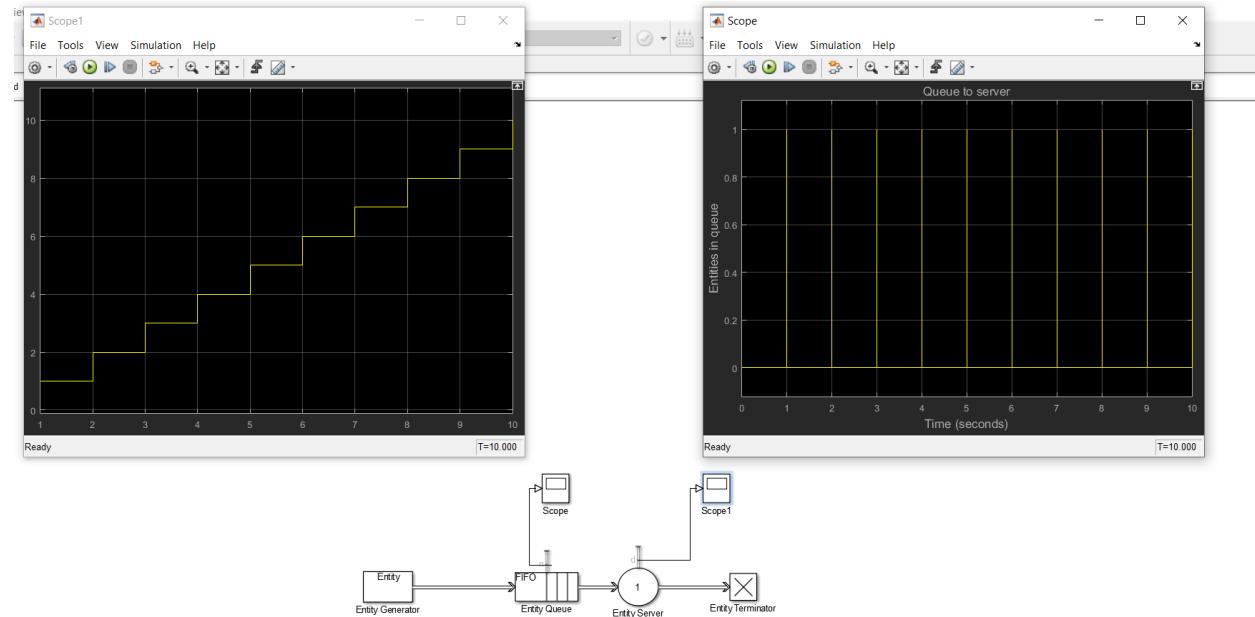
- folosit pentru a vedea evoluția semnalului pe tot parcursul rulării aplicației

EXERCIȚIUL EXPLICAT ÎN PLATFORMA DE LABORATOR

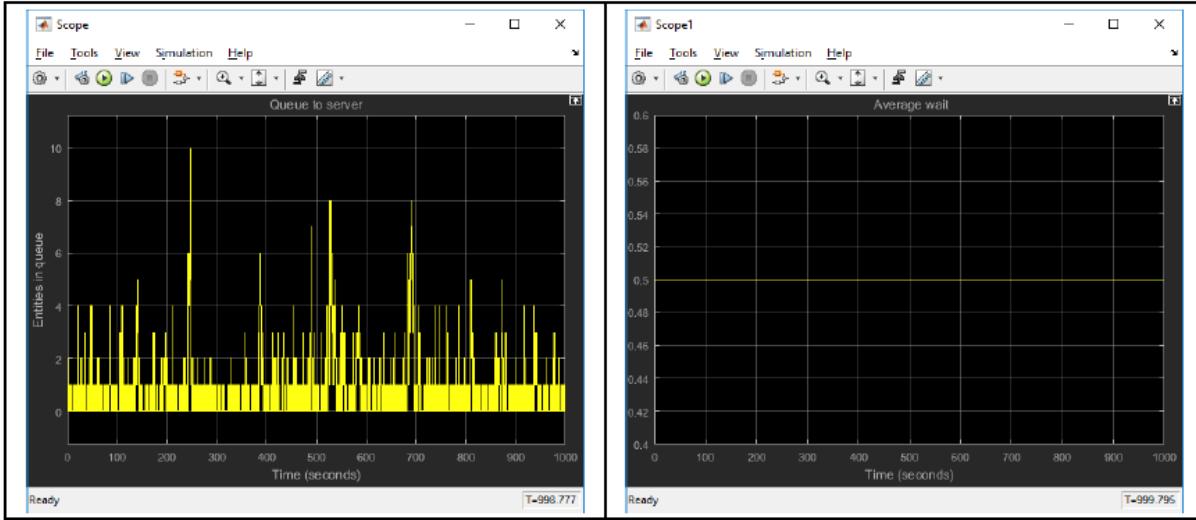
Am lucrat în paralel cu platforma de laborator, iar tabelul 8 corespunde output-urilor semnalelor afişate pe Scope-uri.



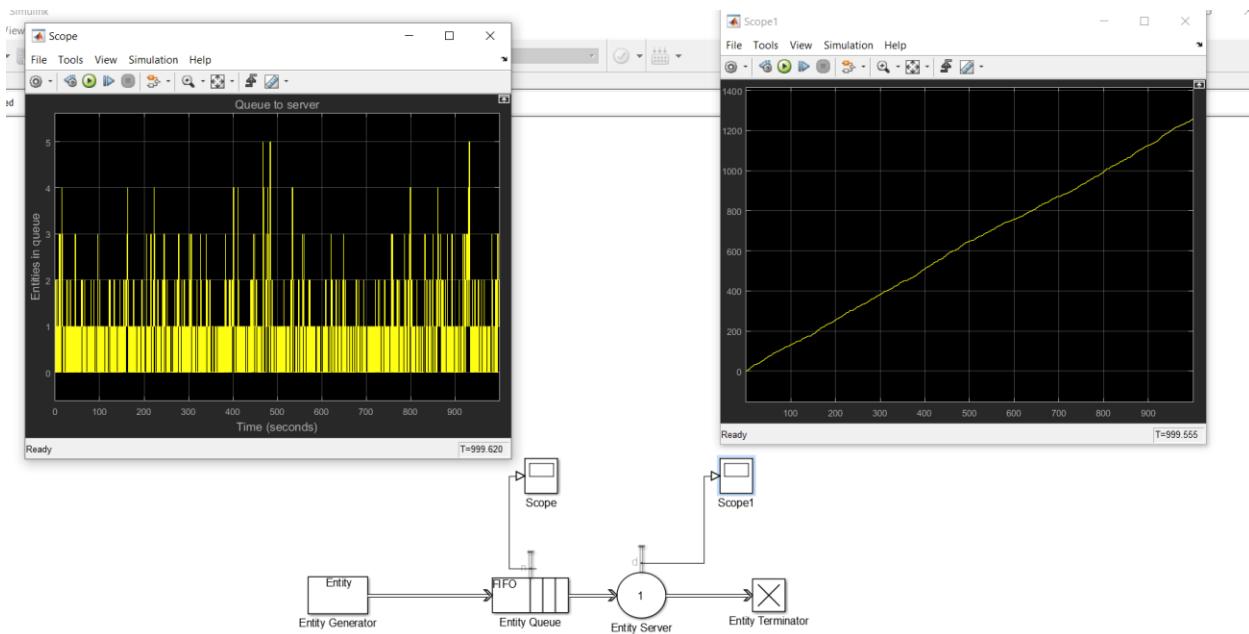
Tabelul 8. Rezultatul simulării modelului din Figura 9



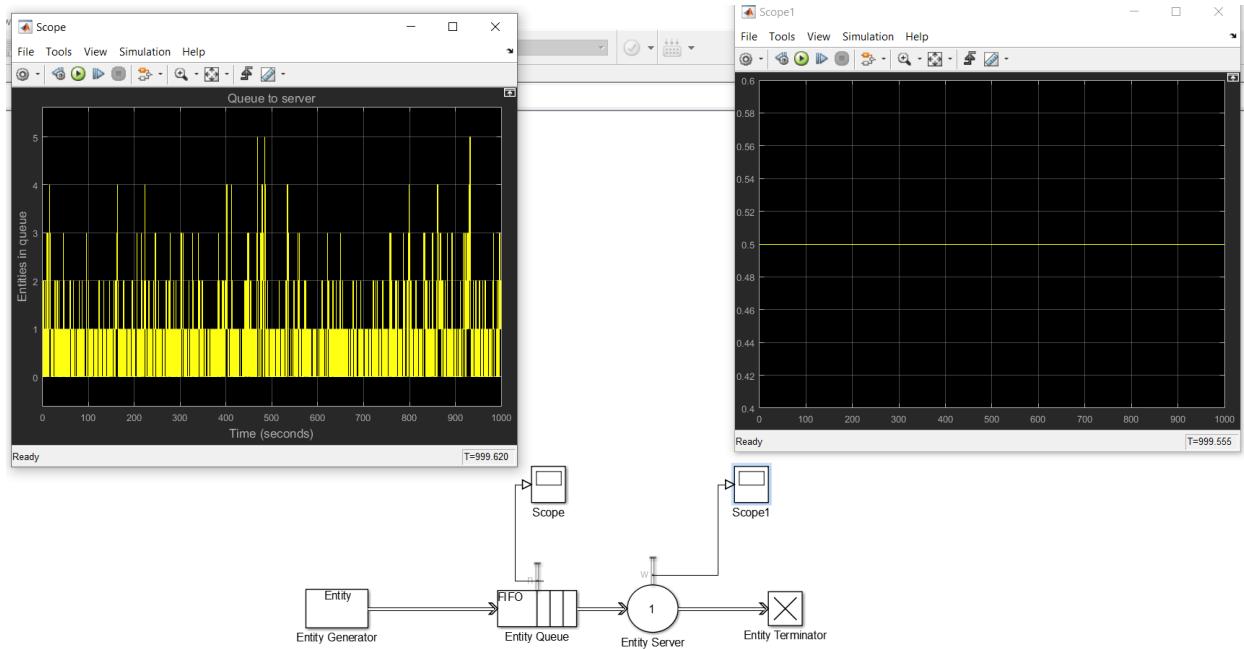
Am modificat în continuare modelul aşa cum cerea platforma de laborator – am setat în blocul Entity Generator timpul între apariția a două entități successive printr-un MATLAB action $dt = \text{exprnd}(0.8)$, iar la Entity Server am setat timpul de deservire 0.5s. Totuși, rezultatul simulării nu corespunde cu cel afișat în tabel.



Tabelul 10. Rezultatul simulării modelului din Figura 9



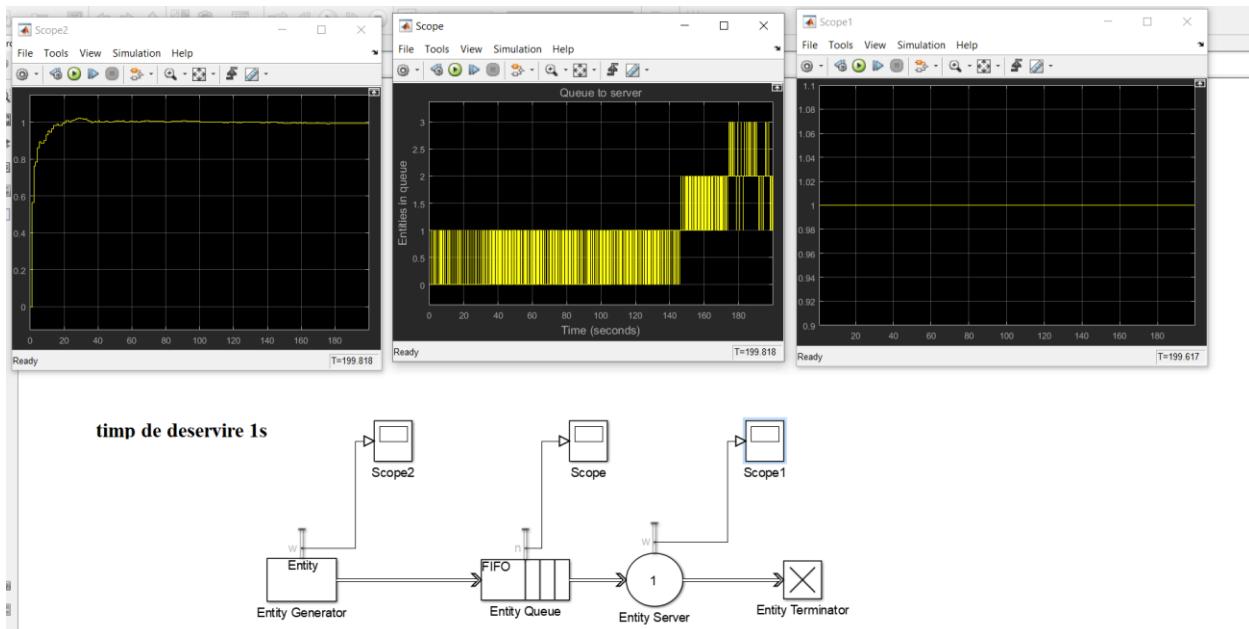
Motivul pentru care cele două simulări nu corespund este că, pe Scope1, ceea ce trebuie afișat nu este *number of entities departed*, d , ci *average wait*, w . După ce am făcut modificarea, tabelele au corespuns.

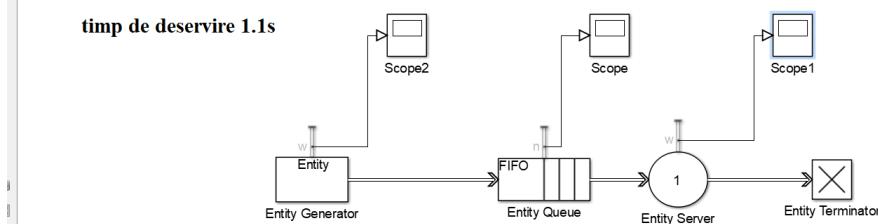
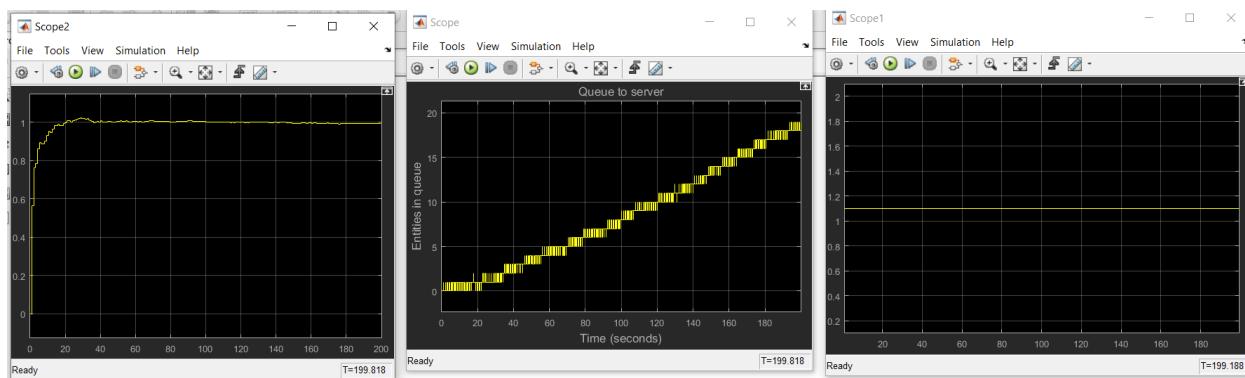
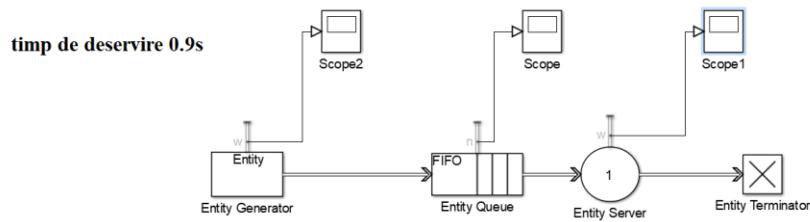
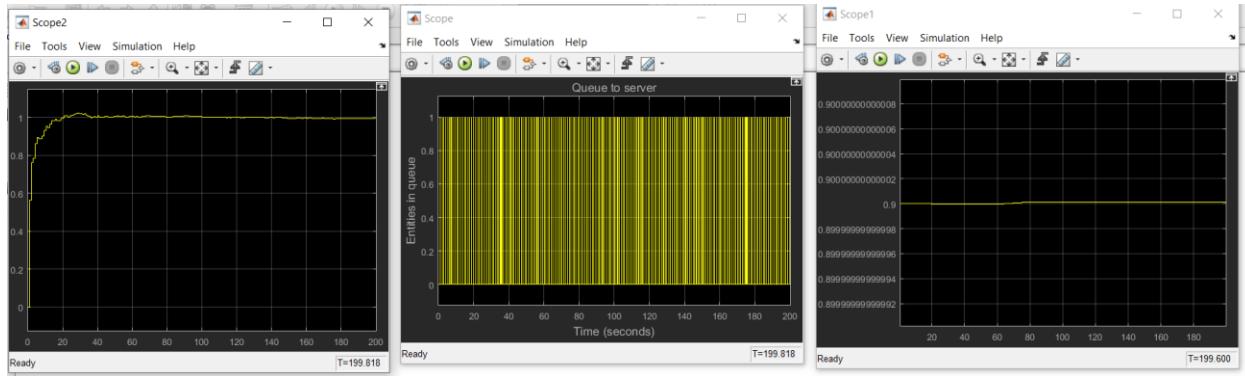


4. Probleme propuse

PROBLEMA 1

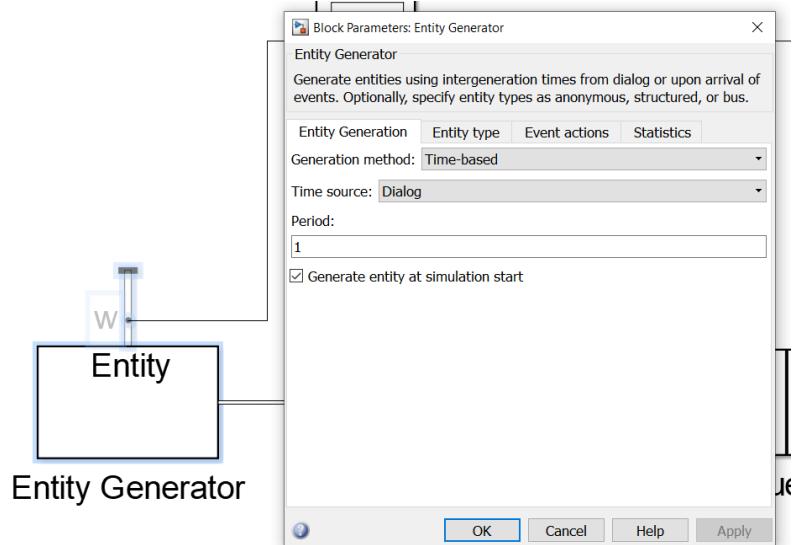
Am să atașez ceea ce se afișează pentru următoarele valori ale timpului de deservire (în această ordine) – 1 , 0.9 , 1.1 .



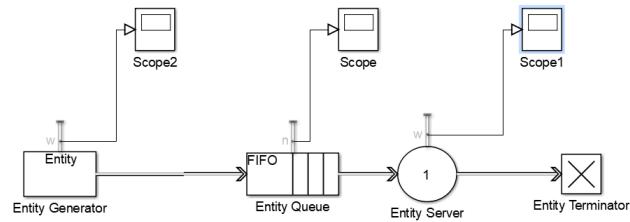
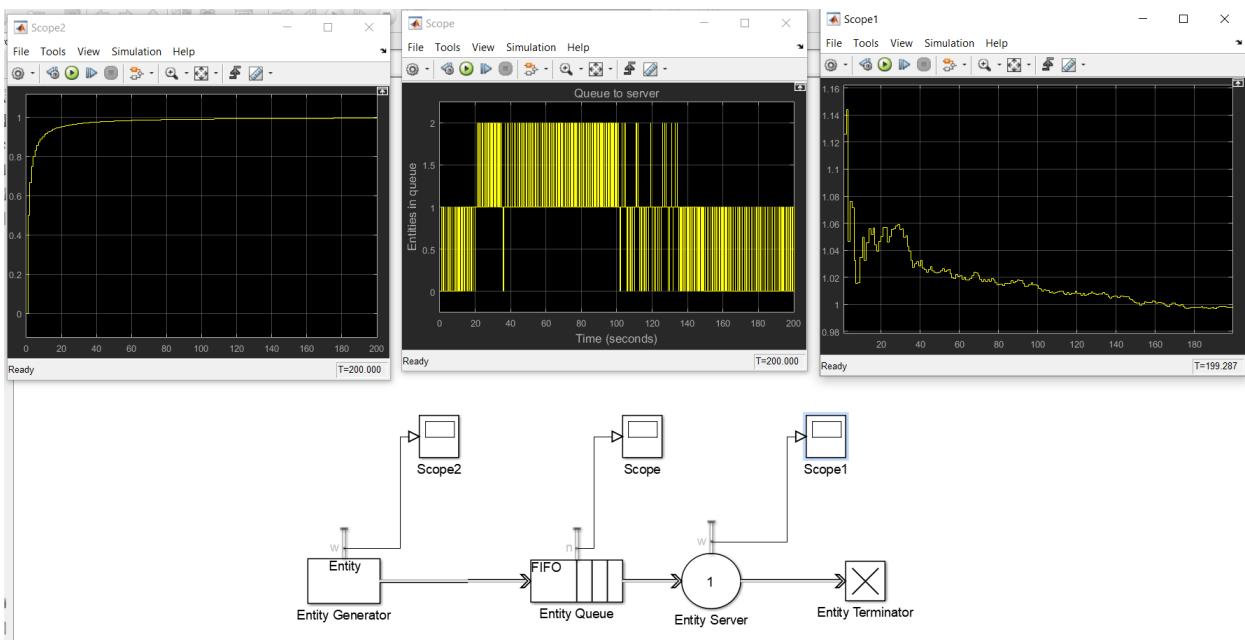


PROBLEMA 2

Intervalul dintre apariția a două entități este 1s, deci

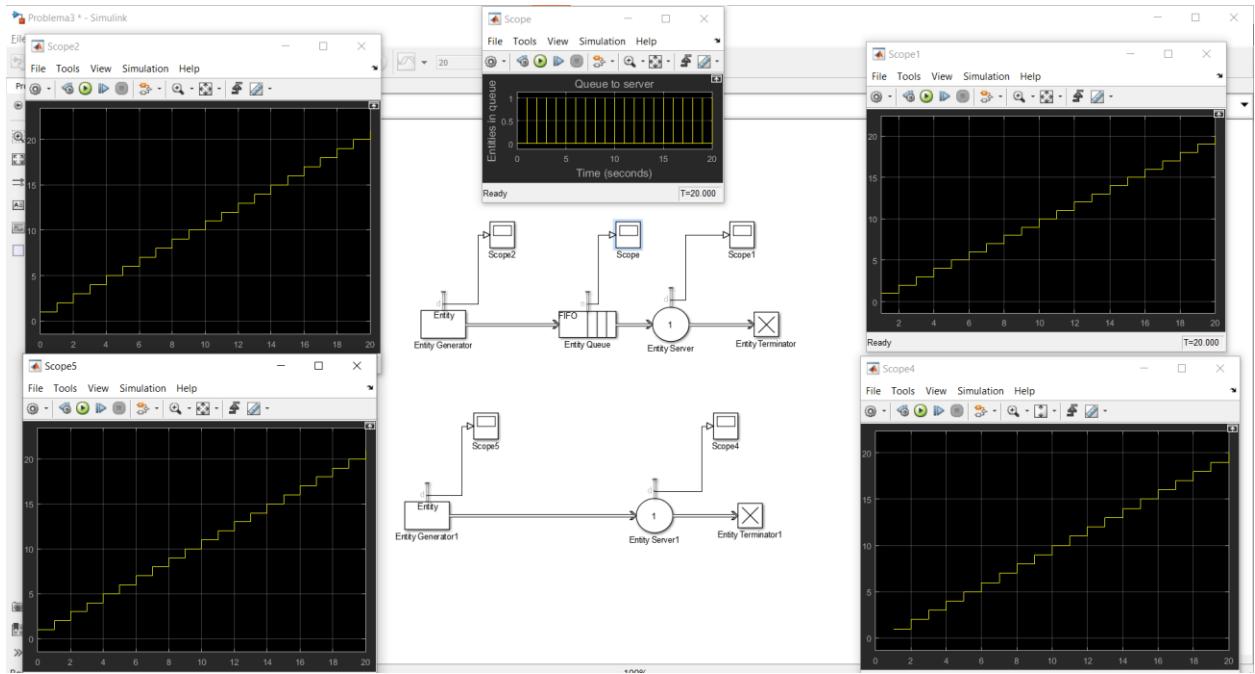


Entity Generator



PROBLEMA 3

După ce am rulat aplicația



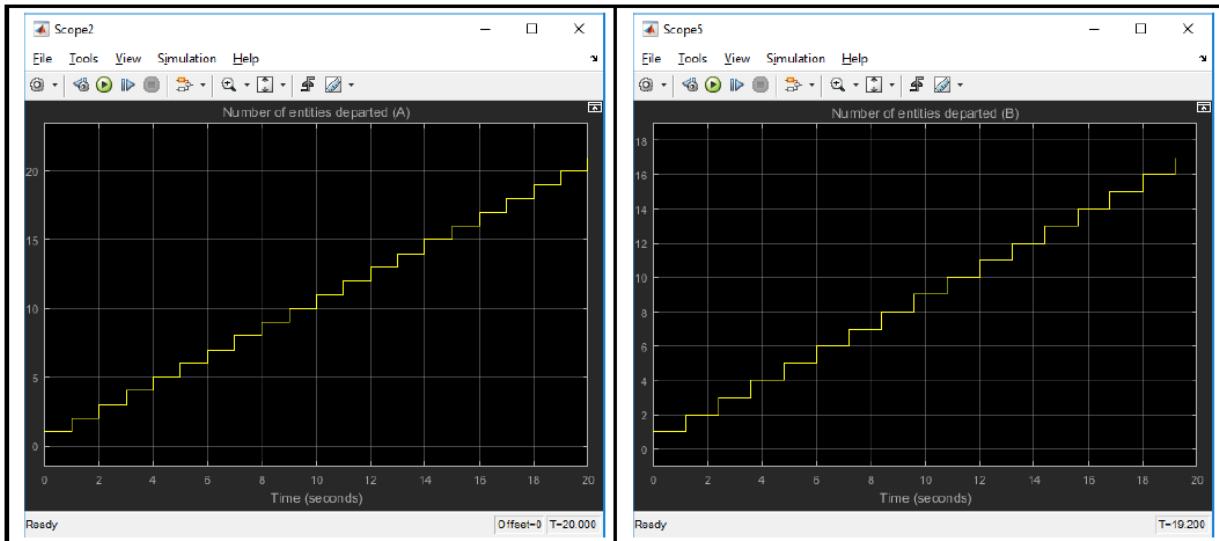
Valorile afișate pe Entity Server sunt identice, deoarece timpul de deservire și timpul între apariția a două entități diferite au aceeași valoare. Acest lucru înseamnă că entitatea generată poate fi deservită imediat (în sample-ul următor). O să explic câteva sample-uri ale aplicației:

0 – 1s - Se deservează prima entitate în Entity Generator

1s – 2s - Prima entitate trece din Entity Generator în Entity Server, iar acum că Entity Generator este gol, se deservează o nouă entitate.

2s - 3s - Entitatea din Entity Server este trimisă către Entity Terminator, astfel eliberându-se blocul. Entity Generator are deja o entitate deservită, iar acum că blocul Entity Server este gol, o trimit către el. Pentru că Entity Server este acum gol, se va deservi o nouă entitate.

2s-3s este recurrent în aplicație, adică toate intervalele de o secundă care îi urmează vor fi pe același principiu.



Tabelul 12. Numărul de entități generate de cele două blocuri Entity Generator în cazul 2

Pentru al doilea caz, diferența de timp dintre timpul de deservire și timpul între apariția a două entități diferite este 0.2. Se produce un fel de delay între Entity Server și Entity Generator. Entitățile sunt generate mai încet decât pot fi procesate de Entity Generator. Acest lucru înseamnă că, chiar dacă se poate genera o entitate la un anumit moment de timp, acest lucru nu se va face pentru că entitatea curentă nu are cum să treacă mai departe.

temp de deservire = service time value; în ENTITY SERVER

timpul între apariția a două entități diferite - în ENTITY GENERATOR

$dt = \text{exprnd}(\text{value})$ – număr aleator cu distribuție exponențială cu valoarea medie value secunde

Lucrarea 2

BLOCURI NOI

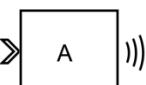
În acest laborator au fost prezentate pentru prima oară blocurile Entity Multicast, Multicast Receive Queue și Sine Wave.

1. MULTICAST RECEIVE QUEUE

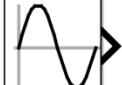
- 
Multicast Receive Queue
- memorează entități într-o coadă, unde pot fi sortate în funcție de prioritate sau momentul de timp în care au intrat în coadă
 - stiva începe să se golească doar atunci când blocul de downstream (cel la care este legat cu săgeata dreaptă) poate primi entități.

-entitățile care sunt memorate în coadă provin de la un Entity Multicast care are setat același tag

2. ENTITY MULTICAST

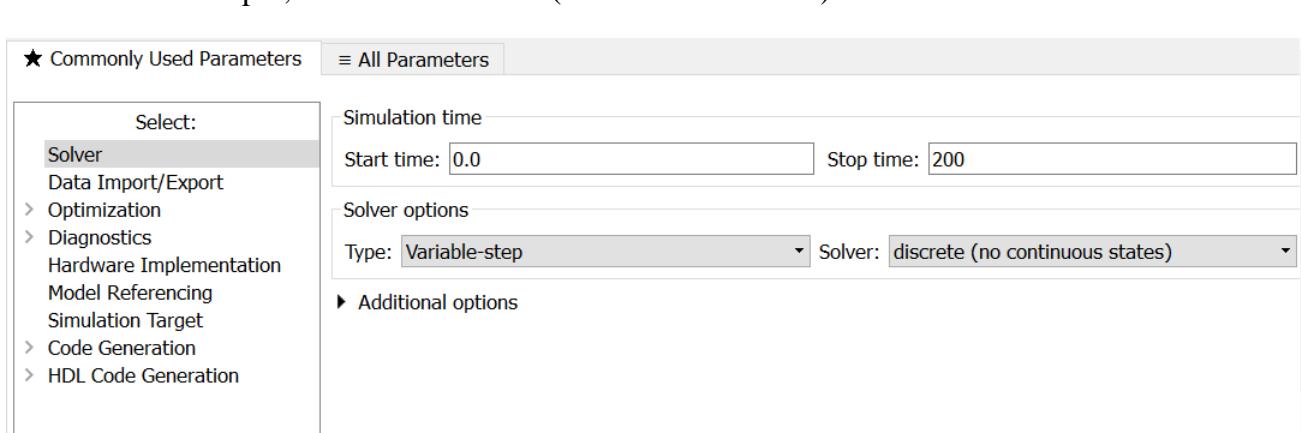
- 
Entity Multicast
- trimite input-ul primit către fiecare Entity Queue/Multicast Receive Queue care are setat același tag
 - este asemănător cu blocurile FROM și GO TO, doar că pentru entități

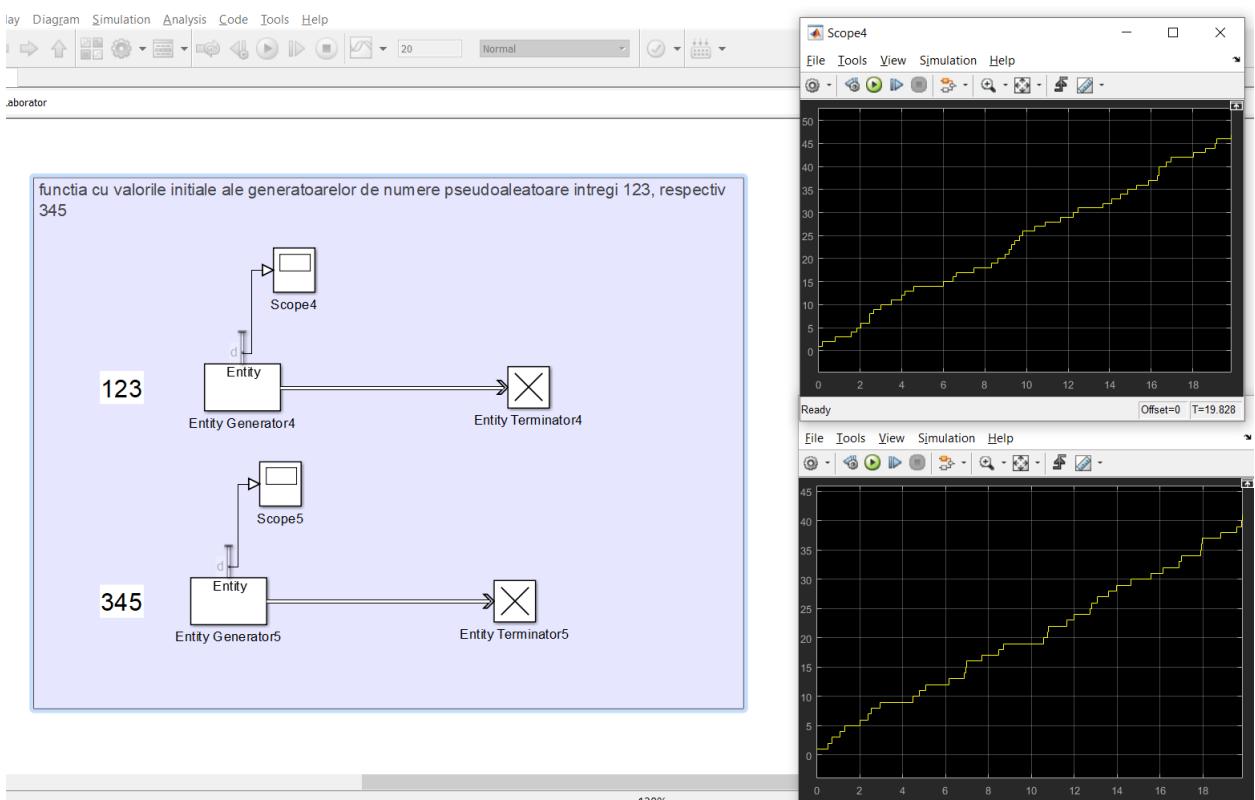
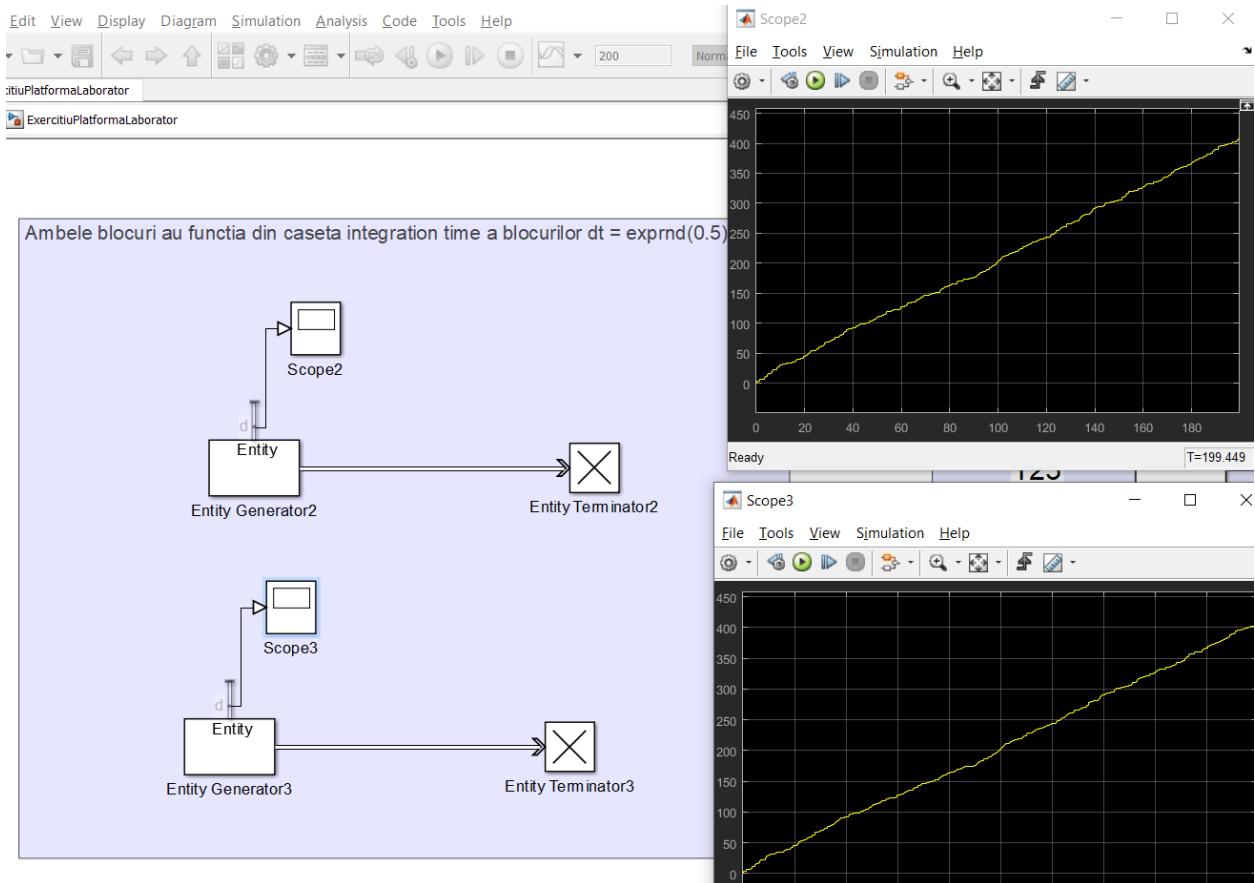
3. SINE WAVE

- 
Sine Wave
- generează o undă sinusoidală pe baza unor valori alese

EXERCIȚII EXPLICATE ÎN PLATFORMA DE LABORATOR

Pentru început, am selectat discrete(no continuous states) la Solver.





Se poate observa că, atunci când funcția are în componență sa numerele pseudoaleatoare 123, pe scope se poate vedea că un număr mai mare de entități au fost trimise mai departe către Entity Terminator decât în cazul în care s-au folosit numerele 345. Am inclus și celalalte figuri explicate în arhivă, dar nu le-am inclus aici pentru că

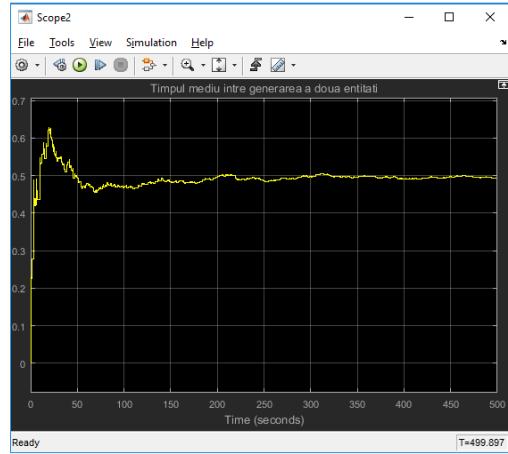
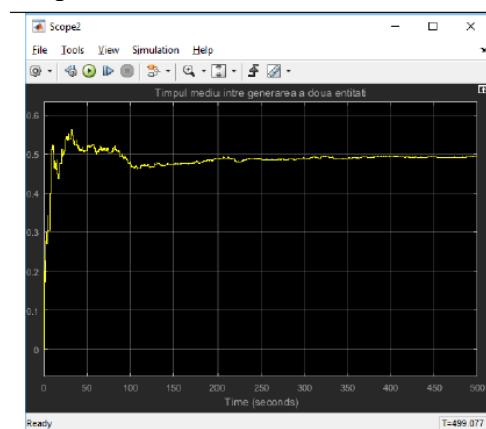


Figura 25. Timpul mediu între generarea a două entități



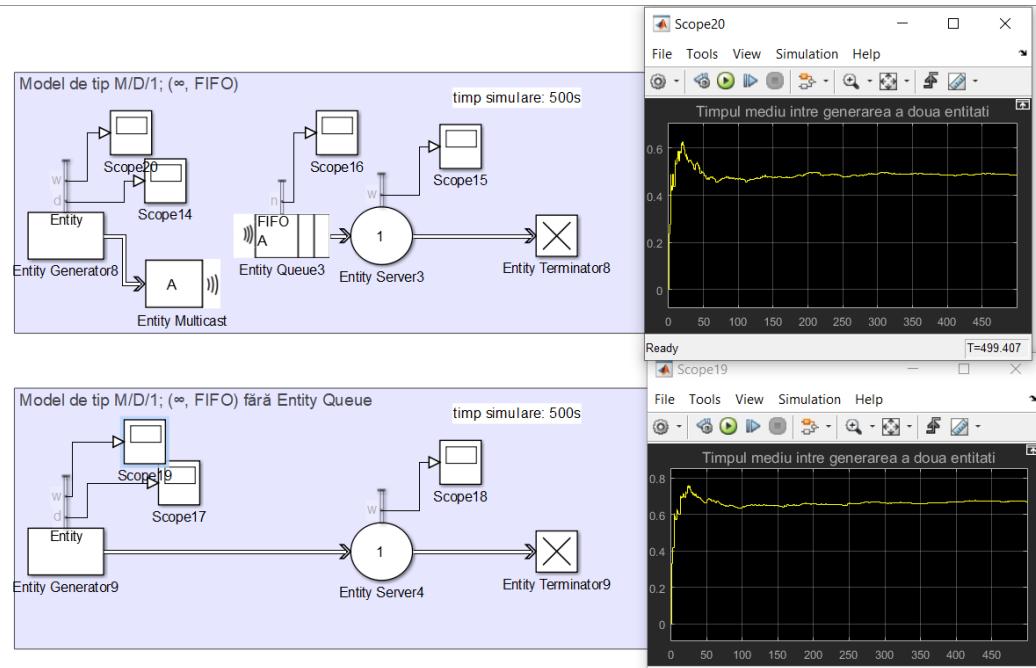
Tabelul 11. Timpul mediu între generarea a două entități pentru diverse valori inițiale ale generatorului de numere pseudoaleatoare întregi

valorile afișate pe scope corespundeau.

În toate cele 3 cazuri media este 0.5. Totuși, atunci când nu se folosește generatorul de numere pseudoaleatoare întregi, rezultatul este unul mai uniform după primele 100s.

Exercițiu. Să se modifice modelul eliminând blocul Entity Queue. Să se compare timpul mediu de așteptare a unei entități în blocul Entity Generator în cele două modele. Să se explice diferențele.

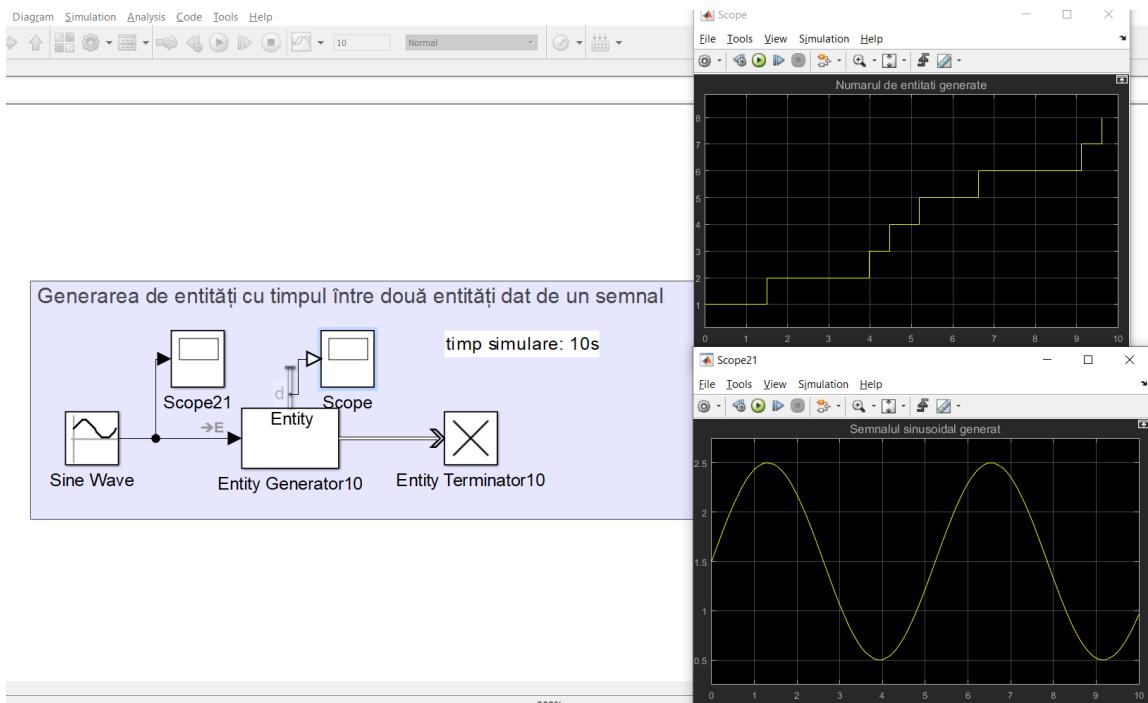
Trebuie să comparăm average waiting time-ul celor două Entity Generator. Pentru a putea vedea acest lucru, am selectat din Statistics căsuța corespunzătoare pentru w pentru ambele Entity Generator, apoi le-am legat către un scope.



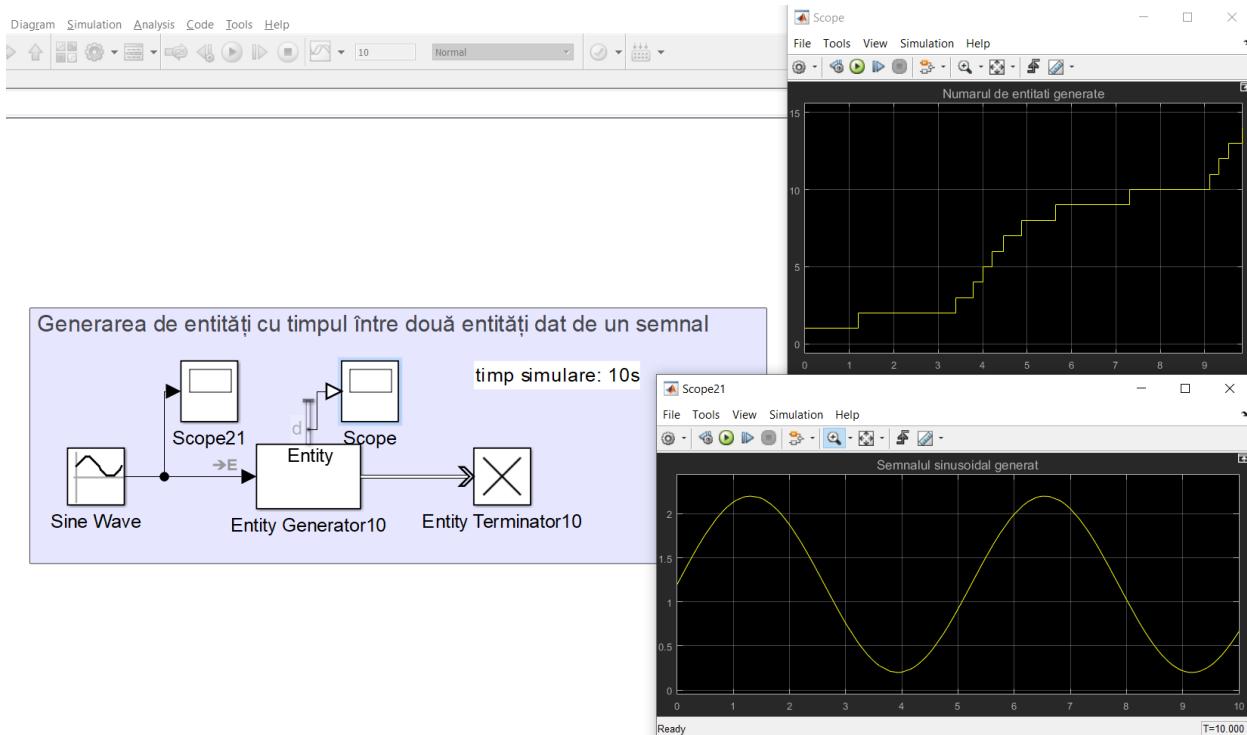
Se poate observa că, atunci când nu avem un Entity Queue, timpul mediu între generarea a două entități crește. Motivul pentru care acest lucru se întâmplă este exact același pe care l-am scris și la primul laborator - Entity Generator poate genera entități decât dacă este gol. Entity Generator poate genera entități mult mai repede decât le poate gestiona Entity Server, dar cum ambele au capacitatea de o entitate, Entity Generator trebuie să aștepte după Entity Server. Acest lucru nu se întâmplă în același mod în cazul în care există și un Entity Queue, deoarece Entity Generator poate genera entități ce mai departe trec în stivă, nu direct în Entity Server. Astfel, deși rata de transfer către Entity Server este mai mică, viteza de generare a entităților nu este afectată atât timp cât este loc în stivă. Cu cât trece mai mult timp, cu atât stiva se umple mai tare, iar din acest motiv, după un anumit moment dat, valorile afișate pe scope-uri încep să corespundă.

Exercițiu. Să se simuleze modelul cu parametrul Bias = 1.2. Se vor explica rezultatele.

modelul cu decalare 1.5



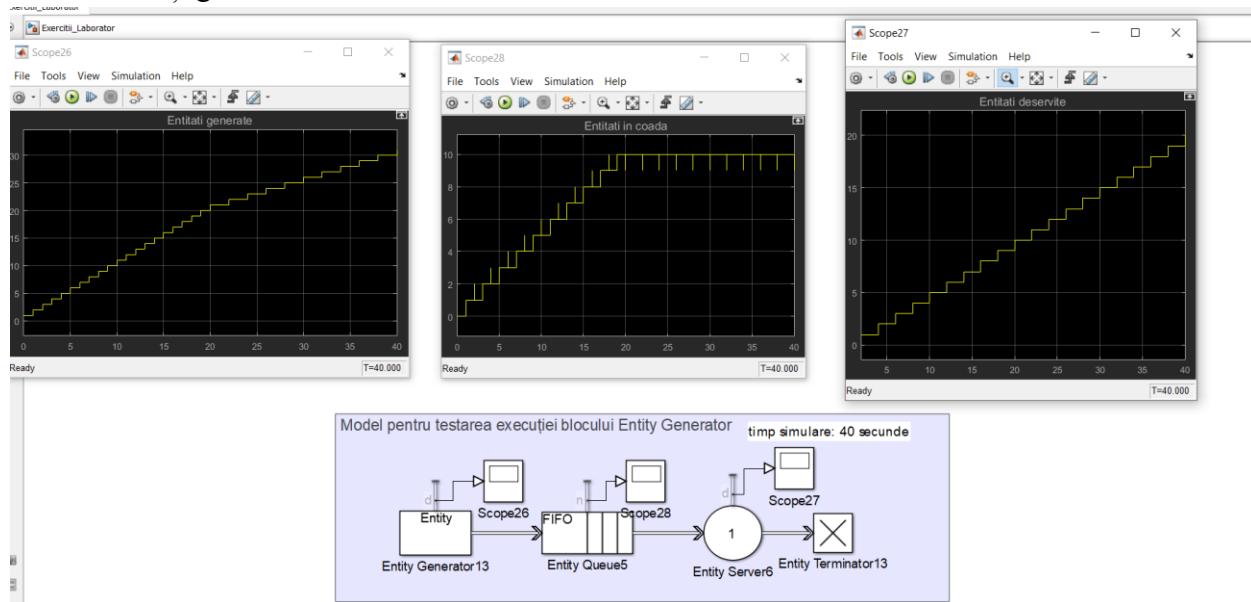
modelul cu decalare 1.2



Se poate observa că semnalul sinusoidal generat, în cazul decalării/bias-ului de 1.2, are valoarea maximă 2.2 și valoarea minimă 0.2. În schimb, semnalul generat cu bias 1.5 are

valoarea maximă 2.5 și valoarea minimă 0.5. De asemenea, numărul de entități generate în cazul în care bias-ul este 1.2 este mai mic decât în cazul în care acesta este 1.5.

Exercițiu. Model pentru testarea execuției blocului Entity Generator. Să se explice de ce panta curbei Entități generate se modifică la momentul $t = 20s$.



Panta curbei Entități generate se modifică după 20s, deoarece atunci este momentul în care stiva este plină. Exact aşa cum am explicitat și înainte, Entity Generator poate genera entități ce mai departe trec în stivă, nu direct în Entity Server. Astfel, deși rata de transfer către Entity Server este mai mică, viteza de generare a entităților nu este afectată atâta timp cât este loc în stivă. Cu cât trece mai mult timp, cu atât stiva se umple mai mult, iar din acest motiv, după un anumit moment dat, când aceasta este plină, Entity Generator este obligat să își micșoreze rata de generare a entităților.

justificare:

Entity Generator - 1s

Entity Server - 2s

Capacitate Entity Queue – 10

2 secunde \Rightarrow 2 evenimente generate, 1 eveniment distribuit \Rightarrow 1 eveniment este stocat în stivă permanent; acest lucru se poate întâmpla de 10 ori, deci $2 \times 10 = 20$ secunde.

5. Probleme propuse

EXERCITIUL 1

se dau

- Loturile sosesc la intervale de 18 ± 6 minute cu distribuție uniformă
*valori din intervalul $(18-6; 18+6) = (12, 24)$. (deci $12 + 12 * \text{rand}$)*
- Prelucrarea unui lot durează 16 ± 4 minute cu distribuție uniformă
*valori din intervalul $(16-4; 16+4) = (12, 20)$. (deci $12 + 8 * \text{rand}$)*
- Durată de opt ore și apoi pe două săptămâni ($60 * 8 * 10$ minute)
se va presupune că un sample al aplicației = 1 minut.
 $8 \text{ ore} = 60 * 8 = 480 \text{ minute}$
 $2 \text{ săptămâni} = 60 * 24 * 14 = 20160 \text{ minute}$

se cer:

- numărul de loturi generate

numărul de loturi generate sunt, de fapt, Number of entities departed din Entity Generator

- numărul de loturi în coadă

numărul de loturi în coadă sunt, de fapt, Number of entities in block din Entity Queue

- numărul de loturi prelucrate.

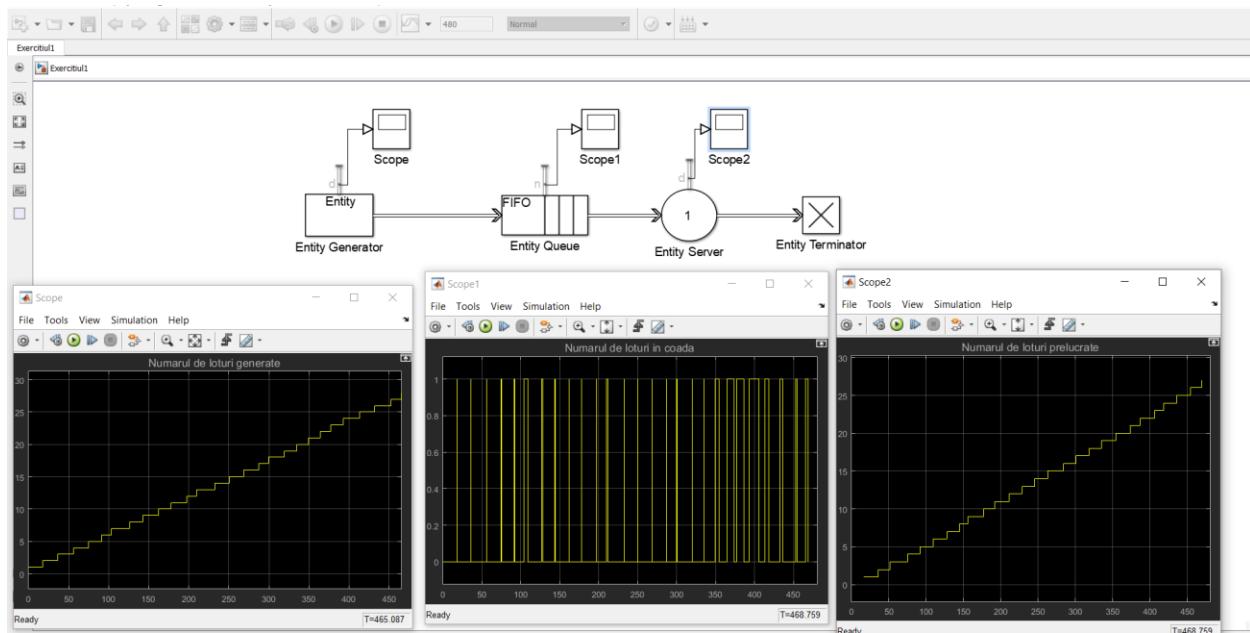
numărul de loturi prelucrate sunt, de fapt, Number of entities departed din Entity Server

rezolvare:

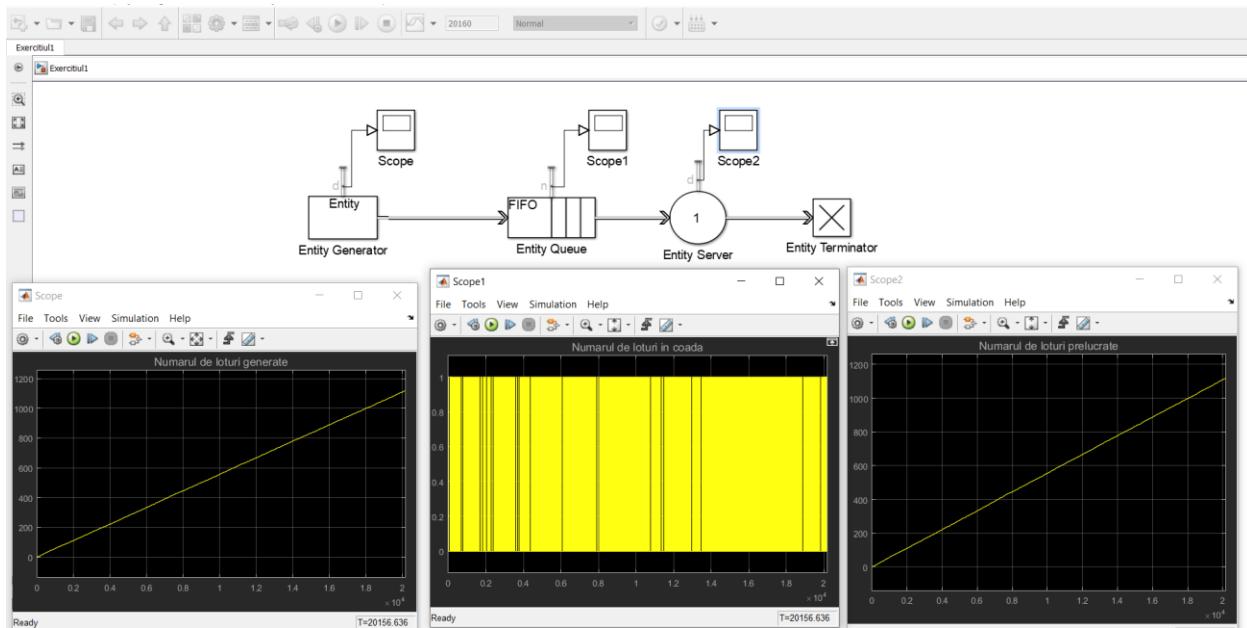
M-am folosit de blocurile Entity Generator (pentru a genera loturile), Entity Queue (pentru a pune în stivă loturile), Entity Server (pentru a prelucra loturile), Entity Terminator (pentru a goli Entity Server, astfel putând prelucra alte loturi) și Scope (pentru a afișa ceea ce mi se cere).

Atât pentru Entity Generator, cât și pentru Entity Server m-am folosit de funcția prezentată în platforma de laborator. Cum nu s-a precizat nimic legat de lungimea cozii în care sunt depozitate loturile pentru piesele pentru prelucrare, am decis ca aceasta să aibă capacitatea 100.

Pentru 8 ore:



Pentru 2 săptămâni:



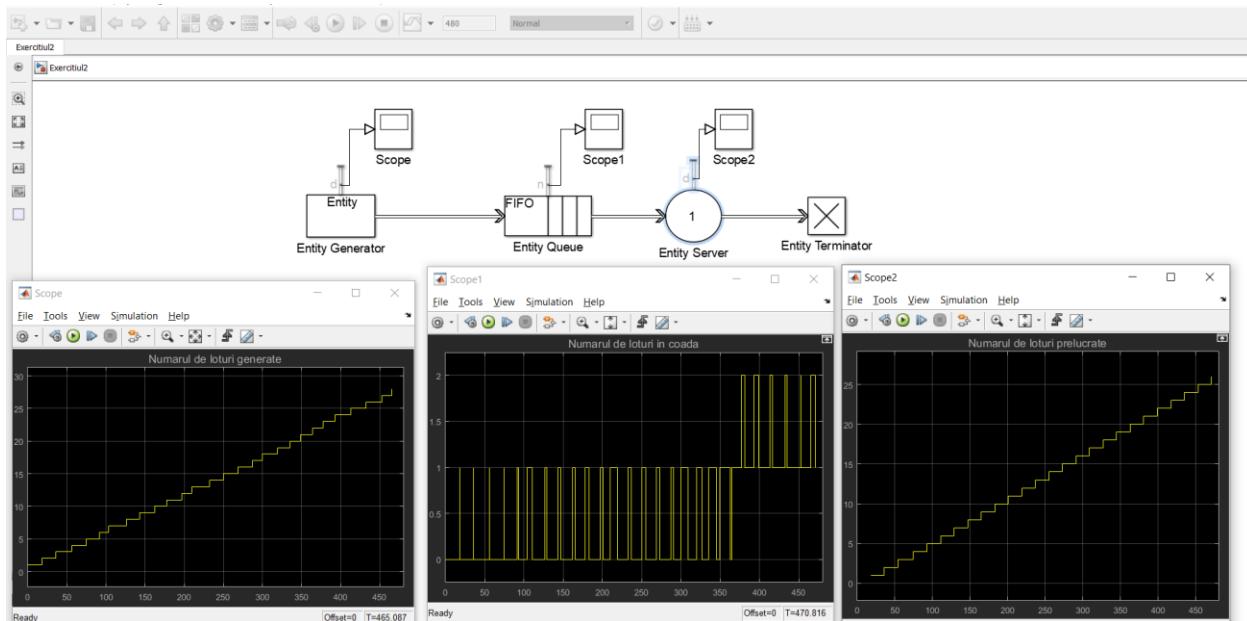
EXERCITIUL 2

Prelucrarea unui lot durează 18 minute, deci trebuie modificat Entity Server.

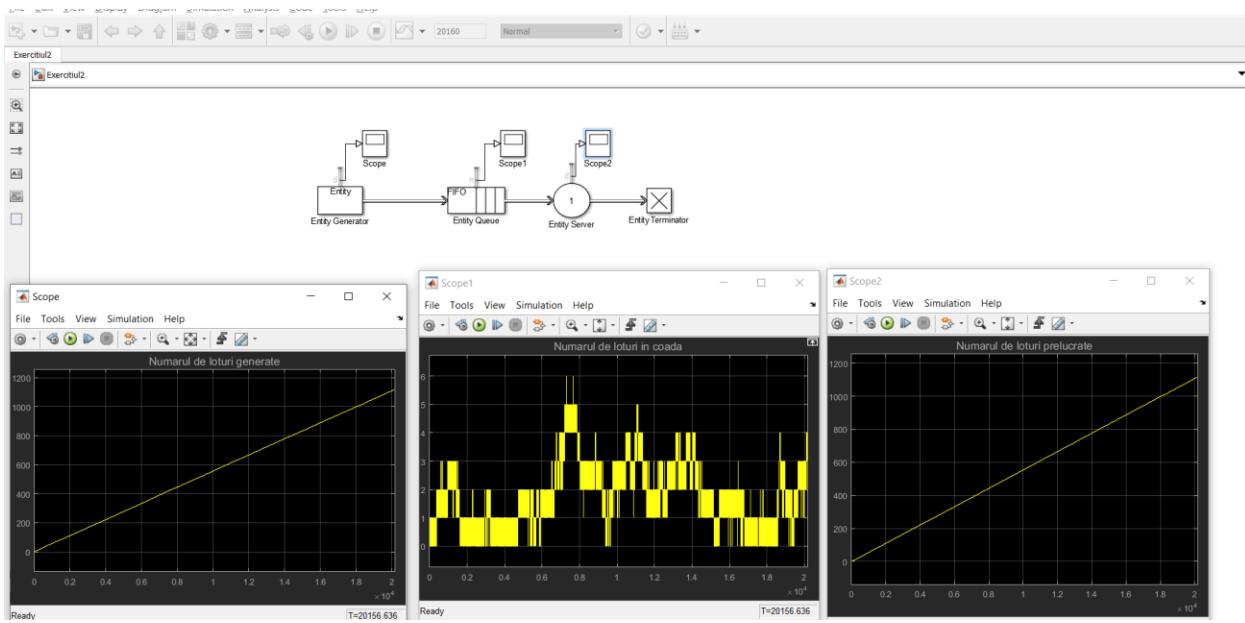
Service time value:

18

pentru 8 ore:



pentru 2 săptămâni:



EXERCITIUL 3

se dau

- Loturile sosesc la intervale de 17 ± 5 minute cu distribuție uniformă
*Entity Generator are valori din intervalul $(17-5; 17+5) = (12, 22)$. (deci $12 + 10 * \text{rand}$)*
- Loturile sunt prelucrate pe o mașină pe o durată 14 minute și apoi pe o altă mașină pe o durată de 3 minute

sunt două Entity Server, ambele având Service Time Source – Dialog; una are Service Time Value 14, iar cealaltă are 3

- Loturile generate așteaptă să fie prelucrate la fiecare mașină într-o coadă
sunt două Entity Queue

- Să se simuleze modelul pe o săptămână
 $7 * 60 * 24 = 10080$ minute

se cer

- numărul de loturi generate
numărul de loturi generate sunt, de fapt, Number of entities departed din Entity Generator

- numărul de loturi în fiecare coadă
numărul de loturi în coadă sunt, de fapt, Number of entities in block din Entity Queue

- numărul de loturi prelucrate
numărul de loturi prelucrate sunt, de fapt, Number of entities departed din Entity Server

- gradul de utilizare al fiecărei mașini.

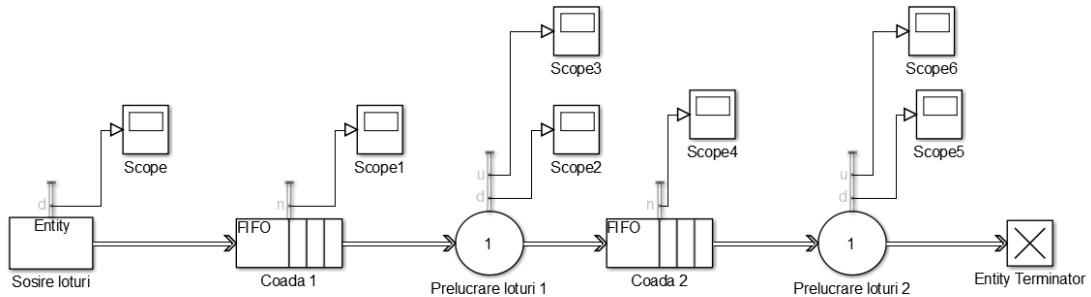
gradul de utilitate al fiecărei mașini este, de fapt, Utilization din Entity Server

rezolvare:

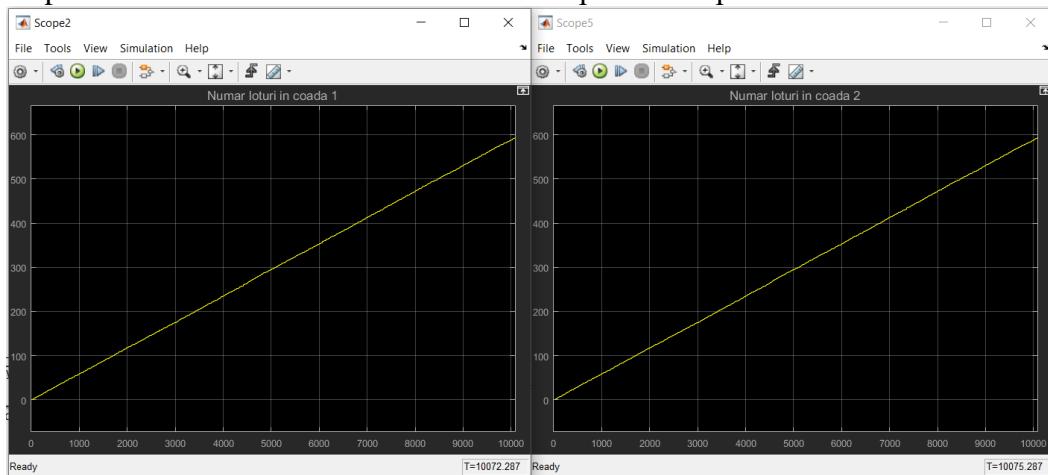
M-am folosit de blocurile Entity Generator (pentru a genera loturile), Entity Queue (pentru a pune în stivă loturile), Entity Server (pentru a prelucra loturile), Entity Terminator (pentru a goli Entity Server, astfel putând prelucra alte loturi) și Scope (pentru a afișa ceea ce mi se cere).

Pentru Entity Generator m-am folosit de funcția prezentată în platforma de laborator. Pentru Entity Server am ales Service Time Source – Dialog, unde am completat cu 14, respectiv

3. Cum nu s-a precizat nimic legat de lungimea celor două cozi în care sunt depozitate loturile pentru piesele pentru prelucrare, am decis ca acestea să aibă capacitatea 100.

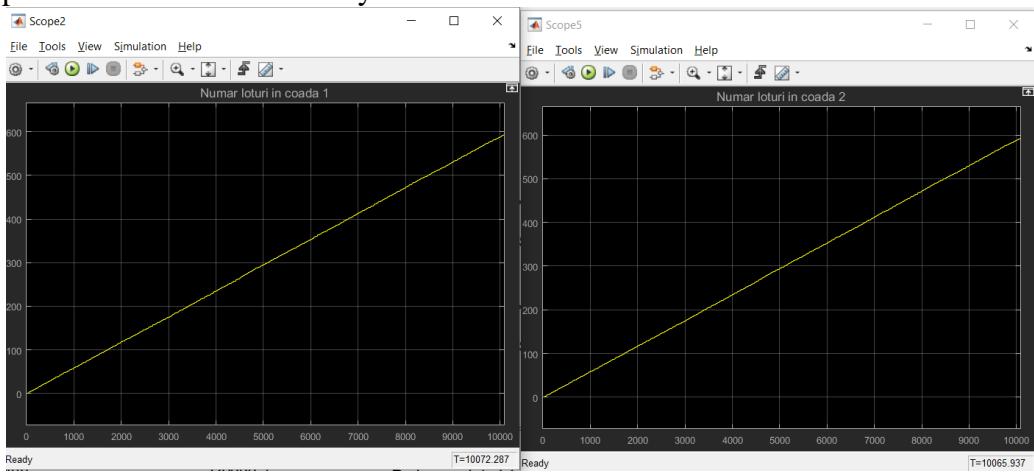


Am pus și acest exercițiu în arhivă. Acolo, în urma rulării, se pot vedea și valorile de pe scope. O să arăt aici decât numărul de loturi prelucrate pentru fiecare coadă.



EXERCITIUL 4

Acest exercițiu este exact ca și exercițiul 3, doar că am modificat valorile introduse pentru cel de-al doilea Entity Server. Acum are valoarea 13.



EXERCITIUL 5

se dă:

- există un serviciu de informații cu cinci operatori.

este un Entity Server cu capacitate 5

- orice operator poate prelua orice cerere.

- cererile apar la intervale cu distribuție exponențială cu valoarea medie 0.1 minute.

valoarea medie 0.1 minute este dată de funcția $dt = exprnd(0.1)$; în Entity Generator

- durata unei con vorbiri este 1 ± 0.2 minute cu distribuție uniformă.

valori din intervalul $(1-0.2; 1+0.2) = (0.8, 1.2)$. (deci $0.8 + 0.4 * \text{rand}$)

- Să se modeleze sistemul pe o durată de o oră.

60 minute

se cere:

- numărul de cereri

numărul de cereri este, de fapt, **Number of entities departed** din Entity Generator

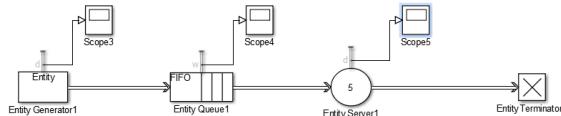
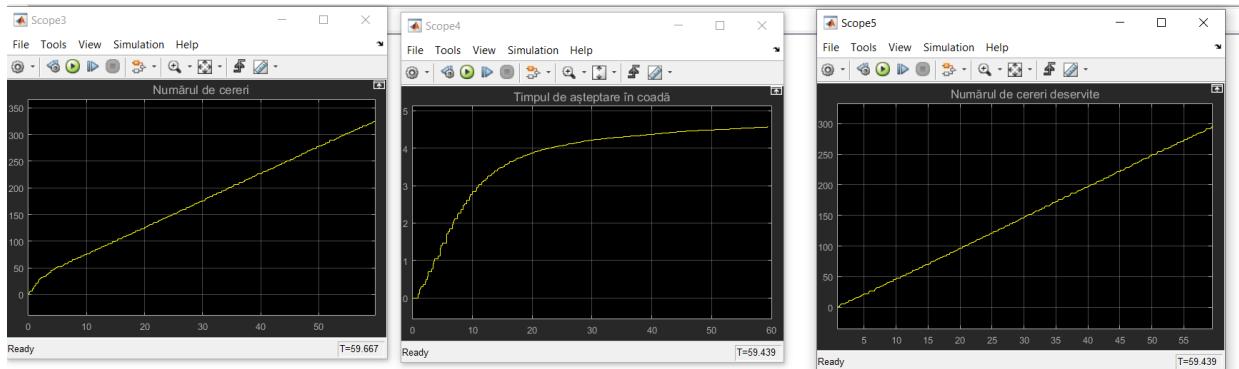
- timpul de așteptare în coadă

timpul de așteptare în coadă este, de fapt, **Average wait** din Entity Queue/Multicast Receive Queue

- numărul de cereri deservite.

numărul de cereri deservite este, de fapt, **Number of entities departed** din Entity Server

rezolvare:



EXERCITIUL 6

se dă:

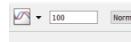
- Fie modelul de mai jos în care blocul Entity Generator generează entități cu distribuție exponențială și valoarea medie 0.2s.

valoarea medie 0.2 secunde este dată de funcția $dt = exprnd(0.2)$

- Blocul Entity Server are capacitatea cinci și deservește entități cu timpul de deservire 1s

timpul de deservire 1s pentru Entity Server este setat din Service time value = 1

- Să se simuleze modelul pe 100s.



- Entitățile vor avea un atribut ServiceTime ce va conține timpul de deservire.
- Blocul Entity Server va citi timpul de deservire din acest atribut

rezolvare:

Am realizat modelul conform instrucțiunilor.



EXERCITIUL 7

se dău:

- se va utiliza un bloc Entity Server cu capacitate infinită.
am setat în Entity Server atributa Capacity cu valoarea inf

- Se va simula modelul pe 100s

se cere:

- se vor compara rezultatele cu cele anterioare.

rezolvare:

În urma rulării modelului, s-a obținut:



Făcând o comparație între cele două modele, se poate observa că
-numărul de cereri/entități este identic

-numărul de cereri/entități în coadă are valoarea fie 0, fie 1 în cazul exercițiului 7. Acest lucru se întâmplă datorită faptului că Entity Server are capacitate infinită, acest lucru însemnând că poate servi orice cantitate de entități pe care o primește. Totuși, pentru ca o entitate să ajungă în Entity Server, conform modelului, este necesar ca aceasta să treacă prin Entity Queue. Din acest motiv, numărul de entități în coadă este necesar să crească, dar imediat după ce intră în queue el este trimis mai departe.

-numărul total de entități ce au trecut prin coadă este mai mare în cazul exercițiului numărul 7, deoarece nu mai sunt "timpuri morți" în care nu se pot genera entități (am explicat la alt exercițiu din platformă).

Lucrarea 3

BLOCURI NOI FOLOSITE

În acest laborator au fost prezentate blocurile Uniform Random Number, Random Number, Simulink Function, Inport, Outport, MATLAB Function, Fcn. De asemenea, în rezolvare exercițiilor m-am folosit de Entity Input Switch și Entity Output Switch .

1. UNIFORM RANDOM NUMBER



Uniform Random Number

-generează numere cu distribuție uniformă

-output-ul său se poate repeta

2. RANDOM NUMBER

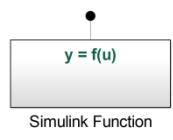


Random Number

-generează numere cu distribuție normală (Gaussiană)

-output-ul său se poate repeta

3. SIMULINK FUNCTION

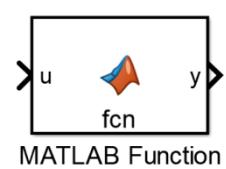


Simulink Function

-folosit pentru definirea de funcții Matlab în Simulink

-conține argument de intrare/ieșire, iar dacă este necesar, și porturi de intrare/ieșire

4. MATLAB FUNCTION



MATLAB Function

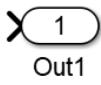
-folosit pentru definirea de funcții Matlab

-cu dublu click se deschide un editor text unde se scriu funcții Matlab

5. IN+OUT



-sunt folosite în subsisteme

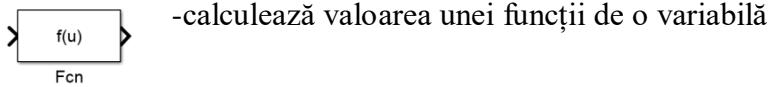


-blocul In transmite mai departe în subsistemului un semnal care intră în interiorul subsistemului de la un nivel mai superior

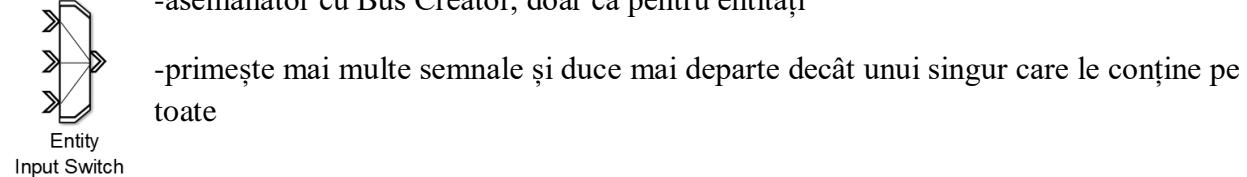


-blocul Out transmite mai departe la un nivel mai superior un semnal careiese din interiorul subsistemului

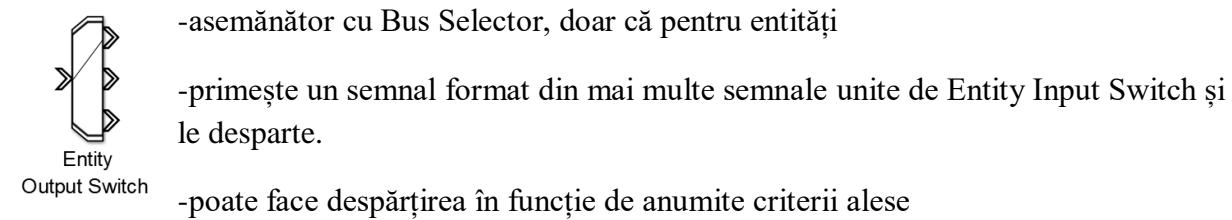
6. FCN



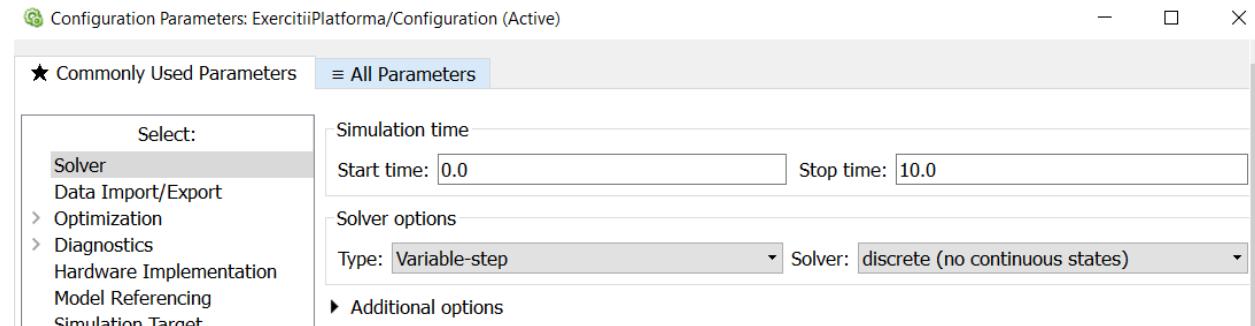
7. ENTITY INPUT SWITCH



8. ENTITY OUTPUT SWITCH



Din nou, primul lucru pe care l-am făcut a fost să setez Solver-ul ca fiind discret din Configuration Parameters.



4. Probleme propuse

PROBLEMA 1

se dau:

- G/G/1; (∞ , FIFO)

Entity Queue are capacitatea infinit (inf) și este de tip FIFO

- blocul Entity Generator generează entități cu timpul între apariția a două entități o variabilă aleatoare uniformă cu valori între 0.2 și 1 secunde

*valori din intervalul (0.2 , 1) (deci $0.2 + 0.8 * \text{rand}$)*

- timpul de deservire al unei entități este o variabilă aleatoare cu distribuție uniformă între 0.3s și 1.2s.

*valori din intervalul (0.3 , 1.2) (deci $0.3 + 0.9 * \text{rand}$)*

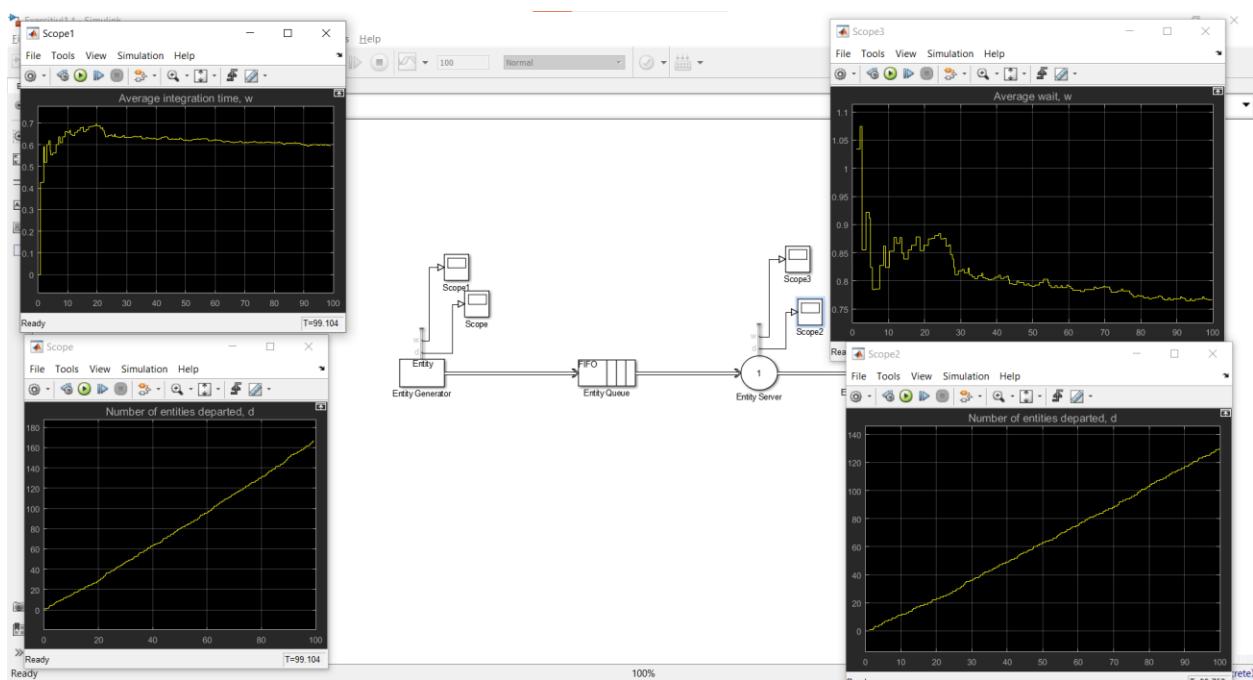
se cere:

- să se simuleze modelul pe 100 secunde



rezolvare:

Rezultatele simulării pe 100 secunde:



PROBLEMA 2

se dau:

- Modelarea unui sistem de calcul. La un sistem de calcul sosesc programe la intervale de 2 ± 1 minute cu distribuție uniformă.

*valori din intervalul $(2-1; 2+1) = (1,3)$. (deci $1 + 2 * \text{rand}$)*

- programele sunt citite de pe disc cu timpul de citire 30 ± 10 s cu distribuție uniformă.

*valori din intervalul $(30-10;30+10) = (20,40)$. (deci $20 + 20 * \text{rand}$) sunt citite de pe disc => este un Entity Queue; cum nu s-a precizat nimic de dimensiunea lui, am pus-o 1000.*

- Timpul de execuție este de 3 ± 1 minute cu distribuție uniformă.

*valori din intervalul $(3-1;3+1) = (2,4)$. (deci $2 + 2 * \text{rand}$)*

- durată de 2 ore, unitatea de timp se alege secunda.

1 minut = 60 secunde

1 oră = 60 minute

*$60 * 60 = 3600$ secunde/oră*

*$2 \text{ ore} = 3600 * 2 = 7200$ secunde* 

- Modelul va conține două blocuri Entity Server, unul pentru citirea programelor de pe disc și altul pentru execuția programelor.

se cer:

- numărul de programe intrate în model

numărul de programe intrate în model este, de fapt, Number of entities departed din Entity Generator

- timpul de citire de pe disc

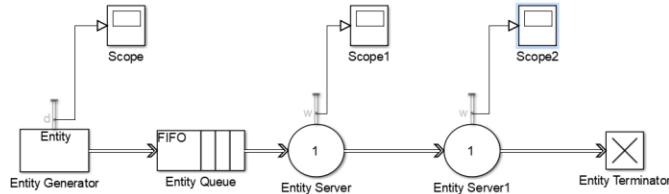
timpul de citire de pe disc este, de fapt, Average wait din primul Entity Server

- timpul de execuție.

timpul de execuție este, de fapt, Average Wait din al doilea Entity Server

rezolvare:

Rezultatele obținute în urma simulării:



PROBLEMA 3

se dă:

- La o companie telefonică există un serviciu de informații cu cinci operatori. Orice operator poate prelua orice cerere.
este un Entity Server cu capacitate 5
- Cererile apar la intervale cu distribuție exponențială cu valoarea medie 0.2 minute.
 $dt = exprnd(0.2)$; la Entity Generator
- Durata unei con vorbiri este 1 ± 0.2 minute cu distribuție uniformă.
*valori din intervalul $(1-0.2; 1+0.2) = (0.8, 1.2)$. (deci $0.8 + 0.2 * rand$)*

- Se va utiliza un multiserver cu capacitatea cinci.
- Să se modeleze sistemul pe o durată de 8 ore.

dacă unitatea de timp aleasă este minutul

$$8 * 60 = 480 \text{ minute}$$

se cere:

- timpul mediu de așteptare în coadă

timpul mediu de așteptare în coadă este, de fapt, Average Wait din Entity Queue

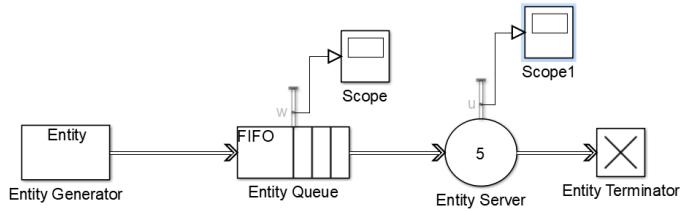
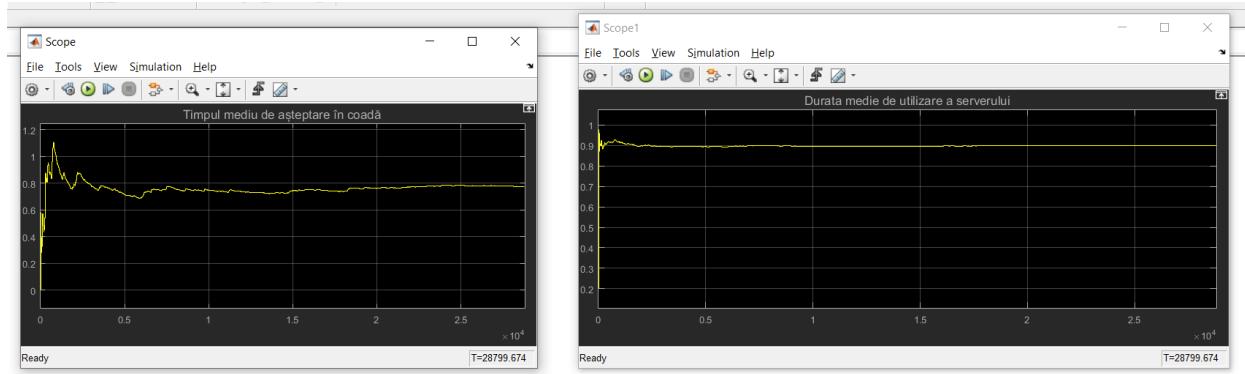
din această afirmație se deduce că înainte de Entity Server se află un Entity Queue

- durata medie de utilizare a serverului

durata medie de utilizare a serverului este, de fapt, Utilization din Entity Server

rezolvare:

În urma rulării modelului am obținut următoarele rezultate:



PROBLEMA 4

se dau:

- Loturile sosesc la intervale de 18 ± 6 minute cu distribuție uniformă.
*din intervalul $(18-6; 18+6) = (12, 24)$. (deci $12 + 12 * \text{rand}$)*
- Prelucrarea unui lot durează 32 ± 4 minute cu distribuție uniformă.
*din intervalul $(32-4; 32+4) = (28, 36)$. (deci $28 + 8 * \text{rand}$)*
- Loturile sunt prelucrate la o stație cu capacitatea 2.

Entity Server are capacitatea 2

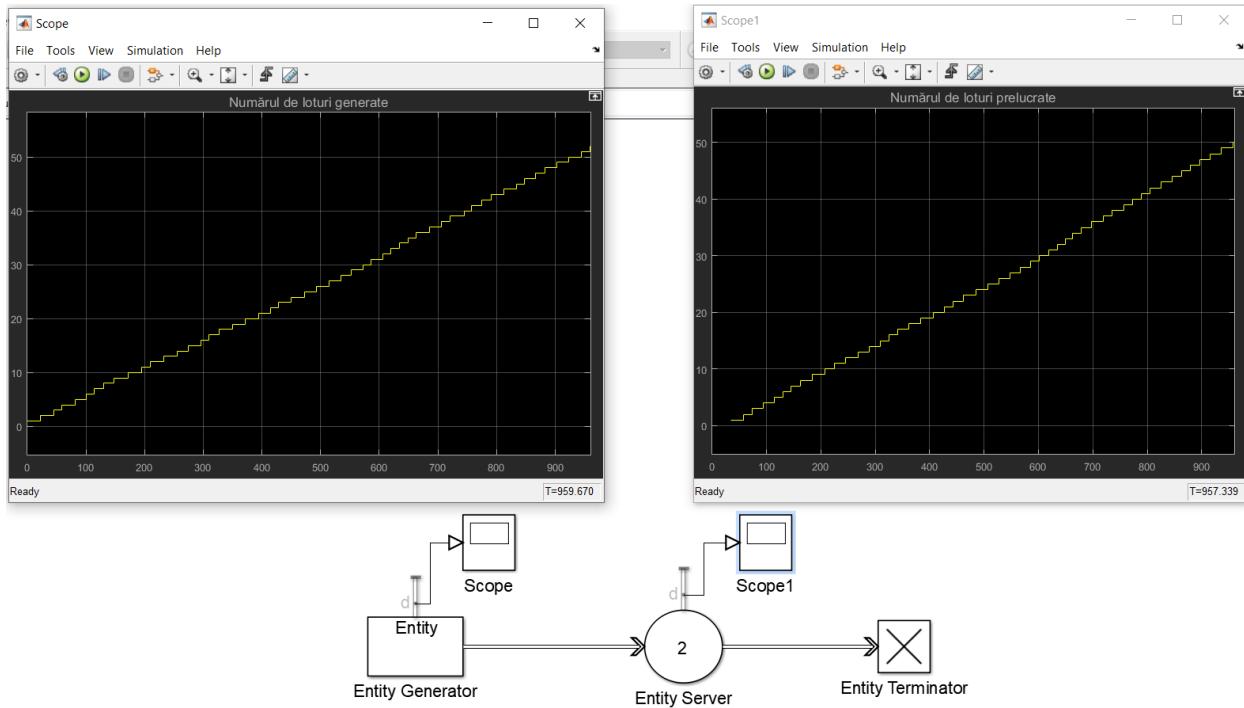
- Să se simuleze modelul pe o durată de 960 minute.
o să aleg unitatea de timp ca fiind minutul. 960

se cer:

- numărul de loturi generate
numărul de loturi generate este, de fapt, Number of entities departed din Entity Generator
- numărul de loturi prelucrate
numărul de loturi prelucrate este, de fapt, Number of entities departed din Entity Server

rezolvare:

În urma rulării modelului am obținut următoarele rezultate:



PROBLEMA 5

se dă:

- La o magazie pentru aprovizionarea cu materiale vin două tipuri de muncitori.

Sunt două Entity Generator, câte unul pentru fiecare tip de muncitor.

- Cei de primul tip vin la intervale de 420 ± 360 s cu distribuție uniformă, iar timpul de deservire este 300 ± 90 s cu distribuție uniformă.

*vin în intervalul $(420-360; 420+360) = (60, 780)$. (deci $60 + 720 * \text{rand}$)*

*timpul de deservire $(300-90, 300+90) = (210, 390)$ (deci $210 + 180 * \text{rand}$)*

- Muncitorii de tipul doi la intervale de 360 ± 240 s cu distribuție uniformă, iar timpul de deservire este 100 ± 30 s cu distribuție uniformă.

*vin în intervalul $(360-240; 360+240) = (120, 600)$. (deci $120 + 480 * \text{rand}$)*

*timpul de deservire $(100-30, 100+30) = (70, 130)$ (deci $70 + 130 * \text{rand}$)*

- Muncitorii de ambele tipuri așteaptă în aceeași coadă să fie deserviți.

ambele Entity Generator intră în același Entity Queue; i-am dat din oficiu capacitatea 300

- durată de 28800s

o să aleg unitatea de timp ca fiind secunda.

se cere:

- numărul de muncitori de fiecare tip care vin pentru aprovizionare

numărul de muncitori de fiecare tip care vin pentru aprovizionare reprezintă, de fapt, Number of entities departed pentru fiecare Entity Generator.

- lungimea cozii

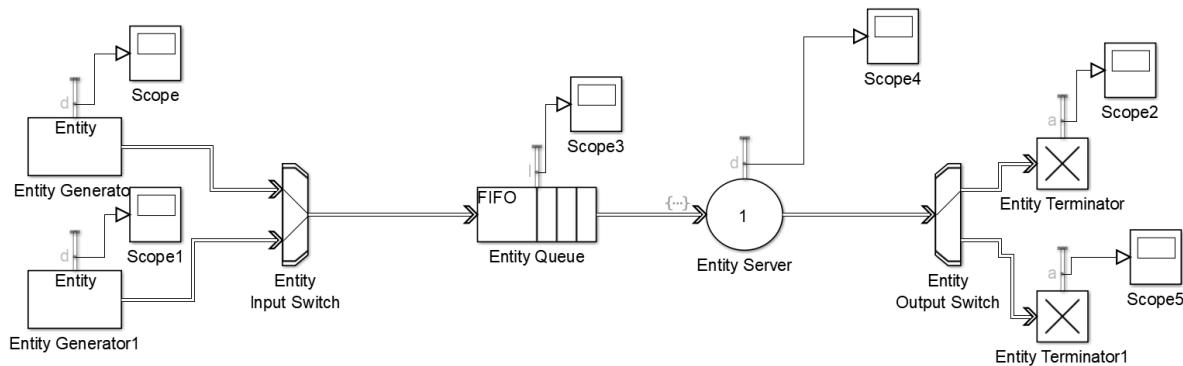
lungimea cozii este Average Queue Length din Entity Queue

- numărul de muncitori deserviți de fiecare tip

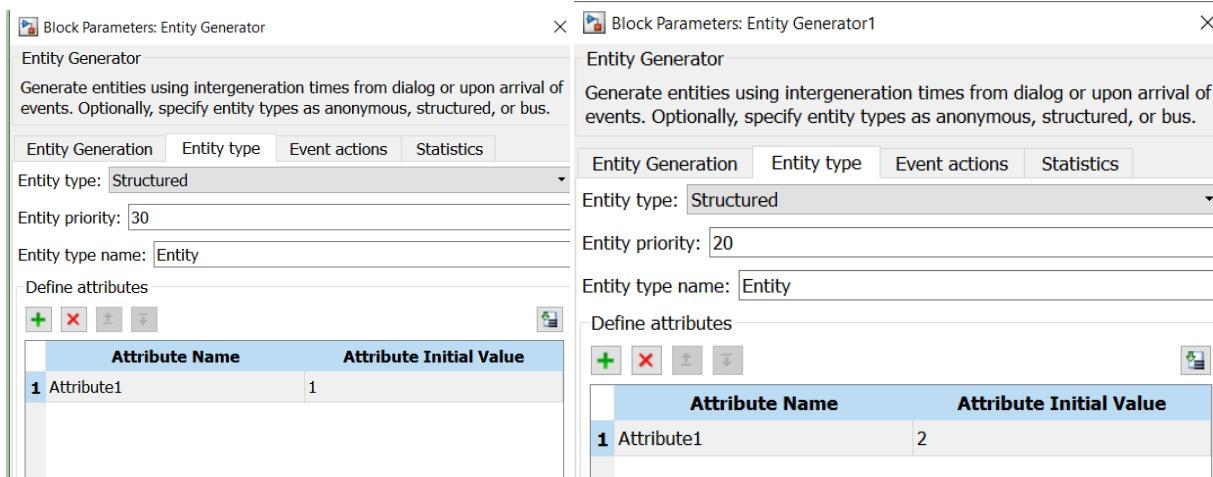
numărul de muncitori deserviți de fiecare tip este, de fapt, Number of entities departed din Entity Server – dar acest lucru e pentru toate tipurile

rezolvare:

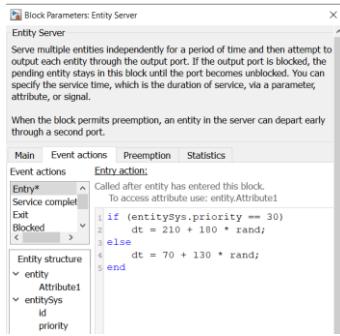
Exercițiul realizat de mine arată astfel:



Sunt două Entity Generator, câte unul pentru fiecare tip de muncitor. Am adăugat, în plus, pentru fiecare Entity Generator ceva distinct – cel de sus are prioritatea 30 și o atribută Attribute1 cu valoarea 1, iar cel de-al doilea, Entity Generator1, are prioritatea 20 și atributa Attribute1 are valoarea 2.

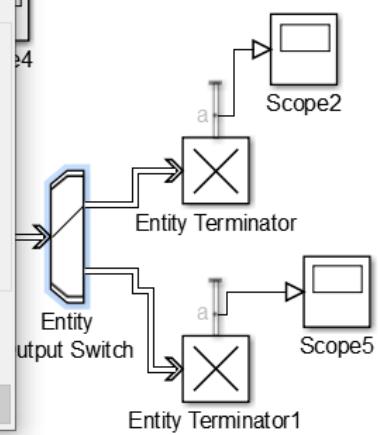
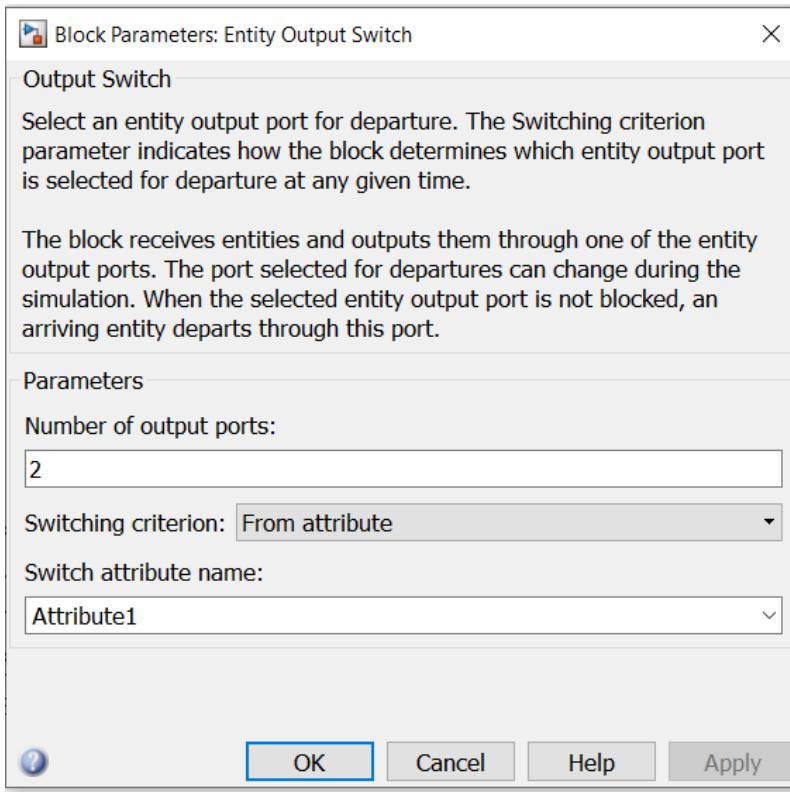


Motivul pentru care au prioritatea entitySys diferită capătă sens în interiorul lui Entity Server, unde am scris o funcție pentru sample-ul în care o entitate intră în acest bloc:



Dacă prioritatea este 30, atunci timpul de deservire este 300 ± 90 s cu distribuție uniformă. Dacă entitatea nu are prioritatea 30 (implicit 20), atunci are timpul de deservire 100 ± 30 s cu distribuție uniformă. Aici se verifică prioritatea principală a entității.

De asemenea, am folosit Entity Input Switch și Entity Output Switch. Entity Input Switch poate primi mai multe semnale și duce mai departe decât unul singur, în interiorul căruia se află toate semnalele pe care le-a primit. Entity Output Switch primește un singur semnal care e format din mai multe semnale unite de un Entity Input Switch și le “desparte”. În cazul meu, modul în care le desparte este bazat pe prioritățile atributelor secundare pe care le au aceste entități = Attribute1.



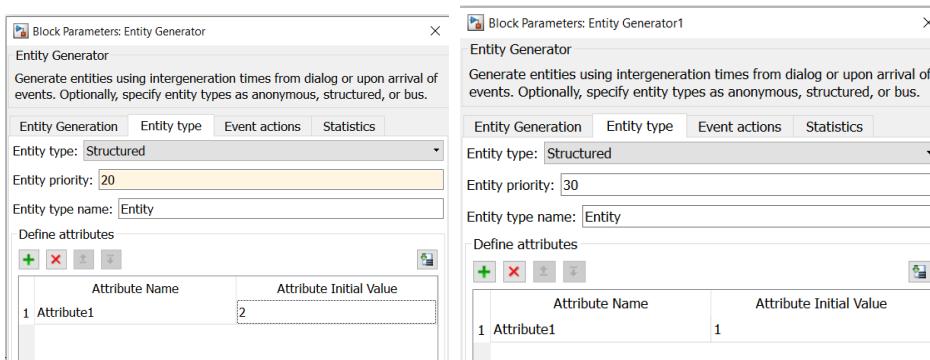
În Entity Terminator am afișat numărul de muncitori deserviți de fiecare tip.

Rezultatul simulării poate fi văzut dacă se rulează exercițiul pus în arhivă.

PROBLEMA 6

Felul în care am făcut eu exercițiul 5 nu ține cont de care dintre muncitori are o prioritate mai mare, ci dacă aceștia au o anumită valoare ca și prioritate. Ca să fac oarecum și acest exercițiu, am modificat:

-Muncitorii 1 au acum prioritatea principală 20, iar la Attribute1 au 2; Muncitorii 2 au acum prioritatea principală 30, iar la Attribute2 au 1 (am inversat).



-Entity Queue este identic, dar acum nu se mai verifică dacă prioritatea este 30, ci 20, pentru ca Muncitorii 1 au acum prioritarea 20, iar formula este pentru ei. Cea de la else este pentru Muncitorii 2.

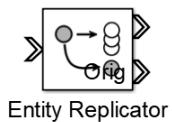
```
1 if (entitySys.priority == 20)
2     dt = 210 + 180 * rand;
3 else
4     dt = 70 + 130 * rand;
5 end
```

Lucrarea 4

BLOCURI NOI FOLOSITE

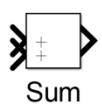
În acest laborator au fost prezentate blocurile Entity Replicator, Sum, Display.

1. ENTITY REPLICATOR



- permite ramificarea unei entități
- output-urile sale sunt copii ale input-ului, adică el permite copierea unei entități.

2. SUM



- realizează operații aritmetice în funcție de simbolurile pe care le are introduce

3. DISPLAY



- folosit pentru a afișa valoarea unui semnal la un singur moment de timp

4. Probleme propuse

PROBLEMA 1

se dau:

- La o mașină unealtă sosesc două tipuri de loturi de piese pentru prelucrare.

sunt două Entity Generator

- Loturile de primul tip sosesc la intervale de 35 ± 10 minute cu distribuție uniformă. Prelucrarea unui lot durează 18 ± 4 minute cu distribuție uniformă.

*loturile de primul tip sosesc în intervalul $(35-10, 35+10) = (25, 45)$. (deci $25 + 20 * rand$)*
*prelucrarea lor durează o valoare din $(18-4, 18+4) = (14, 22)$. (deci $14 + 8 * rand$)*

- Loturile de al doilea tip sosesc la intervale de 45 ± 7 minute cu distribuție uniformă și se fac două prelucrări. Prima prelucrare a lotului durează 18 ± 4 minute cu distribuție uniformă.

*loturile de al doilea tip sosesc în intervalul $(45-7, 45+7) = (38, 52)$. (deci $38 + 14 * rand$)*
*prelucrarea unui lot durează o valoare din $(18-4, 18+4) = (14, 22)$. (deci $14 + 8 * rand$)*

- A doua prelucrare durează 10 ± 2 minute cu distribuție uniformă.

a doua prelucrare e în intervalul $(10-2, 10+2) = (8,12)$. (deci $8 + 4 * \text{rand}$)

- Să se simuleze modelul pe o durată de 480 minute.

din oficiu o să aleg unitatea de timp a modelului ca fiind minutul.

se cer:

- numărul de loturi generate

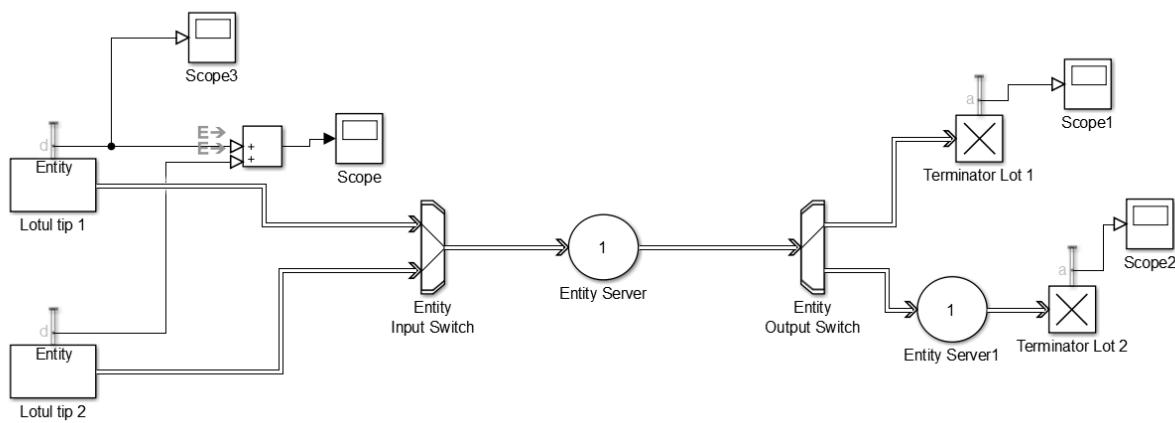
numărul de loturi generate este, de fapt, suma dintre loturile generate de fiecare **Entity Generator**, adică suma dintre **Number of entities departed**

- numărul de loturi prelucrate pentru fiecare tip

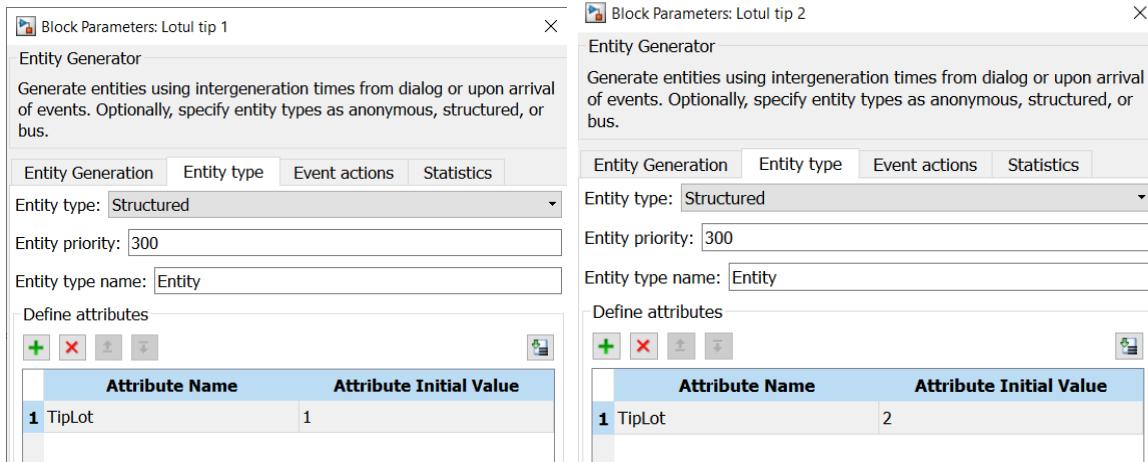
numărul de loturi prelucrate pentru fiecare tip sunt, de fapt, **Number of entities arrived** din **Entity Terminator**-ul corespunzător fiecărei piese.

rezolvare:

Modelul creat de mine arată astfel:



Fiecare lot este generat de propriul său Entity Generator. De asemenea, ambele loturi au atribute pentru a putea fi diferențiate:



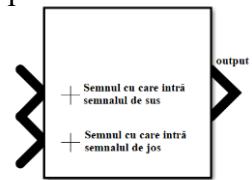
Intergeneration time action:

```

1 persistent init
2 if isempty(init)
3     rng(114);
4     init = true;
5 end
6 dt = 38 + 14 * rand;

```

Motivul pentru care au primit atributa TipLot este ca, atunci când trec prin Entity Output Switch, ele să poată fi împărțite pe câte o ramură a acestuia. Pe ramura de sus merge lotul tip 1, iar pe ramura de jos merge lotul tip 2. De asemenea, ambele loturi au un Entity Server comun – am făcut astfel pentru că era același interval de valori pe care putea fi făcută prima prelucrare pentru amândouă. De asemenea, am adăugat toți timpii din cerință cu funcția prezentată în laboratoare. Pentru a putea afișa numărul de loturi generate, am făcut suma dintre loturile generate de fiecare Entity Generator. Pentru a face suma, m-am folosit de blocul Sum, în care se realizează operații aritmetice:



PROBLEMA 2

se dau:

- Un router conectează mai multe surse de pachete de date cu mai multe destinații.

surse de pachete = Entity Generator

destinații = Entity Terminator

router-ul conectează mai multe surse printr-un Entity Input Port

- Routerul examinează adresa IP destinație a pachetului și îl trimite pe ieșirea corespunzătoare.

router-ul trimite pe ieșirea corespunzătoare = trimite pe o ramură a unui Entity Output Port

- routerul va avea trei surse de date și trei destinații.

3 Entity Generator, 3 Entity Terminator

- Fiecare pachet va avea ca attribute adresa IP sursă, adresa IP destinație și lungimea datelor.

Fiecare Entity Generator are atributele adresa IP sursă, adresa IP destinație, lungimea datelor/pachetului

- Pentru simplificare, adresele IP au valorile 1, 2, 3. Ele sunt memorate în attributele Source și Destination.
- Lungimea pachetului, Length, va fi un număr întreg cuprins între șase și zece cu distribuție uniformă.

*valori în intervalul (6,10), adică **6 + 4 * rand***

- Blocurile Entity Generator generează entități cu distribuție exponențială și valoare medie 8ms.

dt = exprnd(0.008);

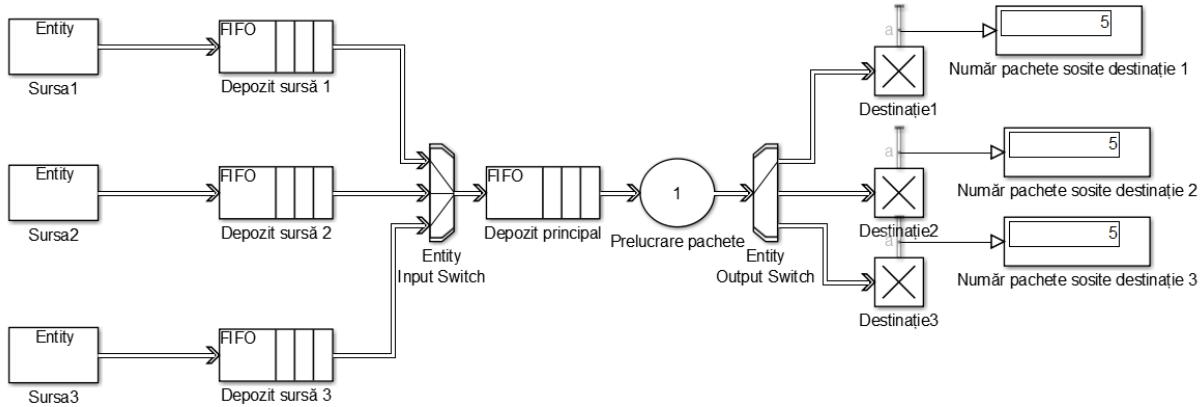
- trei blocuri Entity Queue conectate la cele trei intrări
- un bloc Entity Input Switch ce combină pachetele într-o singură coadă
- un bloc Entity Server cu timpul de deservire dat de atributul Length
- un bloc Entity Output Switch ce va dirija pachetele către cele trei ieșiri conform atributului Destination.
- Ieșirile blocului Entity Output Switch vor fi conectate la trei blocuri Entity Terminator.

se cere:

- Se va simula modelul pe 10s.
- numărul de pachete sosite la fiecare adresă destinație.

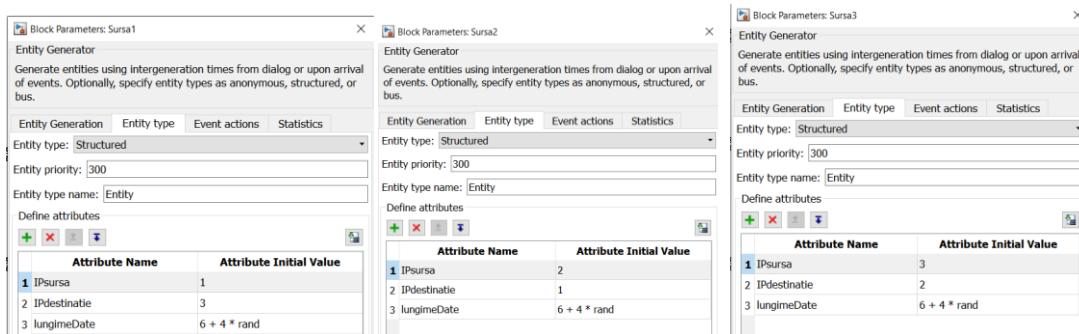
rezolvare:

Exercițiul realizat de mine arată astfel (ceea ce e afișat pe Display e pentru 100 secunde, nu 10):

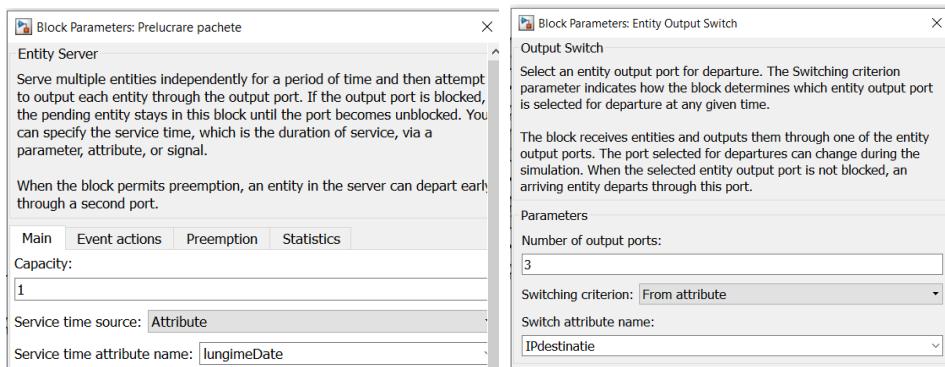


O să atașez poze cu felul în care am configurat blocurile corespunzătoare surselor 1,2 și

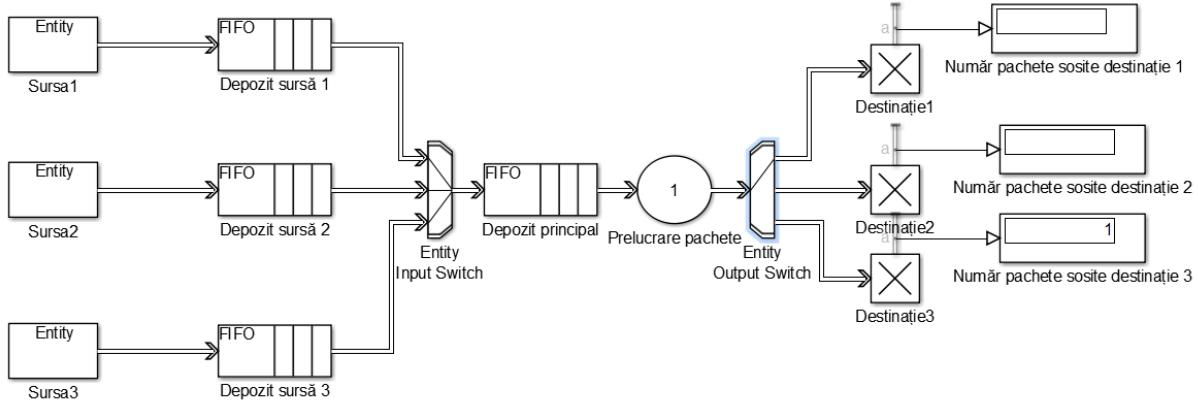
3.



De asemenea, Entity Generator și Entity Output Port au următoarele:



Pentru 10s se vor afișa următoarele:



PROBLEMA 3

se dau:

- se prelucrează piese ce apar la intervale de 2 ± 1 minute cu distribuție uniformă.

*valori din intervalul $(120-60, 120+60) = (60, 180)$. (deci $60 + 120 * \text{rand}$)*

- piesele sunt încărcate pe mașini, iar timpul de încărcare este de 1 minut ± 20 secunde cu distribuție uniformă.

*valori din intervalul $(60-20, 60+20) = (40, 80)$. (deci $40 + 40 * \text{rand}$)*

- timpul de prelucrare pe mașină este 3 ± 1 minute cu distribuție uniformă.

*valori din intervalul $(180-60; 180+60) = (120, 240)$. (deci $120 + 120 * \text{rand}$)*

- 30% din piese suportă încă o prelucrare cu timpul de 2 minute ± 50 secunde cu distribuție uniformă.

*valori din intervalul $(120-50, 120+50)$. (deci $70 + 100 * \text{rand}$)*

30% din piese – event action în Entity Generator

se cere:

- Se va preciza unitatea de timp.

unitatea de timp aleasă este secunda.

- Să se studieze comportarea sistemului pe o durată de 8 ore.

```

if rand < 0.3
    entity.prt = 2;
else
    entity.prt = 1;
end
  
```

$$8 * 60 * 60 = 28800 \text{ secunde sunt în 8 ore.}$$

- numărul de piese ce apar în model

numărul de piese ce apar în model reprezintă Number of entities departed din Entity Generator.

- numărul de piese ce suportă o prelucrare

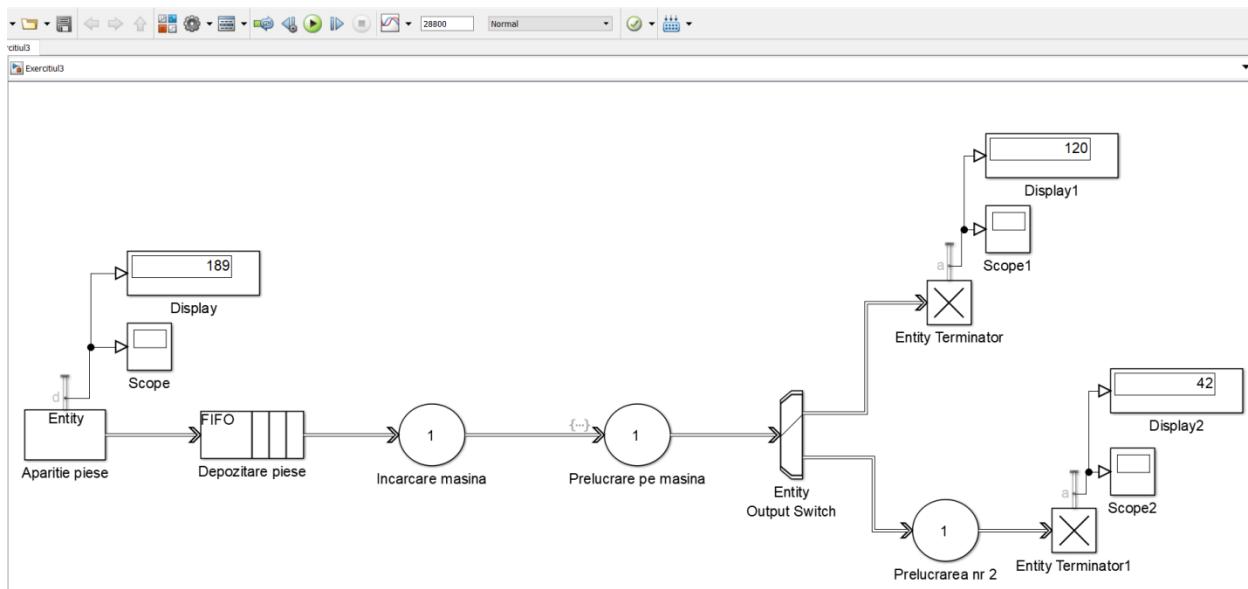
numărul de piese ce suportă o prelucrare reprezintă Number of entities arrived din Entity Terminator (cel de deasupra de switch)

- numărul de piese ce suportă două prelucrări

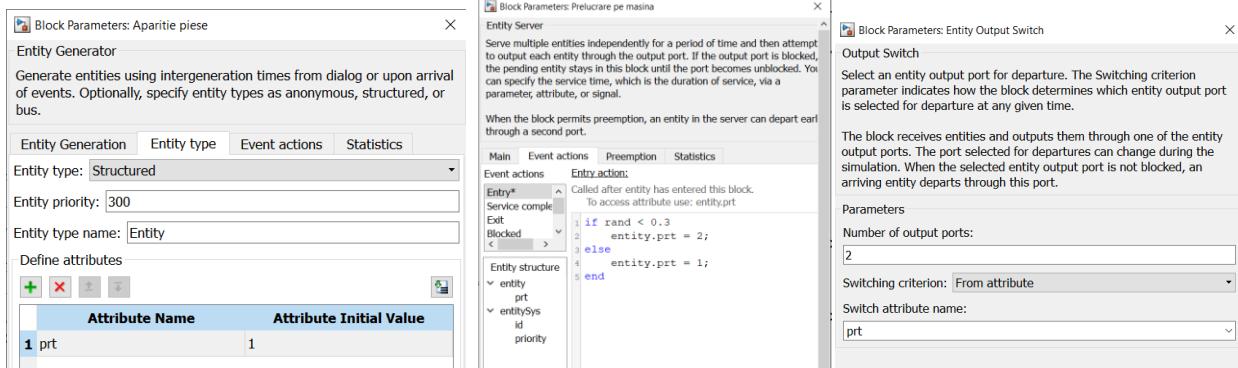
numărul de piese ce suportă două prelucrări reprezintă Number of entities arrived din Entity Terminator (cel de jos de switch)

rezolvare:

Modelul creat de mine arată astfel:



Am setat în Entity Generator/Apariție piese atributa prt, pe care o modific ulterior în Entity Server/Prelucrare pe mașină pentru a putea accesa ambele zone din Entity Outport Switch.



PROBLEMA 4

se dau:

- La o uzină se prelucrează piese ce apar la intervale de 0.5 minute

Entity Generator cu period 0.5

- Piese se pot prelucra pe două mașini

2 Entity Server

- la o mașină prelucrarea unei piese durează 0.9 minute

primul Entity Server – Service Time Value 0.9

- la cealată mașină prelucrarea durează 1.1 minute.

al doilea Entity Server – Service Time Value 1.1

- Piese intră într-o coadă și se alege pentru prelucrare prima mașină liberă.

după ce sunt generate de Entity Generator, piesele trec într-un Entity Import Switch.

se cer:

- Să se studieze comportarea sistemului pe o durată de 480 minute

unitatea de timp aleasă este minutul.

- numărul de piese din coadă

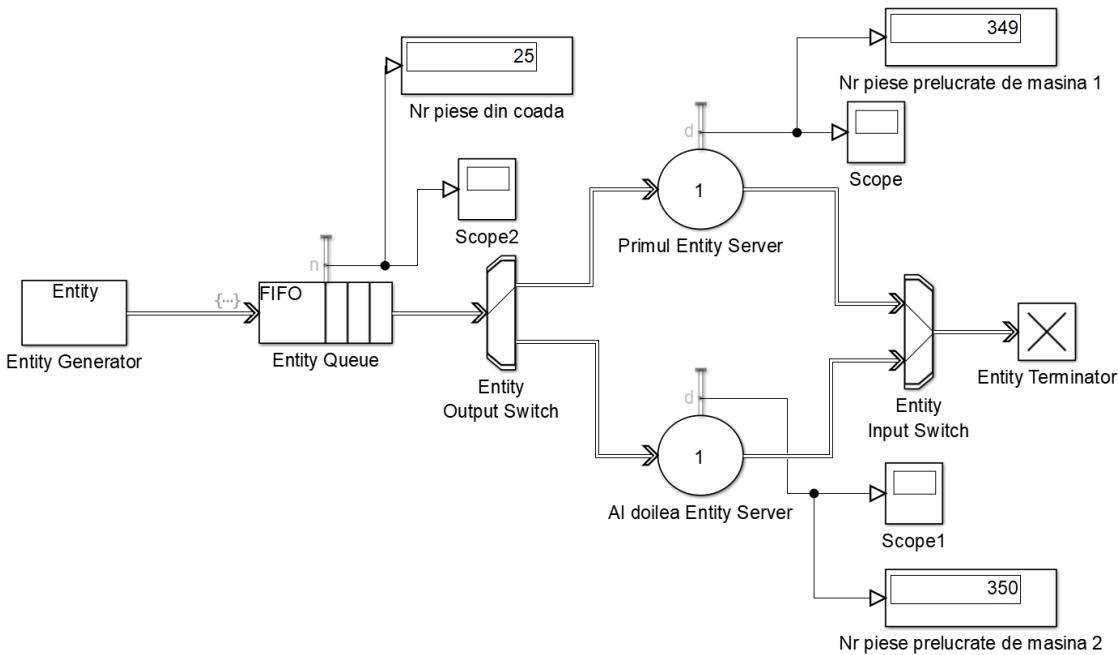
numărul de piese din coadă este

- numărul de piese prelucrate de fiecare mașină.

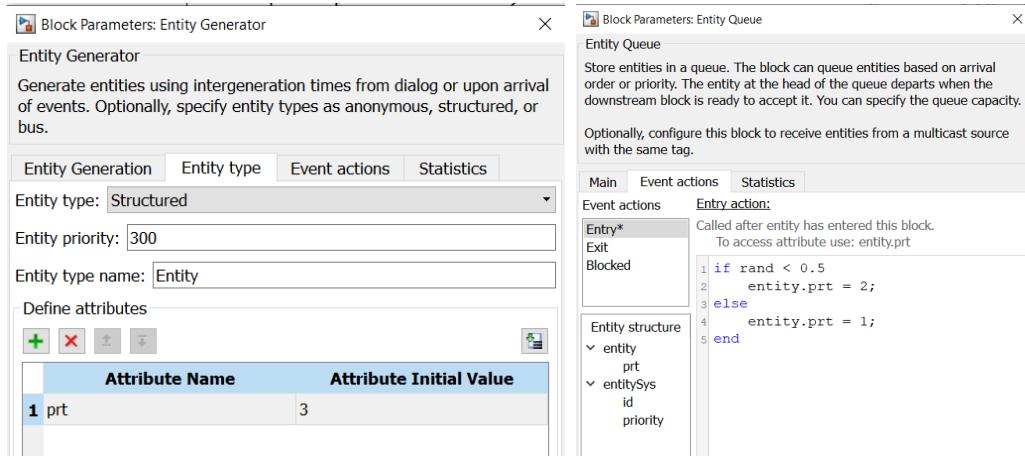
numărul de piese prelucrate de fiecare mașină este

rezolvare:

Modelul realizat de mine arată astfel:



În afară de ceea ce am precizat mai sus legat de Period și Service time value, am mai adăugat încă ceva în Entity Queue și Entity Generator. Atunci când o entitate intră în Entity Queue, i se modifică atributul `prt(priority)`, având șanse egale să meargă către primul Entity Server sau către al doilea.



PROBLEMA 5

se dau:

- La o uzină se produc piese ce presupun două operații.

2 Entity Server

- În prima operație muncitorii asamblează piesele. Operația durează 30 ± 5 minute cu distribuție uniformă.

valori din intervalul $(30-5, 30+5) = (25,35)$. (deci $25 + 10 * \text{rand}$)

- În a doua operație fiecare muncitor utilizează un cuptor pentru a trata termic piesa. Operația durează 8 ± 2 minute.

valori din intervalul $(8-2, 8+2) = (6,10)$. (deci $6 + 4 * \text{rand}$)

- După tratarea termică muncitorii încep construirea unei noi piese.

asamblarea și construirea piesei sunt realizate de aceiași 4 lucrători, doar că, după ce asamblează o piesă, aceasta se duce către tratarea termică; dacă o construiesc, atunci poate fi considerată o piesă terminată și se pune separat.

- Deoarece cuptoarele sunt scumpe, există un singur cuptor în uzină. Există patru muncitori care produc piese.

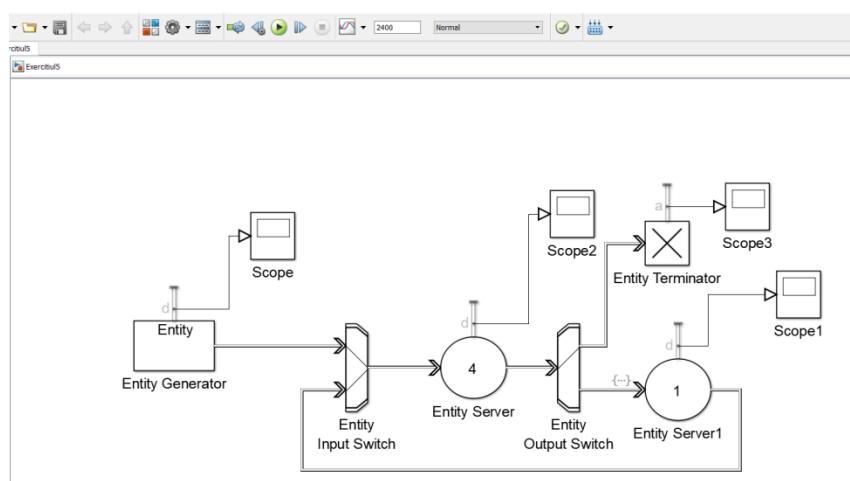
Primul Entity Server are capacitatea 4, iar al doilea are capacitatea 1.

- Se va simula modelul pe 40 ore (cinci zile cu opt ore pe zi).

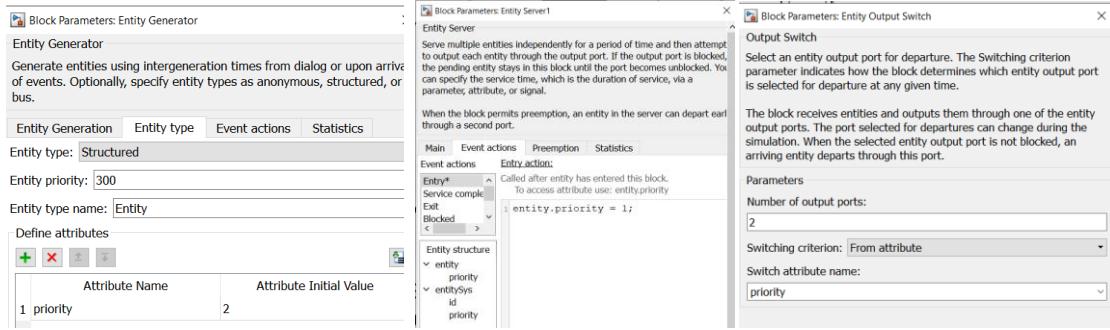
$$5 * 8 * 60 = 40 * 60 = 2400 \text{ ore.}$$

- Piese sunt reprezentate de muncitorii care le produc.
- Se vor genera 4 entități la începutul simulării. Prima operație va fi modelată de un server cu capacitatea patru. După utilizarea cuptorului, entitățile intră înapoi în server.

rezolvare:



În initial, în Entity Input Switch vin valori decât de la Entity Generator. Ele vin cu atributul priority 2. În Entity Server 1, adică după ce piesa este tratată termic, ea primește priority 1, adică o sa mai treacă încă o dată prin Entity Input Switch, va fi prelucrată din nou, iar apoi nu va mai fi tratată termic. În loc să se ducă către Entity Server 1, piesa se va duce către Entity Terminator (pentru că e gata).



PROBLEMA 6

se dau:

- La o mașină unealtă sosesc două tipuri de loturi de piese pentru prelucrare.

sunt două Entity Generator

- Loturile de primul tip sosesc la intervale de 35 ± 10 minute cu distribuție uniformă. Prelucrarea unui lot durează 18 ± 4 minute cu distribuție uniformă.

loturile tip 1 sosesc în intervalul $(35-10, 35+10) = (25, 45)$. (deci $25 + 10 * \text{rand}$)

prelucrarea unui lot tip 1 durează $(18-4, 18+4) = (14, 22)$. (deci $14 + 8 * \text{rand}$)

- Loturile de al doilea tip sosesc la intervale de 45 ± 7 minute cu distribuție uniformă. Prelucrarea unui lot durează tot 18 ± 4 minute cu distribuție uniformă.

loturile tip 2 sosesc în intervalul $(45-7, 45+7) = (38, 52)$. (deci $38 + 14 * \text{rand}$)

prelucrarea unui lot tip 2 durează $(18-4, 18+4) = (14, 22)$. (deci $14 + 8 * \text{rand}$)

- Loturile de ambele tipuri intră în aceeași coadă printr-un bloc Entity Input Switch.

se cer:

- Să se simuleze modelul pe o durată de 960 minute.

unitatea de timp aleasă este minutul

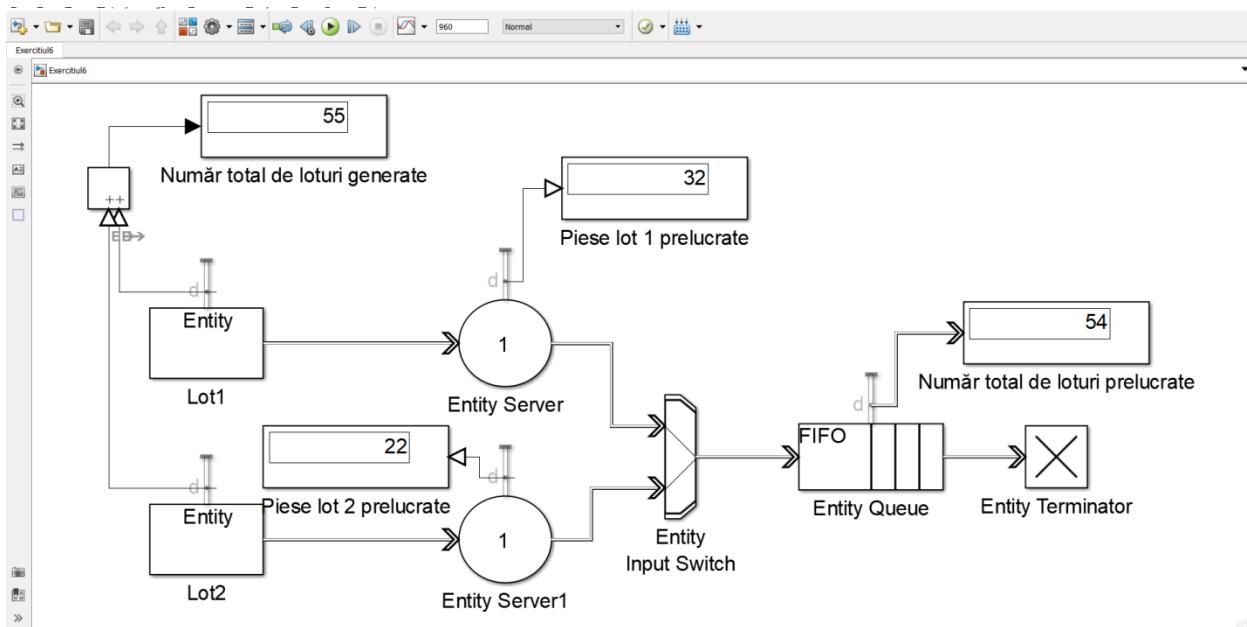
- numărul de loturi generate

numărul de loturi generate este suma dintre loturile tip 1 și loturile tip 2, adică suma dintre Number of entities departed din Entity Generator.

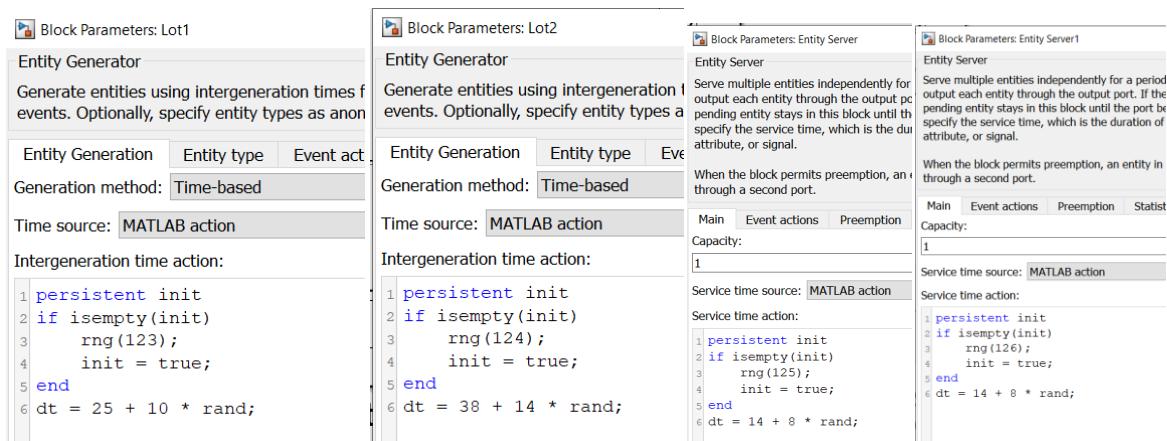
- numărul de loturi prelucrate pentru fiecare tip.

numărul de loturi prelucrate pentru fiecare tip este dat de Number of entities departed din Entity Server-urile corespunzătoare fiecărui lot.

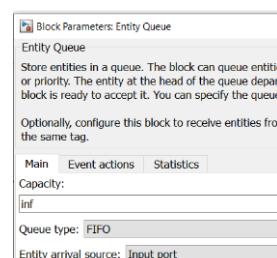
rezolvare:



O să atașez scripturile pentru generarea de loturi și pentru Entity Server:



De asemenea, i-am dat Entity Que-ului capacitatea infinit.



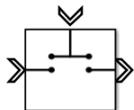
Lucrarea 5

BLOCURI NOI FOLOSITE

În acest laborator au fost prezentate blocurile Entity Gate, Message Send, Message Receive, Pulse Generator, Step, Atomic Subsystem, Subsystem. De asemenea, am făcut exerciții și cu blocurile prezentate în laboratorul 8. Am acolo explicația pentru ele.

Un **mesaj** este o entitate de tip anonim

1. ENTITY GATE



Entity Gate

- are un port de intrare pentru entități și un port de control care primește un mesaj conform căruia se deschide sau închide poarta.
- poate fi folosit ca un enable gate (când primește mesaj pozitiv pe portul de control) sau ca release gate (permite accesul a cel mult o entitate când se primește mesaj, apoi poarta se închide).

2. MESSAGE SEND



Message Send

- crează și trimite un mesaj citind semnalul de intrare la momentele de eşantionare ale blocului sursă

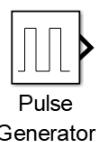
3. MESSAGE RECEIVE



Message Receive

- extrag datele din mesajele primite și le scrie pe output
- dacă nu a primit un mesaj, fie trimit mai departe valoarea default, fie ceea ce a dat când a primit ultima oară un mesaj (mesajul anterior cu delay).

4. PULSE GENERATOR



Pulse Generator

- generează un semnal dreptunghiular în funcție de o formulă ai cărei parametrii sunt aleși de către utilizator.

5. STEP

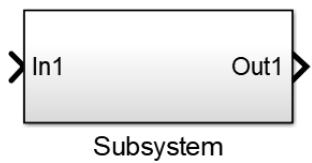


Step

- este folosit pentru a genera un semnal cu rising edge (în general)

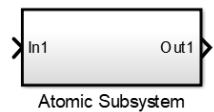
- se poate stabili perioada în care acesta trimită mai departe 0 iar după 1, dar aceste două cifre pot fi modificate.

6.SUBSYSTEM



- folosit pentru a face modelele mai ușor de urmărit
- în interiorul său se realizează anumite modificări ale semnaleului ce provin din In, iar semnalul nou obținut sunt trimite mai departe în Out.

7.ATOMIC SUBSYSTEM



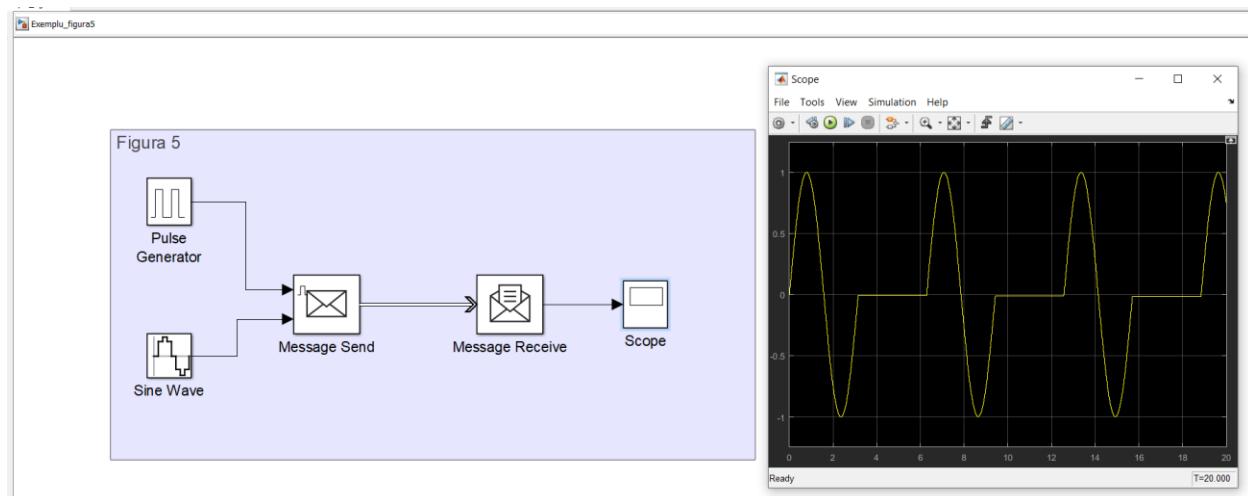
- folosit, în general, pentru modele de dimensiuni mici, dar complexe
- este considerat ca fiind o parte foarte importantă a modelului, indicând un comportament specific în interiorul său.

EXERCITII DIN PLATFORMA DE LABORATOR

PARTEA 1 – UTILIZAREA PORTILOR ÎN MODELE

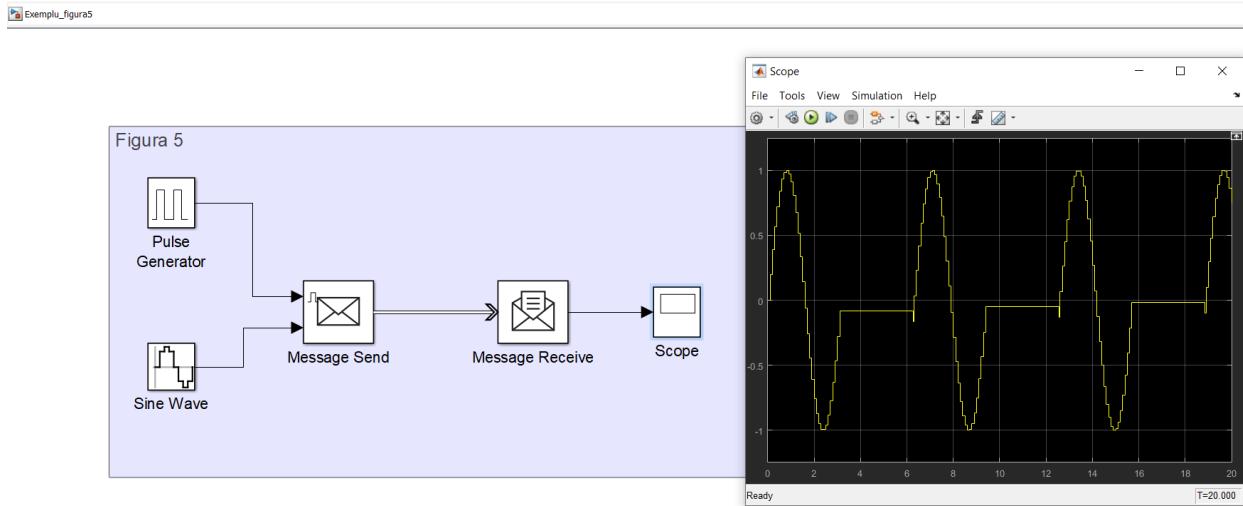
Figura 5

Am realizat figura 5 astfel:



Exercițiu. Se va rula modelul cu perioada de eșantionare de 0.1 s. Se vor compara rezultatele cu cele de la exemplul anterior.

Perioada de eșantionare este, aşa cum s-a precizat și în platforma de laborator, Sample time-ul corespunzător blocului Sine Wave. În urma modificării valorii sale de la 0.01 la 0.1 se obține:



Se poate observa că mesajul generat de Message Receive în cazul în care perioada de eşantionare era 0.01 are o gamă mult mai largă de valori decât atunci când perioada de eşantionare este 0.1. O să explic acest lucru printr-un calcul:

temp total rulare model = 20 unități de timp

număr de dăți în care semnalul își modifică valoarea când sample time-ul este:

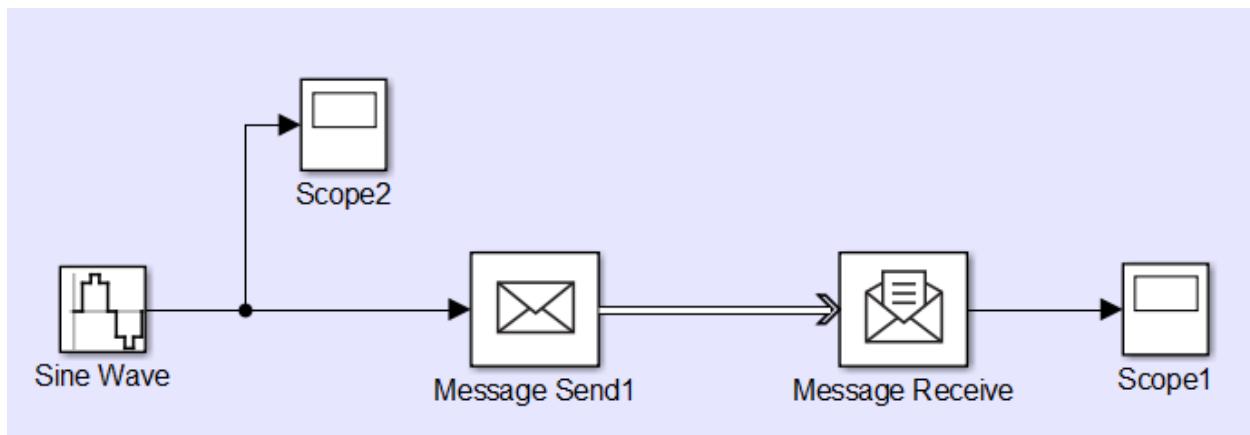
$$0.01 - 20 : 0.01 = 2000$$

$$0.1 - 20 : 0.1 = 200$$

Deci semnalul ia 2000 de valori când sample time-ul setat la Sine Wave este 0.01 și doar 200 când acesta este 0.1 .

Figura 7

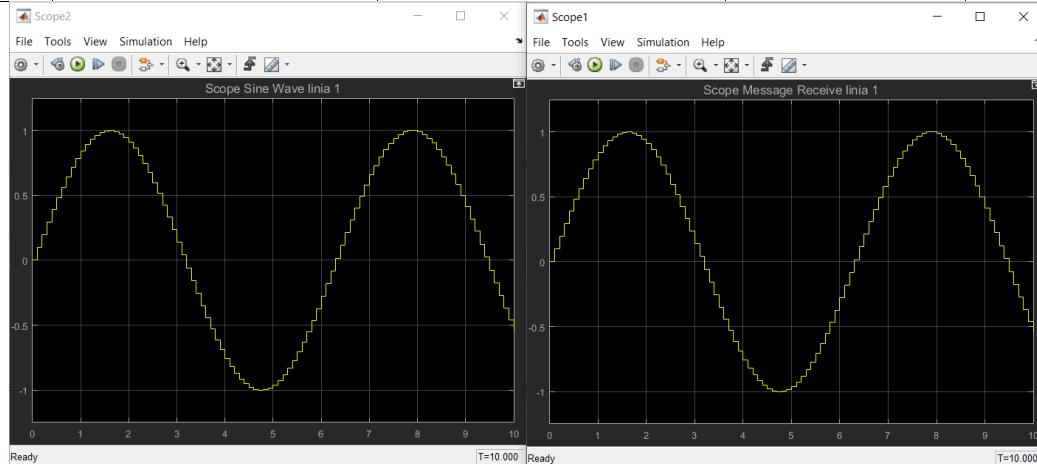
Am realizat modelul din figura 7.



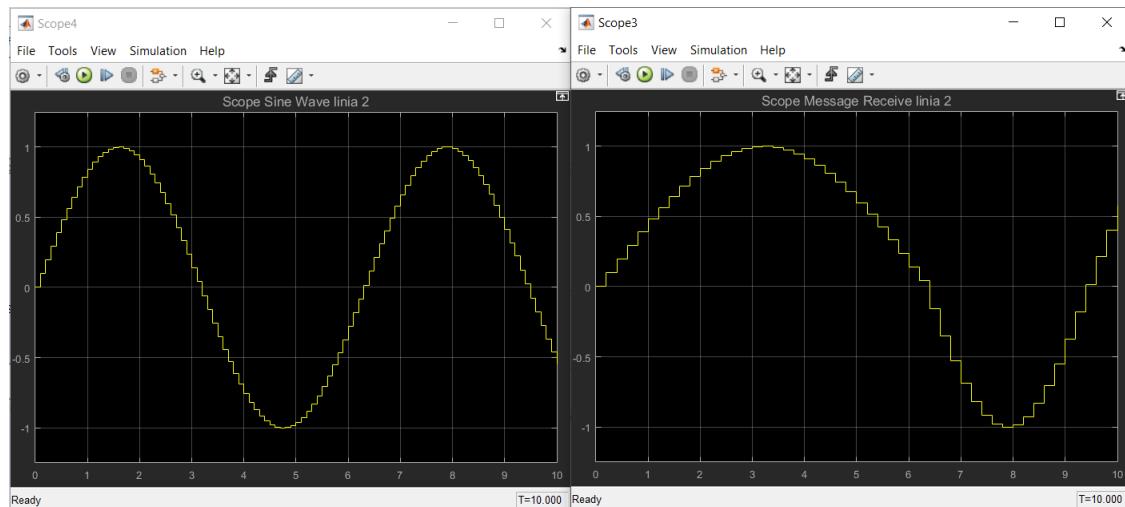
Exercițiu. Fie modelul din Figura 7. Parametrii blocului Sine Wave sunt: Amplitude egal cu unu și Frequency (rad/sec) egal cu unu, vezi Tabelul 2. Se va simula modelul pentru perioadele de eşantionare ale blocurilor Sine Wave și Message Receive și lungimea cozii blocului Message Receive din Tabelul 3. Timpul de simulare va fi 10s. Se vor explica rezultatele.

O să adaug aici semnalele obținute în urma rulării a ceea ce este în tabelul 3.

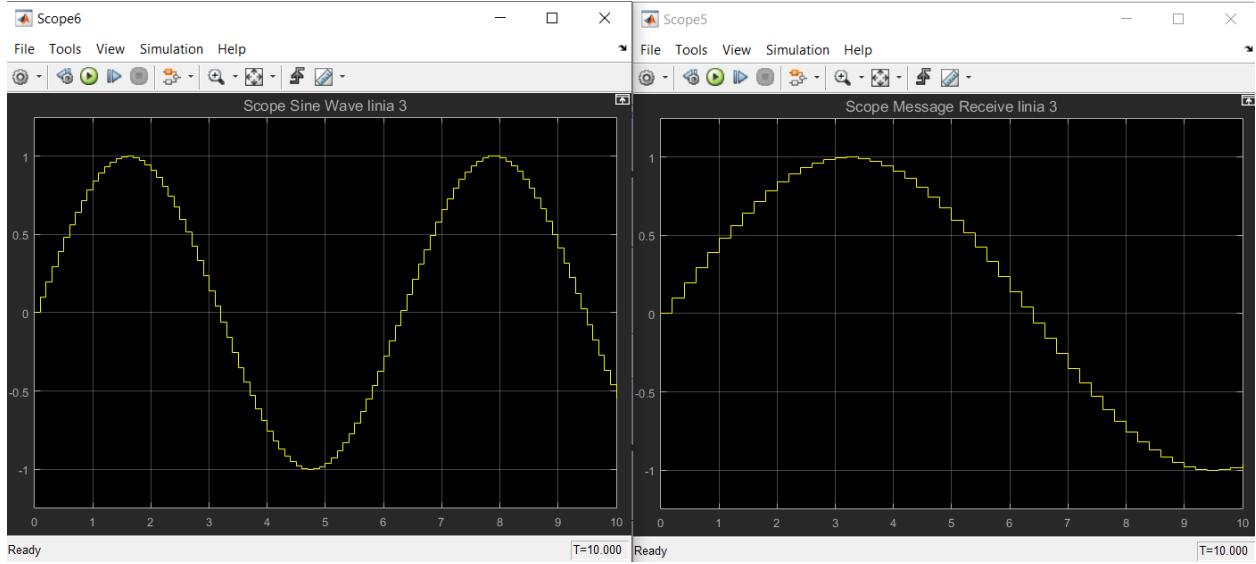
Nr	Sine Wave Sample Time	Message Receive Sample Time	Queue Length
1	0.1	0.1	16
2	0.1	0.2	16
3	0.1	0.2	32
4	0.2	0.2	16
5	0.2	0.1	16



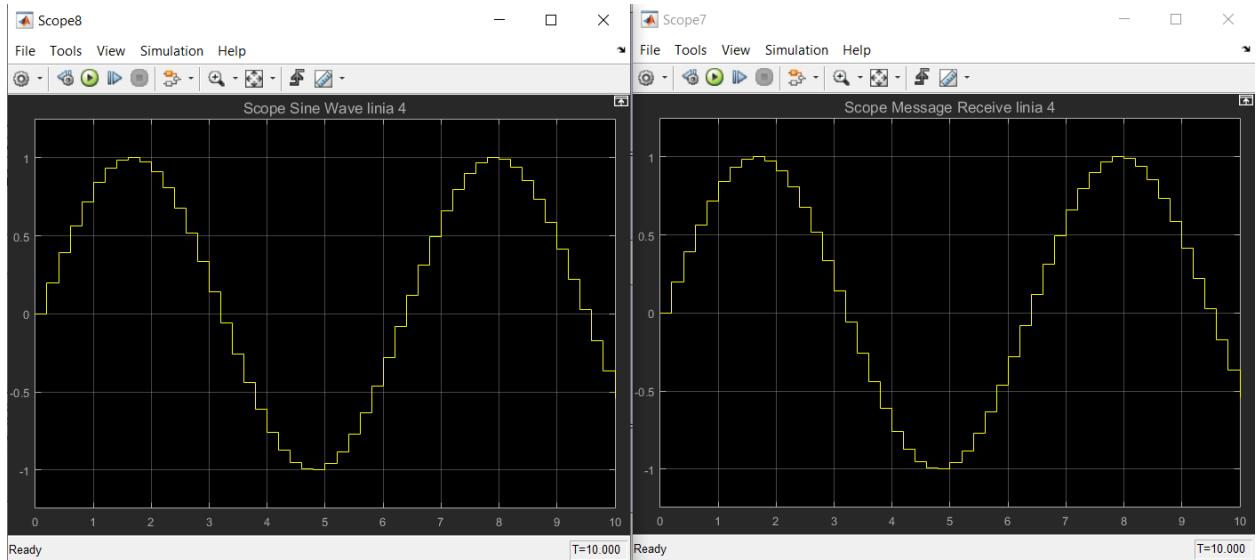
LINIA 1: Cele două semnale corespund, deoarece atât Sine Wave, cât și Message Receive au același sample time.



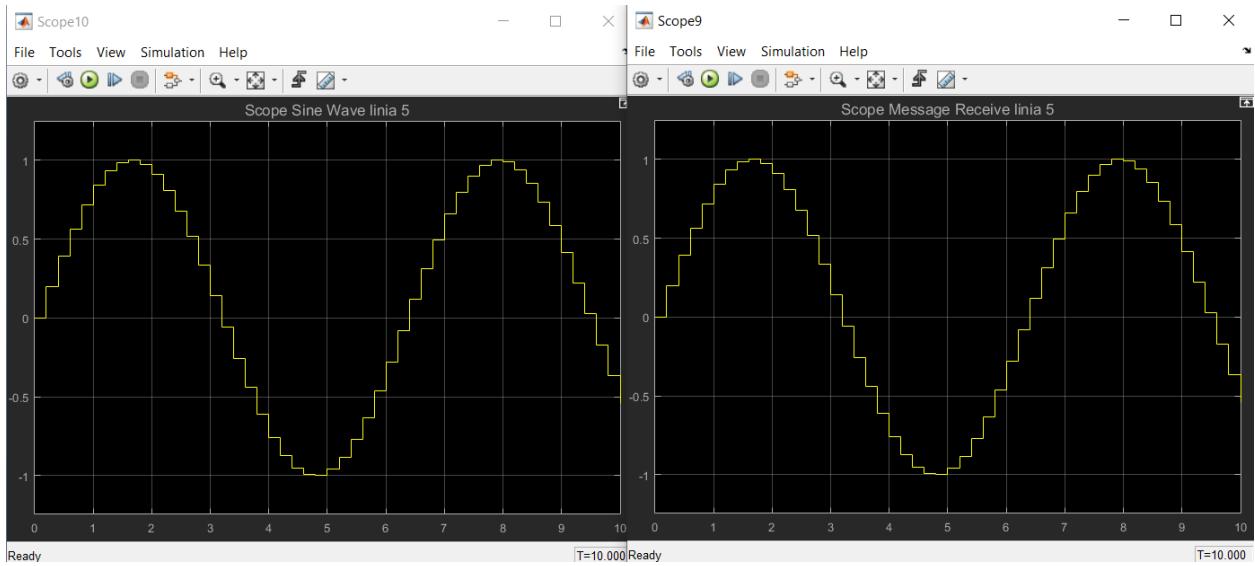
LINIA 2: Semnalul obținut pe Message Receive este semnalul obținut pe Sine Wave alungit. Motivul pentru care arată astfel este că sample time-ul lui Message Receive este jumătate din cel setat pe Sine Wave. Dacă coada din Message Receive ar fi fost plină, automat și Sine Wave-ul ar fi avut sample time-ul "modificat" la 2 până la sfârșitul rulării modelului.



LINIA 3 are aceeași justificare ca linia 2.

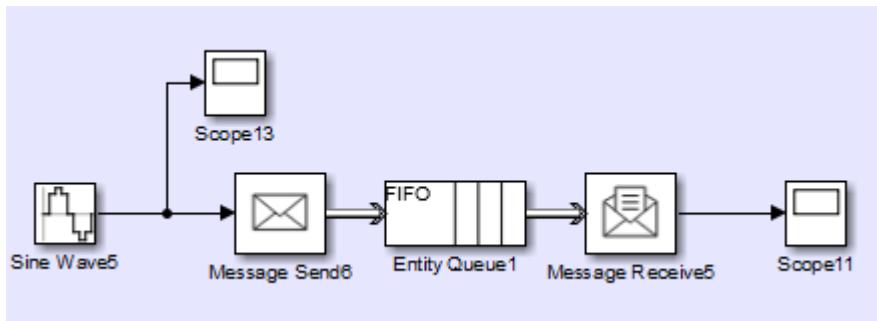


LINIA 4 are aceeași justificare ca linia 1.

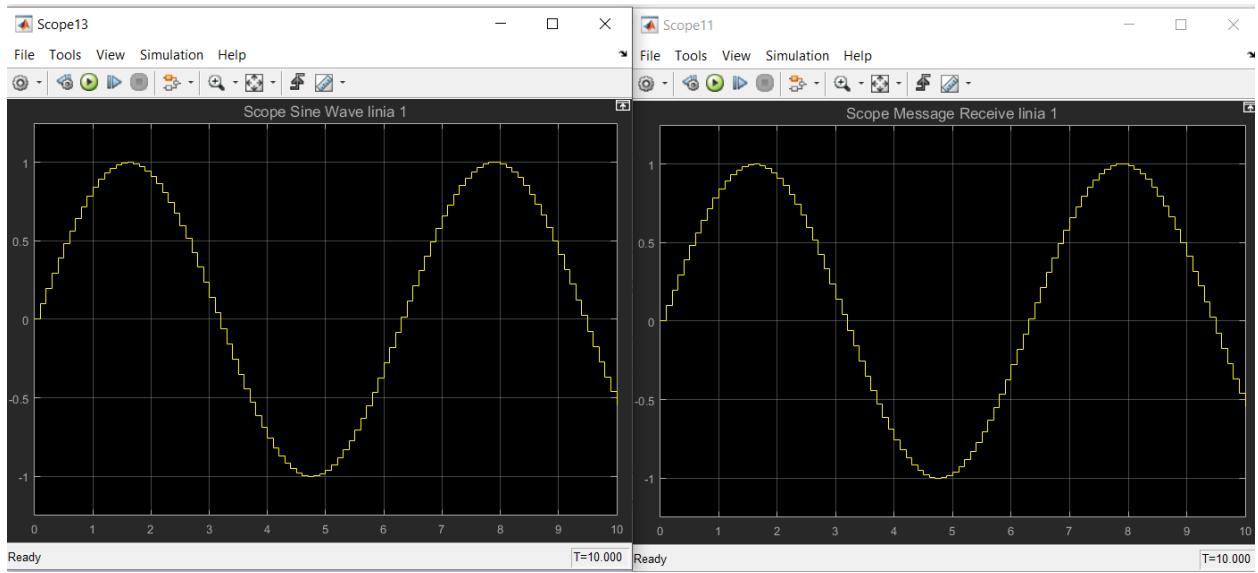


LINIA 5: Deși Message Receive are sample time-ul 0.1, el are ce să trimită mai departe decât odată la 0.2, deoarece Sine Wave are sample 0.2. În acest caz, coada din Message Receive are mereu valoarea cel mult 1, iar imediat ce un semnal a fost generat de Sine Wave, acesta este trimis mai departe de stivă în Message Receive.

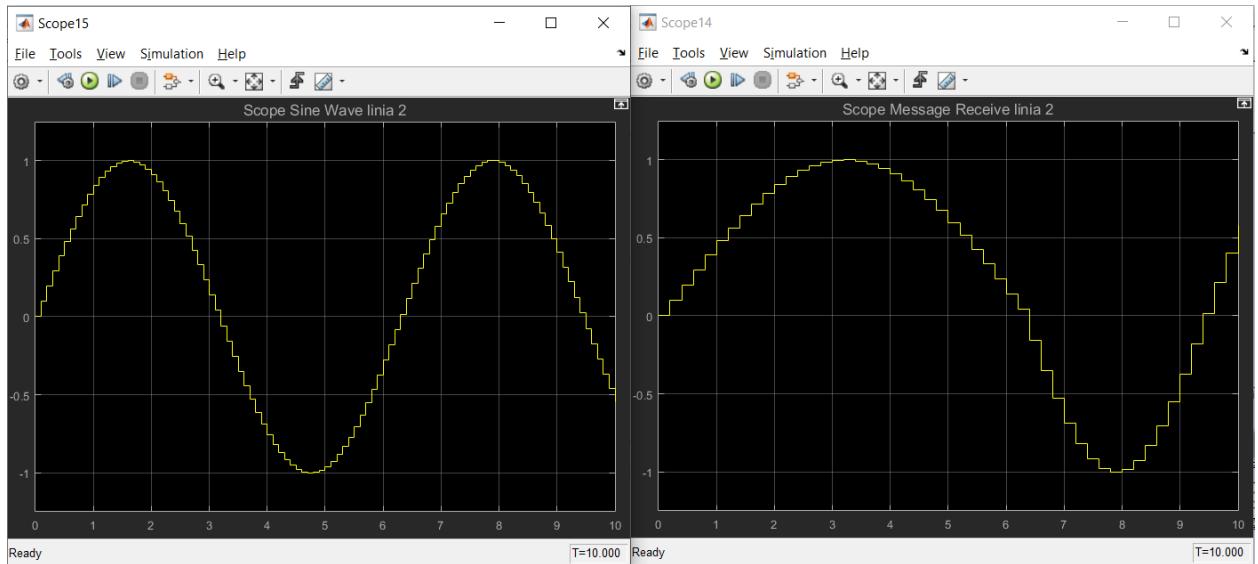
Exercițiu. Se va inseră un bloc Entity Queue între blocurile Message Send și Message Receive și se va simula modelul pentru parametrii din Tabelul 3.



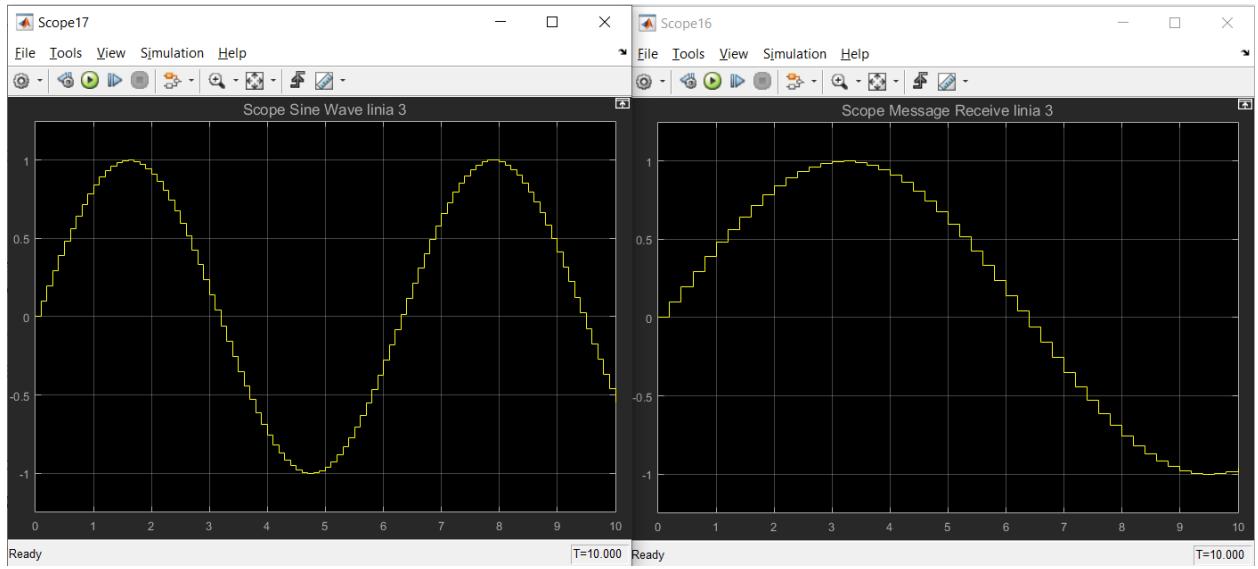
Toate semnalele obținute corespund cu ceea ce era și înainte de Entity Queue.



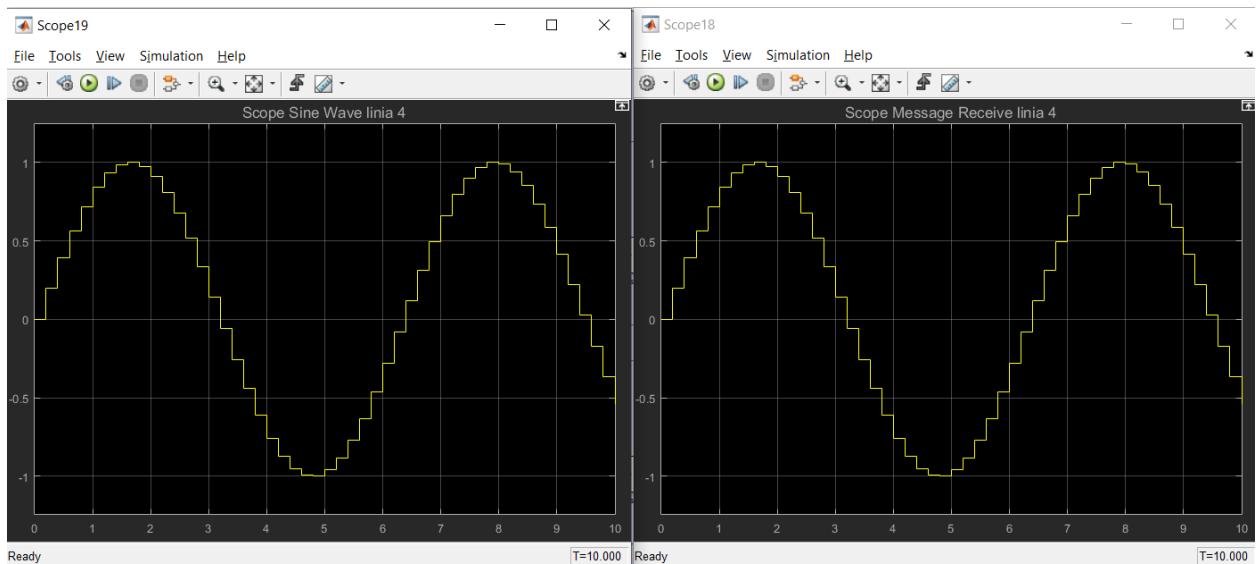
LINIA 1: Cele două semnale corespund, deoarece atât Sine Wave, cât și Message Receive au același sample time. Când au același sample time, semnalul trece imediat prin Entity Queue, astfel că, capacitatea sa nu va depăși niciodată 1.



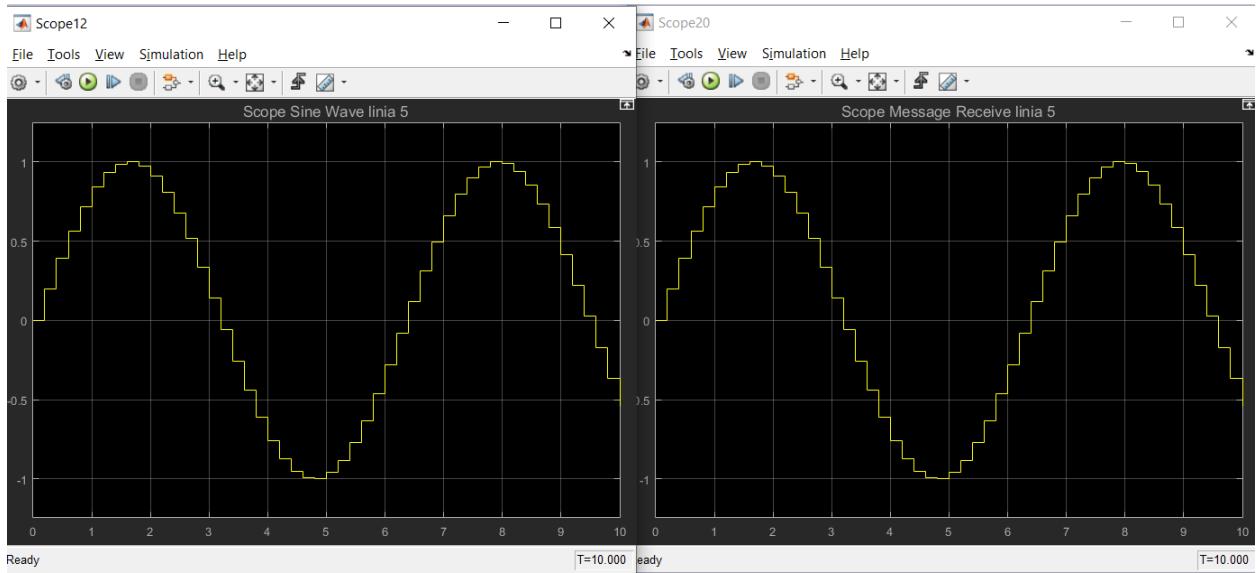
LINIA 2: Semnalul obținut pe Message Receive este semnalul obținut pe Sine Wave alungit. Motivul pentru care arată astfel este că sample time-ul lui Message Receive este jumătate din cel setat pe Sine Wave. Semnalul Sine Wave continuă să se genereze normal până la sfârșitul rulării aplicației, deoarece Entity Queue nu își atinge capacitatea maximă. Dacă Entity Queue ar fi fost plin, automat și Sine Wave-ul ar fi avut sample time-ul "modificat" la 2 până la sfârșitul rulării modelului.



LINIA 3 are aceeași justificare ca linia 2.



LINIA 4 are aceeași justificare ca linia 1.



LINIA 5: Deși Message Receive are sample time-ul 0.1, el are ce ce să trimită mai departe decât odată la 0.2, deoarece Sine Wave are sample 0.2. În acest caz, Entity Queue-ul are mereu valoarea cel mult 1, iar imediat ce un semnal a fost generat de Sine Wave, acesta este trimis mai departe de stivă în Message Receive.

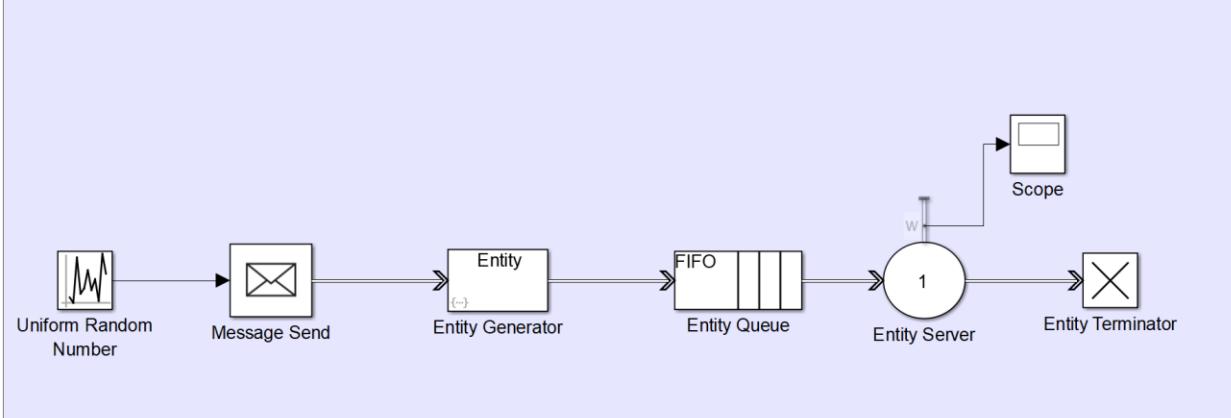
Exemplu. Fie modelul unui atelier din Figura 8 în care sosesc loturi de piese la intervale de 7 ± 6 minute cu distribuție uniformă. Prelucrarea unui lot durează 5 ± 1.5 minute cu distribuție uniformă. Se va simula modelul pe 2400 minute.

intervale de 7 ± 6 minute cu distribuție uniformă – $(7-6,7+6) = (1,13)$

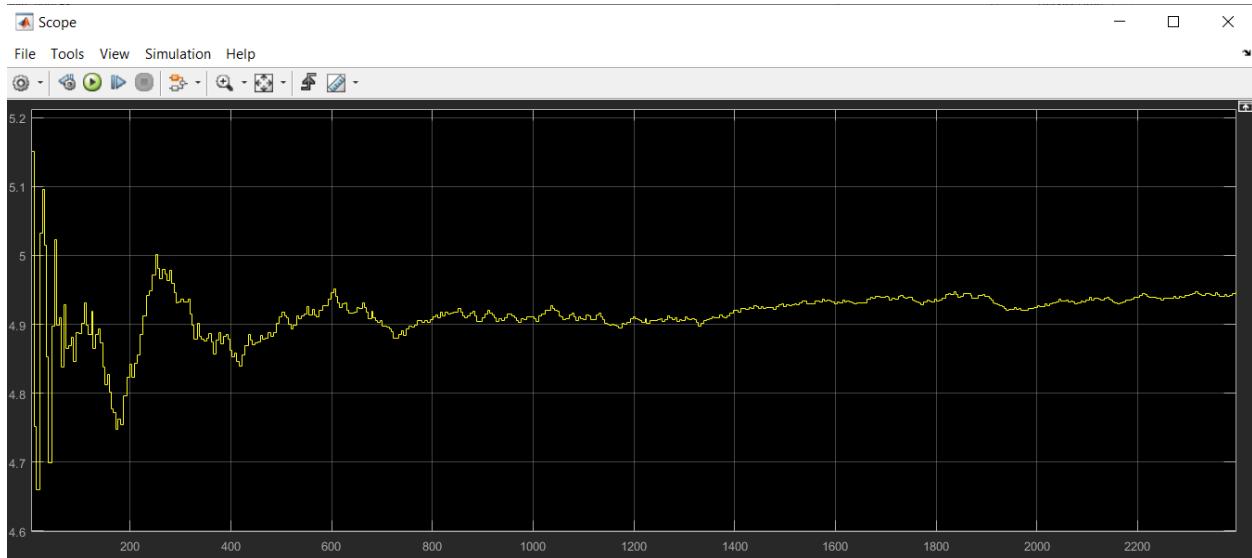
*durează 5 ± 1.5 minute cu distribuție uniformă. – $(5-1.5,5+1.5) = (3.5,6.5)$ ($3.5+3*rand$)*

unitatea de timp aleasă este minutul.

Figura 8

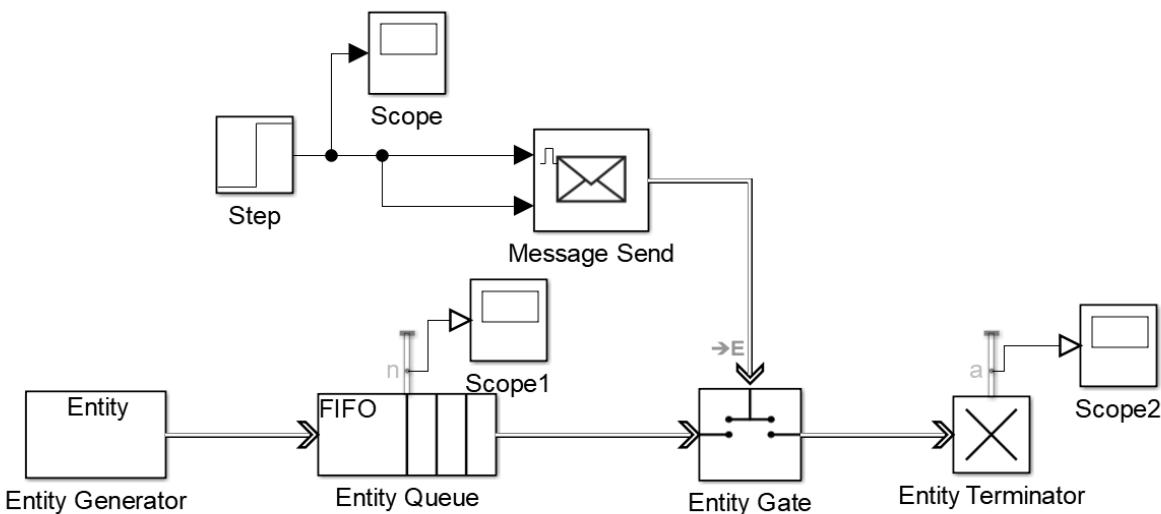


În urma rulării modelului, am obținut timpul mediu de deservire următor:



Exemplu. Fie modelul din Figura 12 în care un semnal treaptă se aplică la intrarea de control a blocului Entity Gate în modul Enable gate, care să deschidă poarta după patru secunde. Semnalul treaptă este generat de blocul Step din biblioteca Simulink/Sources. Parametrul Step time al blocului are valoarea patru pentru ca treapta să apară la 4s, vezi Figura 13. Blocul Entity Generator generează entități cu perioada 1s. Semnalul se aplică prin intermediul un bloc Message Send care convertește semnalul de intrare în entități anonime. Opțiunea Open gate at simulation start din Figura 2 nu este marcată. Blocul Message Send are portul de activare afișat. De ce? Modelul este simulațat pe 10s. Se va observa numărul de entități din coadă înainte și după timpul 4s.

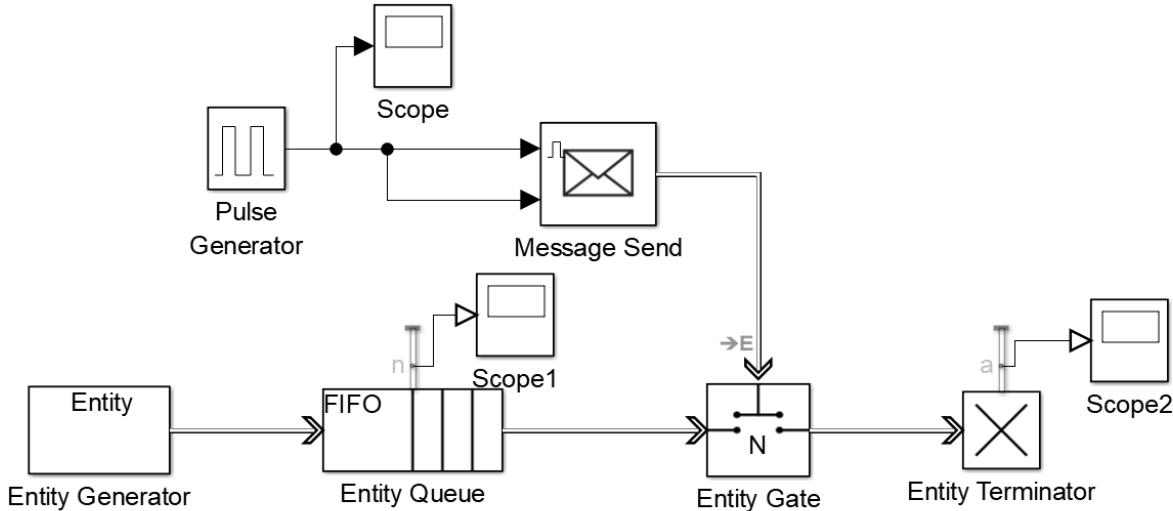
Am făcut și eu modelul din figura 12 și l-am adăugat în arhivă. O să explic felul în care funcționează modelul:



Entitățile se generează din secundă în secundă, iar apoi trec în Entity Queue, a cărei capacitate nu e precizată (am lăsat-o 25 ca fiind valoare inițială). Atâtă timp cât Entity Gate nu primește un semnal anonim cu valoare pozitivă, el nu va permite entităților din Entity Queue să treacă mai departe, astfel că stiva continuă să se umple până când fie ajunge la capacitate maximă, fie până când se dă enable pe Entity Gate. Entity Gate este activat de un Message Send care primește semnale de la Step, dar doar atunci când acesta își ia enable – adică când primește valoarea 1, adică după 4 secunde. El continuă să trimită valori chiar dacă după Step are valoarea 0, deoarece el deja și-a luat enable o dată și trimite ceea ce a primit ultima oară când era enabled. Opțiunea Open gate at simulation start nu este activată, deoarece acest lucru ar fi însemnat că, atunci când se începe execuția programului, Entity Generator ar fi trimis entitățile în Entity Queue și apoi în Entity Terminator, astfel ignorând condiția de enable de pe Entity Gate.

Exemplu. Fie modelul din Figura 14 care controlează accesul entităților cu un bloc Entity Gate. Blocurile Entity Generator generează entități cu intervalul între două entități de 0.5s. Entitățile intră într-o coadă FIFO, iar accesul lor la blocurile Entity Terminator este controlat de un bloc Entity Gate în modul Release gate. Blocul Pulse Generator din biblioteca Simulink/Sources generează impulsuri cu perioada 2s cu factorul de umplere 20% pentru controlul porții. Semnalul este convertit în evenimente de blocul Message Send. Opțiunea Open gate at simulation start din Figura 2 nu este marcată. Blocul Message Send are portul de activare afișat. De ce?

Am făcut și eu modelul din figura 14 și l-am adăugat în arhivă. O să explic ceea ce e diferit față de modelul anterior, de la figura 12:

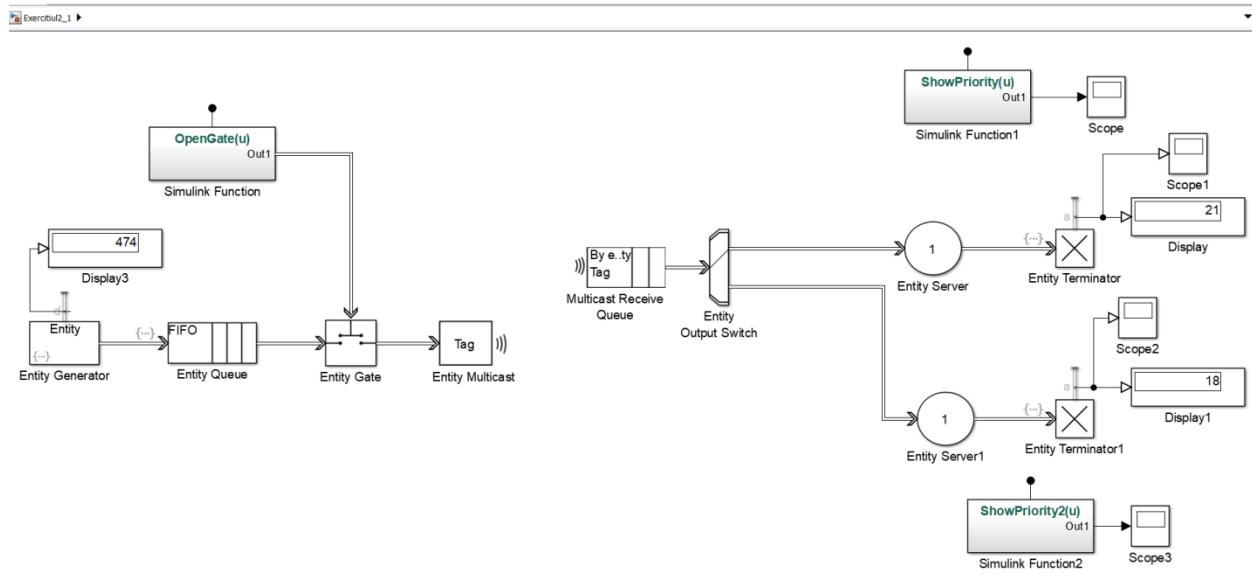


Spre deosebire de ceea ce aveam înainte, acum nu se mai folosește un Step, ci un Pulse Generator. Message Send își ia enable de la Pulse Generator și trimită mai departe valoarea sa. El continuă să trimită valori chiar dacă după Pulse Generator trimite 0, deoarece el deja și-a luat enable o dată și trimite ceea ce a primit ultima oară când era enabled, adică 1. Entity Gate, în

acest caz, este de tip release gate, ceea ce înseamnă că îi permite decât unei singure entități să treacă mai departe când își ia enable. Entity Gate își ia enable de fiecare dată când Pulse Generator devine 1.

2. Probleme propuse

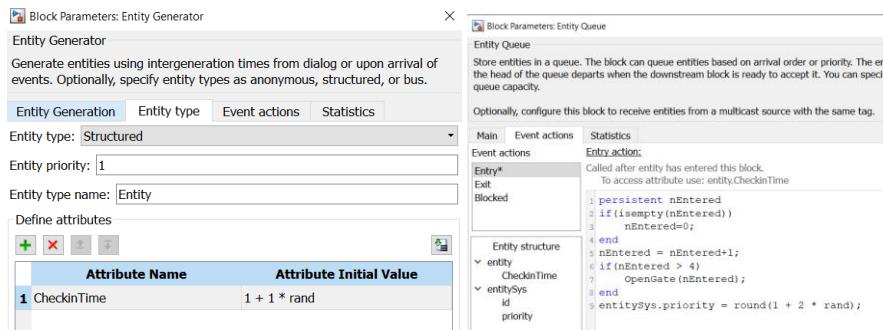
PROBLEMA 1



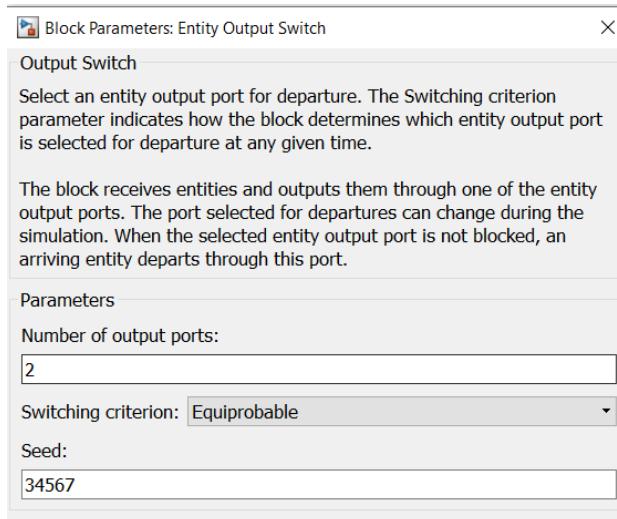
Prin folosirea unui Multicast Receive Queue, se trimit pe ambele Entity Server aceeași entitate. Entity Replicator funcționează pe același principiu ca și un Entity Multicast, astfel că, prin schimbarea făcută, modelul nu își modifică funcționalitatea. Voi explica funcționalitatea modelului:

Entity Generator generează entități care au atributul CheckinTime. Atributul CheckinTime este timpul de verificare la ghișeu și are o valoare randomă în intervalul [1,2].

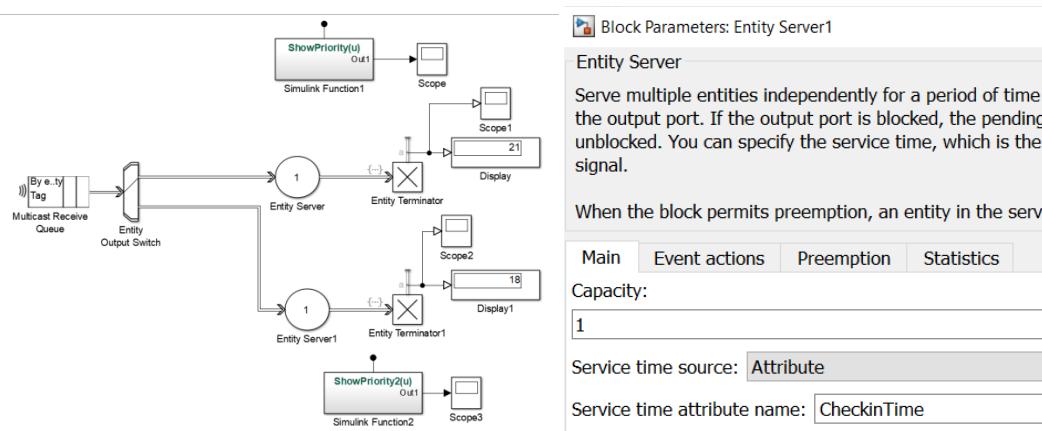
De asemenea, prioritatea entității este 1. După ce intră în Entity Queue, se apelează următorul script care, printre altele, modifică prioritatea main a entității într-un număr între 1 și 3:



Scriptul numără câte entități intră în Entity Queue, iar dacă valoarea este mai mare decât 4 (adică 5), se apelează funcția OpenGate cu valoarea nEntered. Funcția OpenGate are în interiorul său un Message Send care este trimis ca output către Entity Gate, fiind ramura sa de control. Atunci când Message Send trimit un mesaj anonim către Entity Gate, acesta se deschide (este de tip Enable Gate) și permite trecerea entităților în Entity Multicast, ce le trimie mai departe într-o coadă. Din Entity Queue-ul numărul 2 ele intră într-un Entity Output Switch, care are Switching criterior Equiprobable, ceea ce înseamnă că se dă o valoare aleatoare generată de un seed (generator de numere pseudoaleatoare).



Prin folosirea criteriului echiprobabil se îndeplinește ceea ce cere exercițiul, adică ca o entitate să se ducă pe o ramură, iar următoarea să se ducă pe cealaltă ramură a switch-ului.



Ambele Entity Server au timpul de deservire depinzând de atributul CheckinTime. Pentru a afișa prioritatea maină a entității (clasele First Class, Business Class, Economy Class) am folosit încă un script în Entity Terminator-ul de sus:

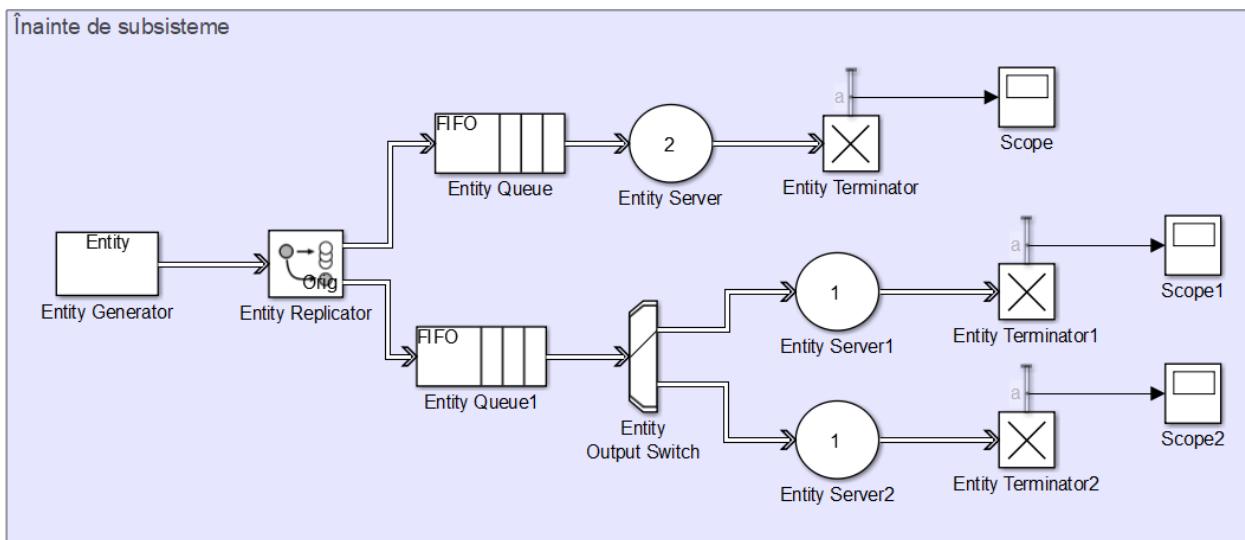


Acest script spune că funcția ShowPriority (din blocul Simulink Function1) are ca și input prioritatea entității care intră în Entity Terminator. Mai departe, am făcut funcția bloc să aibă un output de tip Out, pe care l-am legat la un Scope pentru a putea vedea toate prioritățile entităților ce au trecut prin primul Entity Terminator, iar apoi am duplicat și pentru Entity Terminator1.

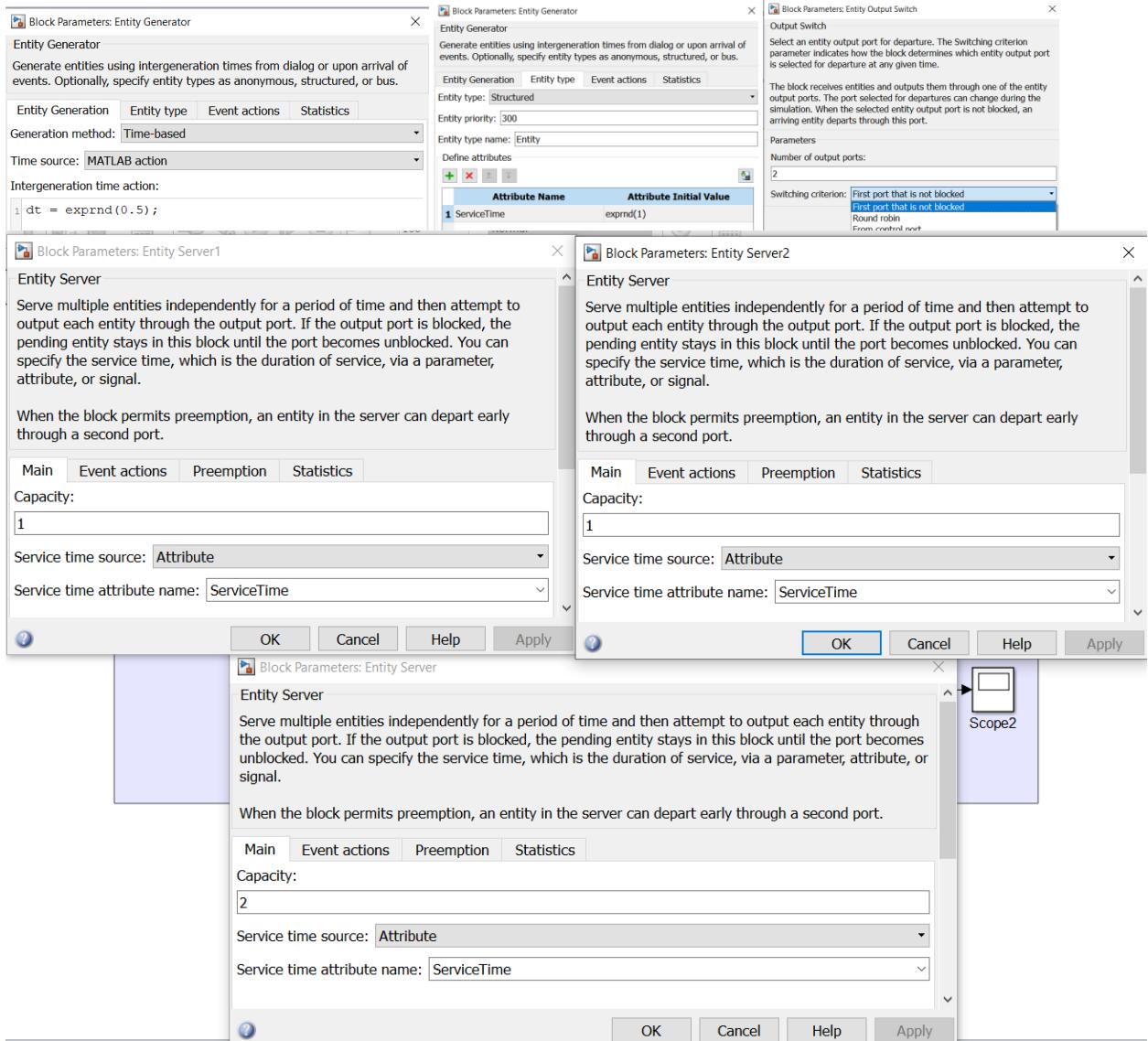
PARTEA 2 – CREAREA DE SUBSISTEME

PROBLEMA 1

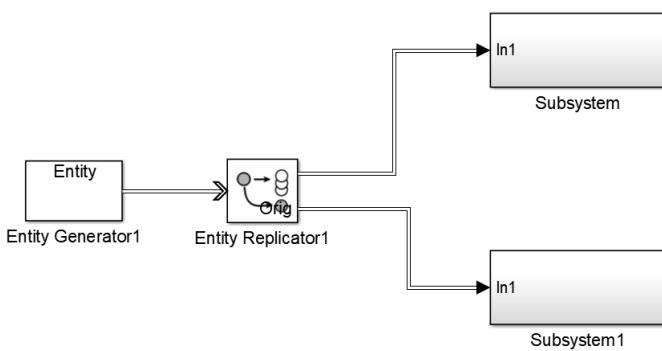
Înainte de a organiza modelul în subsisteme, acesta arăta astfel:

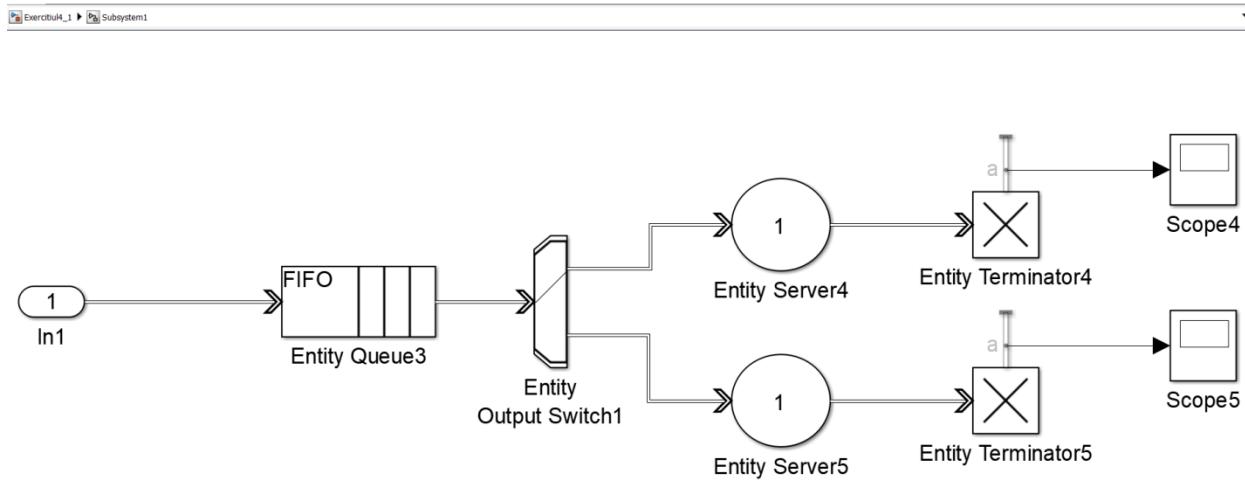
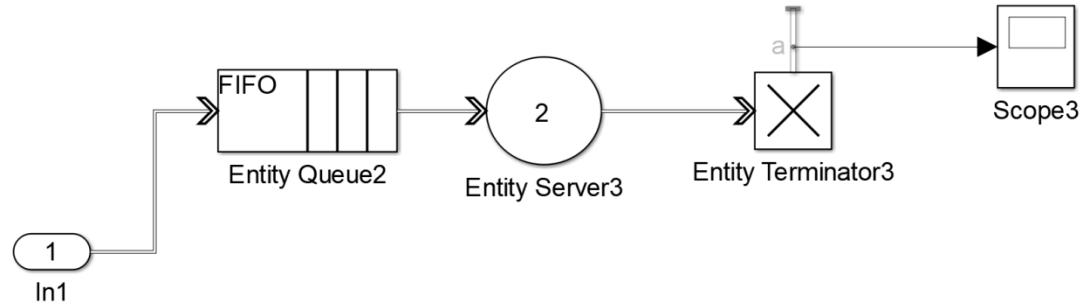


Am ales unitatea de timp ca fiind secunda. O să atașez ceea ce este configurat conform cerinței exercițiului:

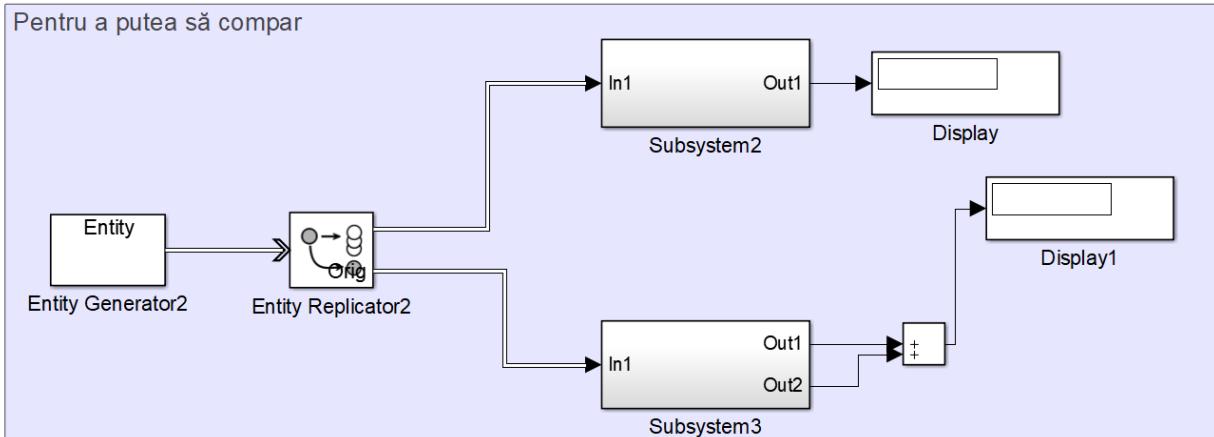


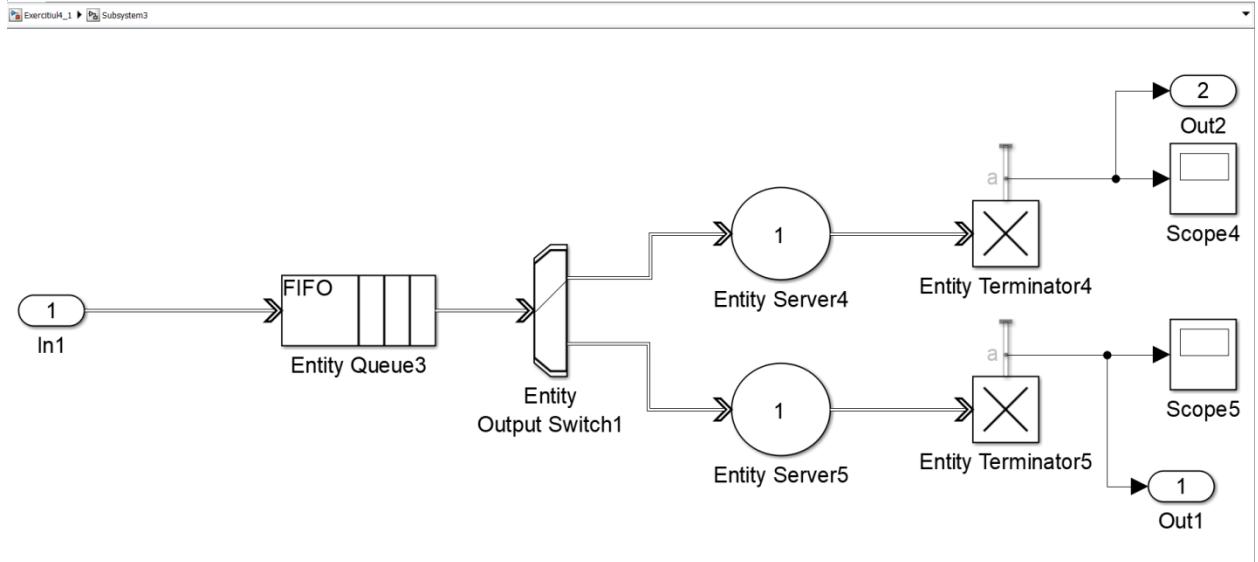
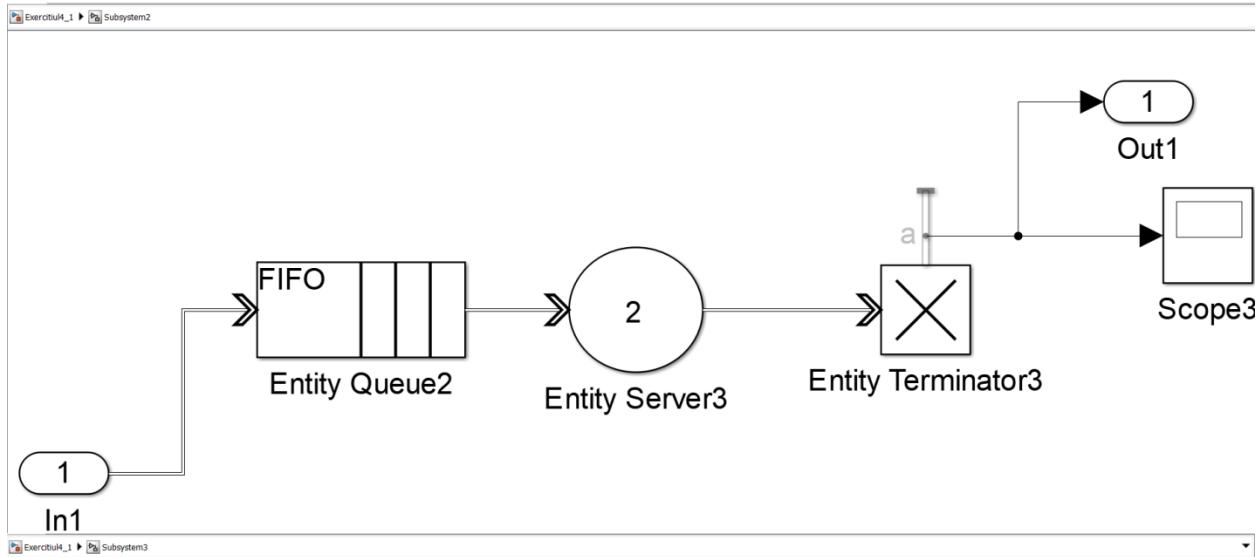
După ce am pus modelul în subsisteme:



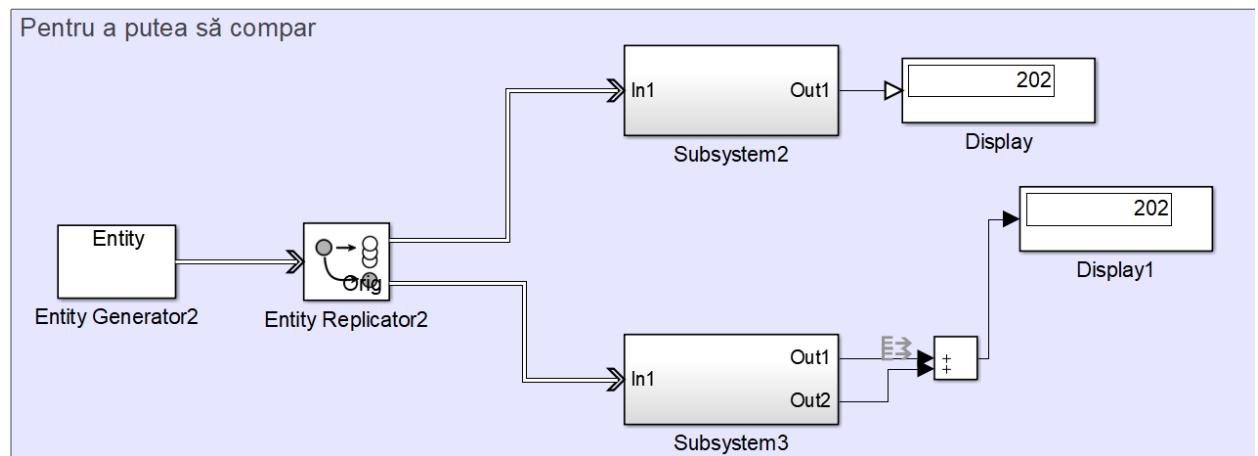


Pentru a putea compara numărul de entități deservite de Entity Server-ul cu capacitate 2 cu suma entităților deservite de cele două Entity Server, am modificat modelul astfel încât output-ul modelului să fie a-ul fiecărui Entity Terminator.





În urma rulării modelului, s-a obținut aceiași valoare pe Display-uri.



Ambele arată 202, deoarece să trimiți 2 entități către un Entity Server cu capacitate 2 este exact ca și cum ai trimite 2 entități într-un switch ce le trimit, mai departe, spre unul din două Entity Server-uri pe criteriul „du-te unde e liber”. Acest lucru nu este general valabil, dar în situația de față toate cele 3 Entity Server au același timp de deservire.

PROBLEMA 2

se dau:

- Într-un port sosesc nave la intervale de 3.2 ± 1.5 h cu distribuție uniformă

navele sunt entitățile generate în Entity Generator

navele sosesc în intervalul $(3.2 - 1.5, 3.2 + 1.5) = (1.7, 4.7)$, adică $1.7 + 3 * \text{rand}$

- Timpul de intrare în port și descărcare este de 8.9 ± 1.5 h cu distribuție uniformă.

intrarea în port și descărcarea se produc în interiorul unui Entity Server

timpul ia valori din intervalul $(8.9 - 1.5, 8.9 + 1.5) = (7.4, 10.4)$, adică $7.4 + 3 * \text{rand}$

- Timpul de ieșire din port este de 0.25 h.

mai este un Entity Server cu Service time value = 0.25

- În port apar furtuni la intervale de 48 ± 4 h cu distribuție uniformă, care durează 4 ± 2 h cu distribuție uniformă

furtunile apar în intervalul de timp $(48-4, 48+4) = (44, 52)$ deci $44 + 8 * \text{rand}$

furtunile durează $(4-2, 4+2) = (2, 6)$ deci $2 + 4 * \text{rand}$

- Intrarea și ieșirea din port se fac dacă există o dană liberă, și nu este furtună. Portul are 3 dane.

conform informațiilor din laboratorul 8, unde au fost prezentate blocurile Resource Pool, Resource Acquirer și Resource Releaser, pot să fac dana ca fiind o resursă.

- Să se simuleze modelul pe o săptămână

unitatea de timp aleasă este ora

$24 * 7 = 168$ ore 

se cer:

- timpul mediu de utilizare a danelor

timpul mediu de utilizare a danelor este dat de Average utilization din Resource Pool

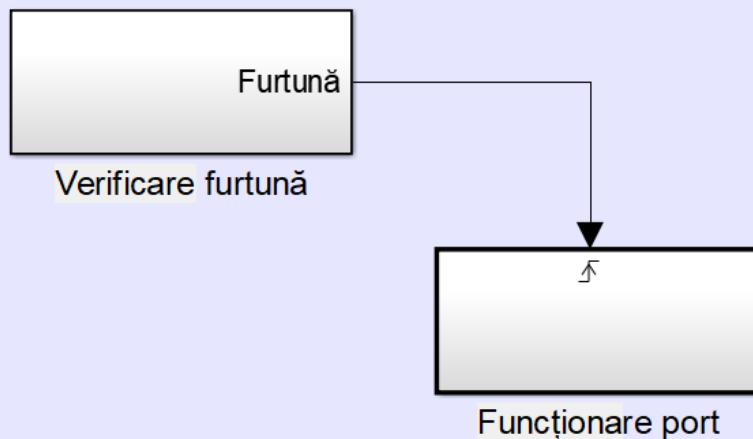
- timpul mediu de așteptare pentru intrarea și ieșirea din port

valoarea este dată de Average wait din Entity Server corespunzătoare intrării și ieșirii din port.

REZOLVARE:

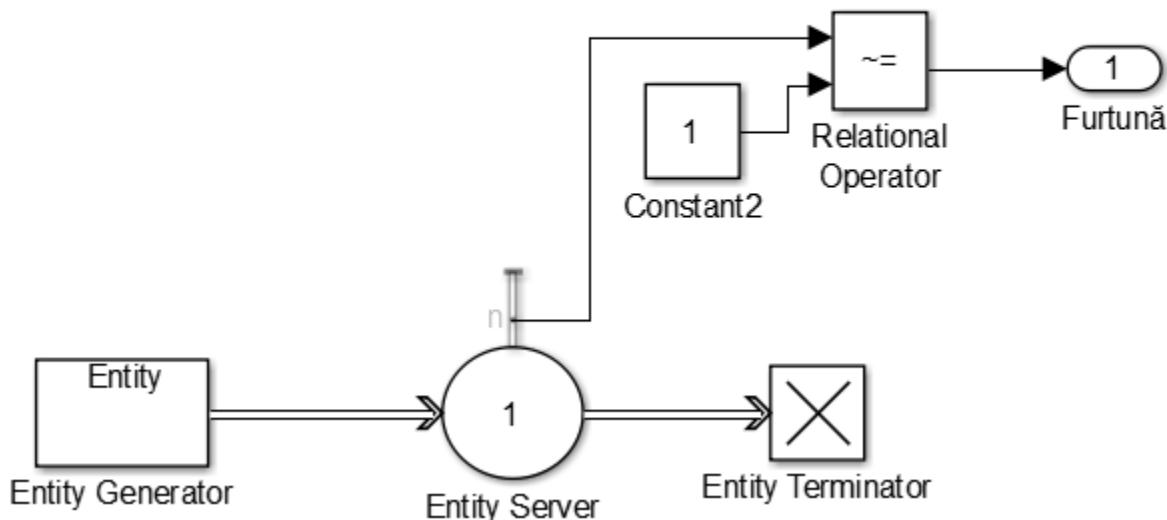
Am făcut exercițiul diferit de cum e în cerință. În loc să mă folosesc de Enable Gate, Message Receiver, Message Sender și Fcn, am ales să fac cu un subsistem trigger. O să explic modul în care funcționează modelul:

Cu furtună



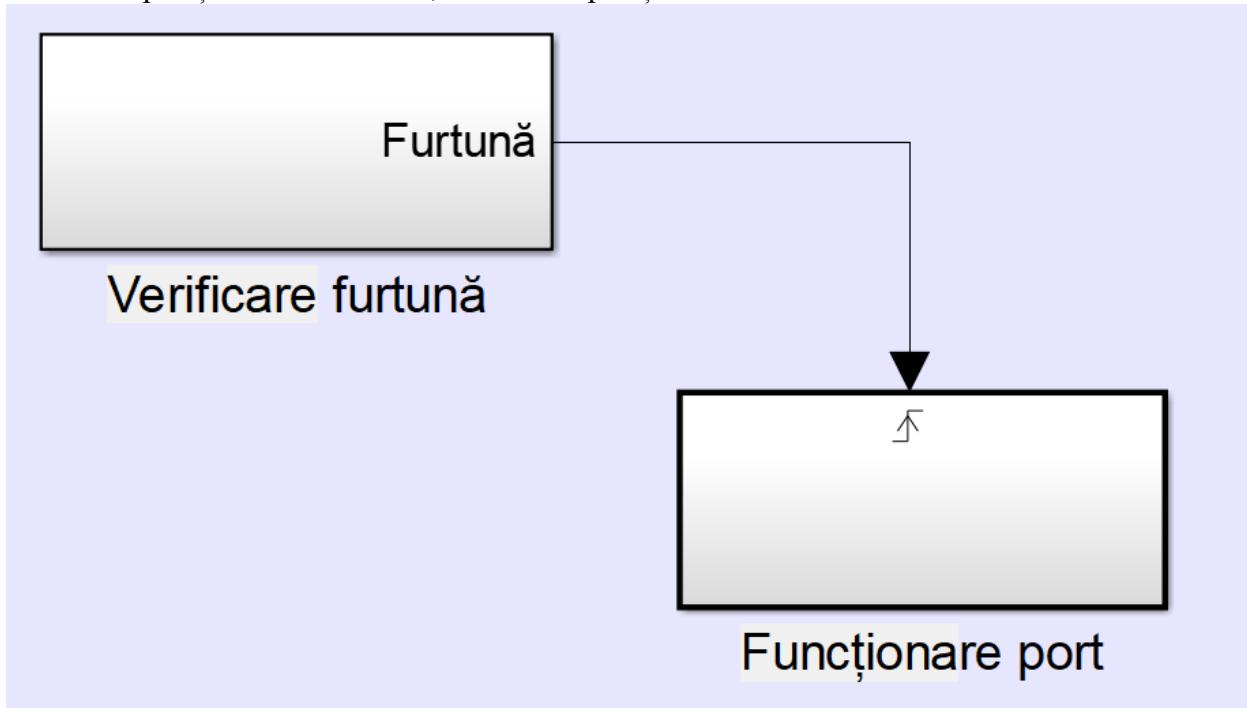
În model sunt două subsisteme, Verificare furtună și Funcționare port. Verificare furtună este un subsistem normal, care funcționează pe tot parcursul rulării modelului.

Exercitiu4_2 ► Verificare furtună

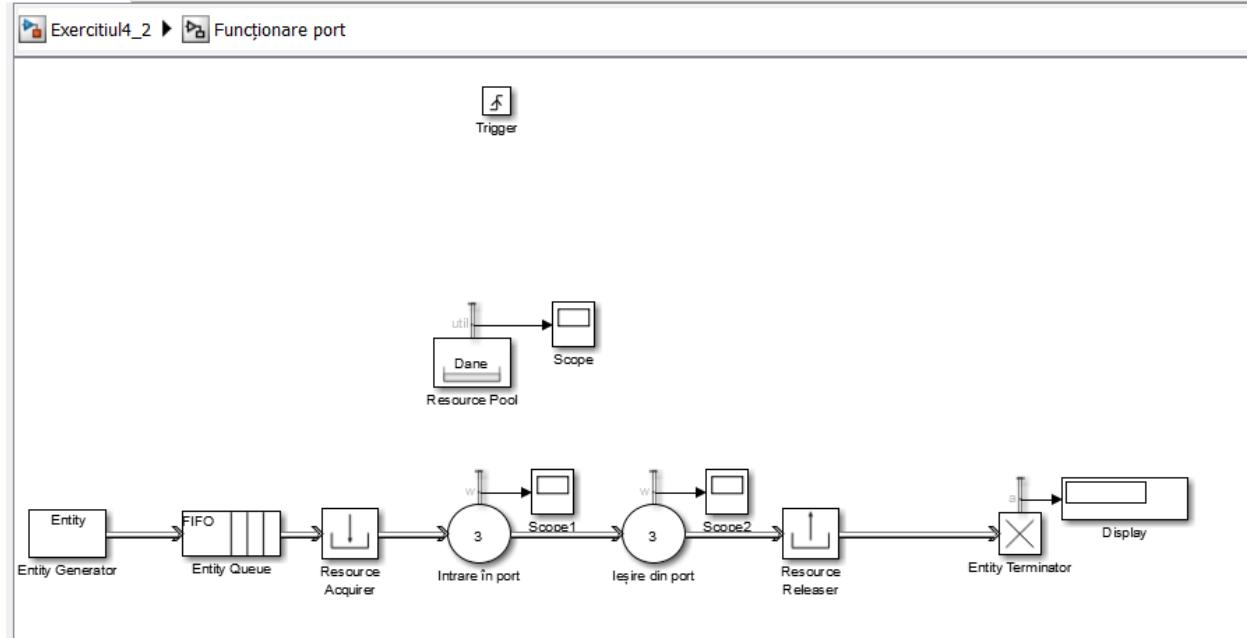


Acest subsistem este responsabil pentru producerea unei furtuni și, implicit, împiedicarea funcționării portului. Nu o să mai adaug timpii de generare a entităților și se deservire a accesora, pentru că i-am explicat mai sus. În Entity Server se afișează Number of entities in block, care este, de fapt, corespondentul luării loc a unei furtuni. Cum Entity Server are capacitate 1, numărul de entități ce pot fi în interiorul său este maxim 1. Atunci când are capacitatea 1, o

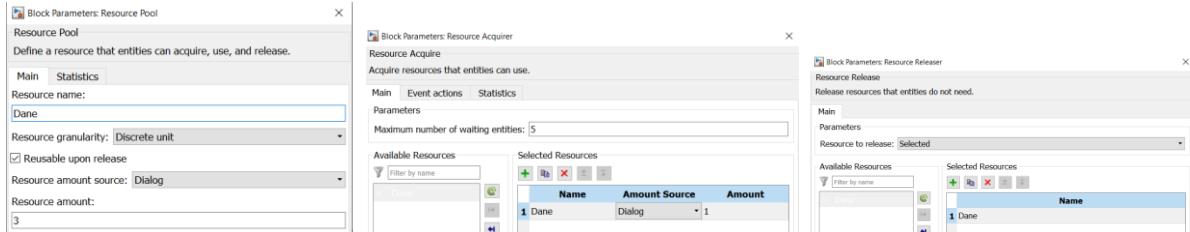
furtună ia loc. Mai departe, eu compar valoarea cu 1. Dacă nu este nicio entitate în Entity Server, atunci comparația dă 1. Dacă este, atunci comparația dă 0.



Funcționare port este un subsistem triggered, adică el funcționează doar atât timp cât primește semnal de enable – adică 1. Conform a ceea ce am explicat mai devreme, subsistemul Verificare furtună trimite semnalul 1 ca output atunci când nu este furtună și 0 atunci când este furtună. În interiorul subsistemului Funcționare port se găsesc:



M-am folosit de resurse pentru a mă asigura că am decât 3 dane, astfel putând să dau capacitatea 3 pentru ambele Entity Server, astfel eficientizând modelul.

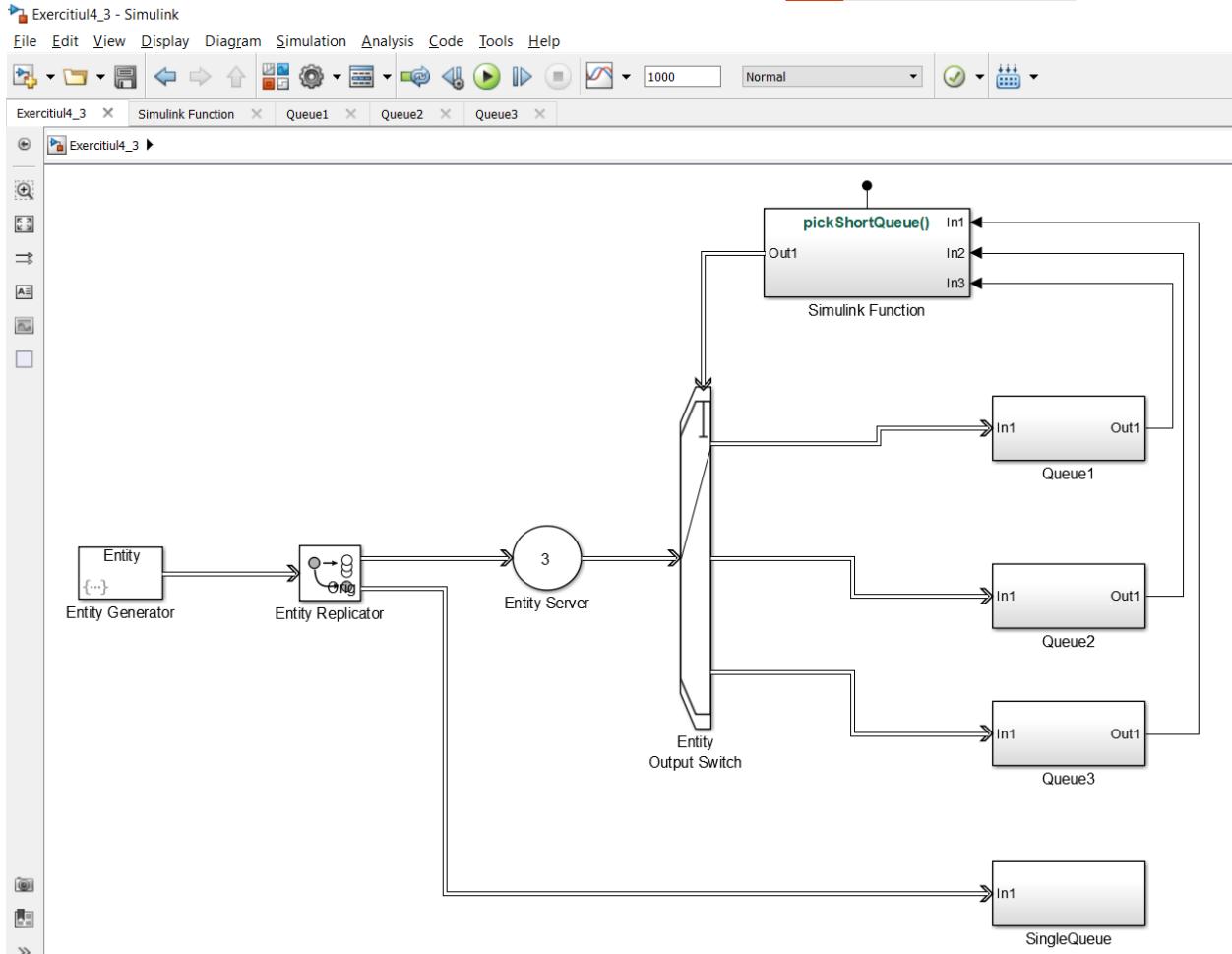


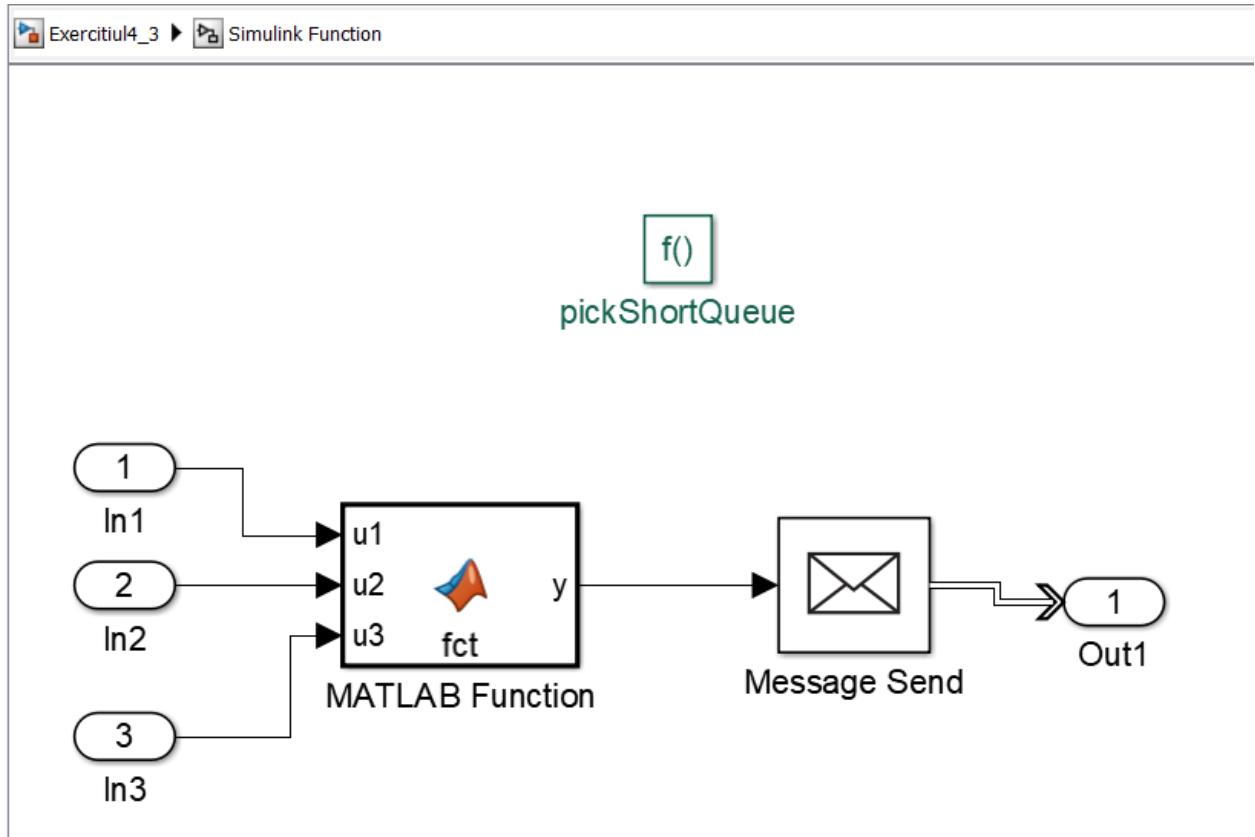
Resource Acquirer alocă o dană unei entități, iar Resource Releaser o ia de la această resursă, trimițând-o înapoi în Resource Pool, de unde poate fi dată din nou unei entități.

Pentru situația fără furtună este îndeajuns să se păstreze în model decât interiorul subsistemului triggered. Am atașat și acest lucru în arhivă, dar cu o singură modificare – timpul de deservire a navelor este 8.4 ± 1.5 h, față de 8.9 ± 1.5 h.

PROBLEMA 3

Am ales unitatea de timp ca fiind minutul. Timpii medii de așteptare în servere este Average wait din Entity Queue-urile corespunzătoare fiecărui subsistem. Modelul a fost deja explicat în platforma de laborator, aşa că doar voi pune poze cu el:





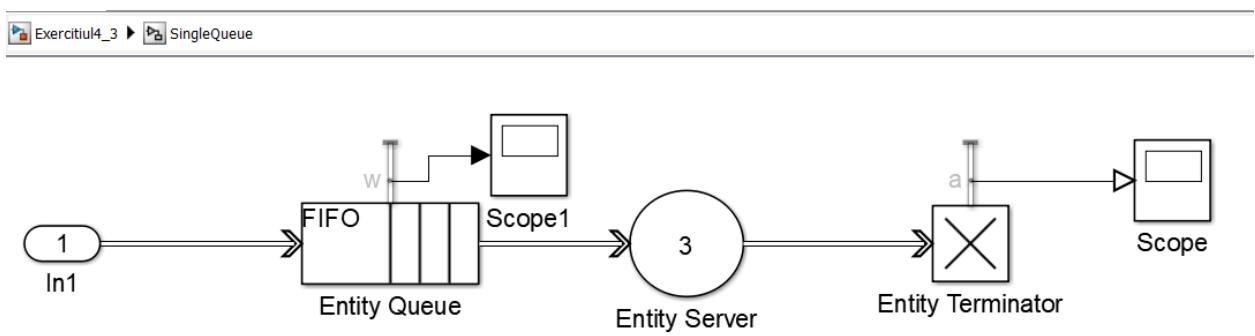
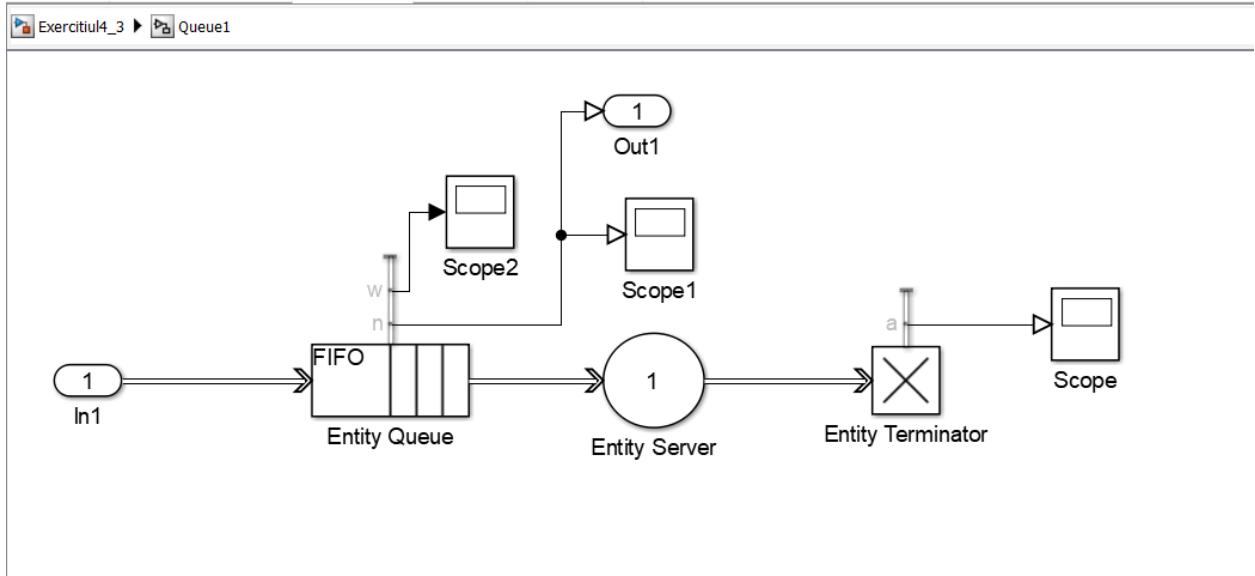
Editor - Block: Exercitiu4_3/Simulink Function/MATLAB Fun...

Simulink Function/MATLAB Function

```

1 function y = fct(u1,u2,u3)
2 %#codegen
3
4 [a,y] = min([u3,u2,u1]);

```

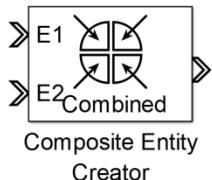


Lucrarea 6

BLOCURI NOI FOLOSITE

În acest laborator au fost prezentate blocurile Composite Entity Creator și Composite Entity Splitter.

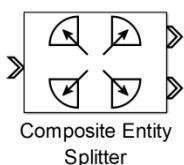
1.COMPOSITE ENTITY CREATOR



- primește mai multe entități de tip diferit și le unește într-o entitate ce le conține pe toate, având de asemenea toate atributele lor
- poate fi privit ca o structură ce conține alte structuri (o entitate ce conține mai multe subentități)

-pentru a putea crea o entitate, este nevoie să primească o entitate din fiecare fel care o alcătuiește.

2.COMPOSITE ENTITY SPLITTER

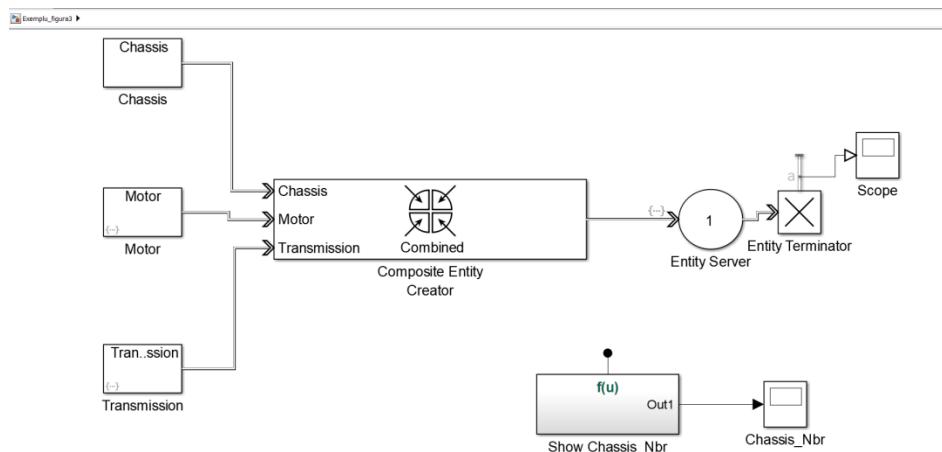


- „desface” o entitate compusă – subentitățile nu mai sunt unite, ci fiecare este acum independentă

EXERCIȚII EXPLICATE ÎN PLATFORMA DE LABORATOR

Exemplul 1 (pagina 4 jos)

Am realizat modelul astfel:



Blocul Entity Generator generează entități astfel:

- Chassis: 1h

- Motor: 0.3h

- Transmission: 0.5h

Fiecare tip de entitate (Chassis, Motor, Transmission) au o atribută cu Numeleentității_Nbr:

Attribute Name	Attribute Initial Value
1 Chassis_Nbr	1

Attribute Name	Attribute Initial Value
1 Motor_Nbr	1

Attribute Name	Attribute Initial Value
1 Transmission_Nbr	1

De asemenea, entitățile Motor și Transmission au un script care se apelează la generarea unei entități folosit pentru a modifica atributul NumeleEntității_Nbr în funcție de a căta entitate a fost generată:

```

Block Parameters: Motor
Event actions
Generate action:
Called after entity is generated.
To access attribute use: entity.Motor_Nbr
1 persistent Motor_count
2 if (isempty(Motor_count))
3 Motor_count = 0;
4 end
5 Motor_count = Motor_count + 1;
6 entity.Motor_Nbr = Motor_count;

Block Parameters: Transmission
Event actions
Generate action:
Called after entity is generated.
To access attribute use: entity.Transmission_Nbr
1 persistent Transmission_count
2 if (isempty(Transmission_count))
3 Transmission_count = 0;
4 end
5 Transmission_count = Transmission_count + 1;
6 entity.Transmission_Nbr = Transmission_count;

```

Apoi, cele 3 entități sunt combinate într-un Composite Entity Creator, ce are rolul de a forma o entitate ce cuprinde toate cele 3 entități și atributele lor. După aceea, entitatea nouă este servită, iar atunci când intră în Entity Server se apelează următorul script:

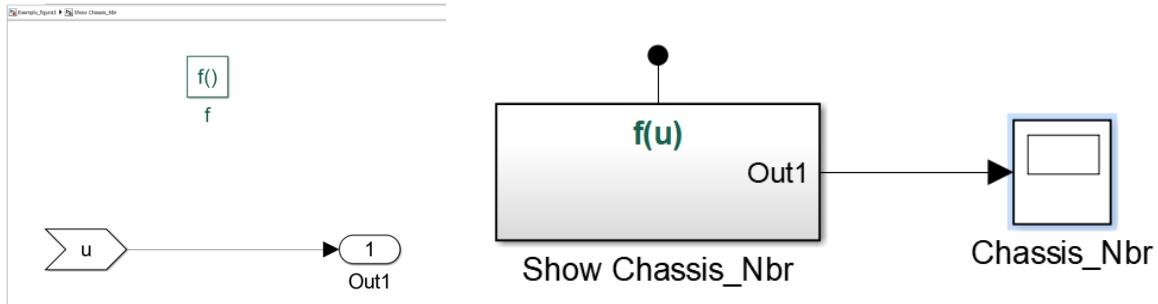
```

entity.Chassis.Chassis_Nbr = entity.Motor.Motor_Nbr + 10 * entity.Transmission.Transmission_Nbr;
2 f(entity.Chassis.Chassis_Nbr);

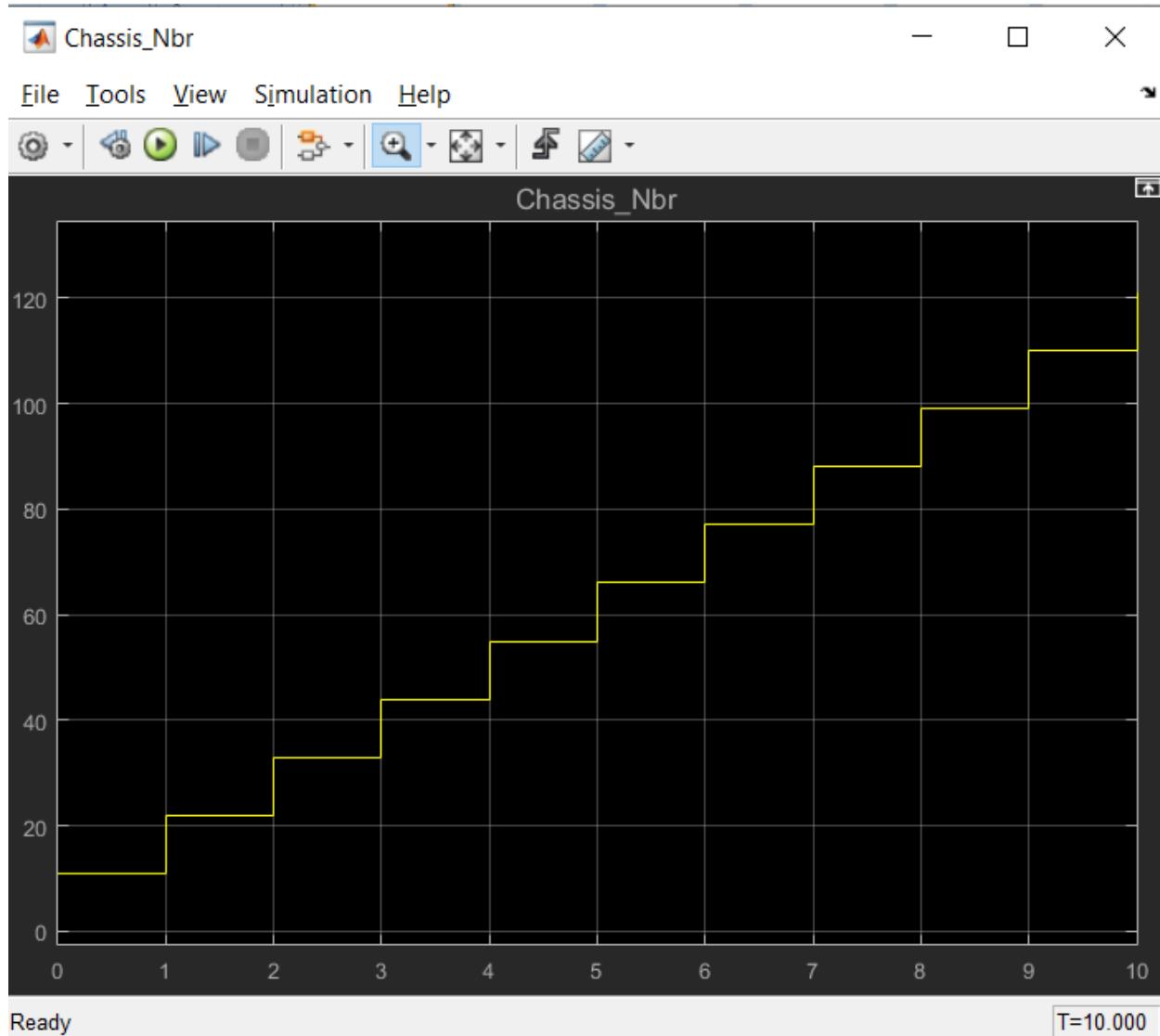
```

Entity	Chassis	Motor	Transmission
entity	Chassis_Nbr	Motor_Nbr	Transmission_Nbr
entitySys	id		
	priority		

Acest script modifică atributa entității Chassis în relația de după egal, iar apoi se apelează funcția `f(entity.Chassis.Chassis_Nbr)`, prin care doar trece input-ul pentru a fi apoi afișat într-un scope.



Practic, prin scope se afișează atributa Chassis_Nbr a entității curente în Entity Server, care evoluează astfel în timp:



Cu zoom in se poate observa că aceasta crește din 11 în 11. Acest lucru se datorează formulei cu care este calculată:

$$\text{Chassis_Nbr} = \text{Motor_Nbr} + 10 * \text{Transmission_Nbr};$$

o să explic primele două cazuri afişate pe scope, 11 și 22, iar celălalte sunt pe exact același raționament:

-s-a creat o entitate ce conține cele 3 entități Chassis, Motor și Transmission:

`Chassis.Chassis_Nbr = 1;`

`Motor.Motor_Nbr = 1;`

`Transmission.Transmission_Nbr = 1;`

-s-a intrat în entity server, unde se aplică formula:

$$\text{Chassis.Chassis_Nbr} = 1 (\text{Motor.Motor_Nbr}) + 10 * 1 (\text{Transmission.Transmission_Nbr}) = 1 + 10 = 11.$$

apoi, se face $f(\text{Chassis.Chassis_Nbr}$, adică se afișează pe scope 11).

-s-a creat o entitate ce conține cele 3 entități Chassis, Motor și Transmission:

$\text{Chassis.Chassis_Nbr} = 1$;

$\text{Motor.Motor_Nbr} = 1 + 1 = 2$;

$\text{Transmission.Transmission_Nbr} = 1 + 1 = 2$;

-s-a intrat în entity server, unde se aplică formula:

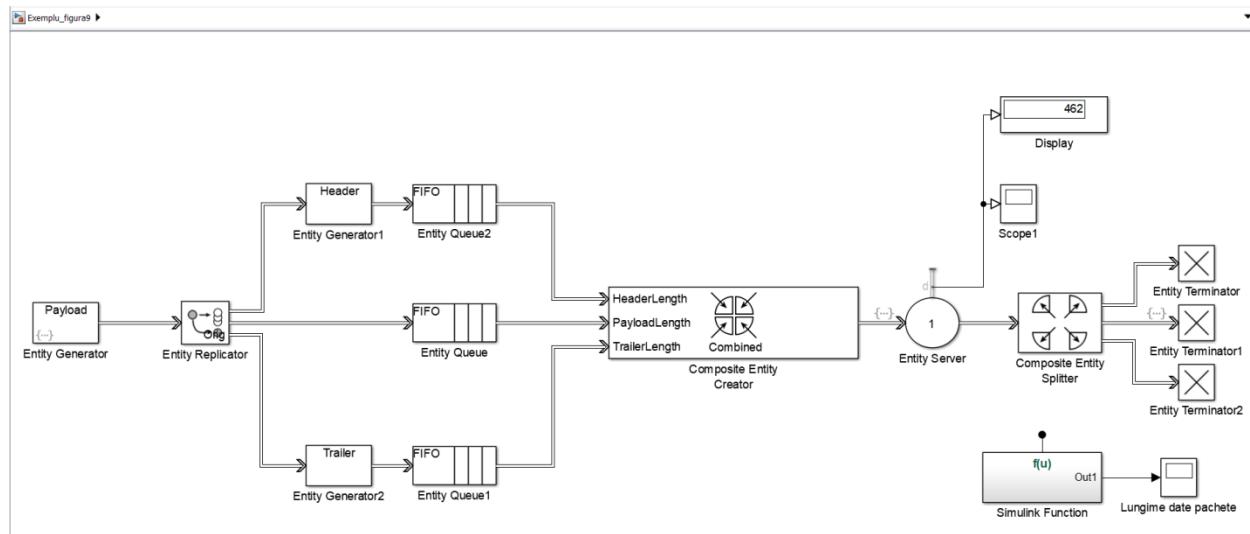
$$\text{Chassis.Chassis_Nbr} = 2 (\text{Motor.Motor_Nbr}) + 10 * 2 (\text{Transmission.Transmission_Nbr}) = 2 + 20 = 22.$$

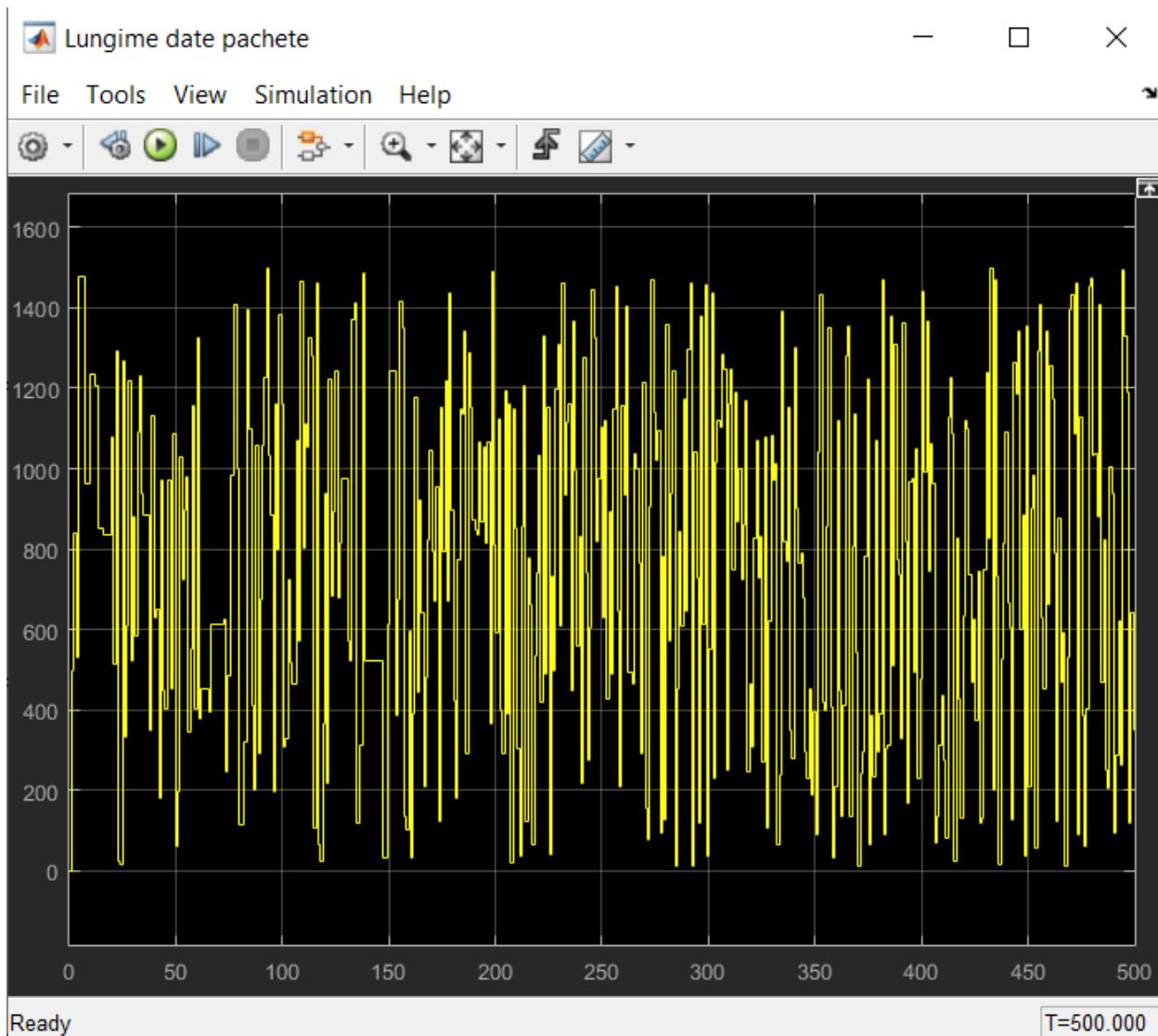
apoi, se face $f(\text{Chassis.Chassis_Nbr}$, adică se afișează pe scope 22).

De aceea numerele asociate ansamblurilor vor fi 11, 22, 33, ..., etc.

Exemplul 2 (pagina 8)

Am făcut și exemplul 2 conform cerințelor din platforma de laborator, iar output-ul afișat pe scope-ul obținut din Simulink Function corespunde:





2. Probleme propuse

PROBLEMA 1

- Să se modeleze o linie de asamblare a automobilelor ce constau din următoarele subansamble: şasiu, motor și transmisie.

sunt 3 Entity Generator, către unul pentru fiecare subansamble: unul cu Entity name type řasieu, unul cu numele Entity name type Motor și unul cu Entity name type Transmisie.

- Fiecare subansamblu este o entitate generată de un bloc Entity Generator la intervale de timp tgen. După generare, subansamblele sunt testate în stații un interval de timp ttest.

deci, după ce fiecare entitate va fi generată de propriul său Entity Generator, fiecare va intra în câte un Entity Server propriu care are durata ttest .

- Asamblarea subansamblelor durează tans = 0.3 h. Unitatea de timp a simulării este ora. asamblarea subansamblelor se produce într-un Entity Server după ce toate 3 entitățile au fost unite cu ajutorul lui Composite Entity Creator.

Subansamblu	t_{gen} [h]	t_{test} [h]
Motor	0.8	0.6
Transmisie	0.6	0.4
Şasiu	0.4	0.4

- Se vor înregistra numărul de subansamble generate, şasiu, motor, transmisie și numărul de automobile ansamblate.

numărul de subansamble generate este, de fapt, Number of entities departed pentru fiecare Entity Generator.

numărul de automobile asamblate este, de fapt, atributul Auto_number obținut în ultimul Entity Server.

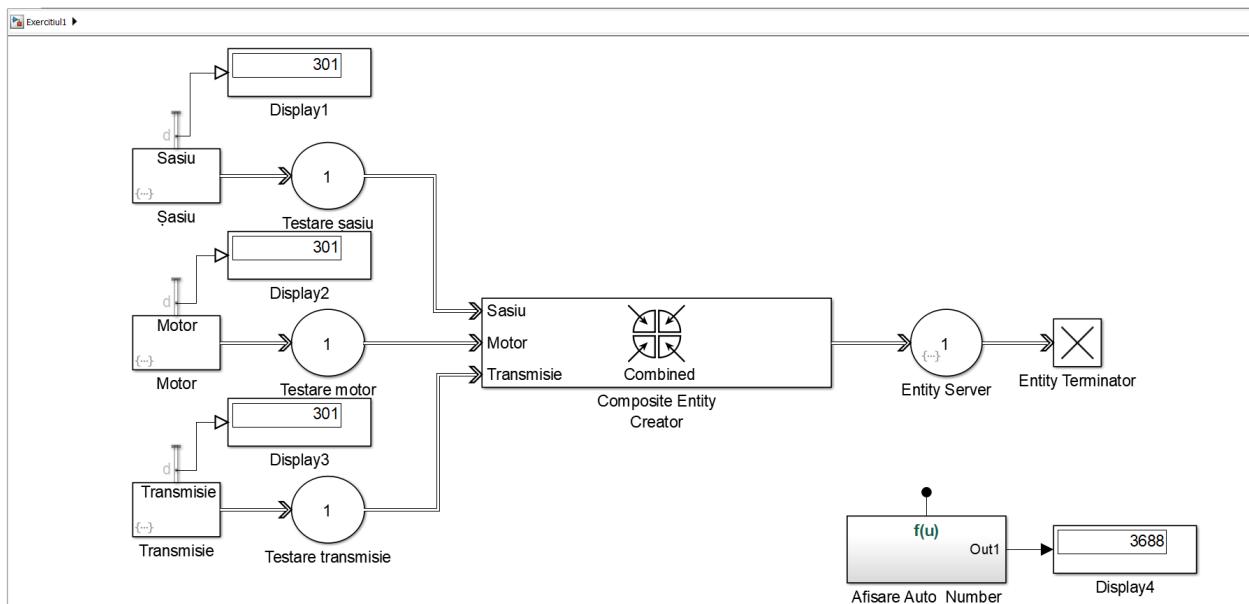
- Să se simuleze modelul pe durata de 240 ore.
- Numărul asociat şasiului se va calcula cu formula

Auto_Number = Motor_Nbr + 10 * Transmission_Nbr + 100 * Chassis_Nbr;

Atributele Motor_Nbr, Transmission_Nbr, Chassis_Nbr vor fi asociate entităților Motor, Transmission, Chassis. Atribuit atributului Auto_Nbr va fi asociat entității Chassis.

REZOLVARE:

Modelul arată astfel:



Şasiu, Motor și Transmisie au, fiecare, atributul Numeentitate_Nbr, iar, în plus, Şasiu are atributul Auto_number, adică numărul de mașini asamblate.

De asemenea, atributele Sasiu_Nbr, Motor_Nbr, Transmisie_Nbr sunt calculate astfel:

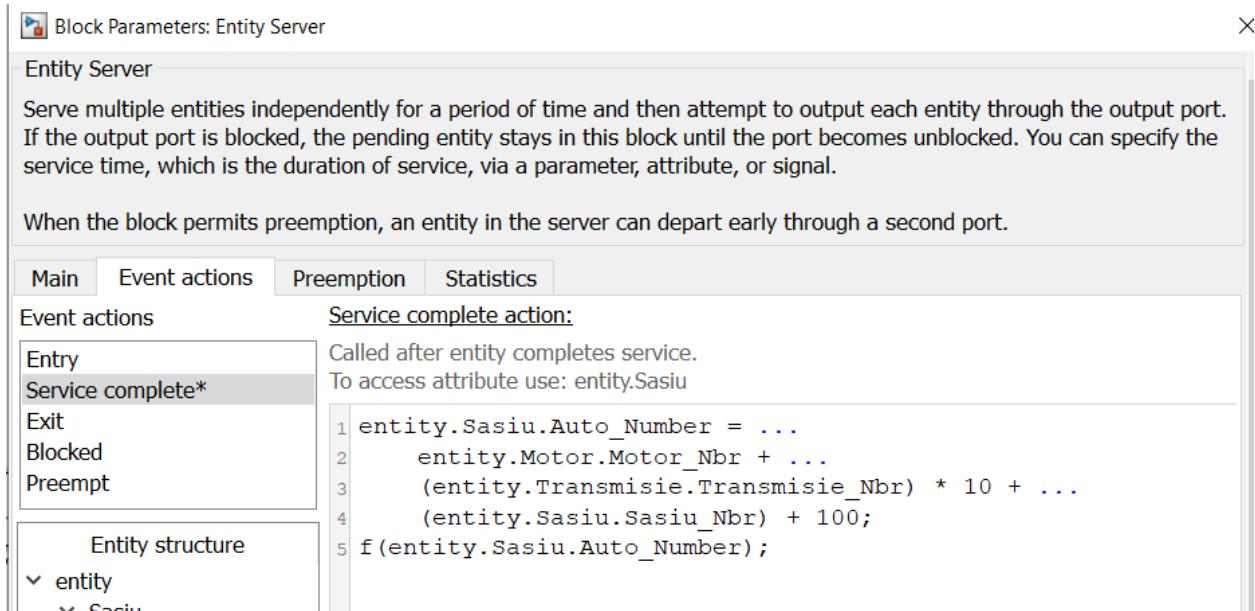
```

Generate action:
Called after entity is generated.
To access attribute use: entity.Transmisie_Nbr

1 persistent Transmisie_count
2 if (isempty(Transmisie_count))
3     Transmisie_count = 0;
4 end
5 Transmisie_count = Transmisie_count + 1;
6 entity.Transmisie_Nbr = Transmisie_count;

```

După ce entitățile au fost testate în Entity Server, ele vor fi combinate de Composite Entity Creator, iar apoi duse într-un alt Entity Server, ce are ca rol să asambleze subansamblele. Atunci când entitatea obținută prin unirea subentităților Motor, Transmisie, Şasiu intră în acest Entity Server, se apelează următorul script:



Prin acest script se calculează Auto_Number (numărul asociat şasiului) conform formulei date în cerință:

Auto_Number = Motor_Nbr + 10 * Transmission_Nbr + 100 * Chassis_Nbr;

iar apoi se apelează funcția f, care doar trimite semnalul pe care îl primește mai departe pentru a putea fi afișat. În urma simulării, am obținut valorile afișate pe display-uri.

PROBLEMA 2

- La o companie telefonică există un serviciu de informații cu cinci operatori. Orice operator poate prelua orice cerere

este un Entity Server cu capacitate 5

- Cererile apar la intervale cu distribuție exponențială cu valoarea medie 0.2 minute. *pentru Entity Generator se folosește MATLAB Action, dt = exprnd(0.2);*

- Durata unei con vorbiri este 1 ± 0.2 minute cu distribuție uniformă.

durata unei con vorbiri este timpul de deservire din Entity Server, și se va seta astfel:

*valori din intervalul (1 ± 0.2) , adică $(1 - 0.2, 1 + 0.2) = (0.8, 1.2)$ adică $0.8 + 0.4 * rand$*

- Se va utiliza un multiserver cu capacitatea cinci.
- Să se modeleze sistemul pe o durată de 8 ore.

*unitatea de timp aleasă este minutul, deci sistemul se va simula pe $8 * 60 = 480$ minute.*

- Se vor afișa timpul mediu de așteptare în coadă

timpul mediu de așteptare în coadă este, de fapt, Average wait din Entity Queue.

- timpul mediu de deservire

timpul mediu de deservire este Average wait din Entity Server

- numărul de operatori ocupați

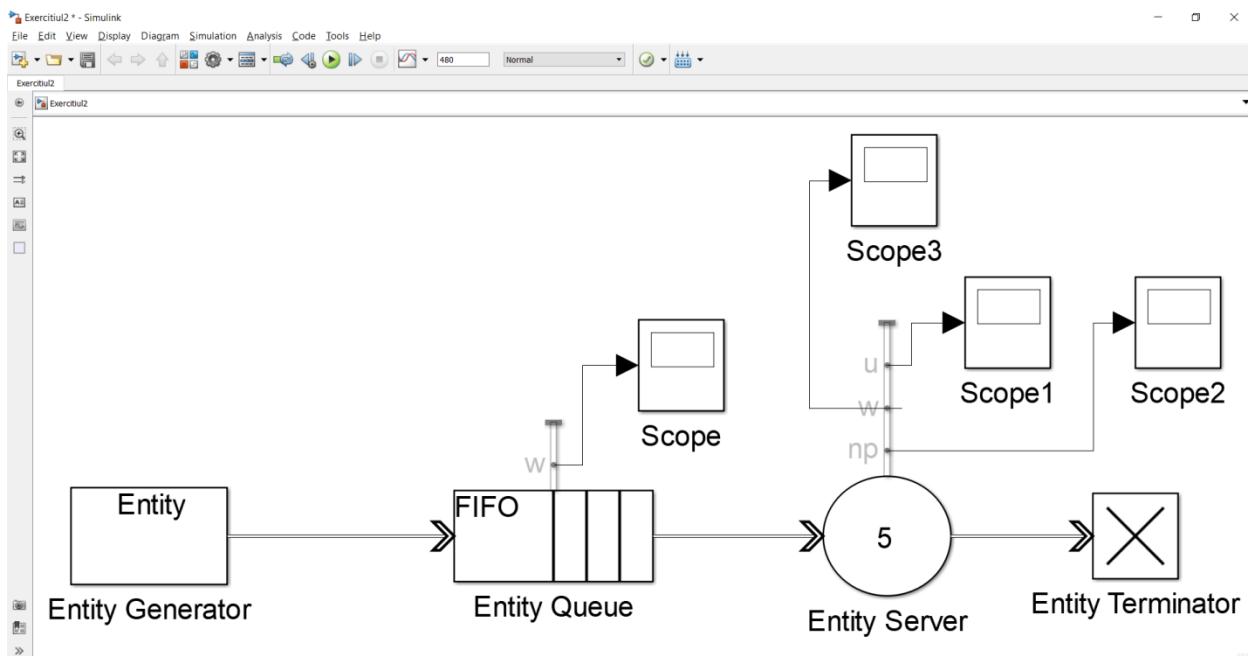
numărul de operatori ocupați este dat de Number of entities in block din Entity Server

- gradul mediu de utilizare a serverului.

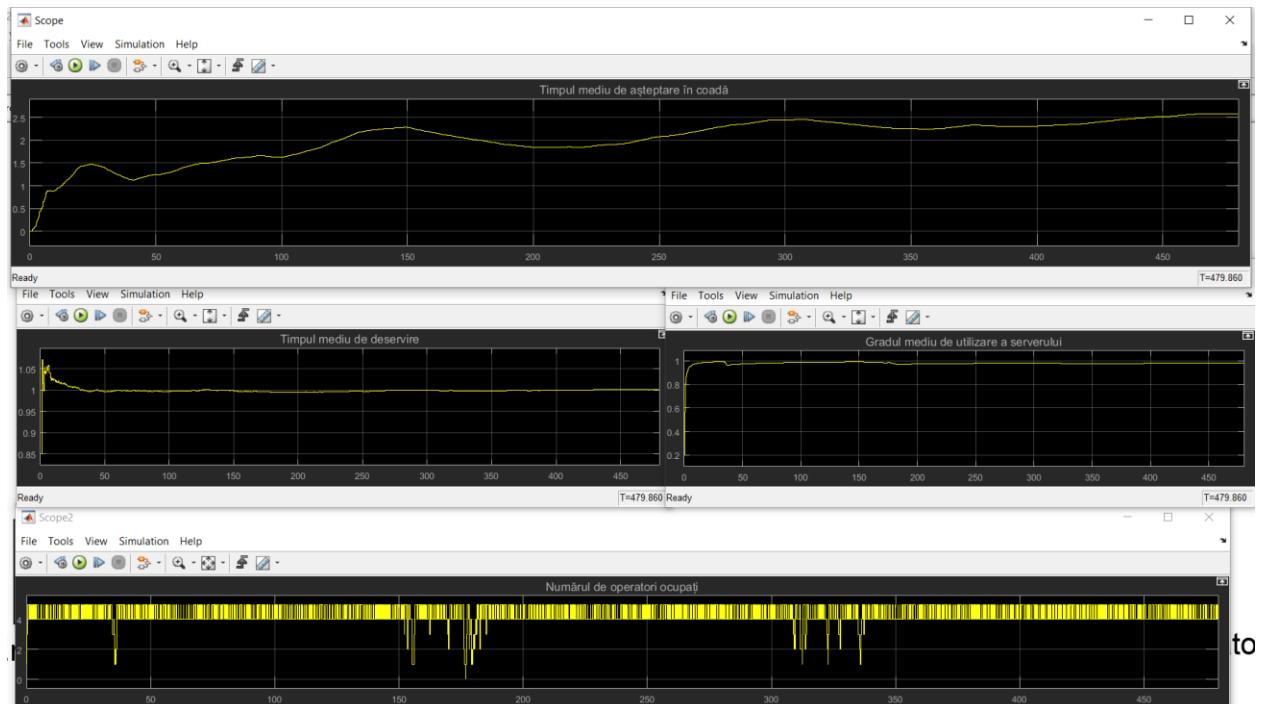
gradul mediu de utilizare a serverului este Utilization din Entity Server

REZOLVARE:

Modelul arată astfel:



Iar, în urma rulării sale, am obținut:



PROBLEMA 3

se dau:

- Loturile sosesc la intervale de 18 ± 6 minute cu distribuție uniformă.
*din intervalul $(18-6; 18+6) = (12, 24)$. (deci $12 + 12 * \text{rand}$)*
- Prelucrarea unui lot durează 32 ± 4 minute cu distribuție uniformă.
*din intervalul $(32-4; 32+4) = (28, 36)$. (deci $28 + 8 * \text{rand}$)*
- Loturile sunt prelucrate la o stație cu capacitatea 2.

Entity Server are capacitatea 2

- Să se simuleze modelul pe o durată de 480 minute.
o să aleg unitatea de timp ca fiind minutul.

se cer:

- numărul de loturi generate

numărul de loturi generate este, de fapt, Number of entities departed din Entity Generator

- timpul mediu de așteptare în coadă

timpul mediu de așteptare în coadă este, de fapt, Average wait din Entity Queue

- timpul mediu de deservire

timpul mediu de deservire este, de fapt, Average wait din Entity Server

- gradul de utilizare a stației

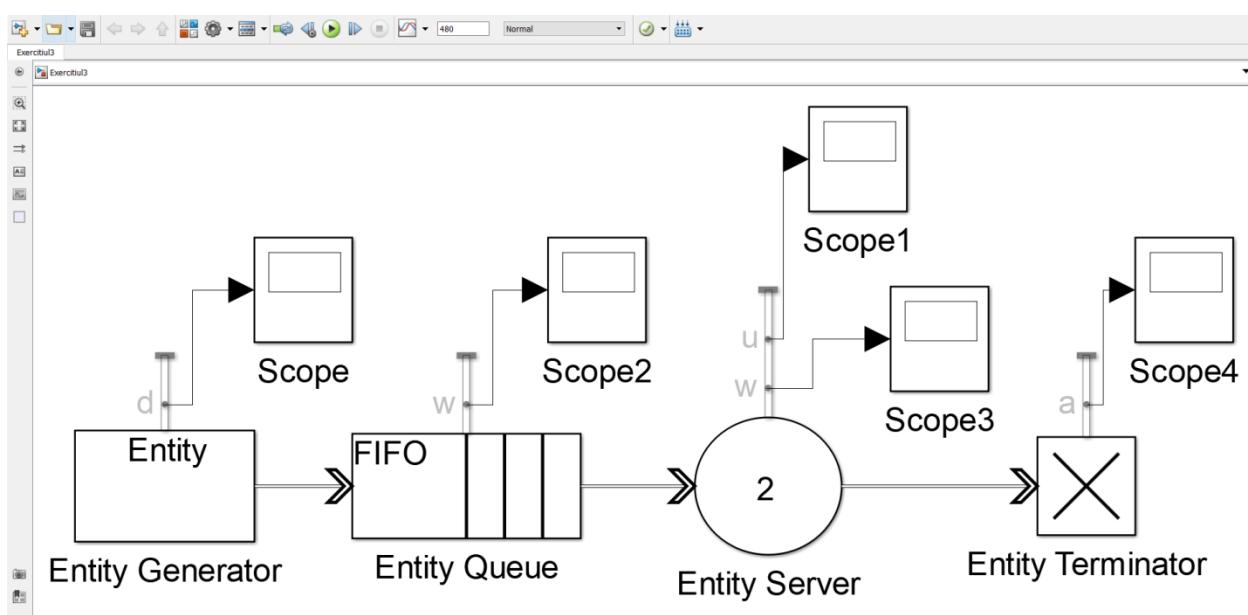
gradul de utilizare a stației este, de fapt, Utilization din Entity Server

- numărul de loturi prelucrate

numărul de loturi prelucrate este Number of entities arrived din Entity Terminator

rezolvare:

Modelul obținut arată astfel:



Valorile obținute în urma rulării sale pot fi obținute apăsând butonul Run. Am pus Title la fiecare Scope cu ceea ce este afișat pe el.

PROBLEMA 4

se dau:

- La o magazie pentru aprovizionarea cu materiale vin două tipuri de muncitori.

Sunt două Entity Generator, câte unul pentru fiecare tip de muncitor.

- Cei de primul tip vin la intervale de 420 ± 360 s cu distribuție uniformă, iar timpul de deservire este 300 ± 90 s cu distribuție uniformă.

*vin în intervalul $(420-360; 420+360) = (60, 780)$. (deci $60 + 720 * rand$)*

*timpul de deservire $(300-90, 300+90) = (210, 390)$ (deci $210 + 180 * rand$)*

- Muncitorii de tipul doi la intervale de 360 ± 240 s cu distribuție uniformă, iar timpul de deservire este 100 ± 30 s cu distribuție uniformă.

*vin în intervalul $(360-240; 360+240) = (120, 600)$. (deci $120 + 480 * rand$)*

*timpul de deservire $(100-30, 100+30) = (70, 130)$ (deci $70 + 130 * rand$)*

- Muncitorii de ambele tipuri așteaptă în aceeași coadă să fie deserviți.

ambele Entity Generator intră în același Entity Queue; i-am dat din oficiu capacitatea 300

- durată de 28800s

o să aleg unitatea de timp ca fiind secunda.

se cer:

- numărul de muncitori de fiecare tip care vin pentru aprovizionare

numărul de muncitori de fiecare tip care vin pentru aprovizionare reprezintă, de fapt, Number of entities departed pentru fiecare Entity Generator.

- lungimea cozii

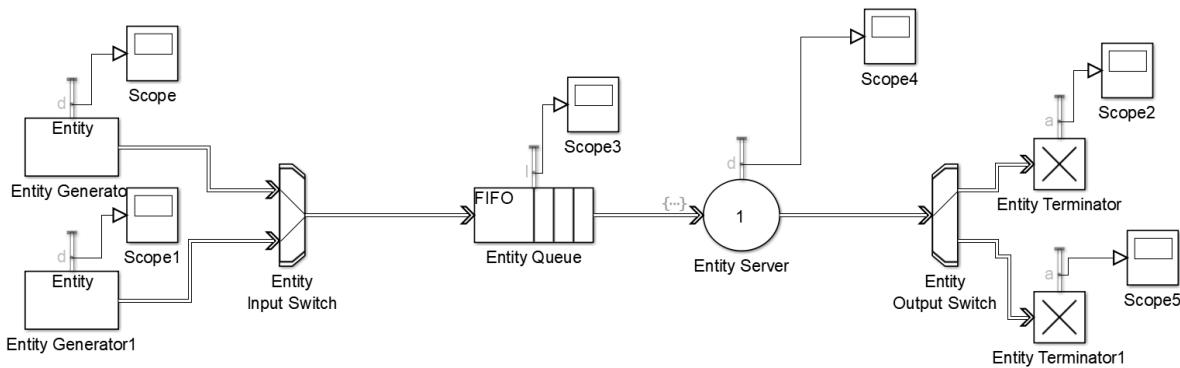
lungimea cozii este Average Queue Length din Entity Queue

- numărul de muncitori deserviți de fiecare tip

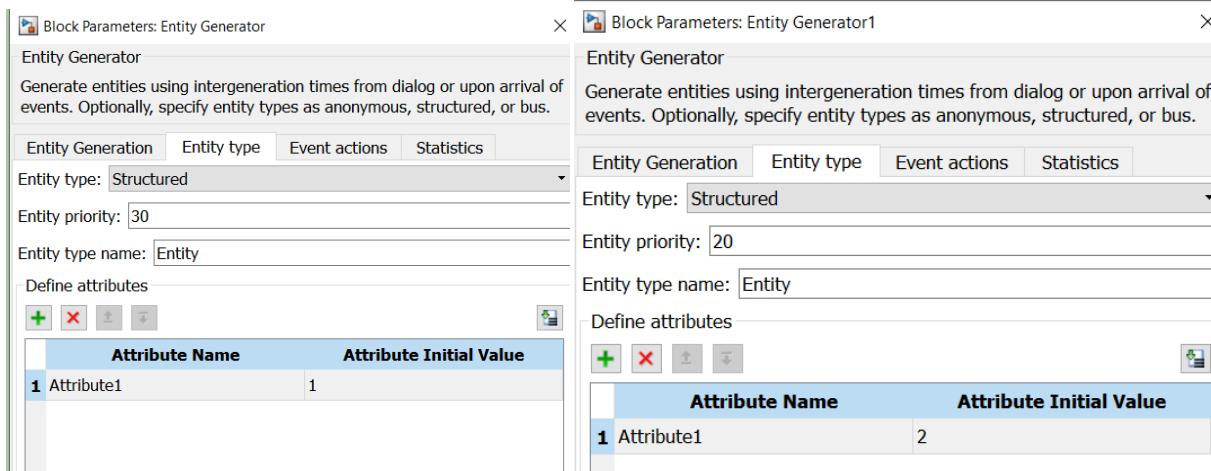
numărul de muncitori deserviți de fiecare tip este, de fapt, Number of entities departed din Entity Server – dar acest lucru e pentru toate tipurile

rezolvare:

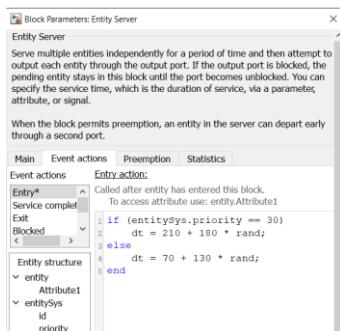
Exercițiul realizat de mine arată astfel:



Sunt două Entity Generator, câte unul pentru fiecare tip de muncitor. Am adăugat, în plus, pentru fiecare Entity Generator ceva distinct – cel de sus are prioritatea 30 și o atribută Attribute1 cu valoarea 1, iar cel de-al doilea, Entity Generator1, are prioritatea 20 și atributa Attribute1 are valoarea 2.



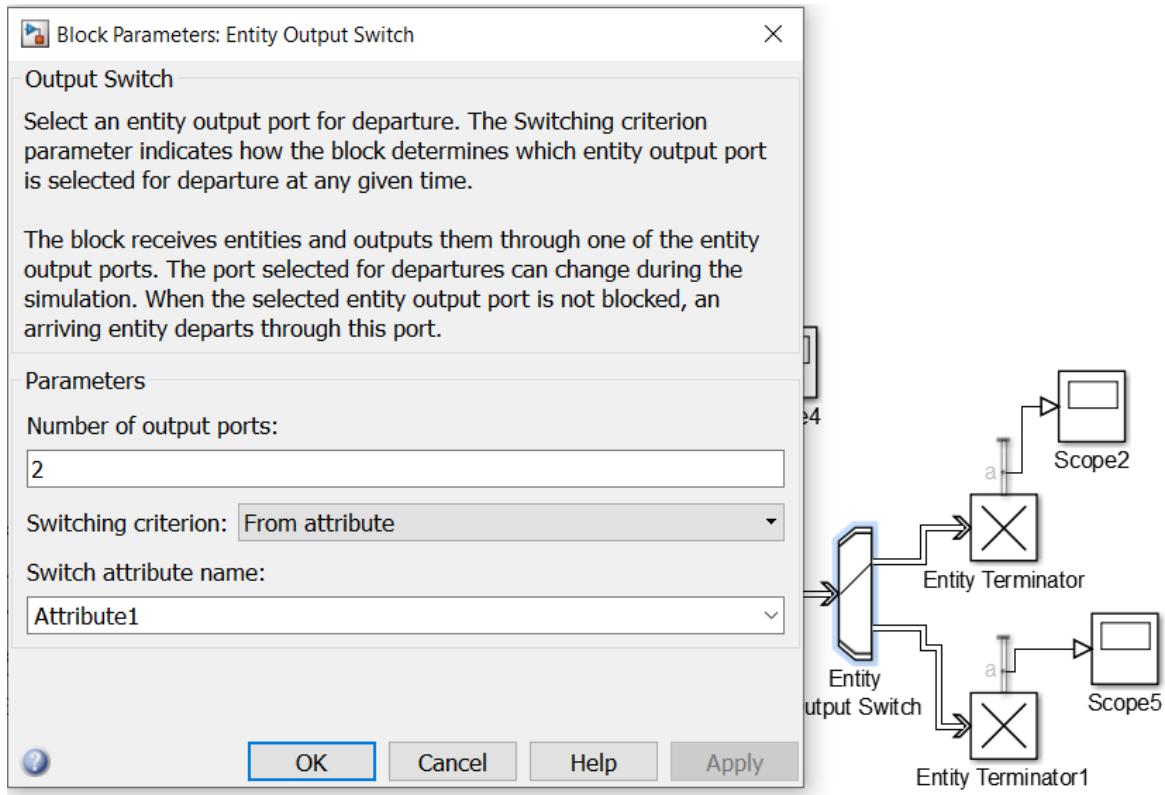
Motivul pentru care au prioritatea entitySys diferită capătă sens în interiorul lui Entity Server, unde am scris o funcție pentru sample-ul în care o entitate intră în acest bloc:



Dacă prioritatea este 30, atunci timpul de deservire este 300 ± 90 s cu distribuție uniformă. Dacă entitatea nu are prioritatea 30 (implicit 20), atunci are timpul de deservire 100 ± 30 s cu distribuție uniformă. Aici se verifică prioritatea principală a entității.

De asemenea, am folosit Entity Input Switch și Entity Output Switch. Entity Input Switch poate primi mai multe semnale și duce mai departe decât unul singur, în interiorul căruia se află toate semnalele pe care le-a primit. Entity Output Switch primește un singur semnal care e format din mai multe semnale unite de un Entity Input Switch și le

“desparte”. În cazul meu, modul în care le desparte este bazat pe prioritățile atributelor secundare pe care le au aceste entități = Attribute1.



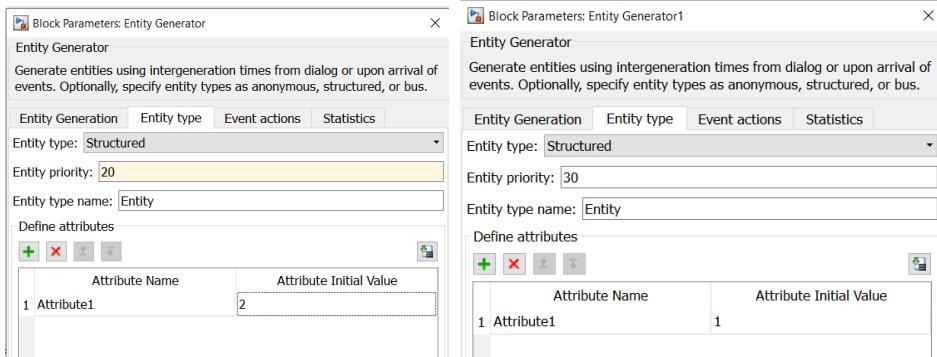
În Entity Terminator am afișat numărul de muncitori deserviți de fiecare tip.

Rezultatul simulării poate fi văzut dacă se rulează exercițiul pus în arhivă.

PROBLEMA 5

Felul în care am făcut eu exercițiul 5 nu ține cont de care dintre muncitori are o prioritate mai mare, ci dacă aceștia au o anumită valoare ca și prioritate. Ca să fac oarecum și acest exercițiu, am modificat:

-Muncitorii 1 au acum prioritatea principală 20, iar la Attribute1 au 2; Muncitorii 2 au acum prioritatea principală 30, iar la Attribute2 au 1 (am inversat).



-Entity Queue este identic, dar acum nu se mai verifică dacă prioritatea este 30, ci 20, pentru ca Muncitorii 1 au acum prioritatea 20, iar formula este pentru ei. Cea de la else este pentru Muncitorii 2.

```

1 if (entitySys.priority == 20)
2     dt = 210 + 180 * rand;
3 else
4     dt = 70 + 130 * rand;
5 end

```

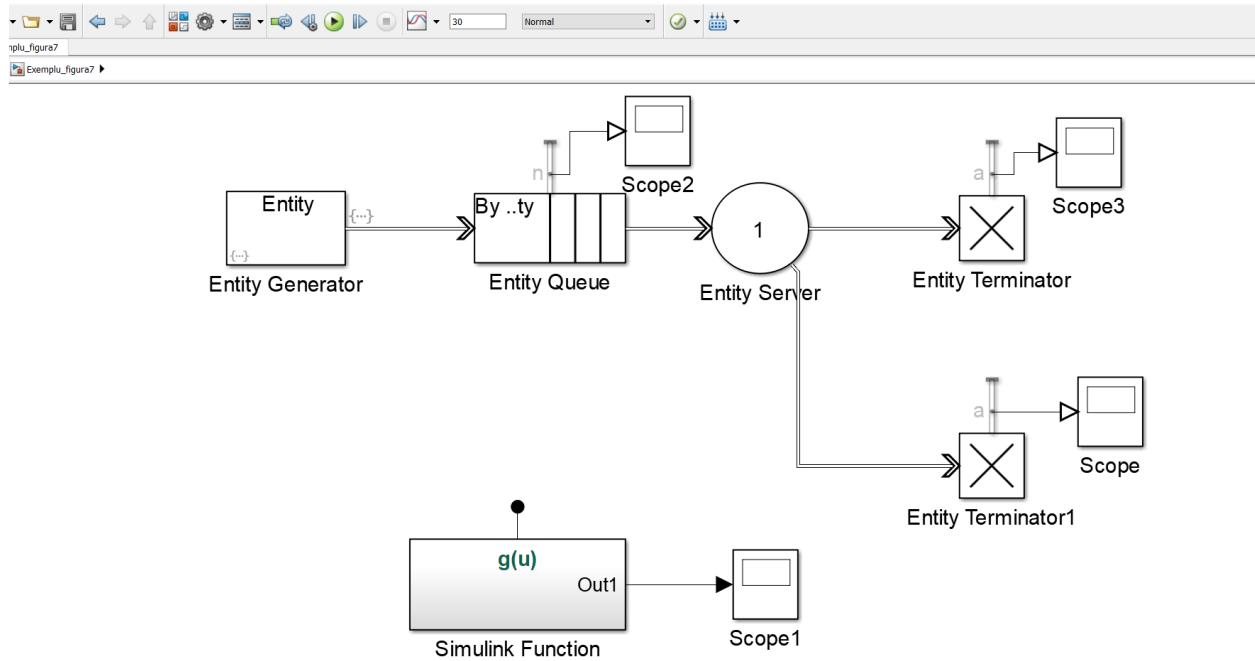
Lucrarea 7

EXERCITII DIN PLATFORMA DE LABORATOR

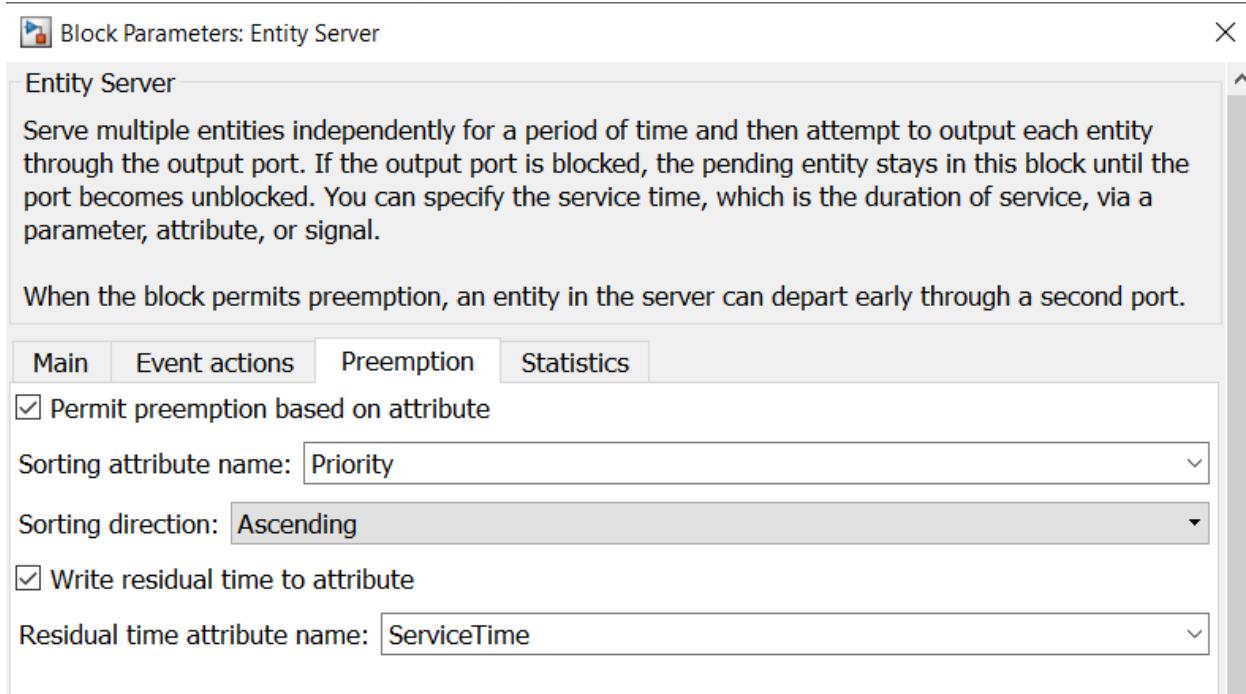
3.1. Entitatea preluată nu reintră în server

Coadă Entity Queue este cu priorități. De ce?

Am atașat exercițiul în arhivă. L-am făcut astfel:



Coadă Entity Queue este cu priorități deoarece această setare a ei este necesară pentru a se putea folosi un Entity Server cu Preemption în funcție de prioritate.



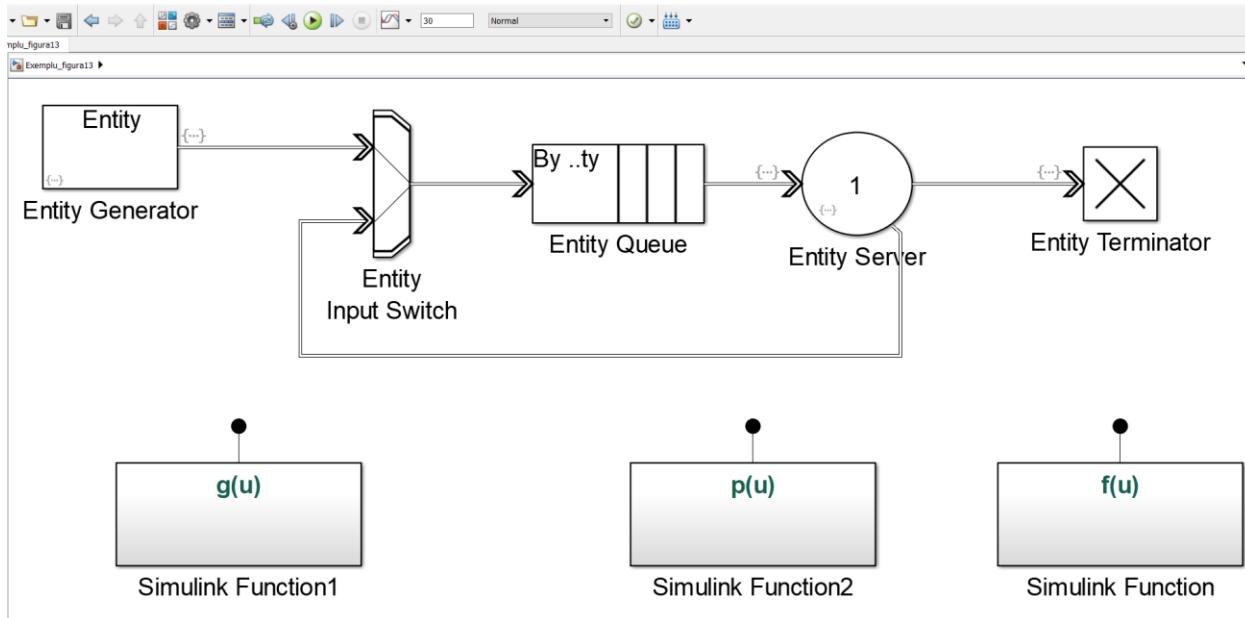
Practic, în acest model, atributul Priority poate lua două valori: 1 sau 2. În funcție de priority, Entity Server alege pe care entitate o prelucrează. Dacă, spre exemplu, la momentul x se prelucrează o entitate cu priority 2, dar în Entity Queue următorul eveniment ce va ieși are prioritatea 1, acesta intră automat în Entity Server, iar prelucrarea entității cu priority 2 este anulată. Dacă Entity Queue-ul ar avea Queue Type-ul altceva în afară de Priority, atunci dacă în interiorul său ar intra o entitate cu priority 1, ea ar trebui să își aștepte rândul în coadă până când poate intra în Entity Server. Acest lucru ar limita-o să facă ceea ce am explicat eu mai sus, când sunt situațiile cu priorități diferite, decât pentru cazul când entitatea care a intrat înaintea ei are prioritate mai mare decât el.

Entitățile sunt preluate la momentele 2, 5, 10, 15, 21 și 29s. De ce?

Entitățile sunt preluate la momentele specificate mai sus pentru că atunci a fost situația explicată mai sus: dacă, spre exemplu, la momentul x se prelucrează o entitate cu priority 2, dar în Entity Queue următorul eveniment ce va ieși are prioritatea 1, acesta intră automat în Entity Server, iar prelucrarea entității cu priority 2 este anulată.

3.2. Entitatea preluată reintră în server cu timpul de deservire rămas

Am atașat exercițiul în arhivă. L-am făcut astfel:



Coada Entity Queue este cu priorități. De ce?

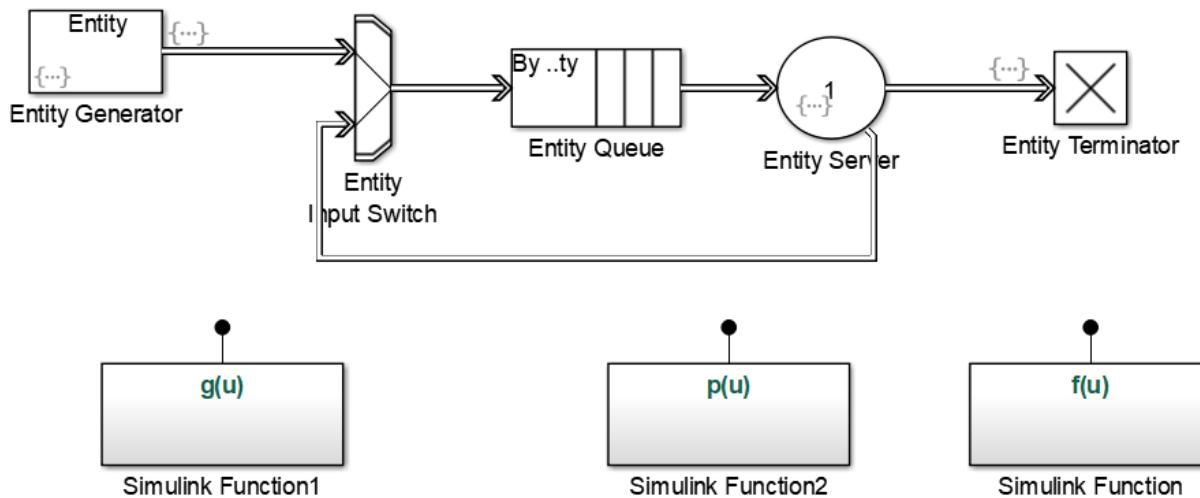
Coada Entity Queue este cu priorități din același motiv pentru care era astfel și la 3.1.

Conform graficului Timpul rezidual al entităților preluate, sunt preluate entități la momentele 2, 5, 6, 10, 15, 29. De ce?

Este același motiv pentru care erau preluate entități la 3.1.

3.3. Entitatea preluată reintră în server cu timpul de deservire inițial

Am atașat exercițiul în arhivă. L-am făcut astfel:



4. Probleme propuse

PROBLEMA 1

Fie un sistem de operare de timp real care trebuie să planifice la execuție două tipuri de taskuri ce vor fi executate la intervale egale. Taskurile de primul tip trebuie executate la intervale de 2s, cele de tipul doi la intervale de 5s. Taskurile vor fi generate de două blocuri Entity Generation. Durata de deservire a taskurilor de primul tip este de 1s, a celor de tipul doi este 2.5s. Se va simula modelul în următoarele cazuri:

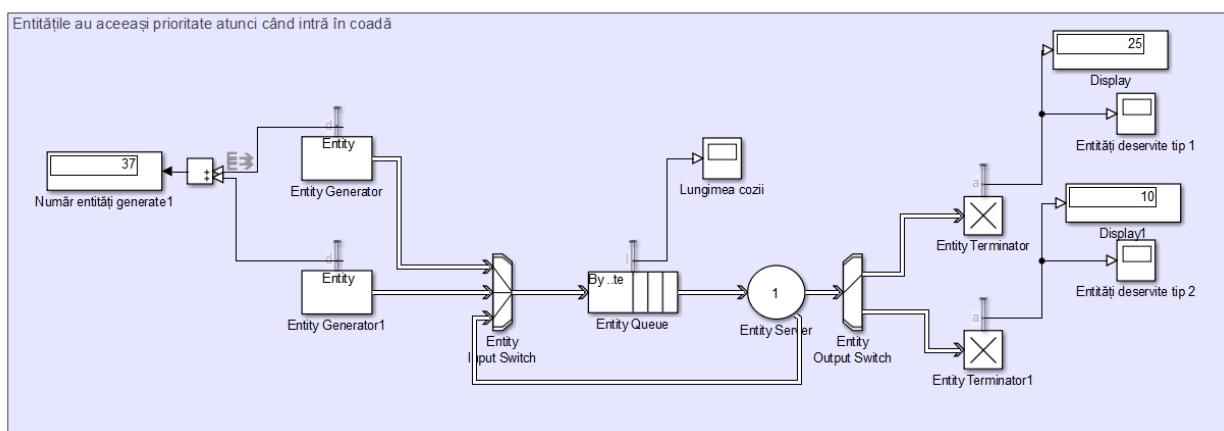
- Ambele tipuri de taskuri au aceeași prioritate.
- Taskurile de primul tip au prioritate mai mare.
- Taskurile de tipul doi au prioritate mai mare.

Modelul se va simula pe o durată de 50s. Unitatea de timp este secunda. Se va considera cazul când entitatea preluată reintră în server cu timpul de deservire rezidual.

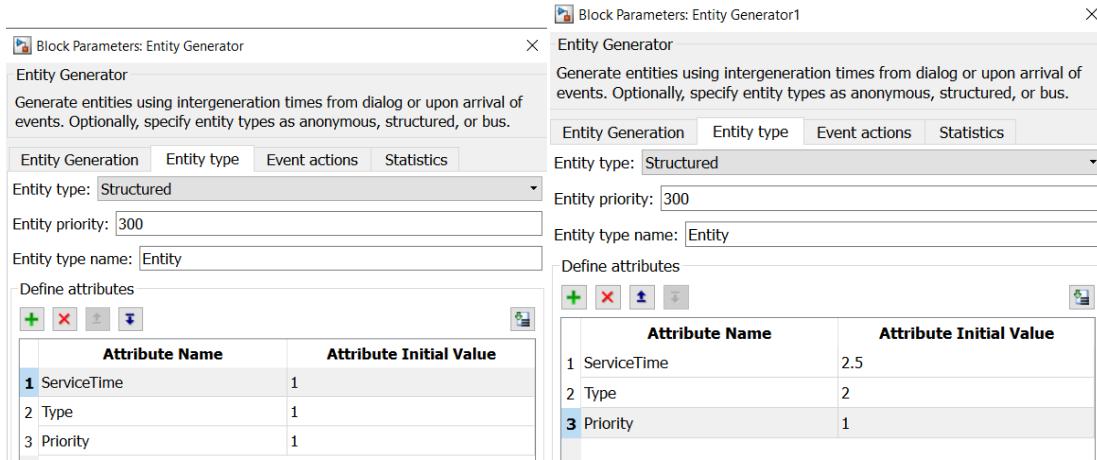
Entitățile vor avea următoarele attribute: Priority, ServiceTime și Type ce va conține tipul taskurilor. Se vor înregistra: numărul de entități generate, lungimea cozii și numărul de entități deservite pentru fiecare caz. Se vor explica rezultatele.

REZOLVARE:

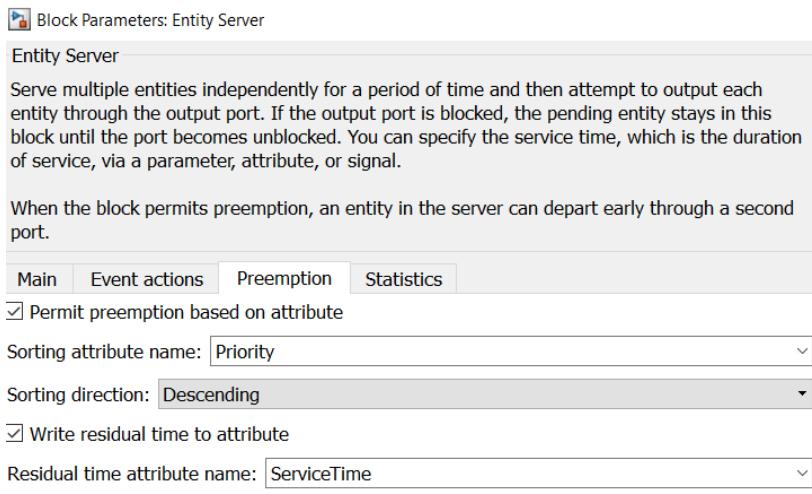
Modelul arată astfel:



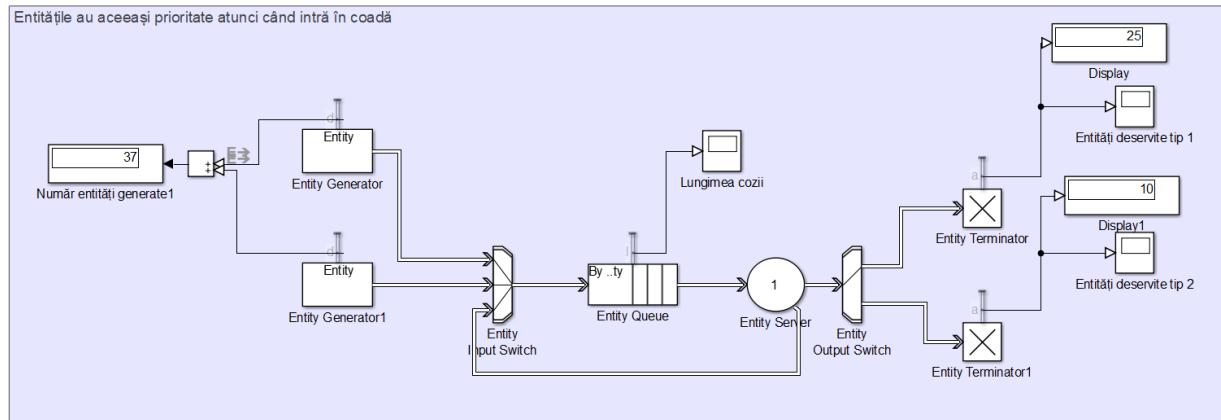
Entity Generator 1 și Entity Generator 2 generează același tip de entități (Entity), doar că la Entity Generator avem Type 1 ca atribută, iar la Entity Generator1 avem Type 2 ca atribută.



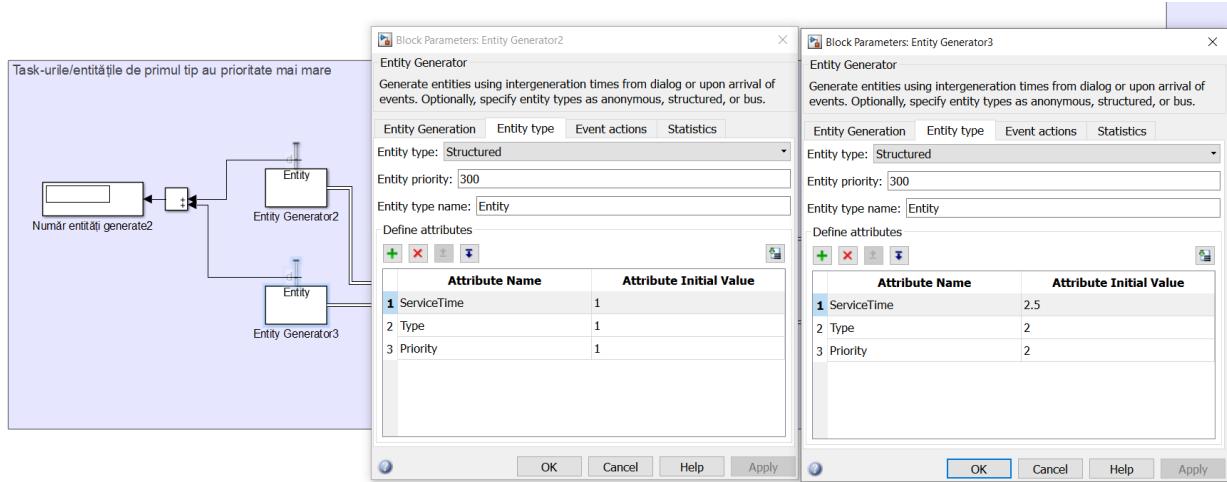
Datorită faptului că **ambele entități au aceeași prioritate Priority**, din Entity Server, care are în Preemption Sorting attribute name Priority, nicio entitate nu va fi preluată.



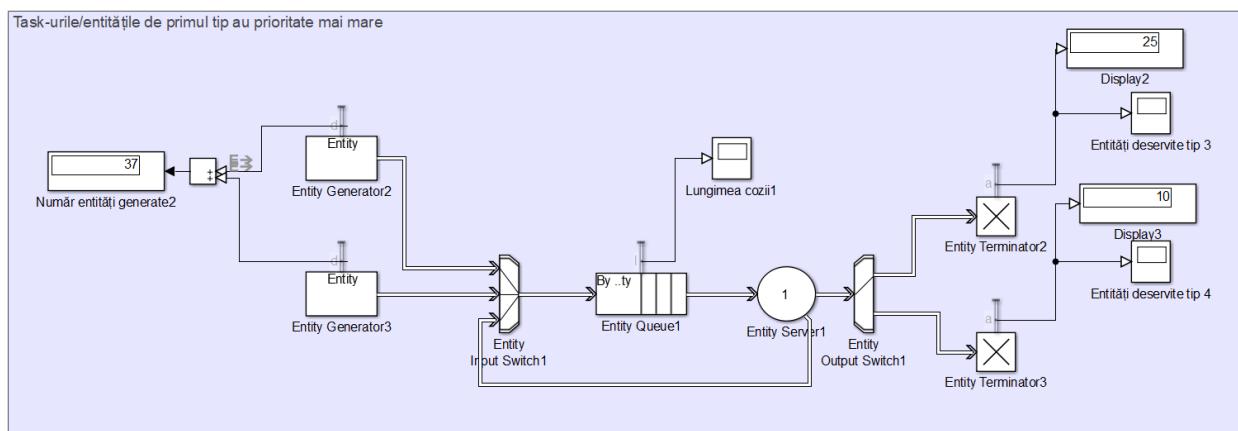
Rezultatele în urma rulării modelului sunt aceleași ca și cum Entity Server nu ar fi avut Preemption-ul configurat:



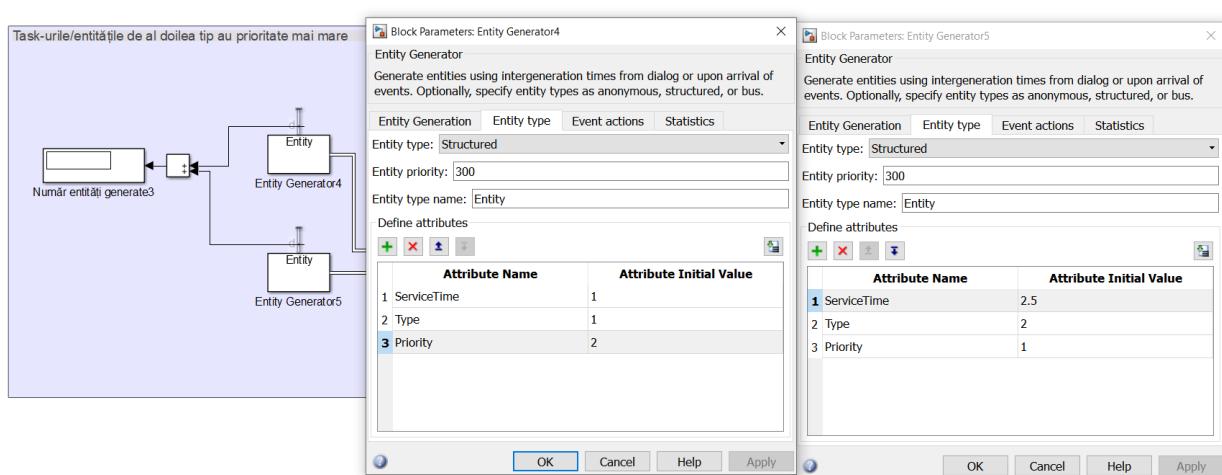
Pentru cazul în care entitățile de primul tip au prioritate mai mare:



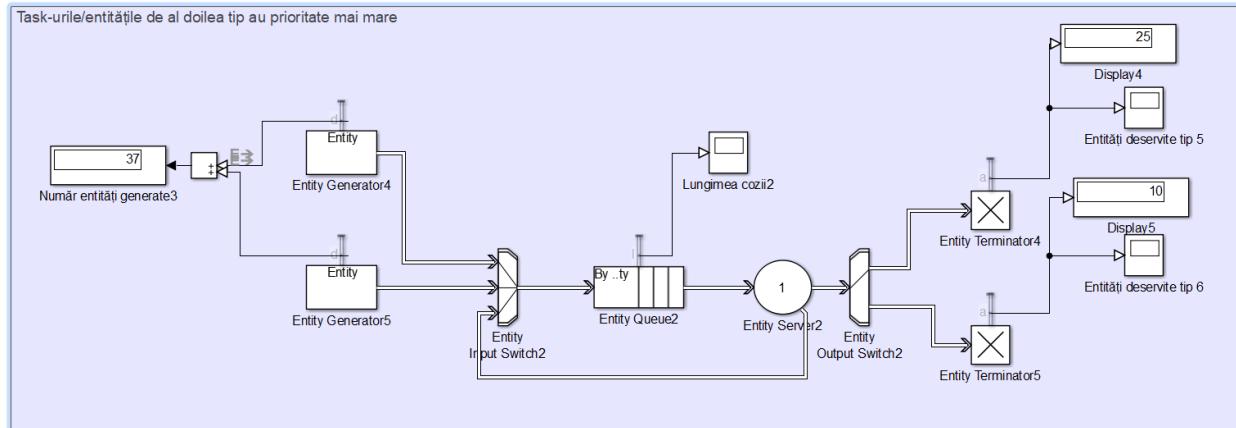
Ceea ce se obține în urma rulării este:



Pentru cazul în care entitățile de al doilea tip au prioritate mai mare:



Ceea ce se obține în urma rulării este:



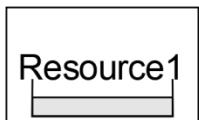
Rezultatele sunt aceleași pentru toate cele 3 cazuri.

Lucrarea 8

BLOCURI NOI FOLOSITE

În acest laborator au fost prezentate blocurile Resource Pool, Resource Acquirer, Resource Releaser.

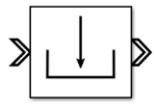
1. RESOURCE POOL



Resource Pool

- definește o resursă ce poate apartine unei entități
- odată obținută de o entitate, ea poate fi folosită și eliberată

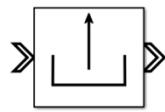
2. RESOURCE ACQUIRER



Resource
Acquirer

- prin intermediul său o entitate poate obține o resursă

3. RESOURCE RELEASER

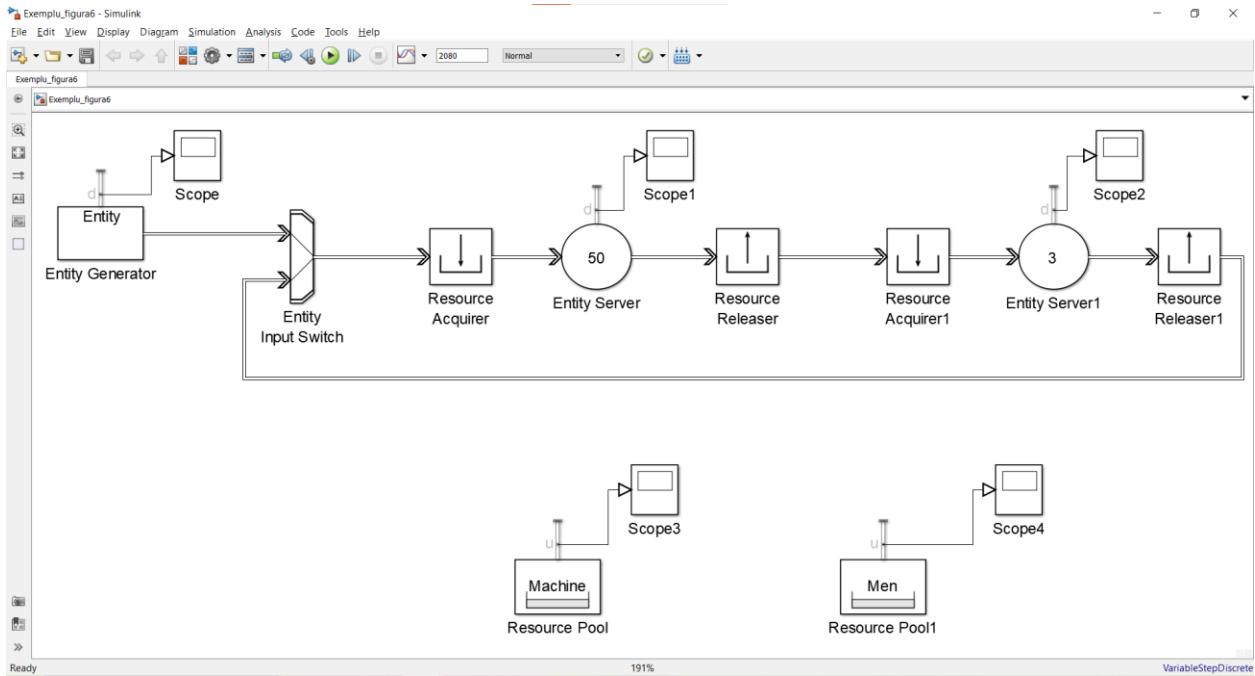


Resource
Releaser

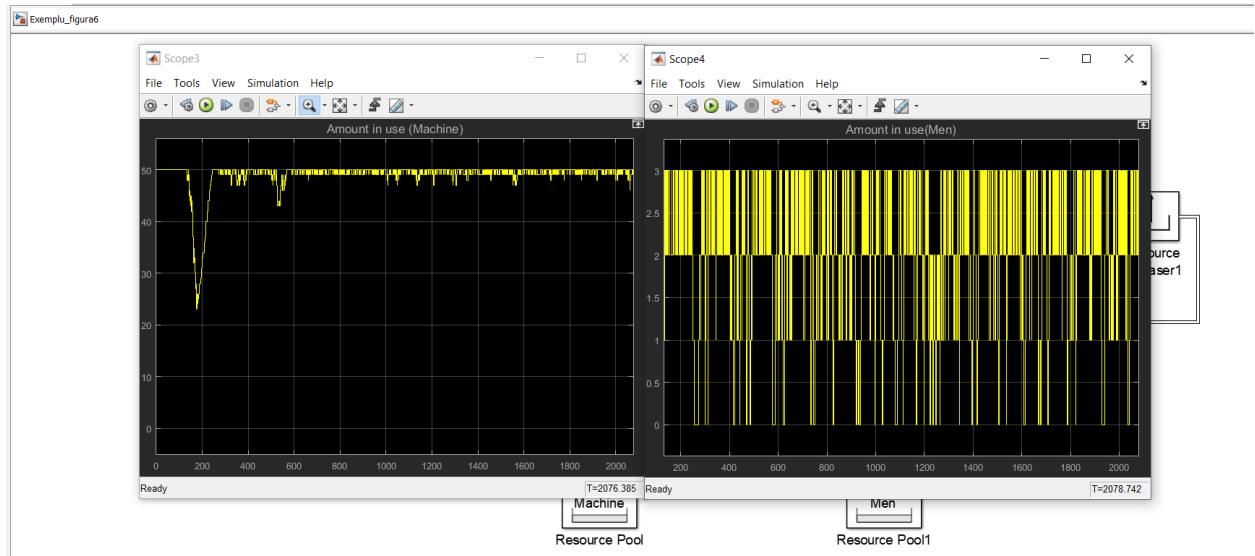
- prin intermediul său entitatea întoarce resursa de unde a primit-o, adică în Resource Pool.

EXEMPLUL PREZENTAT ÎN PLATFORMA DE LABORATOR

Exemplul arată astfel:



Output-urile sale sunt asemănătoare cu cele obținute în platforma de laborator.



3.Probleme propuse

PROBLEMA 1

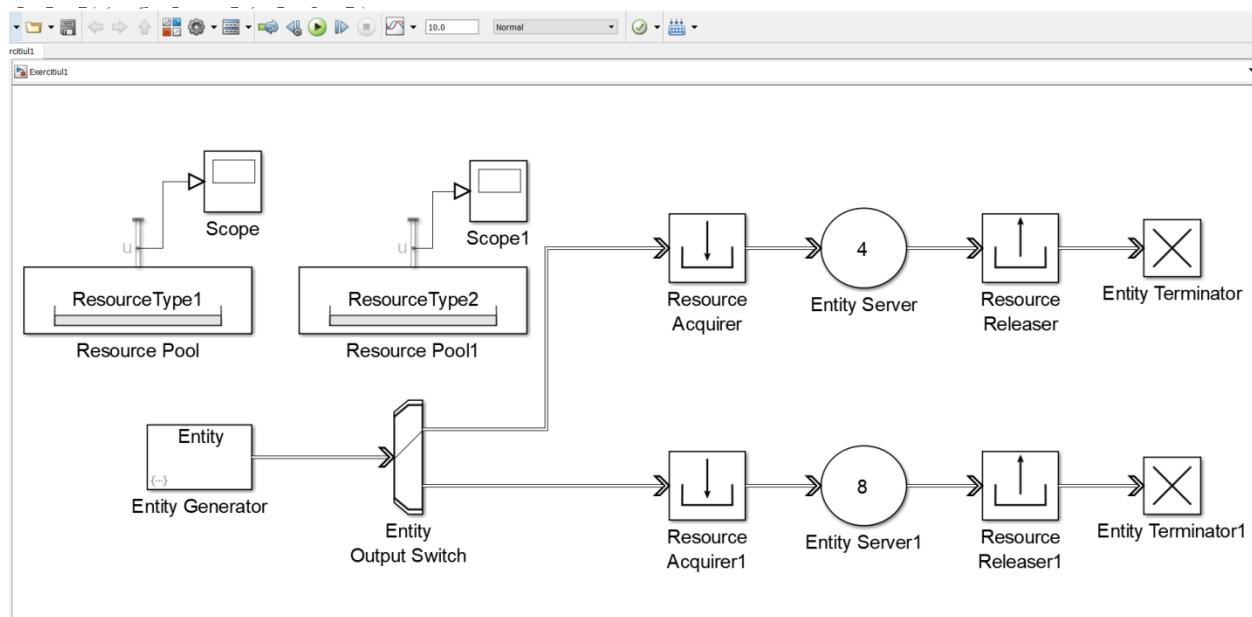
Să se modeleze un sistem ce are două resurse partajate, una cu capacitate patru și alta cu capacitate opt. Un bloc Entity Generator generează entități, cu intervalul între două entități cu distribuție uniformă cu valorile cuprinse între 0.01s și 0.2s. Entitățile generate vor avea un parametru Type, cu valorile unu și doi cu probabilitățile 0.4 și 0.6. Entitățile cu parametrul Type egal cu unu vor partaja resursa cu capacitate patru și au timpul de deservire 0.5s. Entitățile cu parametrul Type egal cu doi vor partaja resursa cu capacitate opt și au timpul de deservire 0.7s. Se va simula sistemul pe 10s. Pentru fiecare

resursă se va înregistra numărul de resurse disponibile. Se vor modifica timpii de deservire ai celor două tipuri de entități și capacitatele resurselor.

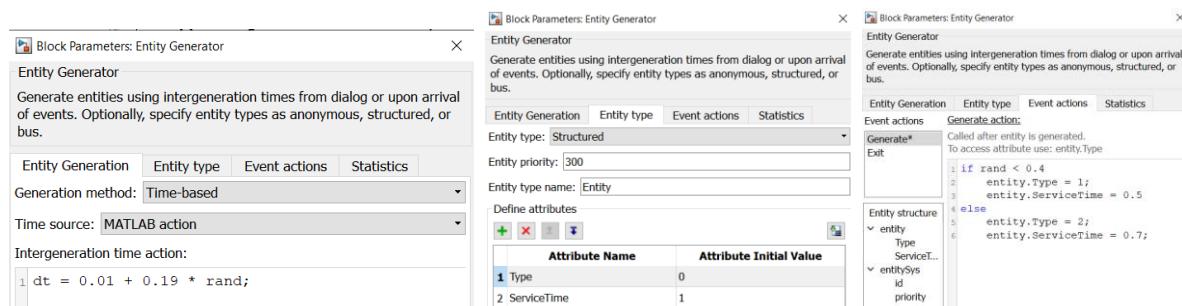
*distribuție uniformă cu valorile cuprinse între $0.01s$ și $0.2s - dt = 0.01 + 0.19 * rand$*

REZOLVARE:

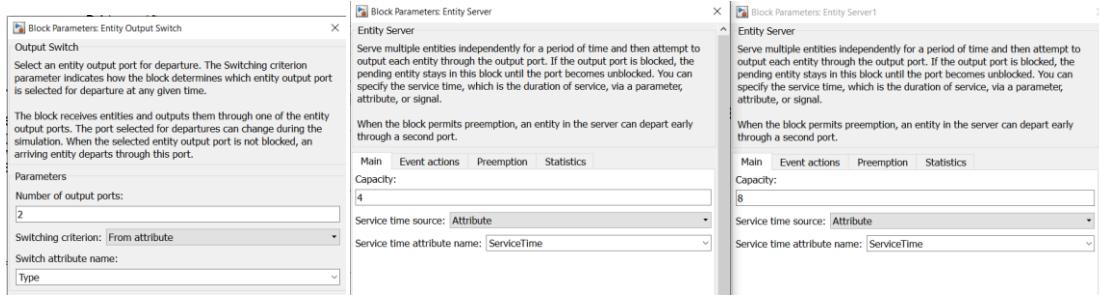
Modelul arată astfel:



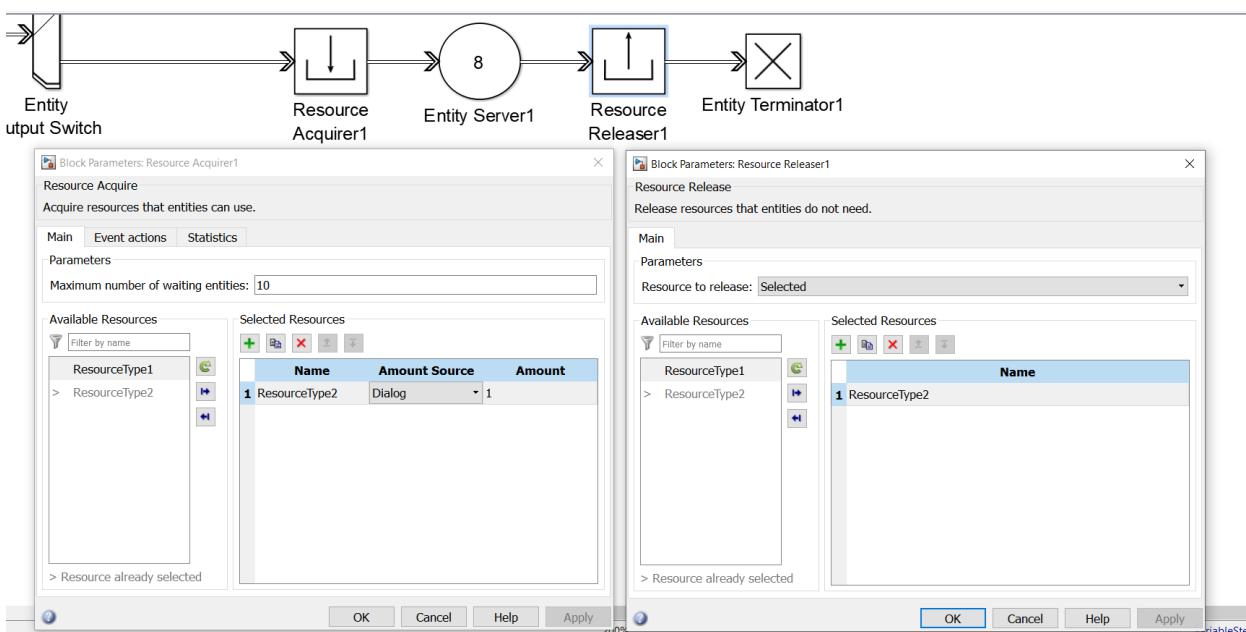
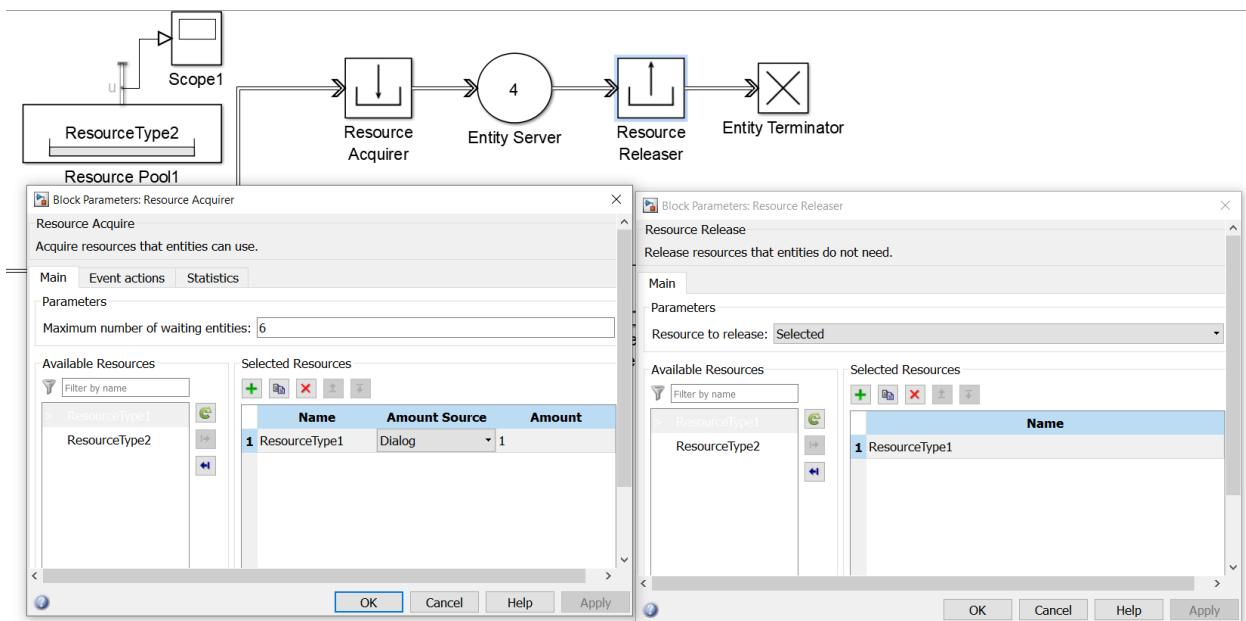
Entity Generator generează entități care au ca atribute Type și ServiceTime.



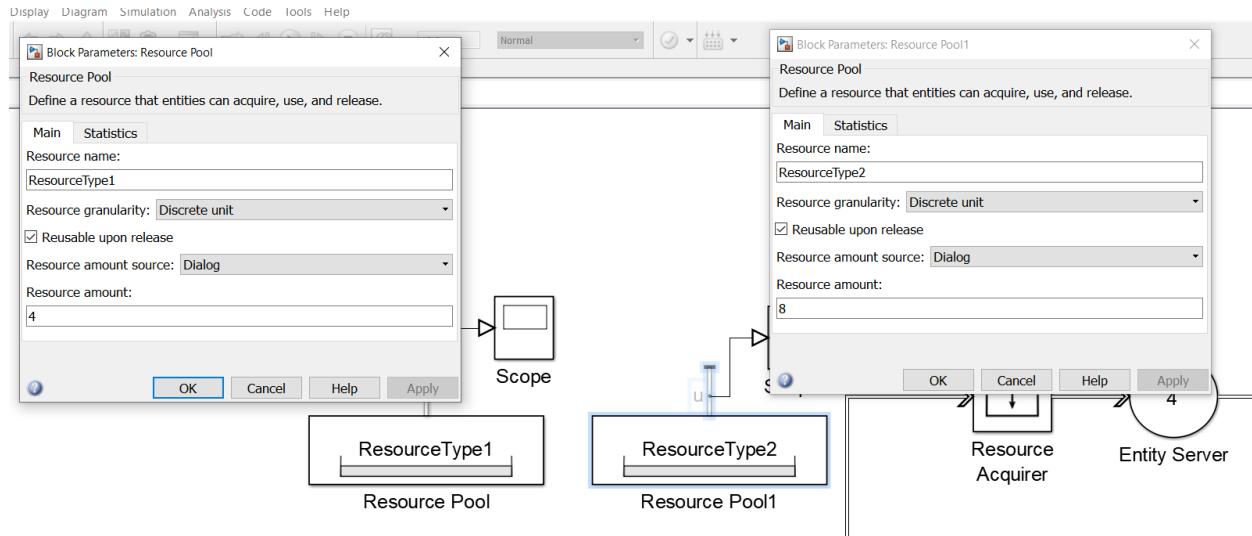
De asemenea, sunt şanse de 40%, respectiv 60% să se genereze entități cu prioritățile 1 sau 2. Fiecare dintre ele are propriul său timp de deservire, deci și acesta va fi modificat în funcție de probabilitate (rand). Blocul Entity Output Switch primește entitățile, pe care le trimite mai departe în funcție de Type. După ce au fost trimise, ele vor intra într-un Resource Acquirer, iar apoi într-un Entity Server ce are ca Service time source atributa ServiceTime a entităților.



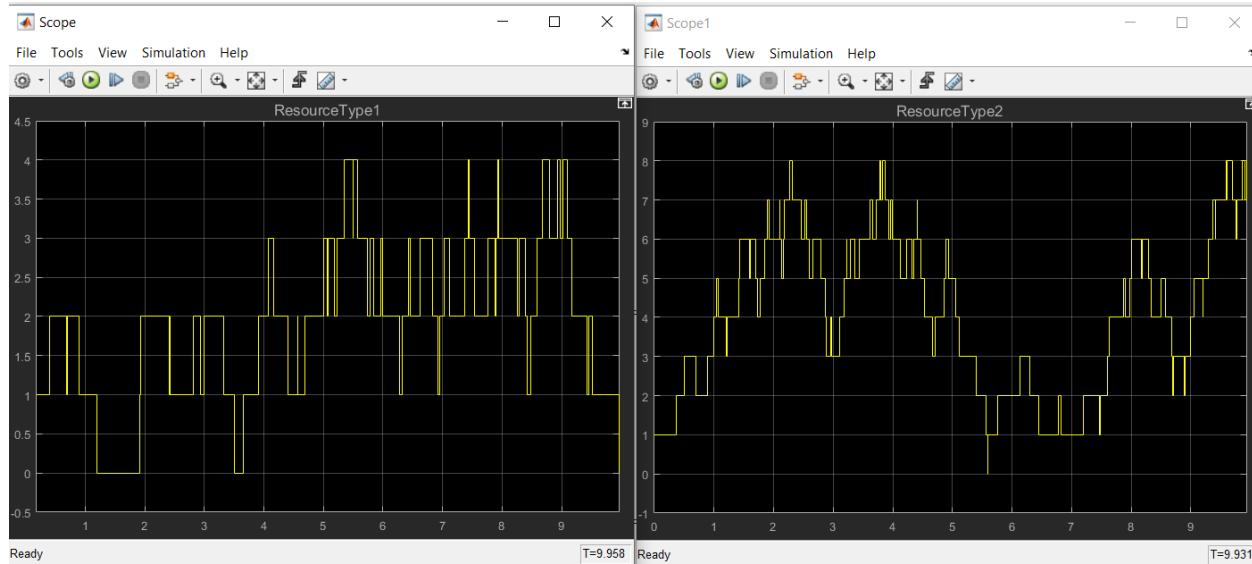
Blocurile Resource Acquirer și Resource Releaser sunt configurate astfel:



În cazul de față, Resource Acquirer au maximum number of pending entities mai mare decât capacitatea Entity Server-urilor, deoarece aşa cere mai încolo exercițiul. De asemenea, blocurile de Resource Pool au Resource amount capacitatea Entity Server-ului corespunzător fiecărui Resource Releaser.



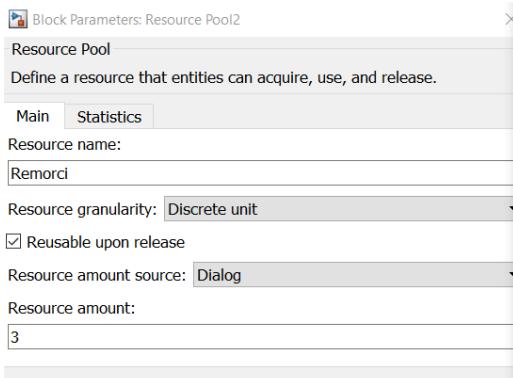
Rezultatele obținute în urma simulării sunt:



PROBLEMA 2

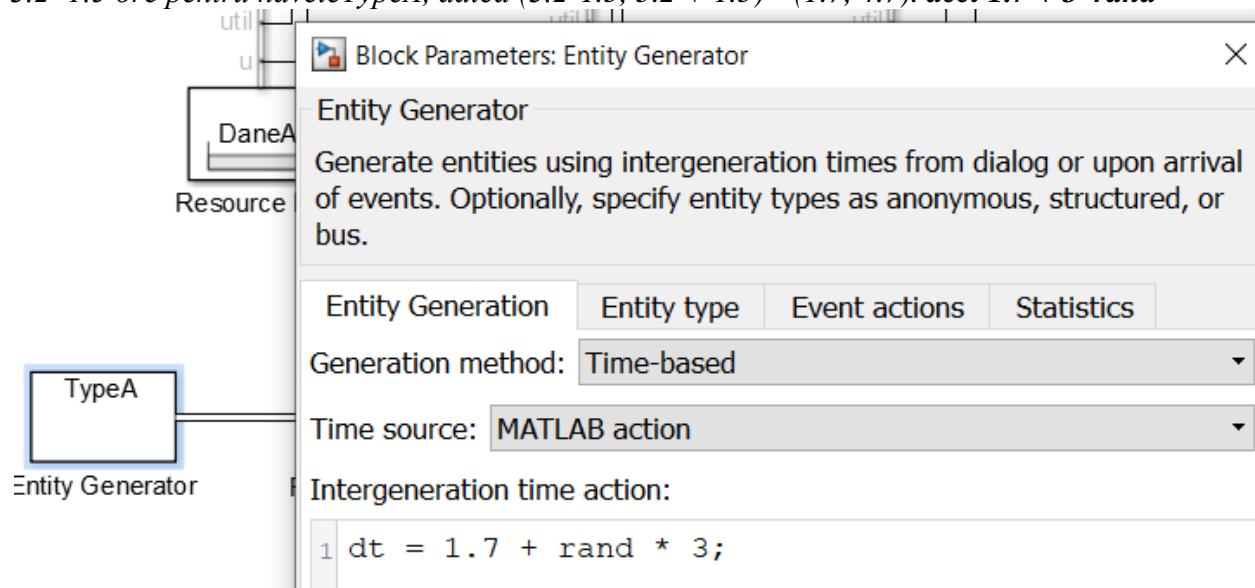
Încărcarea și descărcarea navelor într-un port. Într-un port vin două tipuri de nave, TypeA și TypeB pentru încărcare / descărcare. Portul are trei dane pentru navele TypeA și două dane pentru navele TypeB.

- Există trei remorchere utilizate de nave pentru tractarea navelor în port pentru încărcare / descărcare și după încărcare / descărcare pentru tractarea navelor la ieșirea din port.
- resource pool-ul corespunzător remorcilor are Resource amount 3.*



- Timpii între sosirea navelor în port sunt: 3.2 ± 1.5 ore pentru navele TypeA și 1.5 ± 0.75 ore pentru navele TypeB.

3.2 ± 1.5 ore pentru navele TypeA, adică $(3.2 - 1.5, 3.2 + 1.5) = (1.7, 4.7)$. deci $1.7 + 3 * \text{rand}$



1.5 ± 0.75 ore pentru navele TypeB, adică $(1.5 - 0.75, 1.5 + 0.75) = (0.75, 2.25)$. deci $0.75 + 1.5 * \text{rand}$

- Timpii de intrare în port, încărcare / descărcare și ieșire din port sunt cei din Tabelul 6.

Tip Navă	Intrare în port	Încărcare/descărcare	Ieșire din port
TypeA	0.5h (3 remorcare)	8.4 ± 1.5 h	0.25 h (2 remorcher)
TypeB	0.5h (2 remorcare)	2.1 ± 0.6 h	0.25 h (1 remorcher)

timpul în ore se setează pentru Entity Servers:

-intrare în port – Tracarea navei TypeA

-încărcare/descărcare – Încărcarea/Descărcarea navei Type..

-ieșire din port – Tractarea navei la dană

numărul de remorcare este corespunzător Resource Acquirer-urilor:

-intrare în port – Reținerea remorcherelor pentru intrare în port

-ieșirea din port – reținerea remorcherelor pentru ieșirea din port

- Numărul de resurse în uz pentru dane și remorchere

acesta este reprezentat de **Amount in use**, #u, de la fiecare **Resource Pool**

- Gradul de utilizare pentru dane și remorchere

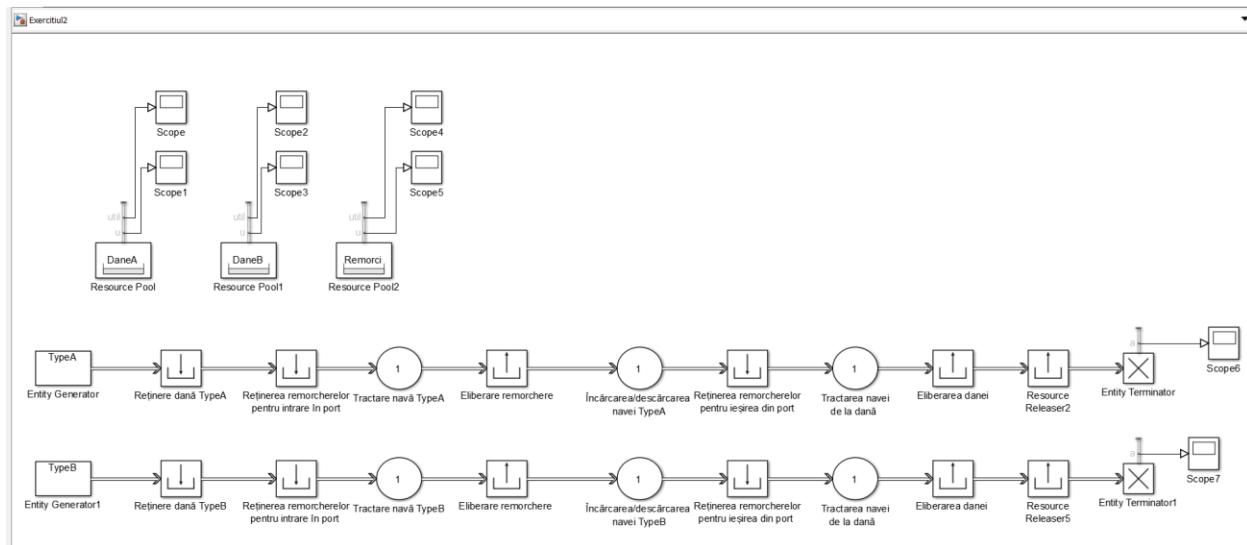
acesta este reprezentat de **Average utilization** de la fiecare **Resource Pool**

- Numărul de nave deservite de fiecare tip.

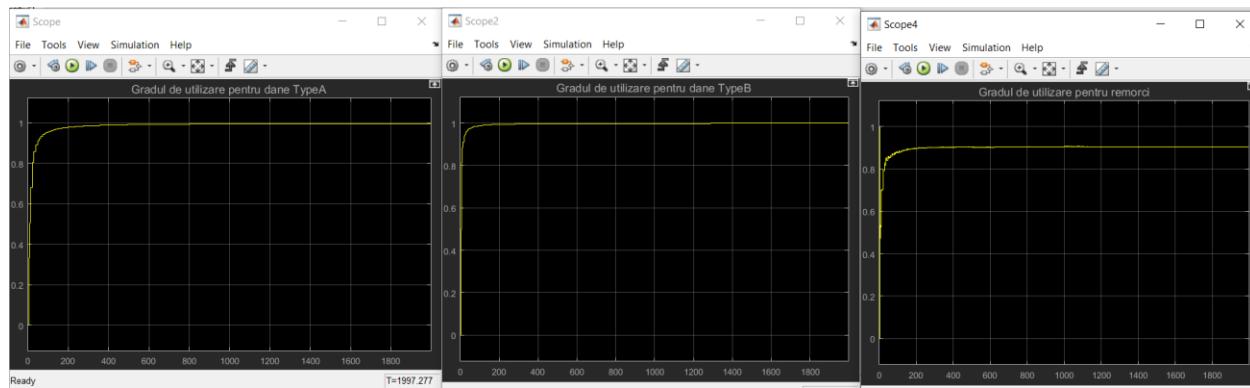
acesta este reprezentat de **Number of entities arrived** din fiecare **Entity Terminator**

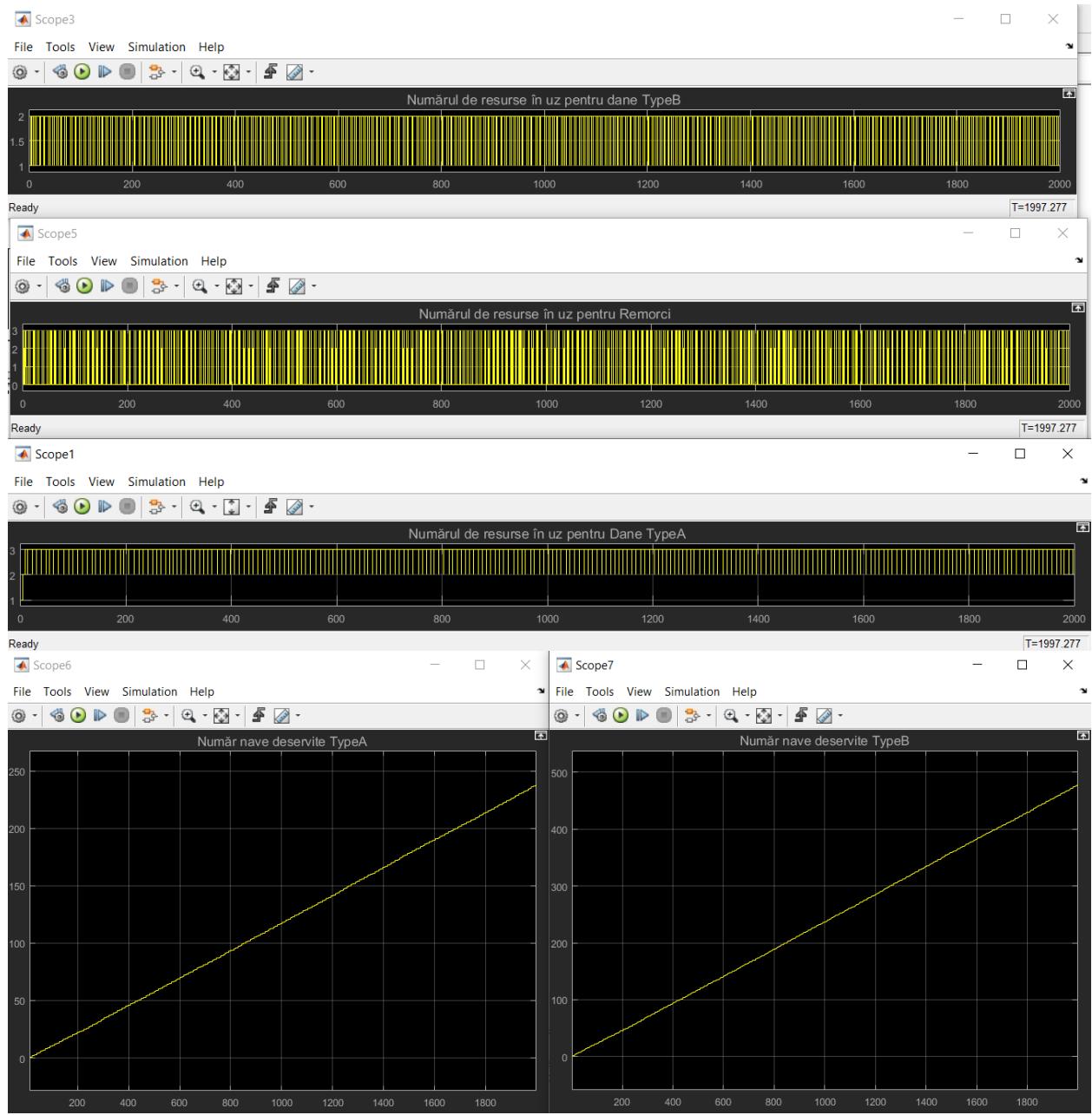
REZOLVARE:

Modelul arată astfel:



Ceea ce se obține în urma rulării programului pentru 2000 ore (unitatea de timp este ora):





PROBLEMA 3

se dau:

- În prima etapă un muncitor asamblează piesele ce constituie o sculă. Etapa durează 30 ± 5 minute cu distribuție uniformă. *durează 30 ± 5 minute cu distribuție uniformă, adică $(30-5, 30+5) = (25,40)$ deci $25 + 10 * rand$*

- În a două etapă același muncitor utilizează un cuptor pentru tratarea termică a sculei. Etapa durează 8 ± 2 minute cu distribuție uniformă. După aceasta scula este terminată și muncitorul începe procesul de fabricare a unei noi scule
durează 8 ± 2 minute cu distribuție uniformă, deci $(8-2, 8+2) = (6, 10)$ deci $6 + 4 * \text{rand}$
- Există patru muncitori și un singur cuptor.
este un singur Entity Server cu capacitatea 4.
- Muncitorii utilizează acest cuptor cu ordinea de deservire primul venit, primul servit. Se va simula modelul pe 40 h.
unitatea de timp aleasă este minutul. - $60 * 40$ minute au $40h = 2400$

se cer:

- timpul mediu de utilizare a cupitorului

acesta este reprezentat de **Average wait** al lui Entity Server1

- gradul de utilizare

gradul de utilizare al cupotorului este reprezentat de **Utilization** al lui Entity Server1

- lungimea cozii

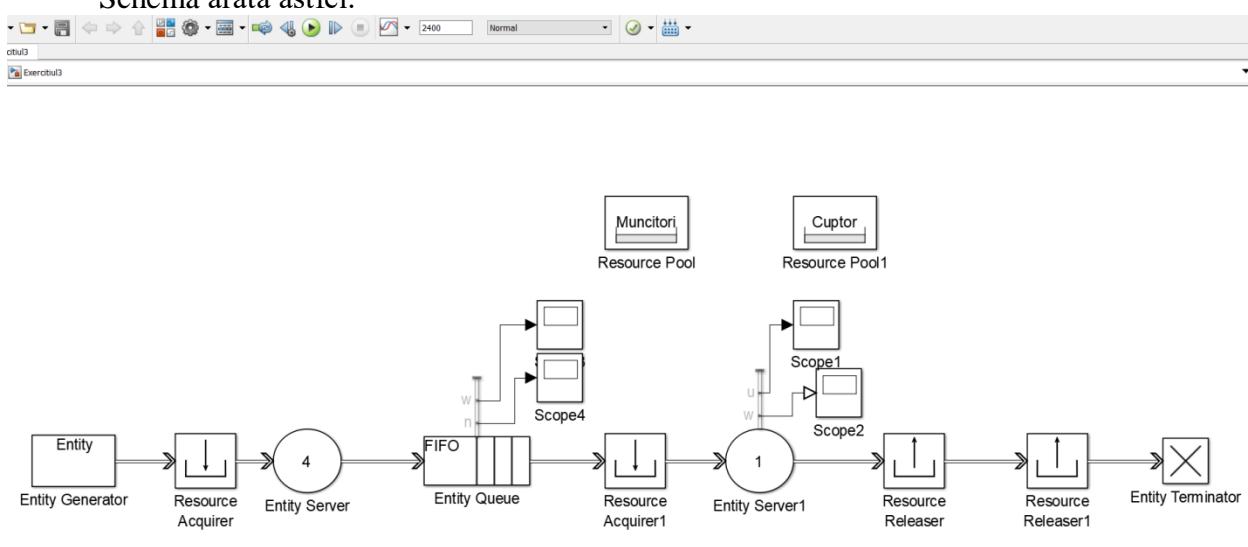
aceasta este reprezentată de **Number of entities in block** din Entity Queue

- timpul mediu de așteptare în coadă

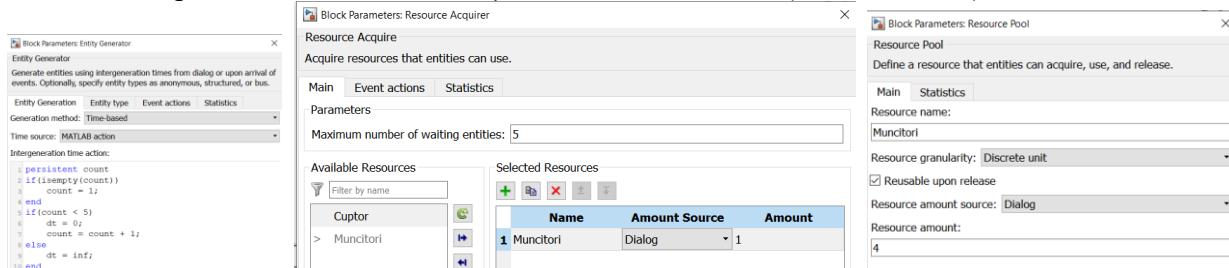
acesta este reprezentat de **Average wait** din Entity Queue.

REZOLVARE:

Schema arată astfel:



În primul sample al aplicației se generează 5 entități, deoarece aşa se cerea într-un exercițiu anterior asemănător cu acesta de la alt laborator. După aceea, entitatea trece prin blocul Resource Acquirer, care îi oferă entității o resursă Muncitori (una din cele 4).



După aceea, entitatea intră în Entity Server, unde este deservită după 25 + rand * 10 minute, iar apoi intră într-un Entity Queue, unde aşteaptă până când îi vine rândul la cupitor. Atunci când iese din Entity queue, intră într-un Resource Acquirer, de unde primește singura resursă Cuptor.

The left screenshot shows the 'Resource Acquire' configuration. It has a 'Parameters' section with 'Maximum number of waiting entities: 5'. Below it is a table titled 'Selected Resources' with one row: 'Cuptor' (Amount: 1). The right screenshot shows the 'Resource Pool' configuration. It has a 'Resource name:' field set to 'Cuptor', a 'Resource granularity:' dropdown set to 'Discrete unit', and a checked 'Reusable upon release' checkbox. The 'Resource amount source:' is set to 'Dialog' and the 'Resource amount:' is set to '1'.

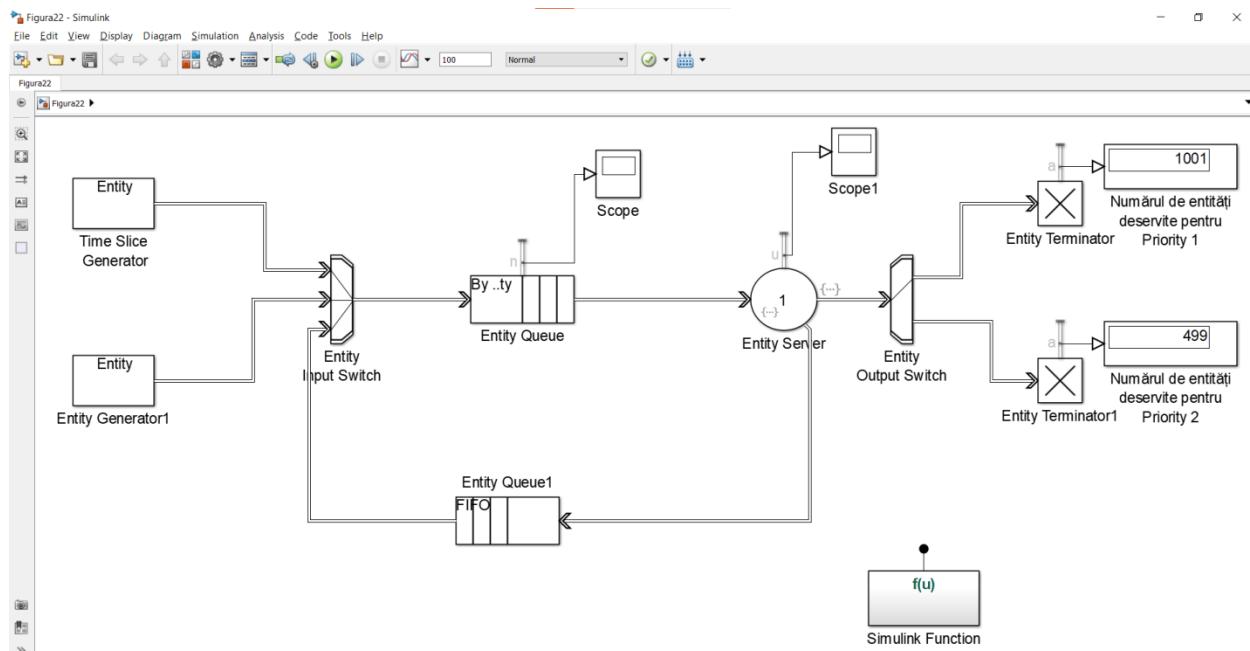
După ce a primit resursa Cuptor, intră într-un Entity Server unde este tratată termic scula de către același muncitor (este același deoarece nu a fost niciun Resource Releaser pentru el până acum). După ce este tratat termic, entitatea/scula trece prin două Resource Releaser, câte unul pentru fiecare resursă pe care o folosește: Cuptor și Muncitor.

The left screenshot shows the 'Resource Release' configuration. It has a 'Parameters' section with 'Resource to release: Selected'. Below it is a table titled 'Selected Resources' with one row: 'Cuptor'. The right screenshot shows the 'Resource Releaser' configuration. It has a 'Parameters' section with 'Resource to release: Selected'. Below it is a table titled 'Selected Resources' with one row: 'Muncitori'.

Lucrarea 9

3.Probleme propuse

figura 22



Se vor înregistra:

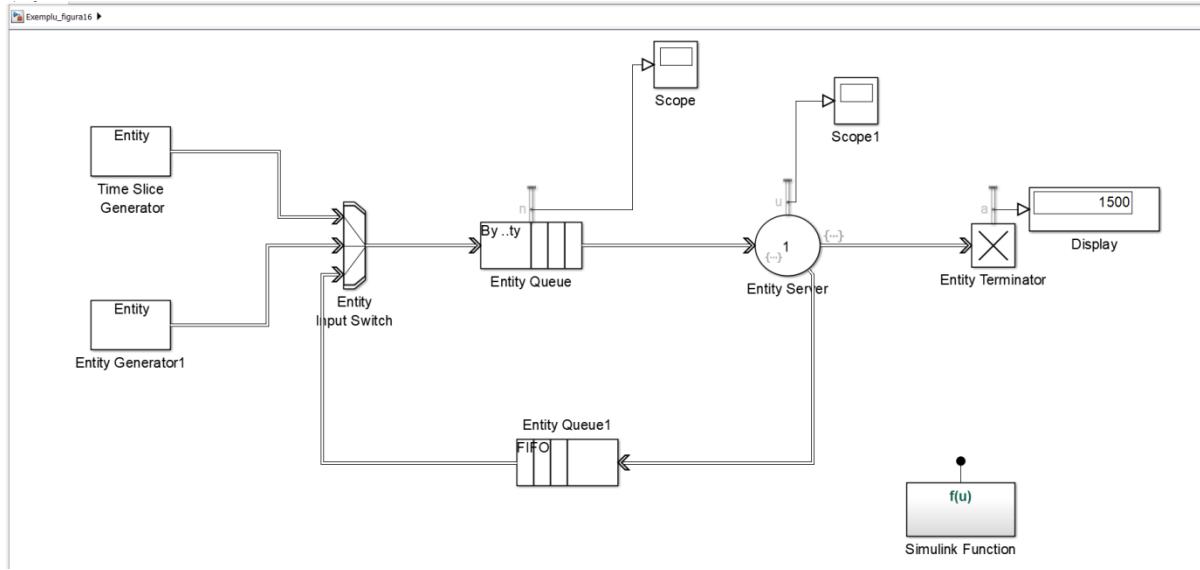
- numărul de entități în coadă și utilizarea serverului;
- numărul de entități deservite pentru fiecare prioritate.

Unitatea de timp a simulării este milisecunda. Se va simula modelul pe o durată de 100ms.

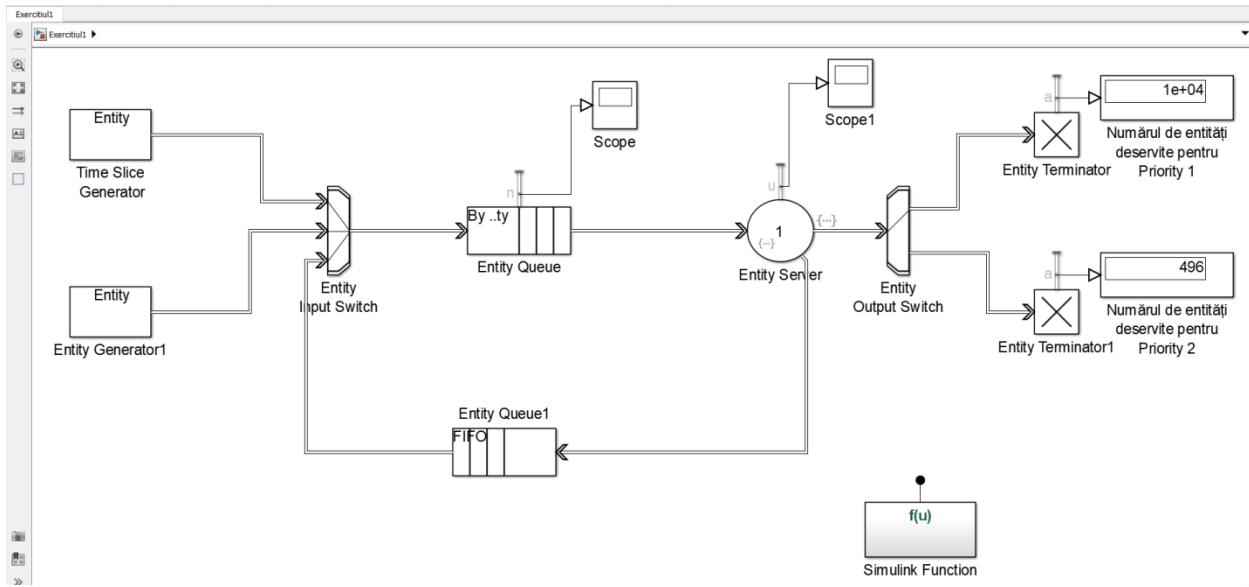
numărul de entități în coadă este Number of entities in block din Entity Queue

numărul de entități deservite pentru fiecare prioritate sunt Number of entities arrived din Entity Terminator-uri

Se poate observa că, la figura 16 se obțin același număr de entități, doar că acolo ele sunt însumate:

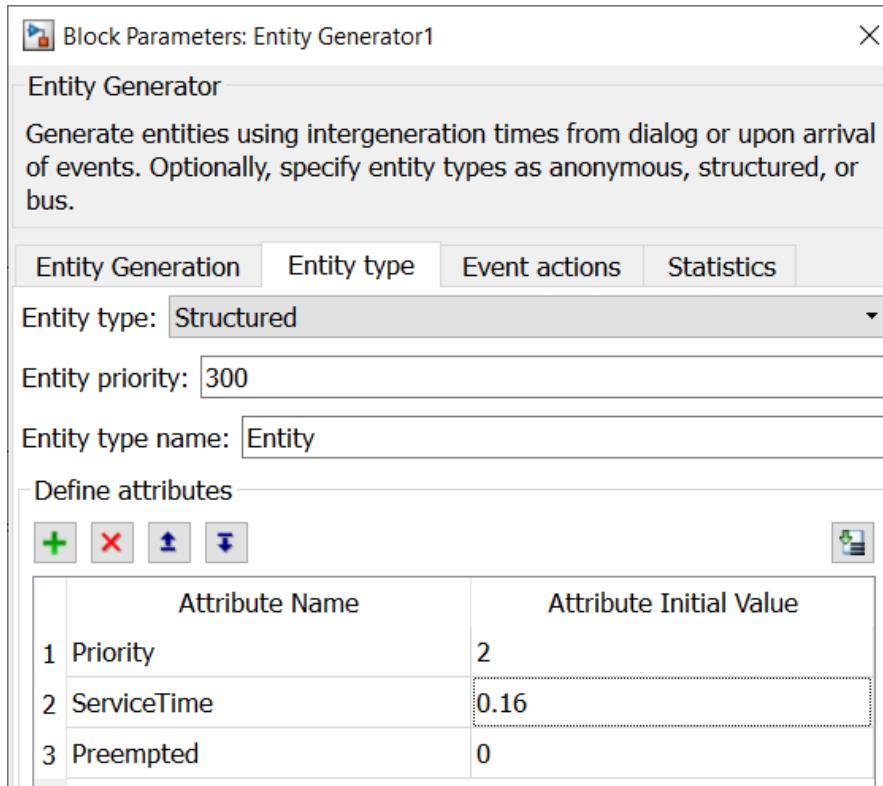


PROBLEMA 1

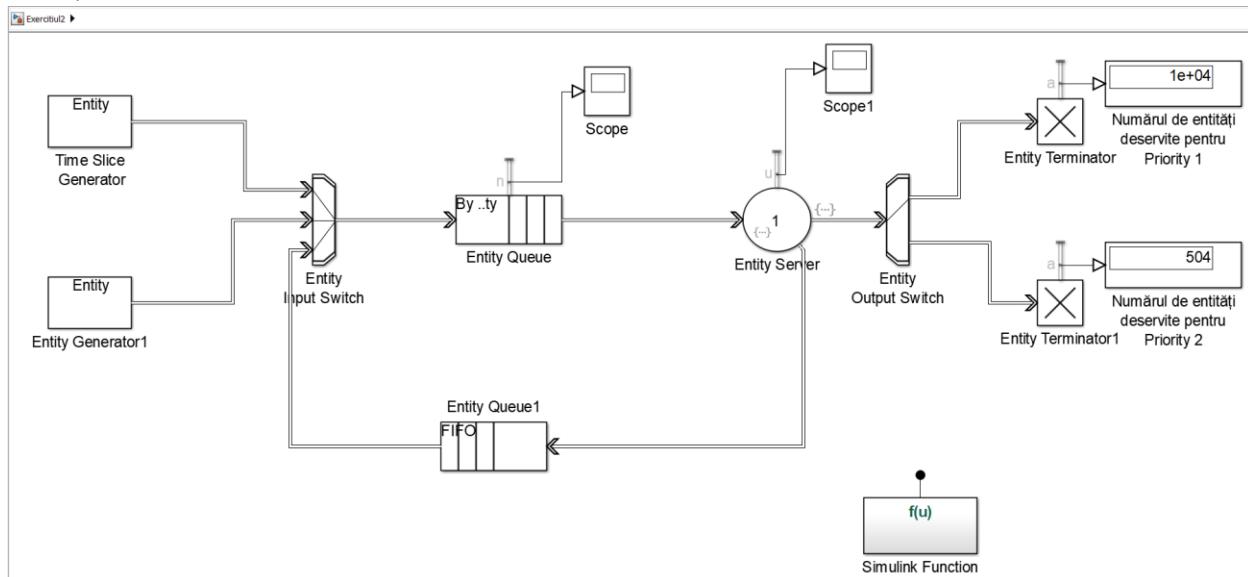


Se poate observa că, prin simpla modificare a perioadei de generare a evenimentelor în cazul lui Time Slice Generator la 0.01ms, se obțin de 10 ori mai multe entități decât atunci când timpul era 0.1ms. Acest lucru se întâmplă pentru că acum entitățile din Time Slice Generator se generează de 10 ori mai repede. Se poate observa, de asemenea, că entitățile generate de Entity Generator au aproximativ aceeași valoare.

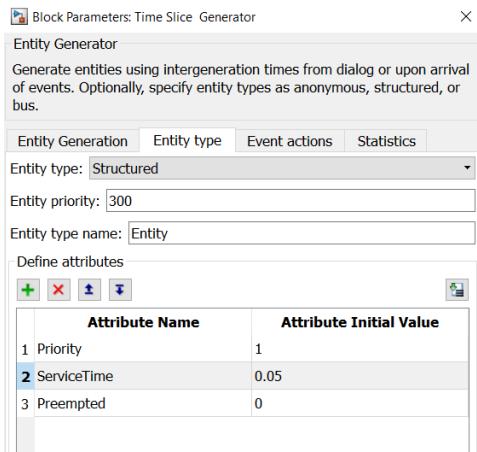
PROBLEMA 2



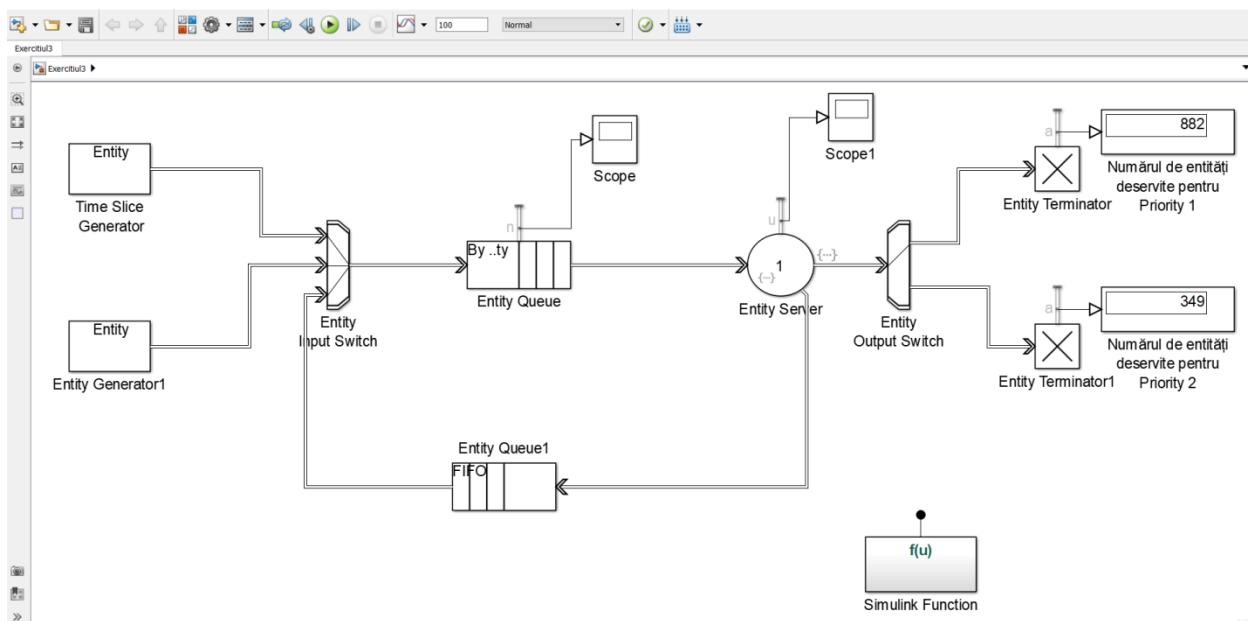
Prin modificarea perioadei pentru blocurile Time Slive Generator (de la 0.1ms la 0.01ms) și pentru Entity Server (de la 0.18 la 0.16ms), se poate observa că acum se deservesc mai multe entități generate de Entity Generator1.



PROBLEMA 3



Odată cu adăugarea unei perioade de servire a entităților generate de Time Slice Generator, se obține:



În exercițiul anterior, numărul de entități cu prioritatea 2 deservite (504) este mai mare decât la exercițiul curent (349). Acest lucru se întâmplă datorită faptului că am adăugat valoarea 0.05ms atributului ServiceTime pentru entitățile generate în Time Slice Generator. Înainte ele treceau prin Entity Server fără să stea acolo, adică în Entity Server singurele entități care intrau și nu ieșeau imediat erau cele generate de Entity Generator. Acum nu mai e cazul. Si ele, la rândul lor, trebuie să își aștepte locul în Entity Server, mai ales că au prioritatea 2, iar coada are ca criteriu de ieșire prioritatea, adică Priority.



Block Parameters: Entity Queue

X

Entity Queue

Store entities in a queue. The block can queue entities based on arrival order or priority. The entity at the head of the queue departs when the downstream block is ready to accept it. You can specify the queue capacity.

Optionally, configure this block to receive entities from a multicast source with the same tag.

Main Event actions Statistics

Capacity:

50

Queue type: Priority

Priority source: Priority

Sorting direction: Ascending

Entity arrival source: Input port