УДК 519.87:519.248:519.682

А. М. Журавлев, Н. В. Медведев, А. Ю. Быков

ИСПОЛЬЗОВАНИЕ МЕЖПЛАТФОРМЕННОЙ БИБЛИОТЕКИ ФУНКЦИЙ ЯЗЫКА СИ++ ДЛЯ РЕАЛИЗАЦИИ ИМИТАЦИОННЫХ МОДЕЛЕЙ ТИПОВЫХ СИСТЕМ МАССОВОГО ОБСЛУЖИВАНИЯ

Рассмотрены имитационные модели классических систем массового обслуживания, реализованные с использованием библиотеки функций языка Си++. Все приведенные системы массового обслуживания имеют аналитические решения. Представлены листинги программ и результаты решений, полученные на имитационных моделях с помощью библиотеки функций языка Си++. Для сравнения также приведены результаты решений, полученные аналитическим путем.

Для исследования сложных систем массового обслуживания (СМО) широко используется имитационное моделирование [1]. В работе [2] для имитационного моделирования СМО предложено использовать межплатформенную библиотеку функций языка Си++, а также рассмотрены особенности данной библиотеки, порядок ее использования и настройки.

Кратко определим основные отличительные особенности данной библиотеки функций:

— библиотека функций является межплатформенной, т.е. может использоваться как для операционных систем (ОС) Windows 98/ME/2000/XP/Vista, так и для UNIX-подобных ОС типа Linux, МС ВС и др.;

— библиотека функций встраивается в стандартные средства разработки приложений на основе языка Си++ типа Microsoft Visual C++ или KDevelop;

— названия основных функций библиотеки напоминают названия соответствующих операторов языка имитационного моделирования GPSS [3], построение имитационных моделей с помощью данной библиотеки функций не более трудоемкое, чем на языке GPSS;

— использовать данную библиотеку могут обычные приложения языка Си++, таким образом, можно создавать комбинированные приложения, в которых кроме задач имитационного моделирования могут решаться и другие прикладные или системные задачи.

Рассмотрим примеры имитационных моделей типовых (классических) СМО [4].

Многоканальная СМО с отказами (задача Эрланга). В качестве примера рассмотрим следующую СМО. Трехканальная СМО обслуживает пуассоновский поток заявок, время между заявками является случайной величиной, распределенной по показательному закону, интенсивность потока заявок составляет 10 заявок в минуту. Время обслуживания одной заявки также распределено по показательному закону, интенсивность потока обслуживания — 4 заявки в минуту. Если заявка приходит в момент, когда все каналы заняты, то она получает отказ в обслуживании. Требуется провести имитационное моделирование СМО в течение 100 ч и определить параметры СМО — вероятность отказа в обслуживании, вероятность того, что заявка будет обслужена, абсолютную пропускную способность (среднее число заявок, обслуживаемых в единицу времени), среднее число занятых каналов. (Примером такой СМО может являться телефонная станция.)

Данная задача имеет аналитическое решение. Основные соотношения следующие:

$$p_0 = \left(1 + \rho + \frac{1}{2!}\rho^2 + ... + \frac{1}{n!}\rho^n\right)^{-1}$$

— вероятность того, что в системе нет ни одной заявки ($\rho = \lambda/\mu$ — приведенная интенсивность, $\lambda = 10$ заявок в минуту — интенсивность входного потока заявок, $\mu = 4$ заявки в минуту — интенсивность обслуживания, $n = 3$ — число каналов обслуживания); $p_n = \frac{1}{n!}\rho^n p_0$ — вероятность того, что в системе n заявок; $P_{\text{отк}} = p_n$ — вероятность отказа в обслуживании; $P_{\text{обс}} = 1 - P_{\text{отк}} = 1 - \frac{1}{n!}\rho^n p_0$ — вероятность того, что заявка будет обслужена; $A = \lambda P_{\text{обс}} = \lambda\left(1 - \frac{1}{n!}\rho^n p_0\right)$ — абсолютная пропускная способность; $\bar{k} = A/\mu = \rho\left(1 - \frac{\rho^n}{n!}p_0\right)$ — среднее число занятых каналов.

Подставляя заданные значения, получаем следующие результаты: $P_{\text{отк}} \approx 0{,}282$; $P_{\text{обс}} \approx 0{,}718$; $A \approx 7{,}178$ заявки в минуту, $\bar{k} \approx 1{,}795$.

Исходный код модели и результаты решения приведены в листинге 1.

Листинг 1

```
#include "SimC.h"
void main()
// Задача Эрланга
{
      PSTORAGE pS; // Указатель на многоканальное устройство
      int N=0, N_Otk=0; // Переменные для подсчета вероятностей
      Start(); // Создание системной среды
      // Создание модельной среды
      NewStorage(pS, "Уст-во", 3);// Создаем трехканальное устройство
      InitGenerate(1, 0);// Начальный транзакт-заявка поступает в событие 1
      while(SysTime<6000) // Цикл обработки событий - время моделирования
                          //100 часов (6000 минут)
      {
            Plan(); // Протяжка модельного времени
            switch(SysEvent) // Обработка событий
            {
            case 1: Generate(RandExp(10, V1)); break; //Поступает транзакт -
                                                      //заявка

            case 2: N++; // Подсчет общего числа транзактов
                  if (pS->SS<3) Enter(pS);  // Если есть место, то занимаем
                                            //многоканальное устройство
                  else {
                        N_Otk++; // Число транзактов, получивших отказ
                        Next(5); // Транзакт получает отказ в обслуживании
                  }
                  break;
            case 3: Advance(RandExp(4, V2)); break;// Задержка - обработка
                                                   // заявки
            case 4: Leave(pS); break; // Освобождаем многоканальное
                                      // устройство
            case 5: Terminate(); break;// Уничтожение транзакта - заявки
            }
      }
    PrintAll(); // Печать результатов моделирования
      fprintf(OUTFILE, "Вер-ть отказа: %f Вер-ть обслуживания %f Абсолютная
пропуск. способность %f",
            (double)N_Otk/N, (double)(N-N_Otk)/N, (N-N_Otk)/SysTime);
}
```

```
     СИСТЕМНОЕ ВРЕМЯ          : SysTime=        6000.056
     СИСТЕМНОЕ СОБЫТИЕ        : SYSEVENT=  1
     НОМЕР ТРАНЗАКТА          : TRANS->NOM=59841
     ВСЕГО ВЫПОЛНЕНО СОБЫТИЙ : EVENTALL=265533
     ЗАТРАЧЕНО ВРЕМЯ          : REALTIME=          0.09 СЕК
     СКОРОСТЬ ИСПОЛНЕНИЯ      : EVESPEED=2855193.548587 СОБ/СЕК
     СРЕДНЕЕ ВРЕМЯ ИСПОЛНЕНИЯ СОБЫТИЯ : MEANTIME=     0.000000 СЕК/СОБ

 СОБ.#  ВСЕГО     СОБ.#  ВСЕГО     СОБ.#  ВСЕГО     СОБ.#  ВСЕГО     СОБ.#  ВСЕГО
    1   59841        2   59840        3   43006        4   43006        5   59840

                        ****************************
                        *                          *
                        *  ----   НАКОПИТЕЛИ  ----  *
                        *                          *
                        ****************************

 НАКОПИТЕЛЬ  ЕМКОСТЬ ЗАГРУЗКА  СРЕДН. ВРЕМЯ  ТЕКУЩЕЕ  МАКС.  СРЕДНЕЕ  ЧИСЛО
                              ПРЕБЫВАНИЯ    СОДЕРЖ. СОДЕРЖ. СОДЕРЖ.   ВХОДОВ
   Уст-во       3     0.598        0.250         0       3      1.795   43006

Вер-ть отказа: 0.281317 Вер-ть обслуживания 0.718683 Абсолютная пропуск.
способность 7.167600
```

Одноканальная СМО с неограниченной очередью. В качестве примера рассмотрим следующую СМО. Одноканальная СМО обслуживает пуассоновский поток заявок, время между заявками является случайной величиной, распределенной по показательному закону, среднее время между заявками составляет 10 с ($\lambda = 0{,}1$ заявки в секунду). Время обслуживания одной заявки также распределено по показательному закону, среднее время обслуживания 8 с ($\mu = 1/8$ заявки в секунду). Если заявка приходит в момент времени, когда канал занят, то она становится в очередь, длина очереди не ограничена. Требуется провести имитационное моделирование СМО при обслуживании 200 000 заявок, определить параметры СМО — параметры очереди задач (среднюю длину очереди и среднее время ожидания задачи в очереди), загрузку канала обслуживания.

По аналогии с предыдущим примером данная задача имеет аналитическое решение [4]. Основные соотношения имеют следующий вид:

$p_0 = 1 - \rho$ — вероятность того, что в системе нет заявок;

$\bar{Q}_{\text{очер}} = \dfrac{\rho^2}{1-\rho}$ — средняя длина очереди;

$\bar{T}_{\text{очер}} = \dfrac{\rho^2}{\lambda\,(1-\rho)}$ — среднее время ожидания в очереди;

$K_{\text{загр}} = 1 - p_0 = \rho$ — коэффициент загрузки канала обслуживания.

Подставляя заданные значения, получаем следующие результаты:

$$\bar{Q}_{\text{очер}} = 3{,}2; \quad \bar{T}_{\text{очер}} = 32\,\text{с}; \quad K_{\text{загр}} = 0{,}8.$$

Исходный код модели и результаты решения представлены в листинге 2.

Многоканальная СМО с неограниченной очередью. В качестве примера рассмотрим следующую СМО. Вычислительный центр состоит из 4 ПЭВМ. Задачи на решение поступают в среднем через 5 мин, время между задачами распределено по экспоненциальному закону. Каждая задача решается любой свободной ПЭВМ или ставится в очередь, если все ПЭВМ заняты, длина очереди не ограничена. Каждая задача решается в среднем 15 мин, время решения распределено по экспоненциальному закону. Требуется провести имитационное моделирование в течение 100 суток, определить загрузку вычислительного центра (среднее число занятых ПЭВМ) и параметры очереди задач (среднюю длину очереди и среднее время ожидания задачи в очереди).

Данная задача имеет аналитическое решение [4]. Основные соотношения по аналогии с рассмотренными примерами имеют следующий

Листинг 2

```
#include "SimC.h"
void main()
{
      // Одноканальная СМО с неограниченной очередью
      PFACILITY pF; // Указатель на канал обслуживания
      PQUEUE pQ; // Указатель на очередь
      Start(200000); //  Создание системной среды, задаем значение счетчика
                     // завершения
      // Создание модельной среды
      NewQueue(pQ, "Очередь"); // Создаем очередь
      NewFac(pF, "Канал"); // Создаем канал обслуживания
      InitGenerate(1, 0); // Транзакт - заявка направляется к первому событию

      while(TG1) //Конец моделирования, когда TG1 (счетчик завершения равен 0)
      {
            Plan();
            switch(SysEvent)
            {
            case 1:
                  Generate(RandExp(0.1, V1)); break; // Генерация заявок
            case 2:
                  Queue(pQ); break; // Занимаем очередь
            case 3:
                  Seize(pF); break; // Занимаем канал обслуживания
            case 4:
                  Depart(pQ); break; // Освобождаем очередь
            case 5:
                  Advance(RandExp(1./8, V2)); break; // Задержка - время
                                                     //обслуживания
            case 6:
                  Release(pF); break; // Освобождаем канал обслуживания
            case 7:
                  Terminate(1); // Транзакт уничтожается, счетчик

                              // завершения уменьшается на 1
            }
      }
      PrintAll();
}
```

```
     СИСТЕМНОЕ ВРЕМЯ          : SysTime=     2002693.038
     СИСТЕМНОЕ СОБЫТИЕ        : SYSEVENT=  7
    TRANS=NULL
     ВСЕГО ВЫПОЛНЕНО СОБЫТИЙ : EVENTALL=1560380
     ЗАТРАЧЕНО ВРЕМЯ          : REALTIME=         0.56 СЕК
     СКОРОСТЬ ИСПОЛНЕНИЯ      : EVESPEED=2771545.293063 СОБ/СЕК
     СРЕДНЕЕ ВРЕМЯ ИСПОЛНЕНИЯ СОБЫТИЯ : MEANTIME=     0.000000 СЕК/СОБ

 СОБ.#  ВСЕГО     СОБ.#  ВСЕГО     СОБ.#  ВСЕГО     СОБ.#  ВСЕГО     СОБ.#  ВСЕГО
     1 200001         3 200000         5 200000         7 200000
     2 200001         4 200000         6 200000         8       0
                      ****************************
                      *                          *
                      *  -----   ОЧЕРЕДИ   -----  *
                      *                          *
                      ****************************

  ОЧЕРЕДЬ  ЧИСЛО  ВХОДОВ МАКС. ДЛИНА СРЕДН. ВР. ОЖ. СРЕДНЯЯ     % ВХОДОВ
           ------------  -----------  --------------   ДЛИНА     В ПУСТУЮ
           С 0 ВРЕМ.ОЖ.  ТЕКУЩ.ДЛИНА  ВЕЗ УЧ. 0 ВХ.              ОЧЕРЕДЬ
  Очередь      200001          34           32.021      3.198     19.811
                39623           1           39.933
                      ****************************
                      *                          *
                      *  ----    ПРИБОРЫ    ----  *
                      *                          *
                      ****************************

 ПРИБОР      ЧИСЛО        СРЕДНЕЕ ВРЕМЯ    ЗАГРУЗКА      ЧИСЛО       СОСТОЯНИЕ
             ВХОДОВ       ОБРАБОТКИ                    ЗАХВАТОВ
    Канал   200000             8.025       0.801          0       FREE
```

вид:

$$p_0 = \left[1 + \rho + \frac{\rho^2}{2!} + \ldots + \frac{\rho^n}{n!} + \frac{\rho^{n+1}}{n!\,(n-\rho)}\right]^{-1}$$

— вероятность того, что в системе нет ни одной заявки (не решается ни одной задачи), ($\rho = \lambda/\mu$ — приведенная интенсивность, $\lambda = 0{,}2$ задачи в минуту — интенсивность входного потока заявок, $\mu = 1/15$ задачи в минуту — интенсивность обслуживания, $n = 4$ — число каналов обслуживания);

$$\bar{k} = \frac{\lambda}{\mu} = \rho$$

— среднее число занятых каналов обслуживания (ПЭВМ);

$$\bar{Q}_{\text{очер}} = p_0 \frac{\rho^{n+1}}{n! n \left(1 - \rho/n\right)^2}$$

— средняя длина очереди;

$$\bar{T}_{\text{очер}} = \frac{1}{\lambda} \bar{Q}_{\text{очер}}$$

— среднее время ожидания заявки (задачи) в очереди.

Подставляя заданные значения, получаем следующие результаты:

$$\bar{k} = 3; \quad \bar{Q}_{\text{очер}} \approx 1{,}5283; \quad \bar{T}_{\text{очер}} \approx 7{,}6415 \text{ мин.}$$

Исходный код модели и результаты решения представлены в листинге 3.

Одноканальная СМО с ограниченной очередью. В качестве примера рассмотрим следующую СМО. Одноканальная СМО обслуживает пуассоновский поток заявок, время между заявками – случайная величина, распределенная по показательному закону, среднее время между заявками составляет 10 с (λ = 0,1). Время обслуживания одной заявки также распределено по показательному закону, среднее время обслуживания 10 с ($\mu = 0{,}1$). Если заявка приходит в момент, когда канал занят, то она становится в очередь, максимальная длина очереди — 3 заявки, если заявка приходит в тот момент времени, когда в очереди уже находится 3 заявки, то она получает отказ в обслуживании. Требуется провести имитационное моделирование СМО в течение 100 ч, определить параметры СМО — вероятность отказа в обслуживании, параметры очереди задач (среднею длину очереди и среднее время ожидания задачи в очереди), загрузку канала обслуживания.

По аналогии с предыдущими примерами данная задача имеет аналитическое решение [4]. Так, в данной задаче $\rho = 1$, поэтому основные

```
#include <SimC.h>
int main()
{
PSTORAGE pS; // Указатель на многоканальное устройство
PQUEUE pQ; // Указатель на очередь
Start(1); // Задаем значение счетчика завершения  равным 1
NewStorage(pS, "Выч. сеть", 4); // Создаем многоканальное устройство
NewQueue(pQ, "Очередь");// Создаем очередь
InitGenerate(1, 0);// Инициализация генератора транзактов – задач
InitGenerate(8, 60*24*100);// Инициализация генератора транзактов -
                // Ограничиваем время моделирования: транзакт поступает через 100 суток
while(TG1) // Цикл обработки событий
{
        Plan(); // Протяжка модельного времени
        switch(SysEvent) // Обработка событий
        {
        case 1: Generate(RandExp(0.2, V2)); break; //Поступает транзакт – задача
        case 2: Queue(pQ); break; // Занимаем очередь
        case 3: Enter(pS); break; // Занимаем многоканальное устройство
        case 4: Depart(pQ); break;// Освобождаем очередь
        case 5: Advance(RandExp(1./15, V3)); break;// Задержка – решение задачи
        case 6: Leave(pS); break; // Освобождаем многоканальное устройство
        case 7: Terminate(); break;// Уничтожение транзакта – задачи
        case 8: Terminate(1); // Уничтожение транзакта
        }
}

PrintAll(); // Печать результатов моделирования
 return 1;
 }
```

```
 СИСТЕМНОЕ ВРЕМЯ          : SYSTIME=     144000.000
 СИСТЕМНОЕ СОБЫТИЕ        : SYSEVENT=  8
TRANS=NULL
 ВСЕГО ВЫПОЛНЕНО СОБЫТИЙ : EVENTALL=215254
 ЗАТРАЧЕНО ВРЕМЯ          : REALTIME=        0.17 СЕК
 СКОРОСТЬ ИСПОЛНЕНИЯ      : EVESPEED=1251476.744196 СОБ/СЕК
 СРЕДНЕЕ ВРЕМЯ ИСПОЛНЕНИЯ СОБЫТИЯ : MEANTIME=    0.000001 СЕК/СОБ

 СОБ.#  ВСЕГО    СОБ.#  ВСЕГО    СОБ.#  ВСЕГО    СОБ.#  ВСЕГО    СОБ.#  ВСЕГО
     1  28661         3  28661         5  28661         7  28659
     2  28661         4  28661         6  28659         8      1

                    ******************************
                    *                            *
                    *  -----   ОЧЕРЕДИ   -----  *
                    *                            *
                    ******************************

  ОЧЕРЕДЬ  ЧИСЛО  ВХОДОВ МАКС. ДЛИНА СРЕДН. ВР. ОЖ. СРЕДНЯЯ      % ВХОДОВ
           ------------  -----------  --------------   ДЛИНА      В ПУСТУЮ
           С 0 ВРЕМ.ОЖ.  ТЕКУЩ.ДЛИНА  БЕЗ УЧ. 0 ВХ.               ОЧЕРЕДЬ
  Очередь       28661           23            7.644      1.521     48.955
                14031            0           14.975

                    ******************************
                    *                            *
                    *  ----   НАКОПИТЕЛИ  ----  *
                    *                            *
                    ******************************

НАКОПИТЕЛЬ    ЕМКОСТЬ  ЗАГРУЗКА  СРЕДН. ВРЕМЯ  ТЕКУЩЕЕ  МАКС. СРЕДНЕЕ   ЧИСЛО
                                  ПРЕБЫВАНИЯ   СОДЕРЖ. СОДЕРЖ. СОДЕРЖ.  ВХОДОВ
Выч. сеть       4       0.749       15.057        2       4    2.997     28661
```

соотношения имеют следующий вид:

$$p_0 = p_1 = p_2 = \ldots = p_n = \frac{1}{n+1},$$

где $n = 4$ — максимальное число заявок в системе (максимальная длина очереди равна $n-1$); $P_{\text{отк}} = p_n$ — вероятность отказа в обслуживании; $\bar{Q}_{\text{очер}} = \bar{Q}_{\text{сист}} - (1-p_0) = p_0 \sum_{k=1}^{n} k\rho^k - (1-p_0)$ — средняя длина очереди; $\bar{T}_{\text{очер}} = \dfrac{\bar{Q}_{\text{очер}}}{(1-p_0)\,\mu}$ — среднее время ожидания в очереди; $K_{\text{загр}} = P_{\text{зан}} = 1 - p_0$ — коэффициент загрузки канала (вероятность того, что канал занят).

Подставляя заданные значения, получаем следующие результаты: $P_{\text{отк}} = 0{,}2$; $\bar{Q}_{\text{очер}} = 1{,}2$; $\bar{T}_{\text{очер}} = 15$ с; $K_{\text{загр}} = 0{,}8$.

Исходный код модели и результаты решения представлены в листинге 4.

Листинг 4

```
#include "SimC.h"
void main()
{
      // Одноканальная СМО с ограниченной очередью
      PFACILITY pF; // Указатель на канал обслуживания
      PQUEUE pQ; // Указатель на очередь
      int N_Otk=0, N=0; // Переменные для подсчета вероятности отказов
      Start(1); //  Создание системной среды, задаем значение
                // счетчика завершения
      // Создание модельной среды
      NewQueue(pQ, "Очередь"); // Создаем очередь
      NewFac(pF, "Канал"); // Создаем канал обслуживания
      InitGenerate(1, 0); // Транзакт-заявка направляется к первому событию
      InitGenerate(8, 3600*100); // Управляющий транзакт появляется в модели
                                 // через 100 часов
      while(TG1)// Конец моделирования, когда TG1 (счетчик завершения равен 0)
      {
            Plan();
            switch(SysEvent)
            {
            case 1:
                  Generate(RandExp(0.1, V1)); break; // Генерация заявок
            case 2:
                  N++;
                  if (pQ->LQ<3) // Если длина очереди меньше 3, то занимаем
                    Queue(pQ);
                  else { N_Otk++; // Транзакт получает отказ в обслуживании
                        Next(7);
                  }
                        break;
            case 3:
                  Seize(pF); break; // Занимаем канал обслуживания
            case 4:
                  Depart(pQ); break; // Освобождаем очередь
            case 5:
                  Advance(RandExp(0.1, V2)); break; // Задержка - время
                                                    // обслуживания
```

```
            case 6:
                  Release(pF); break; // Освобождаем канал обслуживания
            case 7:
                  Terminate(); break;// Транзакт заявка-уничтожается
            case 8:
                  Terminate(1); // Управляющий транзакт уменьшает счетчик
                                // завершения TG1 на 1
            }
      }
      PrintAll();
      fprintf(OUTFILE, "Вероятность отказа: %f", (double)N_Otk/N);
}
```

```
     СИСТЕМНОЕ ВРЕМЯ          : SysTime=     360000.000
     СИСТЕМНОЕ СОБЫТИЕ        : SYSEVENT=  8
    TRANS=NULL
     ВСЕГО ВЫПОЛНЕНО СОБЫТИЙ : EVENTALL=242231
     ЗАТРАЧЕНО ВРЕМЯ          : REALTIME=        0.23 СЕК
     СКОРОСТЬ ИСПОЛНЕНИЯ      : EVESPEED=1035175.213657 СОБ/СЕК
     СРЕДНЕЕ ВРЕМЯ ИСПОЛНЕНИЯ СОБЫТИЯ : MEANTIME=    0.000001 СЕК/СОБ

 СОБ.#  ВСЕГО    СОБ.#  ВСЕГО    СОБ.#  ВСЕГО    СОБ.#  ВСЕГО    СОБ.#  ВСЕГО
     1  35649        3  28515        5  28515        7  35645
     2  35649        4  28515        6  28514        8      1

                        ******************************
                        *                            *
                        *  -----   ОЧЕРЕДИ   -----   *
                        *                            *
                        ******************************

  ОЧЕРЕДЬ  ЧИСЛО  ВХОДОВ МАКС. ДЛИНА СРЕДН. ВР. ОЖ. СРЕДНЯЯ     % ВХОДОВ
           ------------  -----------  --------------   ДЛИНА     В ПУСТУЮ
           С 0 ВРЕМ.ОЖ.  ТЕКУЩ.ДЛИНА  БЕЗ УЧ. 0 ВХ.              ОЧЕРЕДЬ
  Очередь       28518            3          14.973      1.186    25.563
                 7290            3          20.114

                        ******************************
                        *                            *
                        *  ----    ПРИБОРЫ    ----   *
                        *                            *
                        ******************************

 ПРИБОР      ЧИСЛО       СРЕДНЕЕ ВРЕМЯ    ЗАГРУЗКА      ЧИСЛО      СОСТОЯНИЕ
             ВХОДОВ       ОБРАБОТКИ                   ЗАХВАТОВ
    Канал    28515             10.042       0.795          0       SEIZED

Вероятность отказа: 0.200034
```

Замкнутая СМО с одним каналом и m источниками заявок. Рассмотрим следующий пример СМО. Администратор обслуживает компьютерный зал, в котором находится 3 компьютера ($m = 3$). На работающем компьютере в среднем раз в час происходит сбой (время между сбоями распределено по экспоненциальному закону, $\lambda = 1/60$ сбоя в минуту). После сбоя администратор устраняет его последствия в среднем за 20 мин (время распределено по экспоненциальному закону, $\mu = 1/20$). Если сбой происходит, когда администратор устраняет последствия сбоя на другом компьютере, то компьютер становится в очередь на обслуживание. Требуется провести имитационное модели-

рование СМО в течение 20000 ч, определить параметры СМО — вероятность того, что все компьютеры работоспособны, вероятность того, что все компьютеры не работоспособны, загрузку канала обслуживания (администратора), параметры очереди (среднею длину очереди и среднее время ожидания в очереди).

По аналогии с предыдущими примерами данная задача имеет аналитическое решение [4]. Основные соотношения имеют следующий вид:

$p_0 = \dfrac{1}{\sum\limits_{k=0}^{m} \dfrac{\rho^k m!}{(m-k)!}}$ — вероятность того, что все компьютеры работоспособны; $p_i = p_0 \rho^i \dfrac{m!}{(m-i)!}$, $\forall i = 1, 2, \ldots, m$ — вероятность того, что i компьютеров не работоспособно; $K_{\text{загр}} = 1 - p_0$ — коэффициент загрузки канала (администратора); $\bar{Q}_{\text{очер}} = \bar{Q}_{\text{сист}} - (1 - p_0) = m - \dfrac{1 - p_0}{\rho} - (1 - p_0)$ — среднее число заявок (компьютеров) в очереди; $\bar{T}_{\text{очер}} = \dfrac{\bar{Q}_{\text{очер}}}{(1 - p_0)\,\mu}$ — среднее время нахождения заявки (компьютера) в очереди (определяем по формуле Литла).

Подставляя заданные значения, получаем следующие результаты: $p_0 \approx 0{,}346$; $p_m \approx 0{,}077$; $K_{\text{загр}} \approx 0{,}654$; $\bar{Q}_{\text{очер}} \approx 0{,}385$; $\bar{T}_{\text{очер}} \approx 11{,}7647$ мин.

Исходный код модели и результаты решения представлены в листинге 5.

Листинг 5

```
#include "SimC.h"
void main()
{
      // Замкнутая СМО с M источниками заявок
      PFACILITY pF;// Указатель на канал обслуживания
      PQUEUE pQ; // Указатель на очередь
      double T_0=0, T_M=0; // Считаем суммарную длительность, когда не было
                           //заявок и когда было M заявок
      double t1=0, t2; // Вспомогательные переменные
      Start(1); // Создание системной среды,TG1 равен 1
      // Создание модельной среды
      NewQueue(pQ, "Очередь"); // Очередь
      NewFac(pF, "Канал");// Канал обслуживания
      for(int i=0; i<3; i++) InitGenerate(1+3*i, 0);// Три транзакта из 3-х
                                    // источников направляем к событиям
      InitGenerate(15, 60*20000); // Управляющий транзакт входит в модель
                                  // через 20000 часов
      while(TG1) // Цикл обработки событий
      {
            Plan(); // Протяжка модельного времени
            switch(SysEvent)
            {
            case 1:
                  TRANS->PI[0]=2; break;// Номер события, куда должен
                                        // вернуться транзакт
            case 2:
                  Advance(RandExp(1./60, V1)); break;// Моделирование работы
```

```
                                                    // без сбоев
            case 3:
                  Next(9); break; // Отправляем транзакт в событие № 9
            case 4:
                  TRANS->PI[0]=5; break;// Номер события, куда должен
                                        //вернуться транзакт
            case 5:
                  Advance(RandExp(1./60, V2)); break;// Моделирование работы
                                                     // без сбоев
            case 6:
                  Next(9); break;// Отправляем транзакт в событие № 9
            case 7:
                  TRANS->PI[0]=8; break;// Номер события, куда должен
                                        // вернуться транзакт
            case 8:
                  Advance(RandExp(1./60, V3)); break;// Моделирование работы
                                                     // без сбоев
            case 9:
                  // Очередь перестает быть пустой
                  if (pQ->LQ==0 && pF->STATUS==FREE) T_0+=(SysTime-t1);
                  Queue(pQ); //Занимаем очередь
                  if (pQ->LQ==2) t2=SysTime; // В системе M заявок
                  break;
            case 10:
                  Seize(pF); break;// Занимаем канал обслуживания
            case 11:
                  if (pQ->LQ==2) T_M+=SysTime-t2; // Считаем время, когда в
                                                  // системе было M заявок
                  Depart(pQ); break;// Освобождаем очередь
            case 12:
                  Advance(RandExp(0.05, V4)); break;// Моделирование времени
                                                   // устранения сбоя
            case 13:
                  if (pQ->LQ==0) t1=SysTime; // В системе нет заявок
                  Release(pF); // Освобождаем канал обслуживания
                  break;
            case 14:
                  Next(TRANS->PI[0]); break;// Транзакт возвращается обратно
                                            // к соответствующему событию
            case 15:
                  if (pQ->LQ==0 && pF->STATUS==FREE) T_0+=(SysTime-t1);
                  if (pQ->LQ==2) T_M+=SysTime-t2;
                  Terminate(1);// Управляющий транзакт уменьшает TG1 на 1
            }
      }
      PrintAll();
      fprintf(OUTFILE, "P0=%f, P_m=%f", T_0/SysTime, T_M/SysTime);

}
```

```
     СИСТЕМНОЕ ВРЕМЯ          : SysTime=    1200000.000
     СИСТЕМНОЕ СОБЫТИЕ        : SYSEVENT=  15
    TRANS=NULL
     ВСЕГО ВЫПОЛНЕНО СОБЫТИЙ : EVENTALL=318793
     ЗАТРАЧЕНО ВРЕМЯ          : REALTIME=        0.31 СЕК
     СКОРОСТЬ ИСПОЛНЕНИЯ      : EVESPEED=1021772.435904 СОБ/СЕК
     СРЕДНЕЕ ВРЕМЯ ИСПОЛНЕНИЯ СОБЫТИЯ : MEANTIME=    0.000001 СЕК/СОБ

 СОБ.#  ВСЕГО    СОБ.#  ВСЕГО  СОБ.#  ВСЕГО  СОБ.#  ВСЕГО  СОБ.#  ВСЕГО
     1      1        4      1      7      1     10  39180     13  39179
```

```
    2   12987     5   13105     8   13090     11   39180     14   39179

    3   12986     6   13105     9   39180     12   39180     15   1
```

```
                          ****************************
                          *                          *
                          *  -----   ОЧЕРЕДИ   -----  *
                          *                          *
                          ****************************

 ОЧЕРЕДЬ  ЧИСЛО  ВХОДОВ МАКС. ДЛИНА СРЕДН. ВР. ОЖ. СРЕДНЯЯ      % ВХОДОВ
          ------------  -----------  --------------   ДЛИНА     В ПУСТУЮ
          С 0 ВРЕМ.ОЖ.  ТЕКУЩ.ДЛИНА  БЕЗ УЧ. 0 ВХ.              ОЧЕРЕДЬ
 Очередь       39180           2          11.706       0.382     52.940
               20742           0          24.874

                          ****************************
                          *                          *
                          *  ----    ПРИБОРЫ    ----  *
                          *                          *
                          ****************************

 ПРИБОР      ЧИСЛО        СРЕДНЕЕ ВРЕМЯ    ЗАГРУЗКА      ЧИСЛО       СОСТОЯНИ
             ВХОДОВ       ОБРАБОТКИ                    ЗАХВАТОВ
   Канал     39180             20.016       0.654         0        SEIZED

P0=0.346484, P_m=0.076342
```

Заключение. Полученные результаты при достаточно большом интервале времени моделирования близки к результатам, полученным на аналитических моделях. Таким образом, библиотека функций языка Си++, реализованная для различных операционных систем, пригодна для имитационного моделирования СМО.

СПИСОК ЛИТЕРАТУРЫ

1. К е л ь т о н В., Л о у А. Имитационное моделирование. Классика CS. 3-е изд. – СПб.: Питер; Киев: Изд. группа BHV, 2004. – 847 с.
2. М е д в е д е в Н. В., Б ы к о в А. Ю., Г р и ш и н Г. А. Имитационное моделирование систем массового обслуживания с использованием межплатформенной библиотеки функций языка Си++ // Вестник МГТУ им. Н.Э. Баумана. Сер. “Приборостроение”. – 2005. – № 4. – С. 85–93.
3. К у д р я в ц е в Е. М. GPSS World. Основы имитационного моделирования различных систем. – М.: ДМК Пресс, 2004. – 320 с.
4. К л е й н р о к Л. Теория массового обслуживания. – М.: Машиностроение, 1979. – 432 с.

Статья поступила в редакцию 7.12.2007

Алексей Михайлович Журавлев родился в 1981 г., окончил в 2003 г. Государственный университет управления. Специализируется в области безопасности компьютерных сетей.

A.M. Zhuravlev (b. 1981) graduated from the State University for Management in 2003. Specializes in the field of security of computer networks.

Николай Викторович Медведев родился в 1954 г., окончил в 1977 г. МГТУ им. Н.Э. Баумана. Канд. техн. наук, зав. кафедрой "Информационная безопасность" МГТУ им. Н.Э. Баумана. Автор около 50 научных работ в области исследования и разработки защищенных систем автоматической обработки информации.

N.V. Medvedev (b. 1954) graduated from the Bauman Moscow Higher Technical School in 1977. Ph. D. (Eng.), head of "Information Security" department of the Bauman Moscow State Technical University. Author of about 50 publications in the field of study and development of protected systems of automated data processing.

Александр Юрьевич Быков родился в 1969 г., окончил в 1991 г. ВИКИ им. А.Ф. Можайского. Канд. техн. наук, доцент кафедры "Информационная безопасность" МГТУ им. Н.Э. Баумана. Автор около 20 научных работ в области информационной безопасности и исследования систем обработки информации и управления.

A.Yu. Bykov (b. 1969) graduated from the Military Institute for Engineering and Space n.a. A.F. Mozhaiskii in 1991. Ph. D. (Eng.), assoc. professor of "Information Security" department of the Bauman Moscow State Technical University. Author of about 20 publications in the field of study and development of data security and study of systems of data processing and control.

УДК 004.7

С. С. Баскаков

РАСПРЕДЕЛЕННЫЙ АЛГОРИТМ АВТОМАТИЧЕСКОГО ВЫБОРА ОПОРНЫХ УЗЛОВ В БЕСПРОВОДНЫХ МНОГОЯЧЕЙКОВЫХ (MESH) СЕТЯХ

Предложен алгоритм автоматического выбора произвольного числа опорных узлов с различными видами их распределения по площади покрытия беспроводной многоячейковой сети. Полученные оценки сложности по времени и памяти показали, что алгоритм является оптимальным при совместном использовании с протоколами географической маршрутизации по виртуальным координатам.

В последнее время активное развитие получили технологии беспроводных сенсорных сетей, которые имеют множество практических применений, связанных с распределенным сбором, анализом и передачей информации. Беспроводная сенсорная сеть (БСС) представляет собой распределенную, самоорганизующуюся и устойчивую к отказу сеть миниатюрных электронных устройств, обменивающихся информацией по беспроводному каналу связи. Предполагается, что такие сети будут иметь многоячейковую (mesh) топологию и состоять из большого числа (до нескольких десятков тысяч) узлов, которые способны ретранслировать сообщения друг друга.