МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Системы параллельной обработки данных»

Тема: ИСПОЛЬЗОВАНИЕ ФУНКЦИЙ ОБМЕНА ДАННЫМИ «ТОЧКА-ТОЧКА» В БИБЛИОТЕКЕ MPI

Студент гр. 5304 _____ Лянгузов А.А.

Преподаватель _____ Татаринов Ю.С.

Санкт-Петербург

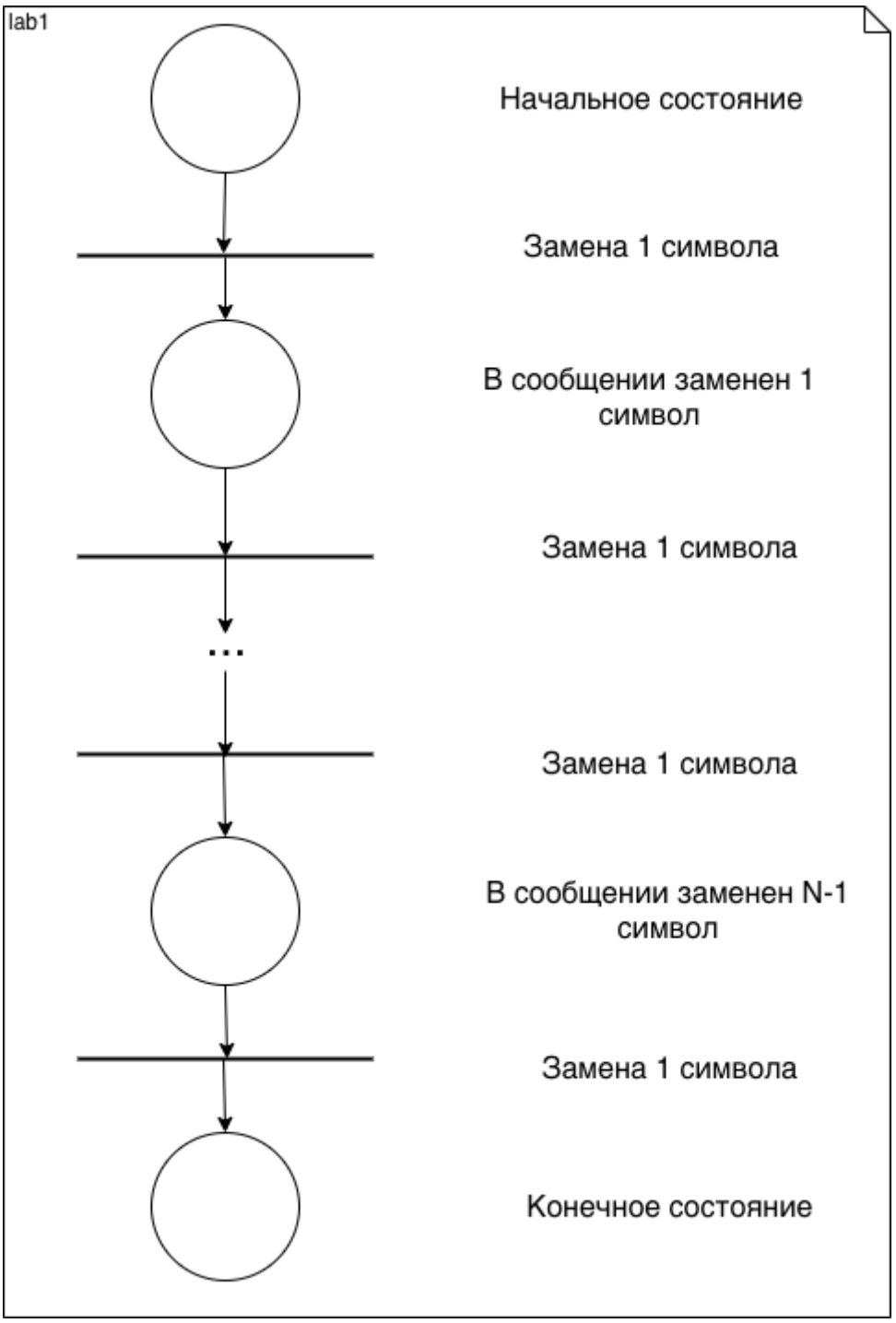
2019

Задание 1. Испорченный телефон :

Процесс 0 генерирует строковое сообщение и передает его процессу со следующим номером. Процесс-получатель случайным образом меняет в сообщении один символ и передает его дальше. Последний процесс передает получившийся результат «ведущему».

Сеть Петри.

Рассматривается последовательный процесс передачи сообщения. Исходя из этого, нулевой процесс передаёт строку первому. Первый процесс меняет строку и передаёт второму. И так N-1 раз (N – количество процессов). Следовательно, сеть Петри будет выглядеть так:



Выполнение работы

За генерацию сообщения отвечает нулевой процесс. Также он ловит конечное сообщение из последнего процесса. Генерируется сообщение заданной длины состоящее из нулей.

Ненулевые процессы ловят сообщение от предыдущего процесса, меняют символ нуля на символ единицы в произвольном месте и отсылают это следующему процессу (последний процесс отсылает это нулевому).

```
(base) xtail@~/Projects/LETI/Parallel/ParallelLabs$ ./assembly.sh launch lab1 4
Child process modified string: 0000000000000000000000000000000000000000000000000000000000000000000000000000100000000000000000000000
Child process modified string: 0000000000000000000000000000000000100000000000000000000000000000000000000000100000000000000000000000
Child process modified string: 0000000000000000000000000000000000100000000000000000000000000000000000000000110000000000000000000000
Result:  0000000000000000000000000000000000100000000000000000000000000000000000000000110000000000000000000000
```

Рис.1. Результат работы программы

Выводы

В ходе выполнения лабораторной работы ознакомились с процессами приёма и передачи сообщений на MPI путем преобразования простейшей программы.

ПРИЛОЖЕНИЕ А.
КОД ПРОГРАММЫ

```
#include <stdio.h>
#include "mpi.h"
#include <stdlib.h>

#define string char*

string makeString(const int len) {
   string s = (char*) malloc(len*sizeof(char));
   if (s == NULL) {
       printf("Allocation error.");
       MPI_Finalize();
       exit(0);
   }

   for(int i = 0; i < len; i++)
   {
       s[i] = '0';
   }

   return s;
}

int main(int argc, char* argv[])
{
   int ProcNum, ProcRank, RecvRank;
   int waiter = 0;

   MPI_Status Status;
   MPI_Init(&argc, &argv);

   //declare size of processes (group id, group size(return))
   MPI_Comm_size(MPI_COMM_WORLD, &ProcNum);

   int len = 100;
   string str = makeString(len);
```

```
    int rands[ProcNum];
    for(int i = 0; i < ProcNum; i++)
    {
        rands[i] = rand() % (len - 1);
    }

    //define process rank in groud (group id, rank(return))
    MPI_Comm_rank(MPI_COMM_WORLD, &ProcRank);
    if ( ProcRank == 0 )
    {
        // (buffer, buffer size, buffer data type, receiver process number,
message tag, group id)
        MPI_Send(str, len, MPI_CHAR, 1, 0, MPI_COMM_WORLD);

        // (buffer, buffer size, buffer data type, sender number,message
tag, group id,status)
        MPI_Recv(str, len, MPI_CHAR, ProcNum - 1, MPI_ANY_TAG,
MPI_COMM_WORLD, &Status);
        printf("Result:  %3s \n", str);
    }
    else
    {
        MPI_Recv(str, len, MPI_CHAR, ProcRank - 1, MPI_ANY_TAG,
MPI_COMM_WORLD, &Status);

        int tmp = ProcRank + 1;
        str[rands[ProcRank - 1]] = '1';
        printf("Child process modified string: %3s \n", str);

        tmp >= ProcNum ?
            MPI_Send(str, len, MPI_CHAR, 0, 0, MPI_COMM_WORLD) :
            MPI_Send(str, len, MPI_CHAR, tmp, 0, MPI_COMM_WORLD);
    }

    MPI_Finalize(); //parallel part of app finish

    free(str);
    return 0;}
```