Учреждение образования

«Белорусский государственный технологический университет»

Кафедра программной инженерии

Лабораторная работа №1

«Синхронизация времени»

Выполнил:

Студент 3 курса 4 группы ФИТ

Карленок Юрий

2020 г.

**Цель**

Изначально в данной лабораторной работе были созданы клиент и сервер, способные синхронизировать время на удаленных машинах. Для этого использовались клиенты и параллельный сервер, способный обслуживать несколько клиентов одновременно.

Второй частью является синхронизация времени клиента и сервера с помощью протокола NTP.

**Синхронизация часов (I)**

Клиент периодически шлет серверу запросы следующей структуры.

struct SYNC

{

SYNC() {}

SYNC(string command, int correction)

{

this->command = command;

this->correction = correction;

}

string command;

int correction;

};

Периодичность клиентского запроса ***Tc*** является параметром программы-клиента и задается в тиках, равных **1/1000 сек**.

Сервер без задержки отвечает клиенту на каждый запрос ответом, имеющим следующую структуру.

struct SYNC

{

SYNC() {}

SYNC(string command, int correction)

{

this->command = command;

this->correction = correction;

}

string command;

int correction;

};

Для каждого эксперимента принимается устанавливается значение ***Tc***. В каждом эксперименте клиент должен делать по 10 запросов

**Код структуры Сервера**

SOCKET serverSocket;

WSADATA wsaData;

if (WSAStartup(MAKEWORD(2, 0), &wsaData) != 0)

throw DefineError("Startup: ", WSAGetLastError());

if ((serverSocket = socket(AF\_INET, SOCK\_DGRAM, NULL)) == INVALID\_SOCKET)

throw DefineError("Socket: ", WSAGetLastError());

SOCKADDR\_IN serverConfig;

serverConfig.sin\_family = AF\_INET;

serverConfig.sin\_port = htons(2000);

serverConfig.sin\_addr.s\_addr = INADDR\_ANY;

if (bind(serverSocket, (LPSOCKADDR)&serverConfig, sizeof(serverConfig)) == SOCKET\_ERROR)

throw DefineError("Bind Server: ", WSAGetLastError());

SYNC receivedData, sentData("SYNC", 0);

SYSTEMTIME systemTime;

int averageResults[10];

clock\_t delta;

int i = 1;

const int MAX\_ITERATION\_COUNT = 11;

for (int i = 1; i != MAX\_ITERATION\_COUNT; i++)

{

SOCKADDR\_IN clientConfig;

int clientConfigLength = sizeof(clientConfig);

int average = 0;

GetSystemTime(&systemTime);

int recvBytes = recvfrom(serverSocket, (char\*)&receivedData, sizeof(receivedData), NULL, (sockaddr\*)&clientConfig, &clientConfigLength);

if (recvBytes == SOCKET\_ERROR)

throw DefineError("Receive: ", WSAGetLastError());

delta = clock();

int correctionDelta = delta - receivedData.correction;

sentData.correction = correctionDelta;

averageResults[i - 1] = correctionDelta;

average = GetAverageCorrection(averageResults, i);

int sentBytes = sendto(serverSocket, (char\*)&sentData, sizeof(sentData), 0, (sockaddr\*)&clientConfig, sizeof(clientConfig));

if (sentBytes == SOCKET\_ERROR)

throw DefineError("Send: ", WSAGetLastError());

cout << "DataTime "

<< systemTime.wMonth << "."

<< systemTime.wDay << ".2018" << " " << endl

<< systemTime.wHour + 3 << " Hours "

<< systemTime.wMinute << " Minuts "

<< systemTime.wSecond << " Seconds "

<< systemTime.wMilliseconds << " MiliSeconds " << endl

<< inet\_ntoa(clientConfig.sin\_addr) << " Correction = "

<< sentData.correction << ", Average Correction = "

<< average << endl;

}

if (closesocket(serverSocket) == SOCKET\_ERROR)

throw DefineError("close socket: ", WSAGetLastError());

if (WSACleanup() == SOCKET\_ERROR)

throw DefineError("Cleanup: ", WSAGetLastError());

**Код структуры клиента**

SOCKET clientSocket;

WSADATA wsaData;

if (WSAStartup(MAKEWORD(2, 0), &wsaData) != 0)

throw DefineError("Startup: ", WSAGetLastError());

if ((clientSocket = socket(AF\_INET, SOCK\_DGRAM, NULL)) == INVALID\_SOCKET)

throw DefineError("Socket: ", WSAGetLastError());

SOCKADDR\_IN serverConfig;

serverConfig.sin\_family = AF\_INET;

serverConfig.sin\_port = htons(2000);

serverConfig.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

SYSTEMTIME systemTime;

SYNC receivedData, sentData("SYNC", 0);

const int TC = 9000;

int serverConfigLength = sizeof(serverConfig);

int maxCorrection = INT\_MIN;

int minCorrection = INT\_MAX;

int averageCorrection = 0;

const int MAX\_ITERATION\_COUNT = 10;

int sentBytes = sendto(clientSocket, (char\*)&sentData, sizeof(sentData), 0, (sockaddr\*)&serverConfig, sizeof(serverConfig));

if (sentBytes == SOCKET\_ERROR)

throw DefineError("Send: ", WSAGetLastError());

int receivedBytes = recvfrom(clientSocket, (char\*)&receivedData, sizeof(receivedData), 0, (sockaddr\*)&serverConfig, &serverConfigLength);

if (receivedBytes == SOCKET\_ERROR)

throw DefineError("Receive: ", WSAGetLastError());

sentData.correction += receivedData.correction;

for (int i = 0; i < MAX\_ITERATION\_COUNT; i++)

{

GetSystemTime(&systemTime);

sendto(clientSocket, (char\*)&sentData, sizeof(sentData), 0, (sockaddr\*)&serverConfig, sizeof(serverConfig));

recvfrom(clientSocket, (char\*)&receivedData, sizeof(receivedData), 0, (sockaddr\*)&serverConfig, &serverConfigLength);

if (maxCorrection < receivedData.correction)

maxCorrection = receivedData.correction;

if (minCorrection > receivedData.correction)

minCorrection = receivedData.correction;

cout << " DataTime " << systemTime.wMonth

<< "." << systemTime.wDay

<< ".2018" << endl << " Hours "

<< systemTime.wHour + 3

<< ":" << systemTime.wMinute

<< ":" << systemTime.wSecond

<< ":" << systemTime.wMilliseconds << " " << endl

<< i + 1 << " " << sentData.correction

<< " curvalue = " << receivedData.correction

<< " max correction / min correction: "

<< maxCorrection << "/" << minCorrection << endl << endl << endl;

sentData.correction += receivedData.correction + TC;

averageCorrection += receivedData.correction;

Sleep(TC);

}

cout << "Average correction: " << averageCorrection / MAX\_ITERATION\_COUNT << endl;

if (closesocket(clientSocket) == SOCKET\_ERROR)

throw DefineError("Closesocket: ", WSAGetLastError());

if (WSACleanup() == SOCKET\_ERROR)

throw DefineError("Cleanup: ", WSAGetLastError());

}

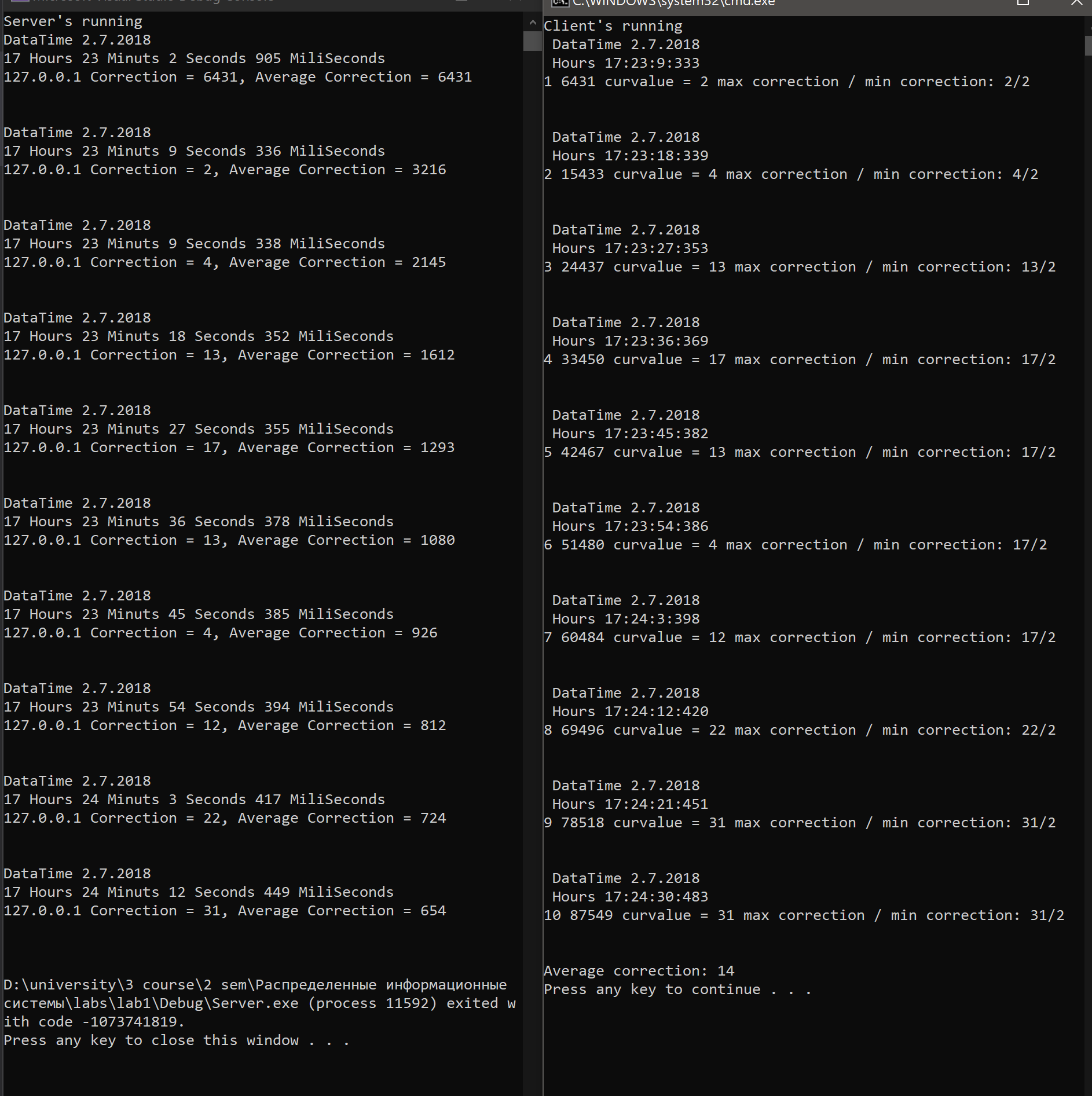


Схема работы сервера:

