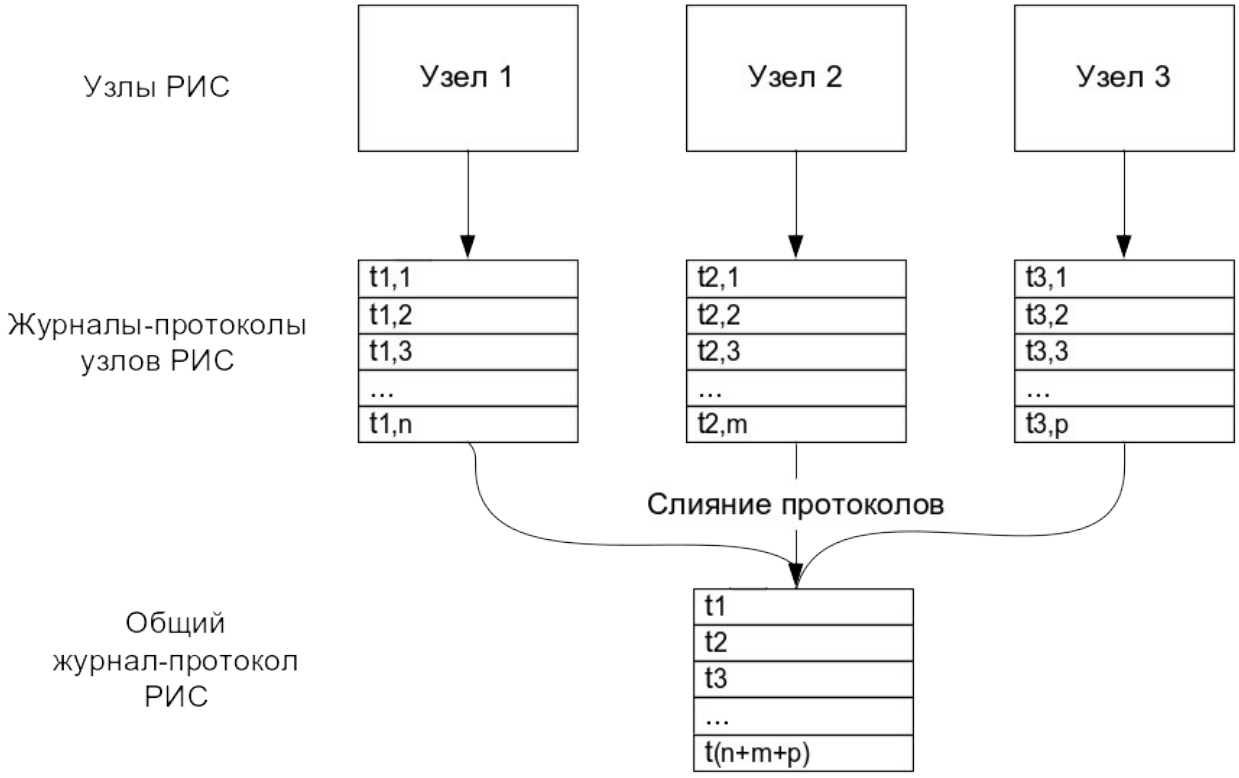
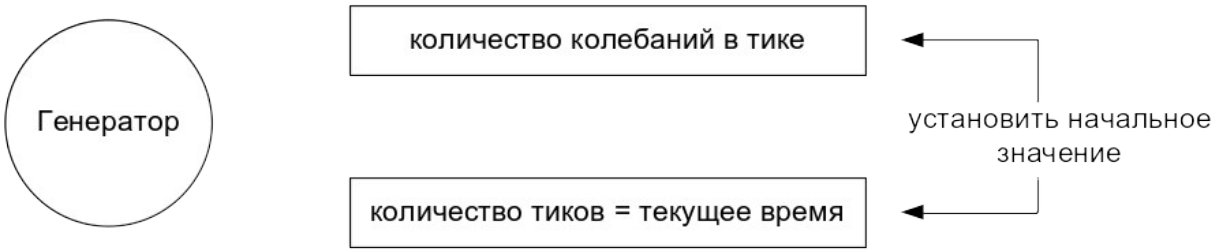
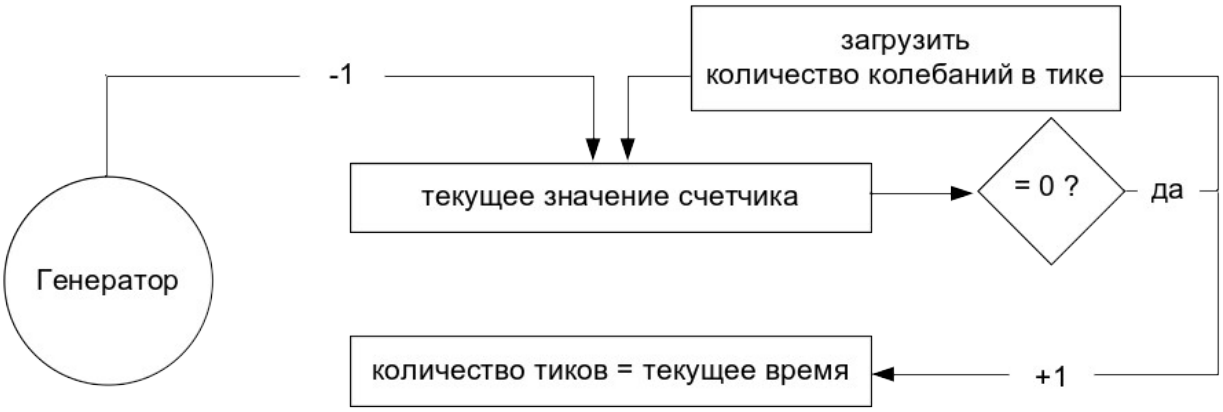
Синхронизация часов

1. Постановка задачи

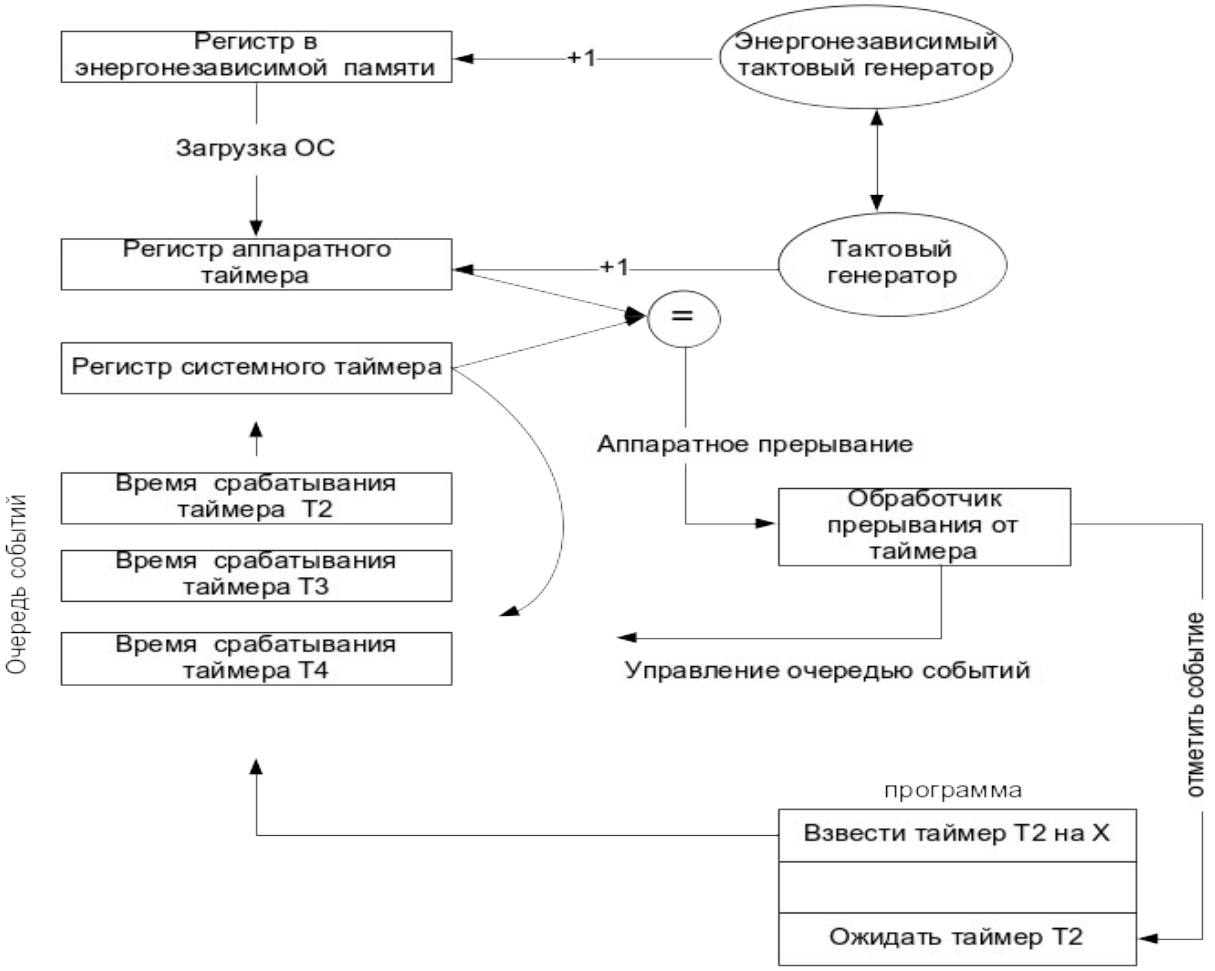


2. **Физические часы на компьютере, таймер:** кристалл кварца, колеблется с постоянной частотой, два регистра-счетчика: подсчет колебаний, подсчет тиков времени.





3. **Ожидающие таймеры**



4. **Социальное время не монотонное:** каждый год солнечный год увеличивается на 3 мс, люди измеряют время не точно, поэтому время от времени делались коррекции, например: в 1582 Папа Григорий XIII пропустил 10 дней календаря. Вычисление даты от Рождества по секундам не получится. Поэтому **эпоха Unix (POSIX-время)** с

01.01.1970 0:00:00 в секундах. Используется 32 бита для представления числа. В 2038 г. счетчик перейдет в область отрицательных чисел. Високосная секунда (повторение последней секунды в каждой минуте).Секунда координация (серверы точного времени): последняя секунда 30.06 или 31.12.

5. **Universal Coordinated Time (UCT):** универсальное согласованное время (на Гринвичском меридиане, раньше GMT – Greenwich Meridian Time), Международное бюро мер и весов (Париж), усредненное значение полученное на основе данных 50 лабораторий, оборудованных атомными часами (цезий-133) - TAI (International Atomic Time), расхождение с солнечными часами примерно 3мс (атомные часы отстают) в сутки, коррекция при ошибке в 800 мс.

6. **Коротковолновые радиостанции с позывным WWV:** выдают сигнал в начале каждой секунды UTC с точностью до ±10мс.

7. **Спутники Geostationary Environment Operational Satellite (GEOS):** выдают сигнал в начале каждой секунды UTC с точностью до ±5мс.

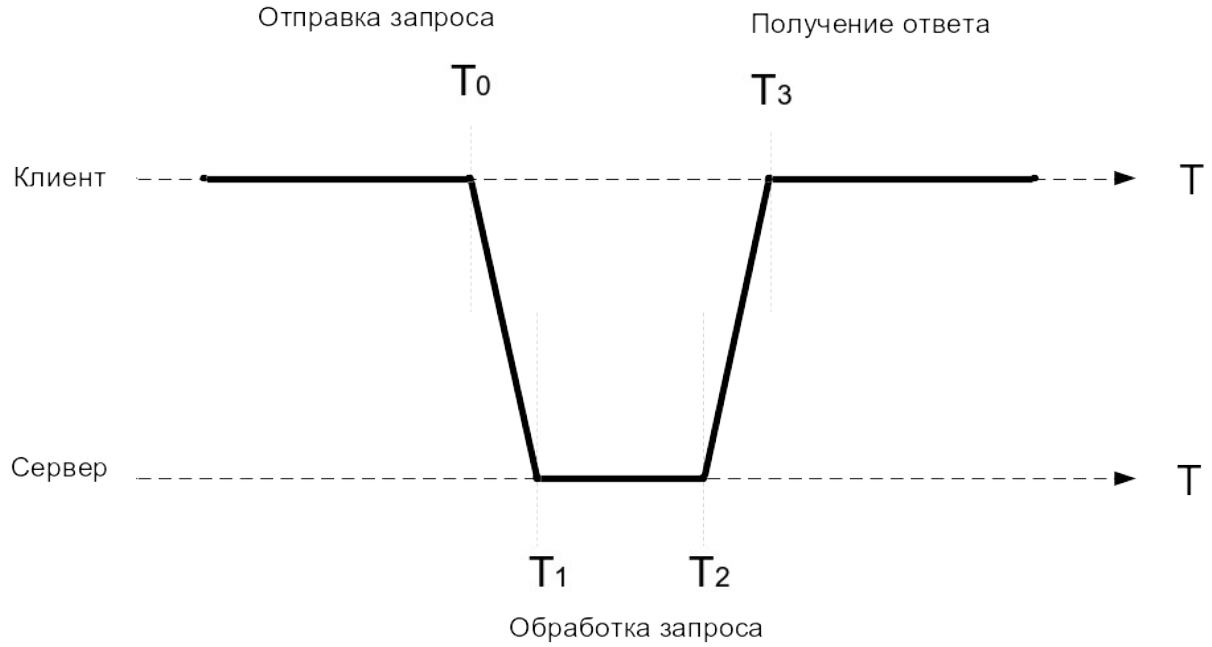
8. **Принципы алгоритмов синхронизации**

t - точное время;
C(t) – время на компьютере;
dt – интервал времени;
(C(t+dt)-C(t))/dt = 1 – часы точные;
(C(t+dt)-C(t))/dt > 1 – часы спешат;
(C(t+dt)-C(t))/dt < 1 – часы отстают;
p = |1-(C(t+dt)-C(t))/dt| - дрейф.

9. **Пример:** пусть 3 компьютера и сервер синхронизации, требуется их синхронизировать с точностью r на время dt: |C(t+dt)-(t+dt)| < r, вычислить p = MAX(p1, p2, p3); периодичность синхронизации r/(2p) с.

10. **Пример:** dt = 1000c, r = 0.5c, p1 = 0.1/1000, p2=0.2/1000, p3=0.05/1000, p = 0.2/1000, период = 0.5/(2*0.2/1000) = (0.5/0.4)*1000 = 1250c.
Требуется синхронизироваться с сервером каждые 1250c.

11. **Алгоритм Кристиана (Cristian):** учитывает время прохождения ответа. Оценить поправку (статистически) и учесть в полученной отсечке времени.



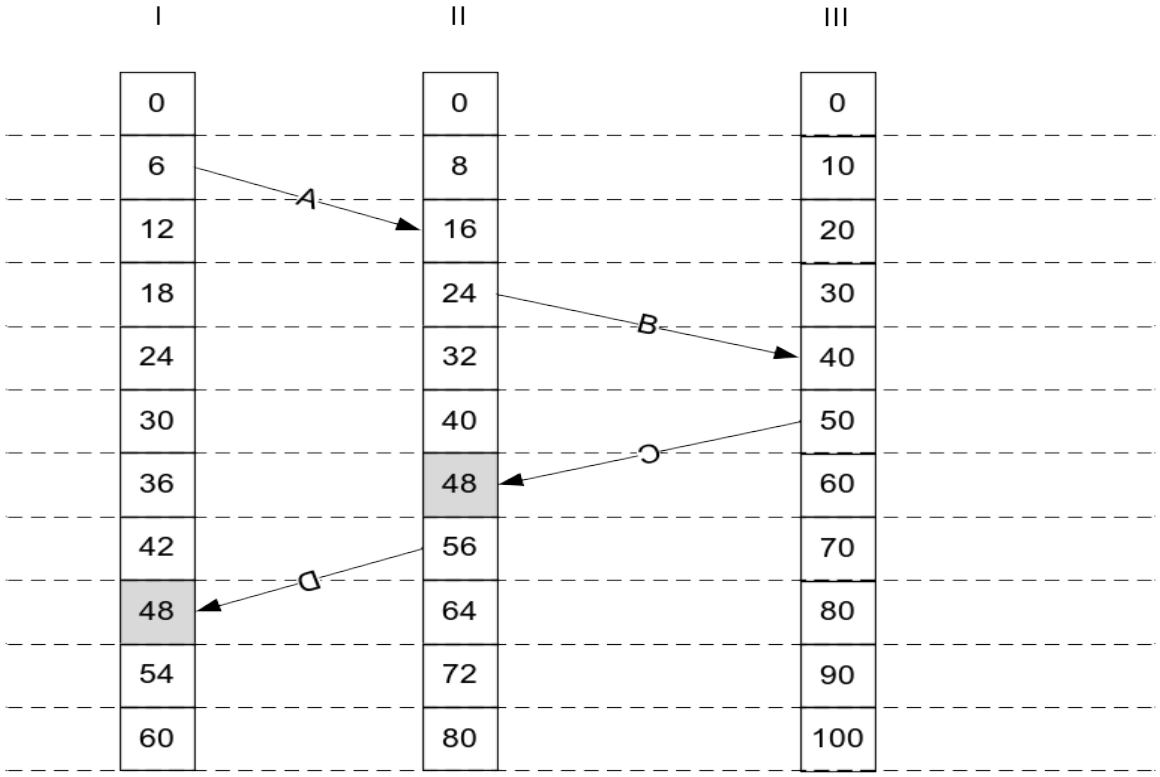
12. В большинстве случаев в РИС не требуется точное астрономическое время, достаточно, чтобы часы на всех компьютерах РИС не были рассинхронизированы более чем на заданную величину T.

13. **Алгоритм Беркли (Berkeley, UNIX):** применяется, если нет точных часов, но несколько машин надо синхронизировать. На машинах агент (клиент/сервер), который регистрируется на сервере времени; с некоторой периодичностью сервер опрашивает агентов и получает текущее время на каждом компьютере; усредняет время и раздает его агентам для установки на каждом компьютере. Основной недостаток: централизованный алгоритм.

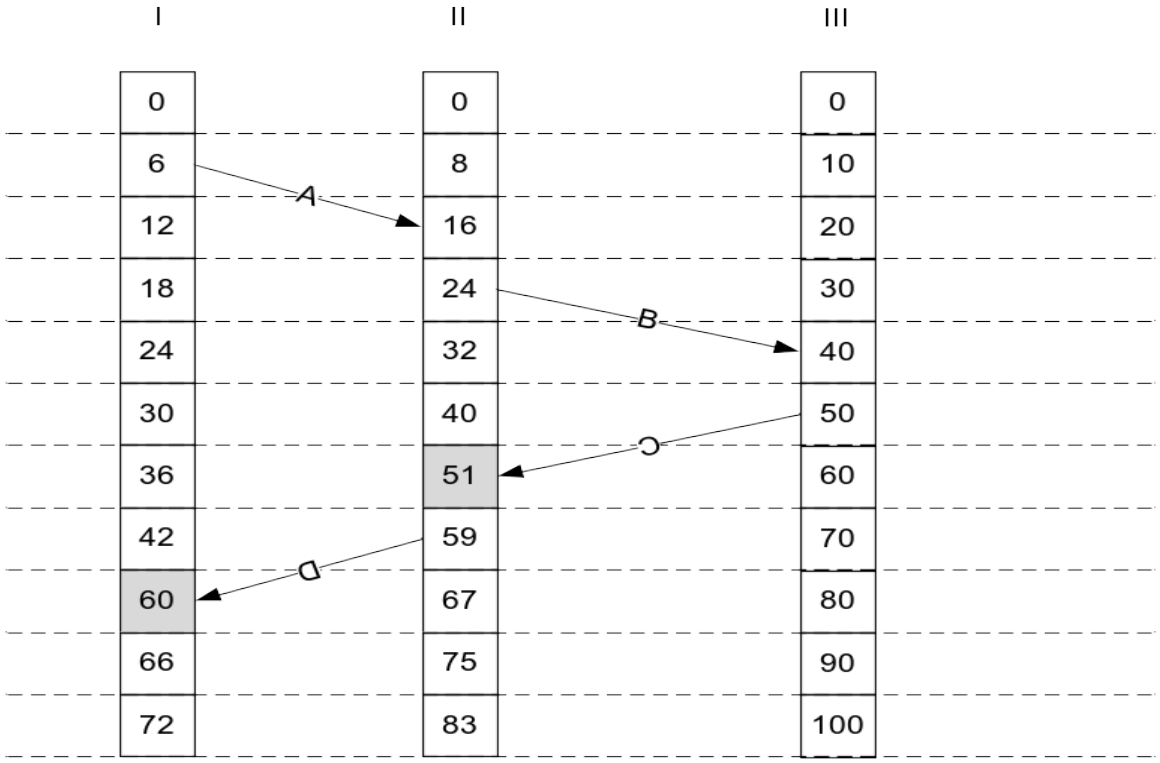
14. **Усредняющий алгоритм:** есть n компьютеров, которые будут синхронизировать время; с интервалом T одновременно, все они рассылают широковещательные сообщения со своим временем; часы у всех не точно синхронизированы, поэтому все они будут отправлены в разное время; на каждом компьютере после отправки сообщения, за определенное время S обрабатываются

пакеты от других компьютеров и засекается время рассогласования с учетом поправки на прохождение в сети сообщения, вычисляется среднее время на каждом компьютере.

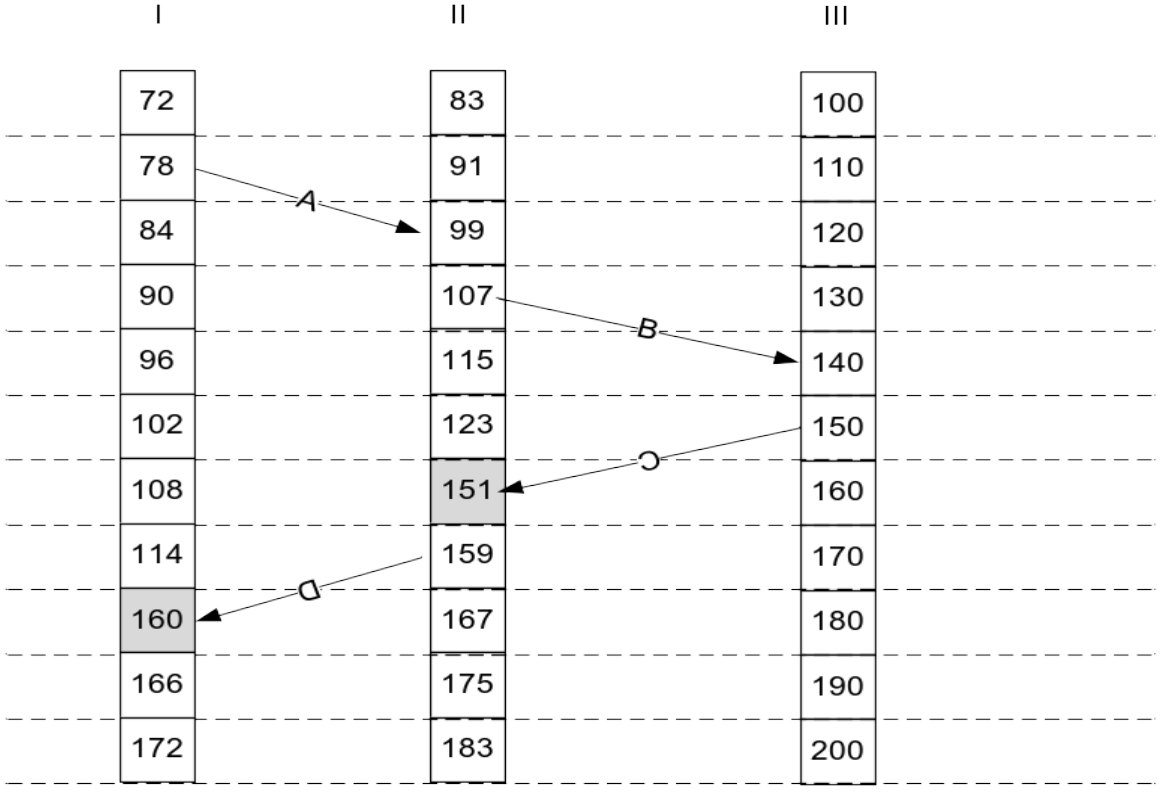
15. Алгоритм **Лампорта**



20, 20

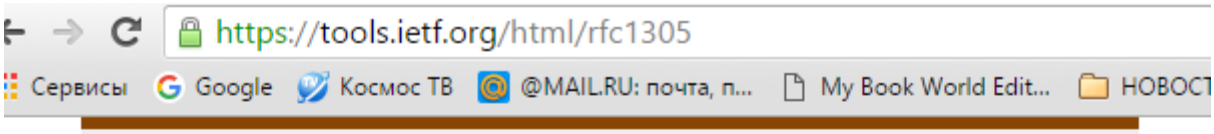


11,17



11,17

16. **Серверы времени (протокол NTP/SNTP)**



[Docs] [txt|pdf] [draft-mills-ntpv3] [Diff1] [Diff2]

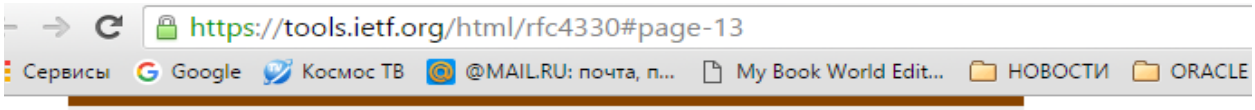
Obsoleted by: 5905 DRAFT STANDARD

Network Working Group
Request for Comments: 1305
Obsoletes RFC-1119, RFC-1059, RFC-958

David L. Mills
University of Delaware
March 1992

Network Time Protocol (Version 3)
Specification, Implementation and Analysis

Note: This document consists of an approximate rendering in ASCII of the PostScript document of the same name. It is provided for convenience and for use in searches, etc. However, most tables, figures, equations and captions have not been rendered and the pagination and section headings



[Docs] [txt|pdf] [draft-mills-sntp-v4] [Diff1] [Diff2] [Errata]

Obsoleted by: 5905 INFORMATIONAL
Errata Exist

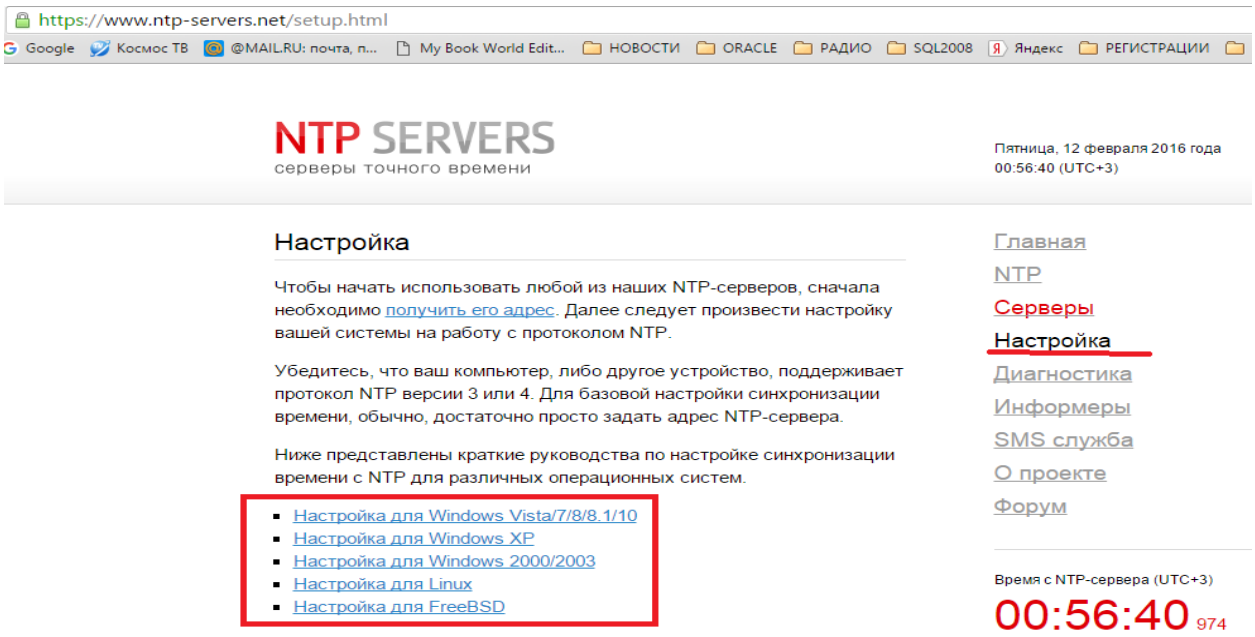
Network Working Group
Request for Comments: 4330
Obsoletes: 2030, 1769
Category: Informational

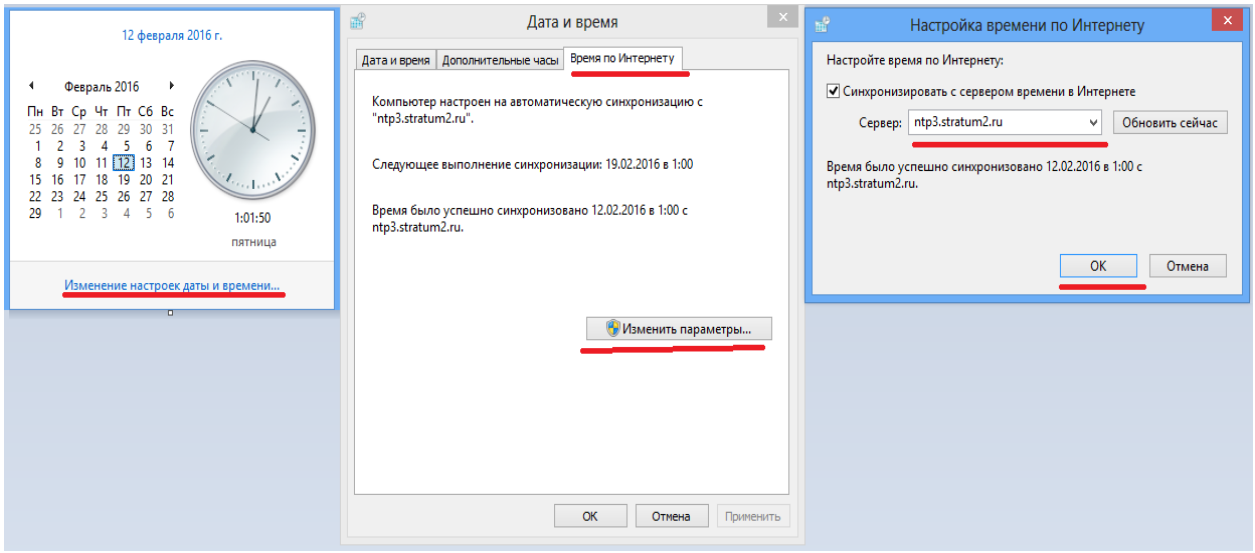
D. Mills
University of Delaware
January 2006

Simple Network Time Protocol (SNTP) Version 4
for IPv4, IPv6 and OSI

Status of This Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

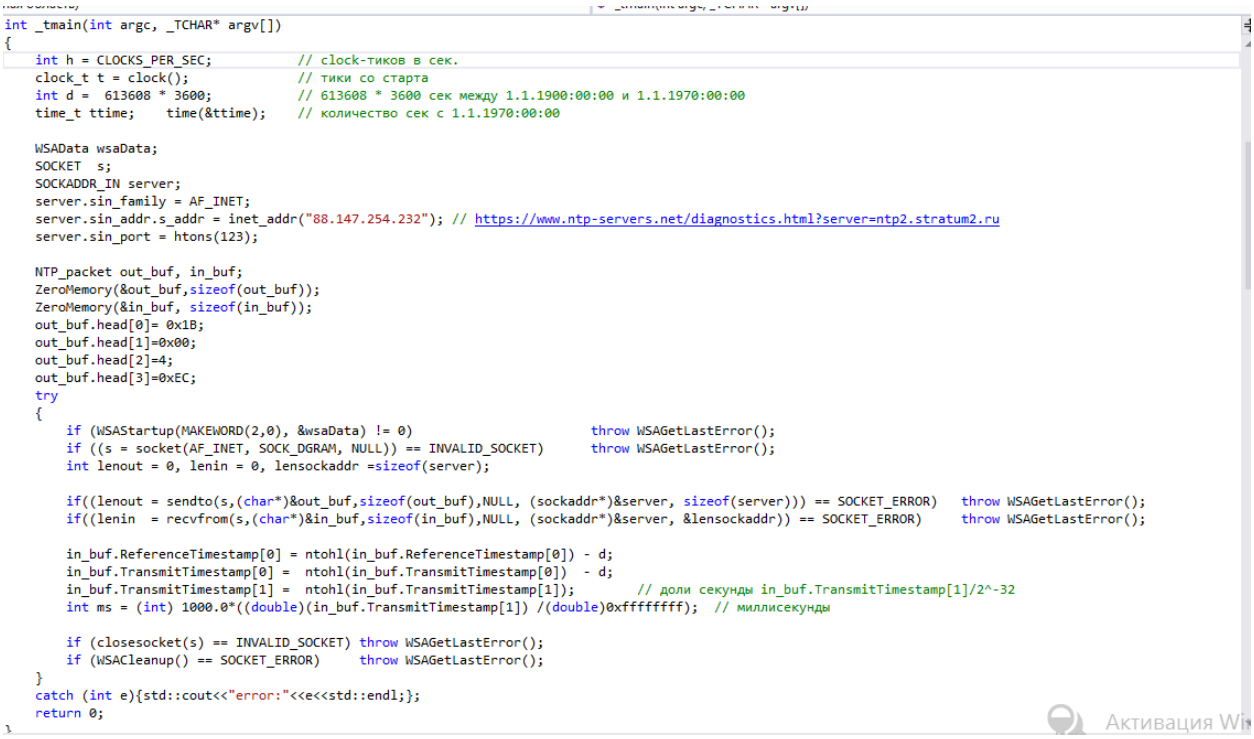




17. **Протокол NTP**

```
#include "stdafx.h"
#include <iostream>
#include <ctime>     //  для clock(), time()
#include "Winsock2.h"
#pragma comment(lib, "WS2_32.lib")

//   https://tools.ietf.org/html/rfc4330
struct NTP_packet                        // пакет   NTP
{
    CHAR     head[4];
    DWORD32  RootDelay;
    DWORD32  RootDispersion;
    CHAR     ReferenceIdentifier[4];    // идентификатор
    DWORD    ReferenceTimestamp[2];     // 32 бита - секунды с   01.01.1900 00:00,  32 бита - доли секунды в 2^-32 единицах
    DWORD64  OriginateTimestamp;
    DWORD32  TransmitTimestamp[2];      //32 бита - секунды с   01.01.1900 00:00,  32 бита - доли секунды в 2^-32 единицах
    DWORD32  KeyIdentifier;              //optional
    DWORD64  MessageDigest[2];           //optional
};
```



```
int _tmain(int argc, _TCHAR* argv[])
{
    int h = CLOCKS_PER_SEC;           // clock-тиков в сек.
    clock_t t = clock();              // тики со старта
    int d =  613608 * 3600;           // 613608 * 3600 сек между 1.1.1900:00:00 и 1.1.1970:00:00
    time_t ttime;    time(&ttime);    // количество сек с 1.1.1970:00:00

    WSAData wsaData;
    SOCKET  s;
    SOCKADDR_IN server;
    server.sin_family = AF_INET;
    server.sin_addr.s_addr = inet_addr("88.147.254.232"); // https://www.ntp-servers.net/diagnostics.html?server=ntp2.stratum2.ru
    server.sin_port = htons(123);

    NTP_packet out_buf, in_buf;
    ZeroMemory(&out_buf,sizeof(out_buf));
    ZeroMemory(&in_buf, sizeof(in_buf));
    out_buf.head[0]= 0x1B;
    out_buf.head[1]=0x00;
    out_buf.head[2]=4;
    out_buf.head[3]=0xEC;
    try
    {
        if (WSAStartup(MAKEWORD(2,0), &wsaData) != 0)                    throw WSAGetLastError();
        if ((s = socket(AF_INET, SOCK_DGRAM, NULL)) == INVALID_SOCKET)   throw WSAGetLastError();
        int lenout = 0, lenin = 0, lensockaddr =sizeof(server);

        if((lenout = sendto(s,(char*)&out_buf,sizeof(out_buf),NULL, (sockaddr*)&server, sizeof(server))) == SOCKET_ERROR)   throw WSAGetLastError();
        if((lenin  = recvfrom(s,(char*)&in_buf,sizeof(in_buf),NULL, (sockaddr*)&server, &lensockaddr)) == SOCKET_ERROR)    throw WSAGetLastError();

        in_buf.ReferenceTimestamp[0] = ntohl(in_buf.ReferenceTimestamp[0]) - d;
        in_buf.TransmitTimestamp[0] =  ntohl(in_buf.TransmitTimestamp[0])  - d;
        in_buf.TransmitTimestamp[1] =  ntohl(in_buf.TransmitTimestamp[1]);        // доли секунды in_buf.TransmitTimestamp[1]/2^-32
        int ms = (int) 1000.0*((double)(in_buf.TransmitTimestamp[1]) /(double)0xffffffff);  // миллисекунды

        if (closesocket(s) == INVALID_SOCKET) throw WSAGetLastError();
        if (WSACleanup() == SOCKET_ERROR)     throw WSAGetLastError();
    }
    catch (int e){std::cout<<"error:"<<e<<std::endl;};
    return 0;
}
```


18. **Как с помощью СУБД построить логические глобальные часы?**