

Основы программирования сетевых приложений

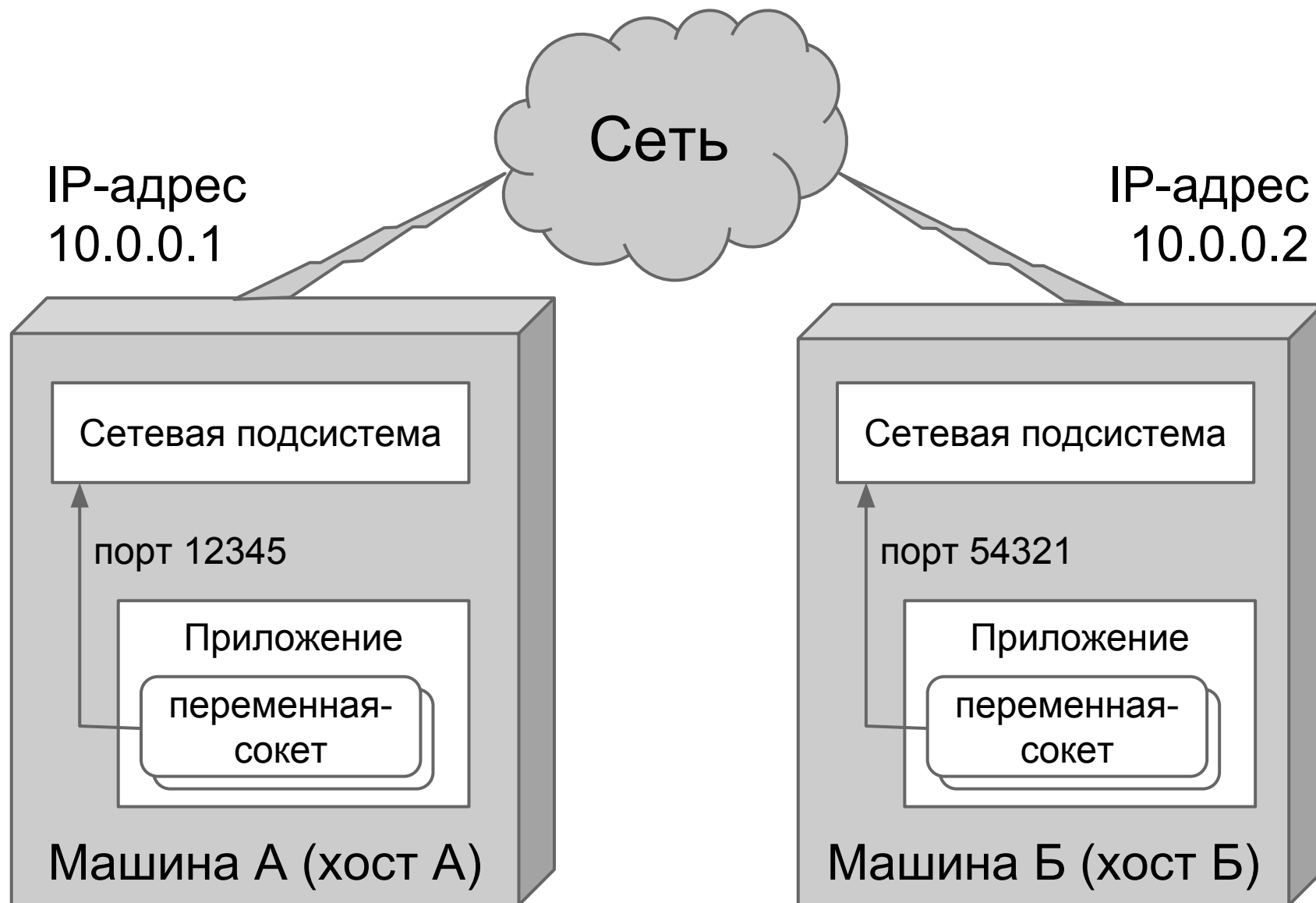
Курс «Информационные сети и телекоммуникации»

весенний семестр 2016 г.

кафедра Управления и информатики НИУ «МЭИ»

Сокет (socket)

- Устройство для связи в системе.
- Дескриптор такого устройства в программе.
- Сокет **НЕ** является:
 - соединением (*его может не быть*);
 - файлом (*нельзя закрыть его и открыть заново, перемотать, узнать длину*);
 - общей сущностью на связывающихся узлах
(*2 системы: в каждой 1 сокет и 1 программа, в которой 1 дескриптор для этого сокета*).



Адрес узла (host) и порт (port)

- ❑ Хост идентифицируется адресом IP.
 - ❑ Адрес IP: 4 байта, записываются через точку
 - ❑ Особые адреса IP:
 - ❑ 0.0.0.0 — текущий (loopback)
`INADDR_ANY` (0x00000000),
 - ❑ 255.255.255.255 — для рассылки всем узлам (broadcast)
`INADDR_BROADCAST` (0xFFFFFFFF).
- ❑ У каждого узла 65536 портов
 - ❑ Порт идентифицирует приложение.
 - ❑ Порты 1 — 1023 зарезервированы для системы.
 - ❑ Сокет связан только с одним портом.
- ❑ Данные передаются между точками **(хост, порт)**.

Простейшая схема работы

- 1) Создать сокет.
- 2) Передать данные:
 - а) отправить:
 - указать адрес получателя,
 - выполнить отправку;
 - б) принять:
 - указать собственный адрес,
 - выполнить прием,
 - *(можно)* определить адрес отправителя.
- 3) Заккрыть сокет.

Интерфейс программирования

- Berkley Sockets (BSD sockets), 1989 г. — набор функций в BSD UNIX 4.2.
- POSIX sockets (UNIX, Linux, Mac — *nix) — стандарт на интерфейс программирования:
 - Является «наследником» BSD sockets.
 - Используется не только для сетей.
- Windows Sockets (winsock):
 - Не полностью соответствует стандарту POSIX.
 - Много дополнительной функциональности (элементы WSA*), связанной с Windows API.

Создание сокета

Тип возвращаемого значения: UNIX, Linux, Mac — `int`
Windows — `SOCKET`



```
auto channel = socket(  
    AF_INET, SOCK_DGRAM, IPPROTO_UDP);
```

The diagram illustrates the components of the `socket` function call. An arrow points from the text 'Тип возвращаемого значения' to the `auto` keyword. Three arrows point upwards from the parameter descriptions to the arguments in the function call: `AF_INET`, `SOCK_DGRAM`, and `IPPROTO_UDP`.

Семейство адресов:

в каких сетях
расположены узлы,
будут задаваться их
адреса

Вид сокета:

как будет вестись
работа с сокетом
из программы

Протокол:

как система будет
передавать данные
по сети

Семейство адресов

- Определяет способ адресации узлов, с которыми будет осуществляться связь.
- Address family (AF_xxx), или protocol family (PF_xxx).
- Популярные семейства:
 - **AF_INET** — «интернет», адреса IPv4 (127.0.0.1)
 - **AF_INET6** — «интернет», адреса IPv6 (:::1)
 - **AF_LOCAL** — локальные сокеты ОС *nix
 - **AF_BTH** (Bluetooth), **AF_IRDA** (ИК) в Windows

Вид сокетов

- Определяет, как приложению работать с сокетом.
- Пример: дейтаграммные (**SOCK_DGRAM**)
 - Соединения не устанавливается (connectionless)
 - Передача ведется блоками данных (datagram):
 - размер блока ограничен (< 512...1500 байт);
 - блоки или приходят сразу целиком, или не приходят (не может прийти часть блока);
 - блоки могут прийти в ином порядке, чем были отправлены;
 - возможны потери дейтаграм (unreliable — ненадежная доставка).

Протокол

- Определяет, как система будет отправлять данные по сети
- Совместимы только определенные протоколы и виды сокетов:
 - SOCK_DGRAM, IPPROTO_UDP
 - SOCK_STREAM, IPPROTO_TCP
 - ~~○ SOCK_STREAM и IPPROTO_UDP~~

Адрес

Перечисленное объявлено в библиотеках,
в своих программах этого писать не нужно

```
struct sockaddr_in {  
    short  
    unsigned short  
    struct in_addr  
    char  
};
```

Семейство адресов (AF_INET, ...)

sin_family;

Номер порта
(в сетевом порядке байт).

sin_port;

sin_addr;

Адрес узла (4 байта)

sin_zero[8];

Не используются.

```
struct in_addr {  
    unsigned long s_addr;  
};
```

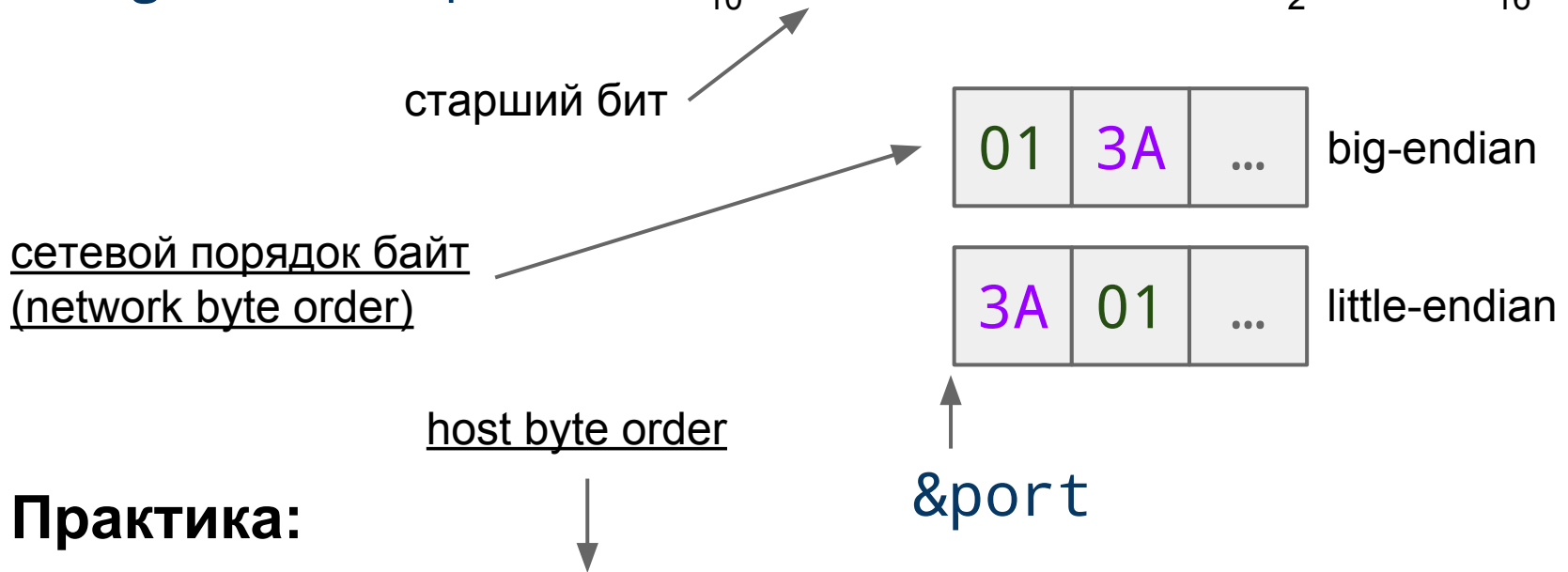
Необходимо преобразование
("127.0.0.1" → 0x7f000001):

```
unsigned long inet_addr(const char* address)
```

```
const char* inet_ntoa(unsigned long address) ← (Обратное.)
```

Порядок байт (byte order, endianness)

`unsigned short port` = 314_{10} = 0000 0001 0011 1010₂ = 01 3A₁₆



Практика:

- ❑ Порядок байт на разных машинах отличается.
- ❑ `htons()`, `htonl()`, `ntohs()`, `ntohl()`
h — host, n — network, s — short, l — long

Отправка данных

Целое со знаком:

`int` (Windows), `ssize_t` (*nix).

↓
`auto result = sendto(`

`channel,` ← Ранее созданный сокет.

`buffer,` ← Адрес данных для отправки.

`size,` ← Количество байт для отправки.

`0,` ← Флаги настроек (см. документацию).

`(const sockaddr*)&destination),`

`sizeof(destination));`

Количество отправленных
байт

`(0 <= result <= size)`

при успехе;

`SOCKET_ERROR` (Windows)

или `-1` (*nix) при ошибке.

Адрес получателя
и его размер.

Привязка сокета (указание собственного адреса)

Необходима для приема, необязательна для отправки. Отменить нельзя.

Успех: 0,

ошибка: `SOCKET_ERROR` (Windows) или `-1` (*nix).

↓

```
int result = bind(  
    channel,  
    (const sockaddr*)&address,  
    sizeof(address));
```

↑
Размер структуры-адреса.

← Ранее созданный
сокет.

↑ Структура-адрес `sockaddr_in`,
на который должно быть отправлено
сообщение, чтобы сокетом `channel`
можно было его принять.

Прием данных (SOCK_DGRAM)

Размер принятой датаграммы, если она уместилась в буфер (`result <= size`).

Код ошибки:

WSAEMSGSIZE — датаграмма была длиннее `size`, буфер заполнен, остаток утерян.

другой — прием не удался.

SOCKET_ERROR или (-1)

```
auto result = recvfrom(
    channel,
    buffer,
    size,
    0,
    nullptr,
    nullptr);
```

Буфер под принятые данные.

Количество байт, которое можно записать в буфер.

Флаги настроек (см. документацию).

} Нужны для получения адреса отправителя (необязательные параметры).

Прием данных (SOCK_DGRAM): получение адреса отправителя

```
sockaddr_in source; ← Структура-адрес.  
int source_size = ...; ← Размер в байтах  
auto result = recvfrom(  
    channel, buffer, size, 0,  
    (sockaddr*)&source, &source_size);
```

↑
или переменной source.

Адрес переменной, содержащей:

Windows;
`socklen_t` в *nix
(беззнаковый).

перед вызовом —

количество байт, которое можно записать
в структуру-адрес;

после вызова —

количество байт, записанных в структуру-адрес.

N. B.: вызовы блокирующие:

функции не вернут управление программе, пока не произойдет некоторое событие.

- `recvfrom()` выполняется, пока не будут получены данные по сети.
- `sendto()` выполняется, пока ОС не примет данные для отправки.
 - Ожидание только при загрузенности системы.
 - Отправка может произойти не сразу или не произойти вообще (unreliable!).

Заккрытие сокета

- Функция:
 - Windows: `closesocket(channel);`
 - *nix: `close(channel);`
- Сокеты — дефицитный ресурс ОС!
 - По исчерпанию доступных сокетов становится невозможно создать новые.
 - Процессу выделяется несколько тысяч.
(Сетевые программы могут работать непрерывно, долго и под большой нагрузкой.)

Настройка сокета

Значение опции: 1 — включить (разрешить), 0 — выключить (запретить).

`int value = 1;`

`setsockopt(`

`channel,`

`SOL_SOCKET,`

`SO_BROADCAST,`

`(const char*)&value,`

`sizeof(value));`

Описание опций и возможных значений — в документации ([MSDN](#) или [man setsockopt](#)).

Опция задается для указанного сокета (не для сеанса, подсистемы и т. п.).

Указатель на переменную со значением опции и её размер.

Диагностика ошибок

- Можно получить код последней ошибки:
 - Windows:
 - `int error_code = WSAGetLastError();`
 - расшифровка кодов — в [MSDN](#);
 - *nix:
 - переменная `errno`;
 - расшифровка кодов — `man errno`;
 - проверять нужно *сразу* после ошибки.
- Возвращаемое значение при ошибках:
 - Windows: `SOCKET_ERROR` (есть исключения)
 - *nix: всегда `-1`

Используемые библиотеки

- Windows:

- `#include <winsock2.h>`
- КОМПОНОВАТЬ с `ws2_32.lib`
 - Visual Studio:
`#pragma comment(lib, "ws2_32")`
 - Code::Blocks:
Project → Build options... → Linker

- *nix:

```
#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>
```

Запуск и останов библиотеки сетевой подсистемы (Windows)

Структура с данными
о сетевой подсистеме

Передается указатель на структуру.
Поля будут заполнены системой.

/ Запуск: */*

WSADATA wsa;

WSAStartup(0x0202, &wsa);

успех: 0

неудача: не-0

Версия Windows Sockets 2.2

MAKWORD(2, 2) = 0x0202

/ Останов: */*
WSACleanup();