Sockets and Networks

Operating Systems II

Viktor Iakovlev (Victor Yacovlev)

Inter-Process Communications

- Files I/O
- Shared Memory Pages (mmap)
- Unnamed Pipes (pipe)
- Names Pipes (fifo)
- Sockets - allows one process to interact to many processes

Socket

- Just a file descriptor
- Has an unique name
- Several kinds of name domains:
  - UNIX-socket: the name of special file in the Virtual File System
  - TCP/IP-socket: host address + port number

Socket Creation

`int socket(int domain, int type, int protocol)`

Just creates a socket as file descriptor

- `domain` – naming conventions (`AD_UNIX`, `AD_INET`, `AF_IPX`, `AF_APPLETALK`, `AF_BLUETOOTH` and so on)
- `type` – `OSCK_STREAM` or `SOCK_DGRAM`
- `protocol` – `0` – automatic; `IPPROTO_TCP` and so on

- Just creates file descriptor. Not ready to interact yet
- Might be inherited by child process (fork) or cloned to another file descriptor (dup2)
- Must be closed after use to free resources (close)

Socket Setup

- For client-side: connect to someone
  - *system call* **`connect`**
- For server-side: declare a name and begin listening for incoming connections
  - register a name using **`bind`**
  - create incoming queue and switch to listening mode **`listen`**
  - accept pending connection **`accept`**

Socket Name

```
int bind(int socket,
     const struct sockaddr *addr,
     socklen_t address_len)
```

Several types of addresses:

- `struct sockaddr_in` – IPv4 address + port
- `struct sockaddr_in6` – IPv6 address + port
- `struct sockaddr_un`– local file name

Name Registration

- Required to allow incoming connections
- Might be used for outcoming connections (for connectionless message sending)

Switch to Listen Mode

```
listen(int sockfd, int backlog)
```

- backlog - incoming queue size
- if too many (>backlog) connections then incoming connection refuses
- `SOMAXCONN` constant (128 for Linux) stores maximum queue size (depends on Kernel build configuration)

Connection creation

- **`connect`** - connect to another socket
- **`accept`** - wait for the next incoming connection and then create it

Socket I/O

```
ssize_t recv(int socket, void* buffer, size_t buf_size, int flags)
```
Reads data from socket. Parameters:

- `MSG_PEEK` – just skip data
- `MSG_OOB` – get out-of band priority data
- `MSG_WAITALL` – read for all data to be transmitted

```
ssize_t send(int socket, const void *buffer, size_t size, int flags)
```
Write data to socket. Parameters:

- `MSG_OOB` – set out-of-band priority flag
- `MSG_NOSIGNAL` – do net send SIGPIPE in case if socket closed

```
 read(Socket, Buffer, Size) → recv(Socket, Buffer, Size, 0).
write(Socket, Buffer, Size) → send(Socket, Buffer, Size, 0).
```

Incoming Connections

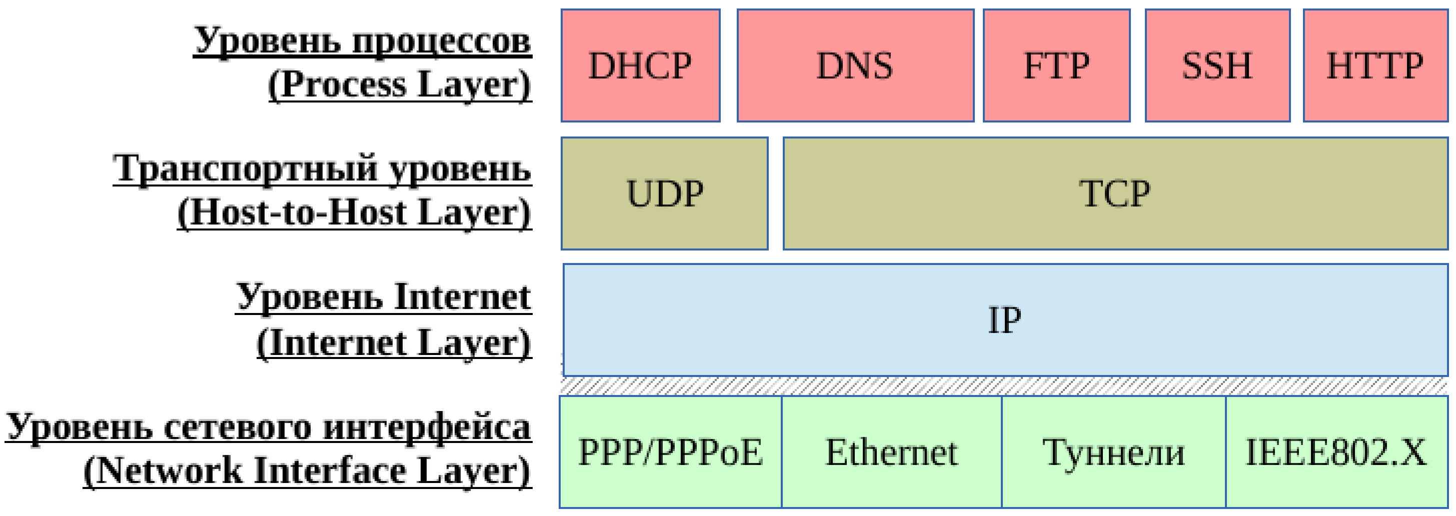
1. Create the socket [socket]
2. Bind to some name [bind]
3. Switch to listen mode [listen]
4. Accept the next pending connection [accept] as a new socket
5. Use it using [read/write] or [recv/send]
6. Don’t forget to close socket that created by [accept]
7. goto step 4

Inter-Process Communications (UNIX-domain sockets)

- Desktop Usage
  - X-server
  - DBUS-service
  - PulseAudio-service
- Server Usage
  - Web Applications
  - SQL-server
- Default Sockets Location
  - `/var/run`
  - `/tmp/*`

NETWORKS

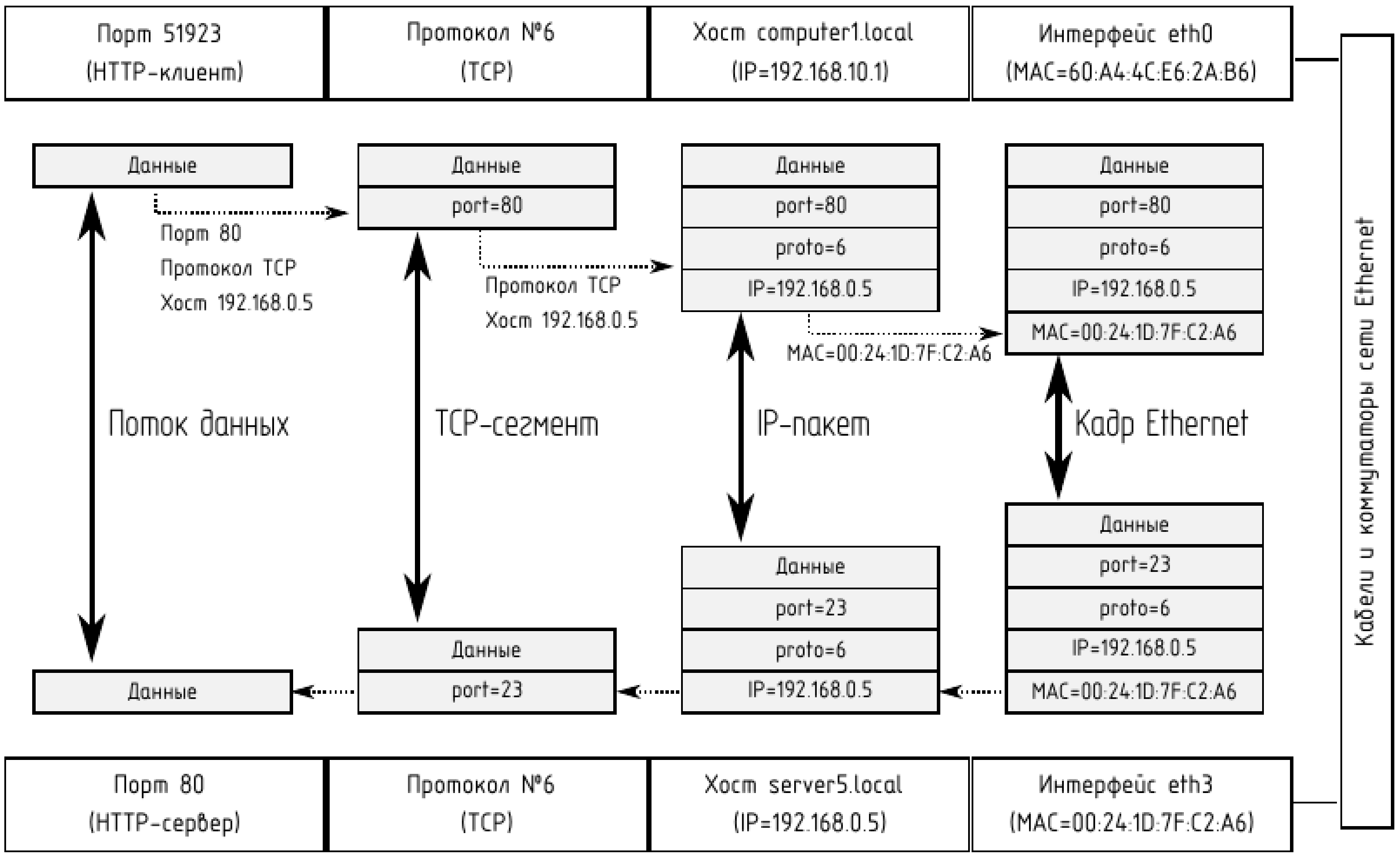
TCP/IP


Уровень процессов
(Process Layer)
DHCP
DNS
FTP
SSH
HTTP
Транспортный уровень
(Host-to-Host Layer)
UDP
TCP
Уровень Internet
(Internet Layer)
IP
Уровень сетевого интерфейса
(Network Interface Layer)
PPP/PPPoE
Ethernet
Туннели
IEEE802.X

OSI

Стек TCP/IP		Модель OSI (Open Systems Interconnections)	Примеры
Уровень процессов		Уровень приложений (Application)	HTTP, FTP, SSH, Telnet
		Уровень представления (Presentation)	ASCII, GZIP, binary
		Уровень сеанса (Session)	NetBIOS, SSL
Транспортный уровень		Уровень транспорта (Transport)	TCP, UDP
Уровень Internet		Уровень сети (Network)	IPv4, IPv6, IPX, AppleTalk
Уровень сетевого интерфейса		Уровень канала (Data Link)	PPP, IEEE 802.2 (Ethernet)
		Физический уровень (Physical)	USB, IEEE 802.11 IEEE 802.3 (Ethernet)

Data Transmission TCP/IP



Ethernet

MAC-адрес получа-теля	MAC-адрес отпра-вителя	Допол-нитель-ные опции	Длина	Данные	Конт-роль-ная сумма
6 байт	6 байт	4 байта	2 байта	от 46 до 1500 байт (параметр MTU)	4 байта

IPv4

Байты	**0**	**1**	**2**	**3**
0...3	**Версия и размер заголовка**	**Тип службы**	**Размер IP-пакета**	
4...7	**ID группы пакетов**		**Флаги и смещение**	
8...11	**TTL**	**Номер протокола**	**Контрольная сумма заголовка**	
12...15	**Адрес отправителя**			
16...19	**Адрес получателя**			
20...24	**Дополнительные опции**			

Hostnames v.s. IPv4 Addresses

- IP-address is a 32-bit (IPv4) or 128-bit (IPv6) unsigned integer value
- The names are stored at DNS databases
- 127.0.0.1 (localhost) means *current computer*
- Each host name might have several IP-addreses
- Each IP-address might have several host names

UDP

Байты	0	1	2	3
0..4	Порт отправителя		Порт назначения	
5..8	Длина пакета		Контрольная сумма	

TCP

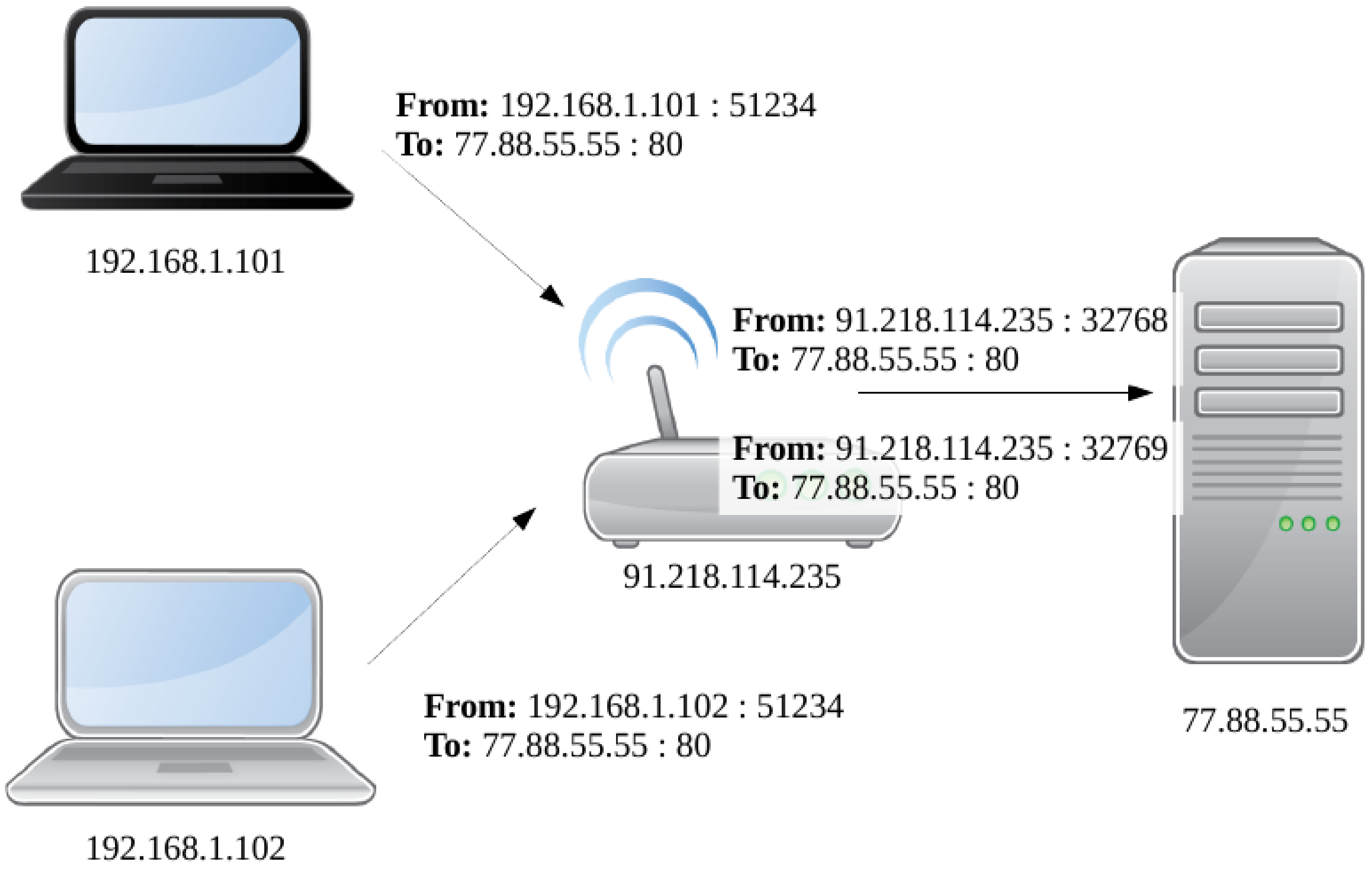
Байты	0	1	2	3
0...3	Порт отправителя		Порт получателя	
4...7	Порядковый номер пакета			
8...11	Порядковый номер подтверждаемого пакета (с флагом ACK)			
12...15	Размер заголовка в 32-битных словах, 000, NS	CWR, ECE, URG, **ACK**, PSH, RST, **SYN**, **FIN**	Размер окна (буфера для приема данных, ожидаемых при ответе)	
16...19	Контрольная сумма заголовка и данных		Указатель на порядковый номер пакета, в котором заканчивается приоритетных блок данных	
20……	Дополнительные опции			

Номера портов

```
0               – (unused)
20, 21          – FTP
22              – SSH
25              – SMTP
80              – HTTP
137,138,138     – NetBIOS
143             – IMAP
443             – HTTPS
465             – SMTPS
631             – CUPS
993             – IMAPS

1024...65535  – outgoing port numbers
(legacy UNIX-systems and Windows XP)
32768...65535 – outgoing port numbers (Linux)
49152...65535 – outgoing port numbers (BSD, Windows Vista+)
```

Port Forwarding



From: 192.168.1.101 : 51234
To: 77.88.55.55 : 80

From: 91.218.114.235 : 32768
To: 77.88.55.55 : 80

From: 91.218.114.235 : 32769
To: 77.88.55.55 : 80

From: 192.168.1.102 : 51234
To: 77.88.55.55 : 80

Ports <1024

- Used by standard services
- Only root user can bind
- **It is DANGEROUS!**

Solution

- Create socket
- Create child process to inherit file descriptor
- Child process to elevate privileges to root and then bind

Done by authbind for Debian/Ubuntu