Лекция №6

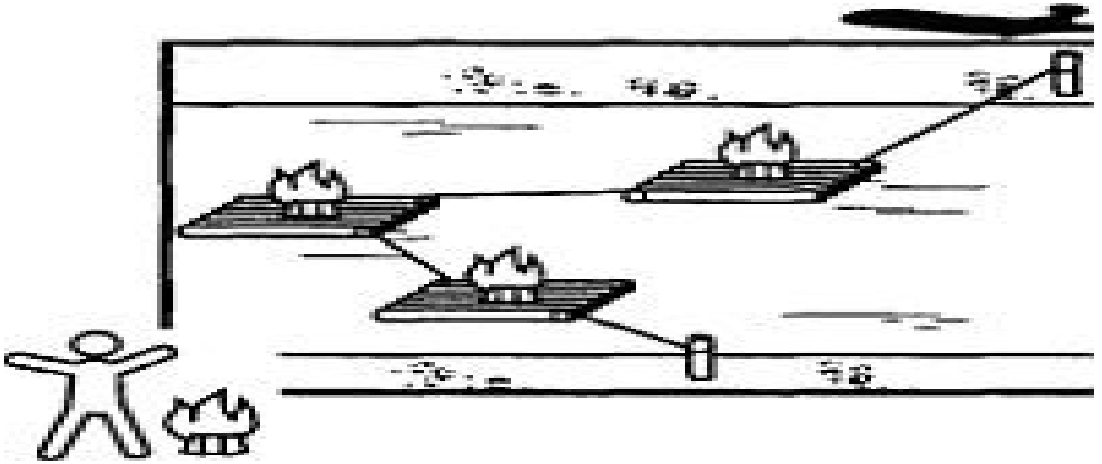
Передача информации

Передача информации

Передача информации — физический процесс, посредством которого осуществляется перемещение информации в пространстве. Записали информацию на диск и перенесли в другую комнату. Данный процесс характеризуется наличием следующих компонентов:

- Источник информации.
- Приёмник информации.
- Носитель информации.
- Среда передачи.

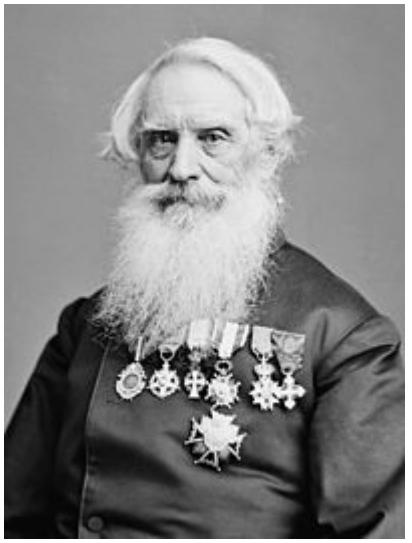
Исторические примеры
Сигнальные костры



Технические характеристики:

- Дальность передачи информации - до 50Км
- Скорость передачи = скорости света
- Количество передаваемой информации за один интервал времени — 1бит
- Тип связи — широковещательный
- Возможно применение кодирования информации

Исторические примеры Азбука Морзе



Передаваться и приниматься азбука Морзе может с различной скоростью — это зависит от возможностей и опыта радистов. Обычно средней квалификации радист работает в диапазоне скоростей 60 — 100 знаков в минуту. Достижения по скоростным приёму-передаче находятся в диапазоне скоростей 260—310 знаков в минуту.

A	А	·−	N	Н	−·
B	Б	−···	O	О	−−−
C	Ц	−·−·	P	П	·−−·
D	Д	−··	Q	Щ	−−·−
E	Е	·	R	Р	·−·
F	Ф	··−·	S	С	···
G	Г	−−·	T	Т	−
H	Х	····	U	У	··−
I	И	··	V	Ж	···−
J	Й	·−−−	W	В	·−−
K	К	−·−	X	Ь	−··−
L	Л	·−··	Y	Ы	−·−−
M	М	−−	Z	З	−−··

Способы передачи информации

- Между переменными;
- Между объектами;
- Между модулями;
- Между приложениями;
- Между компьютерами.

Передача содержимого переменной

1. Передача значения

a = b

2. Передача адреса

f(&a)

3. Передача ссылки

int *a = &b

Преобразование / приведение типов данных

```
double a = 1.1;
int b = a;

int b = (int) a;

String a = "a" + "b";
```

Автоматическое преобразование

```
byte a = 40;
byte b = 50;
byte c = 100;
int d = a* b / c;
```

Передача содержимого объекта

```
MyClass a,b;
a = new MyClass();
b = a;

a.clone();
```

Видимость переменной

+ public

- private

= protected

static

final

Обращение к переменным объекта

myObject.var1

myObject→var1

myObject.getName()

myObject.setName("New name")

Передача информации между модулями

В распоряжении программистов есть несколько методов IPC (Inter-process communication):

- полудуплексные каналы UNIX
- FIFO (именованные каналы)
- Очереди сообщений в стиле SYSV
- Множества семафоров в стиле SYSV
- Разделяемые сегменты памяти в стиле SYSV
- Сетевые сокеты (в стиле Berkeley)
- Полнодуплексные каналы (каналы потоков)

Каналы (pipelines, pipes)

Канал - это средство
процесса со стандартны
метод односторонних
duplex) между процесса

ls | losrt | lp

stdin / stdout / stderr

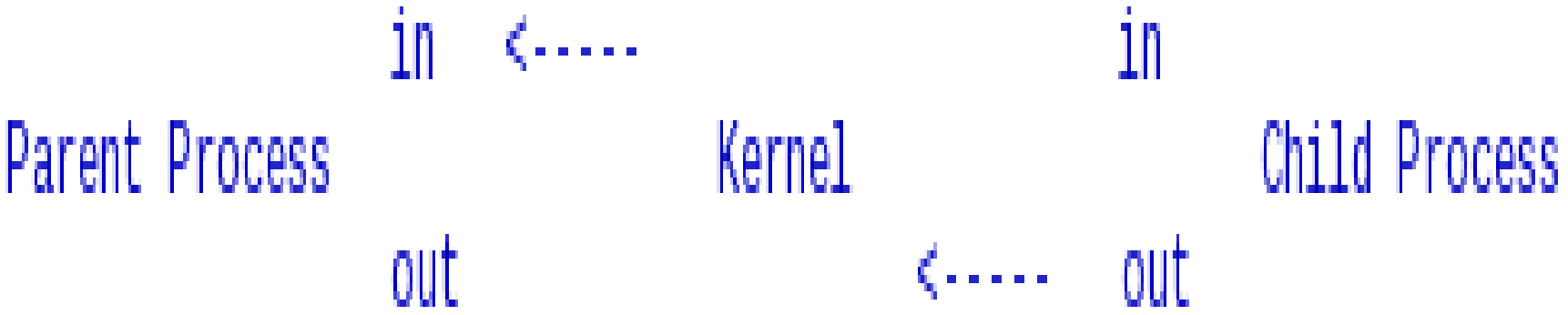
Constant	Description

STDIN	An already opened stream to ***stdin***. This saves opening it with
	`<?php`
	`$stdin = fopen('php://stdin', 'r');`
	`?>`
	If you want to read single line from ***stdin***, you can use
	`<?php`
	`$line = trim(fgets(STDIN)); // reads one line from STDIN`
	`fscanf(STDIN, "%d\n", $number); // reads number from STDIN`
	`?>`

STDOUT	An already opened stream to ***stdout***. This saves opening it with
	`<?php`
	`$stdout = fopen('php://stdout', 'w');`
	`?>`

STDERR	An already opened stream to ***stderr***. This saves opening it with
	`<?php`
	`$stderr = fopen('php://stderr', 'w');`
	`?>`

Каналы (pipelines, pipes)



```c
/*************************************************************************
 Excerpt from "Linux Programmer's Guide - Chapter 6"
 (C)opyright 1994-1995, Scott Burkett
 *************************************************************************
 MODULE: pipe.c
 *************************************************************************/
  #include <stdio.h>
  #include <unistd.h>
  #include <sys/types.h>

  int main(void)
  {
          int     fd[2], nbytes;
          pid_t   childpid;
          char    string[] = "Hello, world!\n";
          char    readbuffer[80];
          pipe(fd);
          if((childpid = fork()) == -1)
          {
                  perror("fork");
                  exit(1);
          }
          if(childpid == 0)
          {
                  /* Потомок закрывает вход */
                  close(fd[0]);
                  /* Посылаем "string" через выход канала */
                  write(fd[1], string, strlen(string));
                  exit(0);
          }
          else
          {
                  /* Родитель закрывает выход */
                  close(fd[1]);
                  /* Чтение строки из канала */
                  nbytes = read(fd[0], readbuffer, sizeof(readbuffer));
                  printf("Received string: %s", readbuffer);
          }
          return(0);
  }
```

FIFO

First In — First Out

Именованные каналы во многом работают так же, как и обычные каналы, но все же имеют несколько заметных отличий.

- Именованные каналы существуют в виде специального файла устройства в файловой системе.
- Процессы различного происхождения могут разделять данные через такой канал.
- Именованный канал остается в файловой системе для дальнейшего использования и после того, как весь ввод/вывод сделан.

System V IPC

Каждый объект IPC имеет уникальный IPC идентификатор. (Когда мы говорим "объект IPC", мы подразумеваем очередь единичных сообщений, множество семафоров или разделяемый сегмент памяти.) Этот идентификатор требуется ядру для однозначного определения объекта IPC. Например, чтобы сослаться на определенный разделяемый сегмент, единственное, что вам потребуется, это уникальное значение ID, которое привязано к этому сегменту.

System V IPC

Очереди сообщений представляют собой связный список в адресном пространстве ядра. Сообщения могут посылаться в очередь по порядку и доставаться из очереди несколькими разными путями. Каждая очередь сообщений однозначно определена идентификатором IPC.

System V IPC

Семафоры лучше всего предствлять себе как счетчики, управляющие доступом к общим ресурсам. Чаще всего они используются как блокирующий механизм, не позволяющий одному процессу захватить ресурс, пока этим ресурсом пользуется другой.

System V IPC

Разделяемая память может быть наилучшим образом описана как отображение участка (сегмента) памяти, которая будет разделена между более чем одним процессом. Это гораздо более быстрая форма IPC, потому что здесь нет никакого посредничества (т.е. каналов, очередей сообщений и т.п.). Вместо этого, информация отображается непосредственно из сегмента памяти в адресное пространство вызывающего процесса. Сегмент может быть создан одним процессом и впоследствии использован для чтения/записи любым количеством процессов.

Unix socket

Unix Domain Sockets — двухсторонний канал для межпроцессорного обмена данными.
Принцип работы:

- Создать сокет (поток) (socket)

- Связать поток с адресом (bind)
- Включить прослушивание (listen)
- Принять соединение (accept)

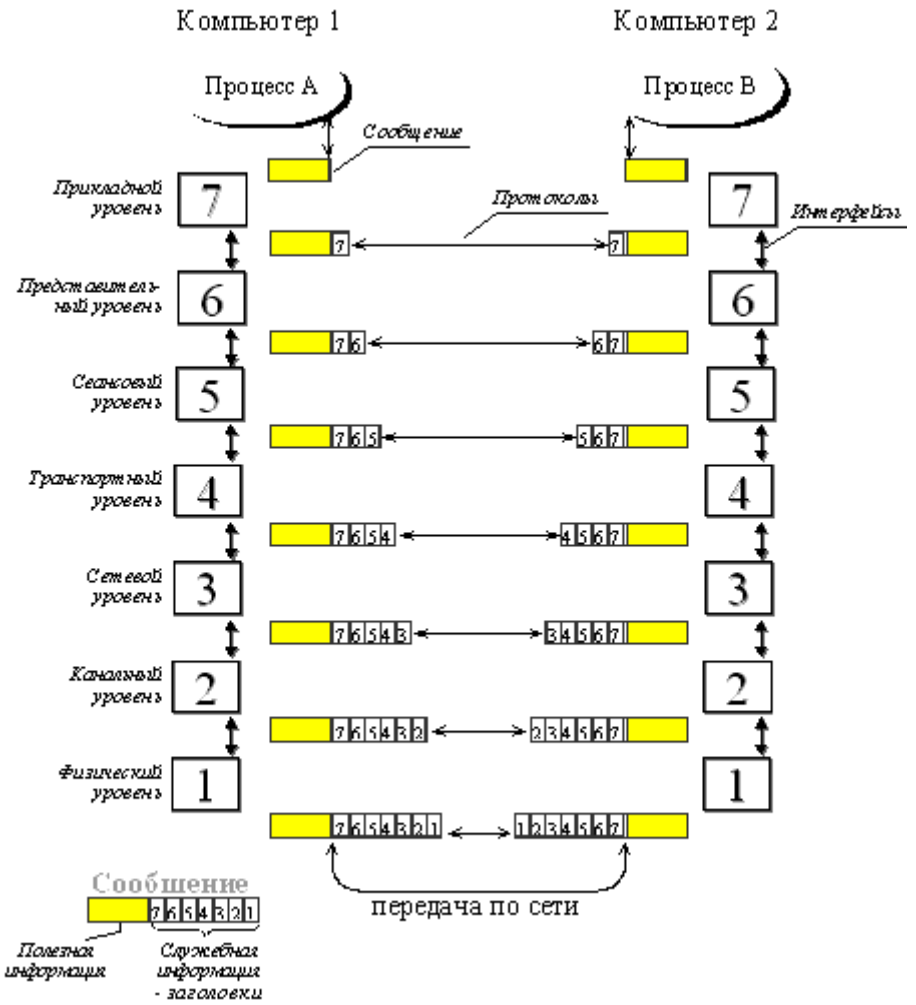
- Подсоединиться к потоку (connect)

- Получить данные из потока (recv) или отправить в поток (send)
- Закрыть соединение (close, shutdown)

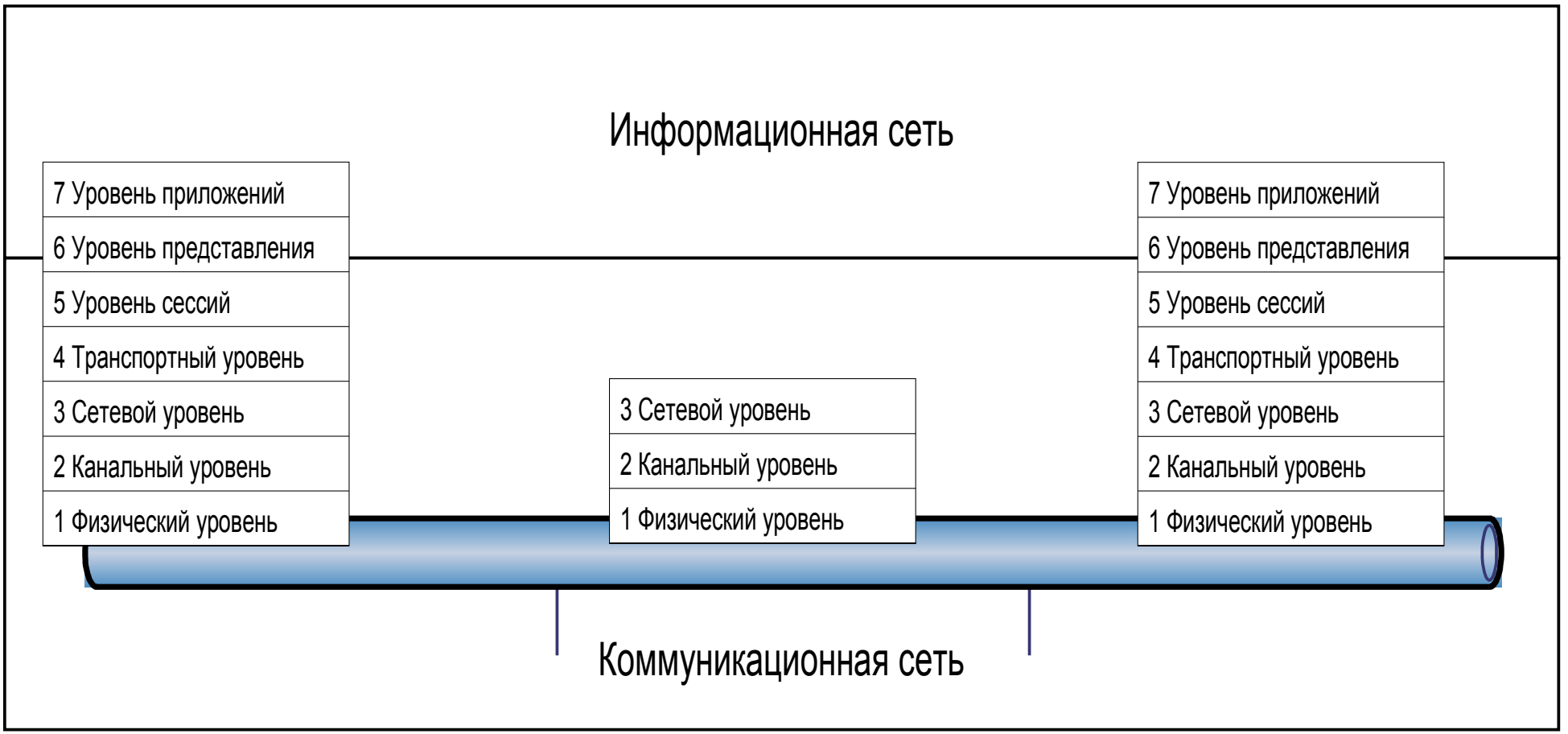
OSI

Open System Interconnection

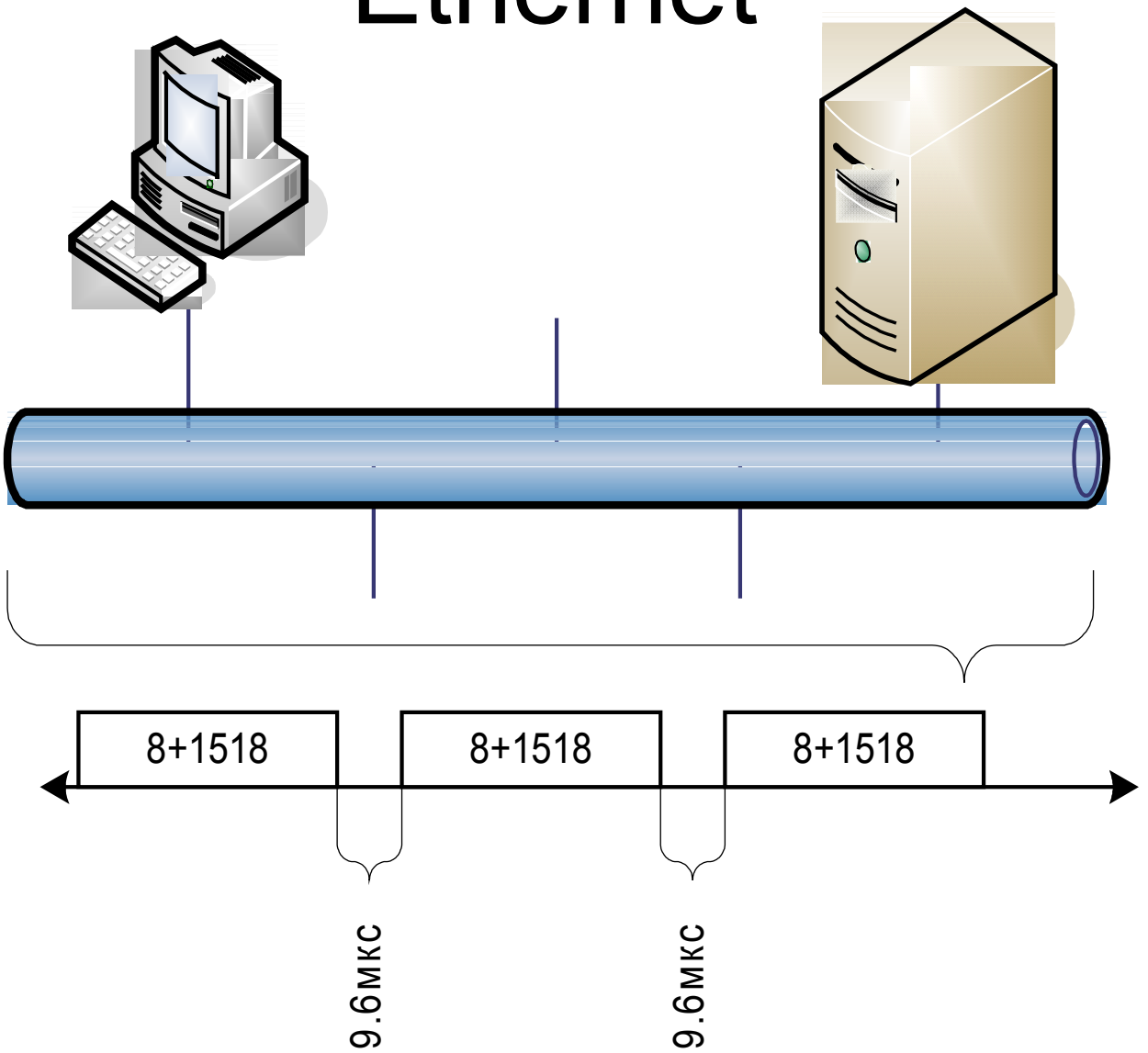
OSI

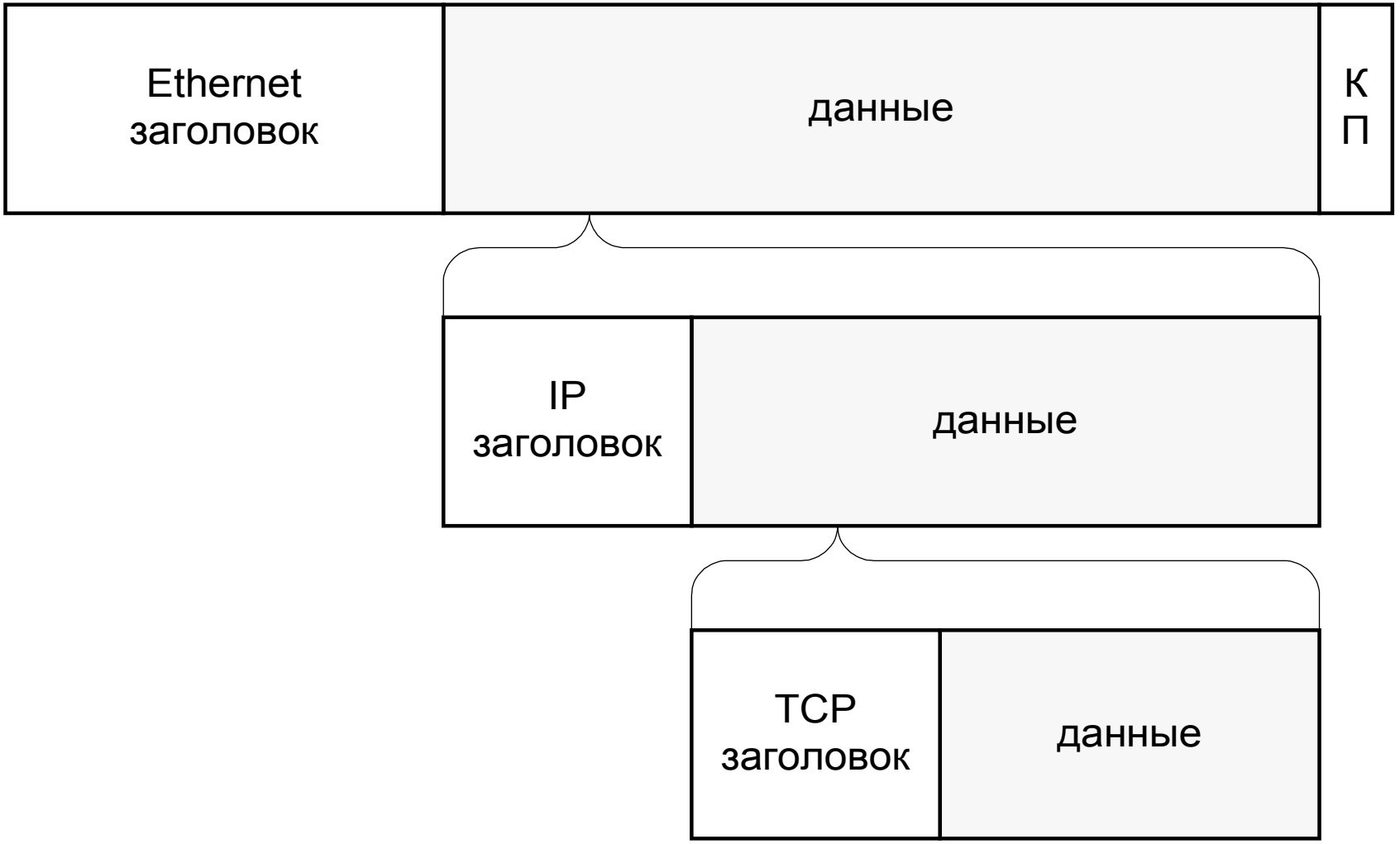

Компьютер 1
Компьютер 2
Процесс А
Процесс В
Сообщение
Протоколы
Интерфейсы
Прикладной уровень
Представительный уровень
Сеансовый уровень
Транспортный уровень
Сетевой уровень
Канальный уровень
Физический уровень
7
6
5
4
3
2
1
передача по сети
Сообщение
Полезная информация
Служебная информация - заголовки

OSI

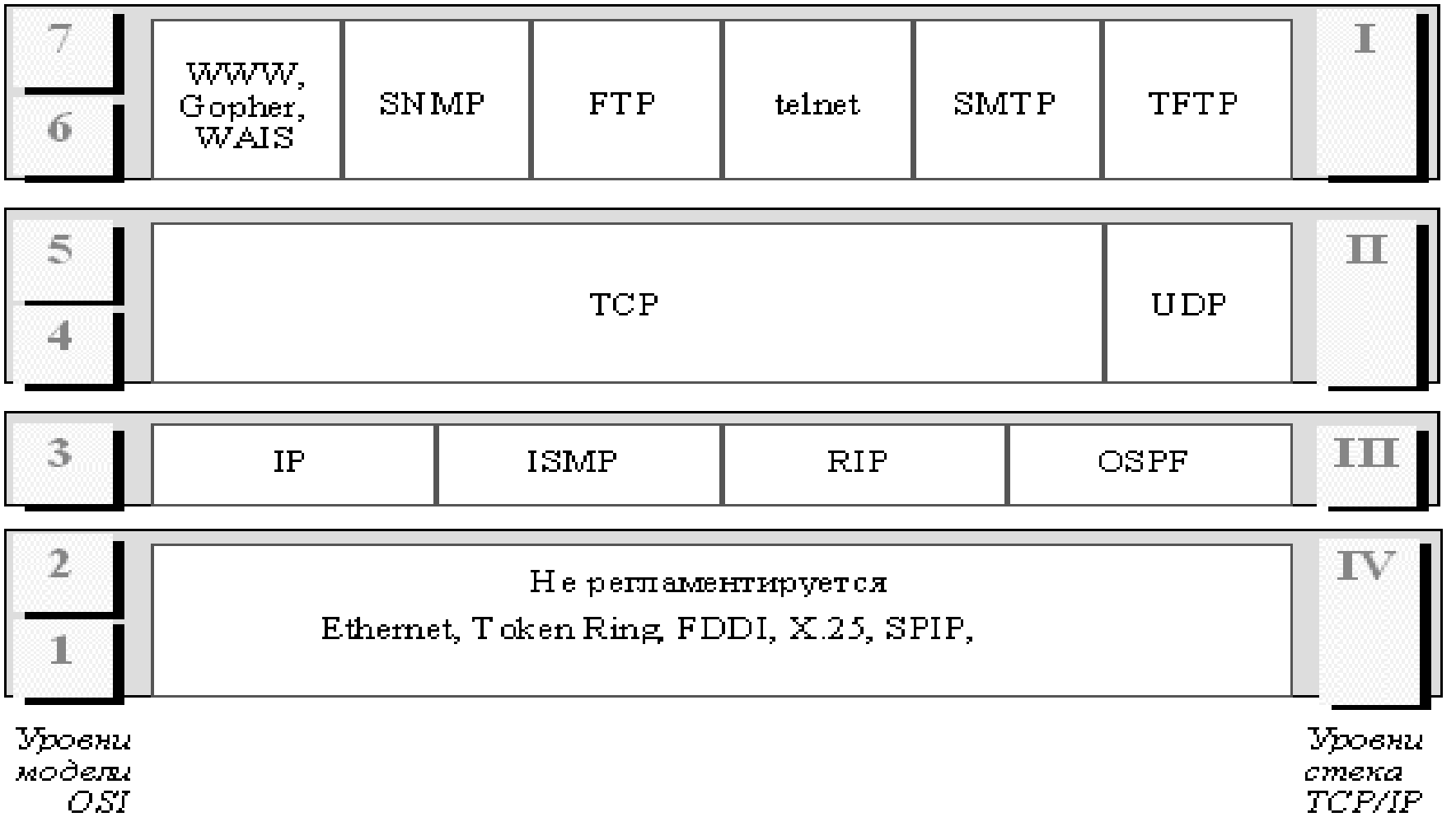

Информационная сеть
7 Уровень приложений
6 Уровень представления
5 Уровень сессий
4 Транспортный уровень
3 Сетевой уровень
2 Канальный уровень
1 Физический уровень
3 Сетевой уровень
2 Канальный уровень
1 Физический уровень
7 Уровень приложений
6 Уровень представления
5 Уровень сессий
4 Транспортный уровень
3 Сетевой уровень
2 Канальный уровень
1 Физический уровень
Коммуникационная сеть

Ethernet

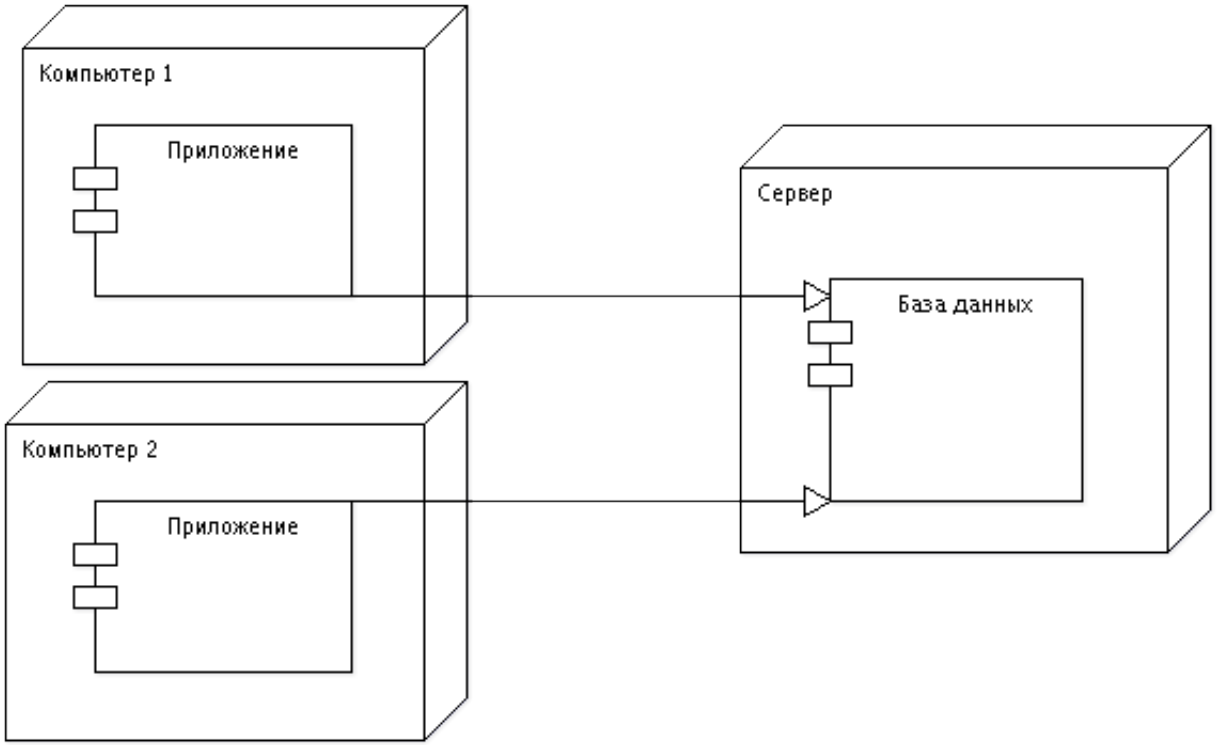



Ethernet
заголовок
данные
К
П
IP
заголовок
данные
TCP
заголовок
данные

Стек TCP/IP


7
6
WWW, Gopher, WAIS
SNMP
FTP
telnet
SMTP
TFTP
I
5
4
TCP
UDP
II
3
IP
ISMP
RIP
OSPF
III
2
1
Не регламентируется
Ethernet, Token Ring, FDDI, X.25, SPIP,
IV
Уровни модели OSI
Уровни стека TCP/IP

Архитектура ПО



Windows

В среде Win32 существует два основных типа объектов — объекты ядра (kernel) и объекты GDI/USER (графического интерфейса и интерфейса пользователя).

Объекты ядра представляют собой базовые объекты Win32 и включают события, отображения файлов, файлы,почтовые гнезда, блоки совместного использования, каналы, семафоры, процессы и потоки.

Процесс можно представить как выполняющееся приложение или отдельный экземпляр приложения. В среде Win32 одновременно могут быть активными сразу несколько процессов. Каждый процесс получает собственное 4-гигабайтовое адресное пространство для кода и данных. В этих 4 гигабайтах памяти размещается вся распределяемая приложением память, все его потоки, карты файлов и многое другое.

Windows

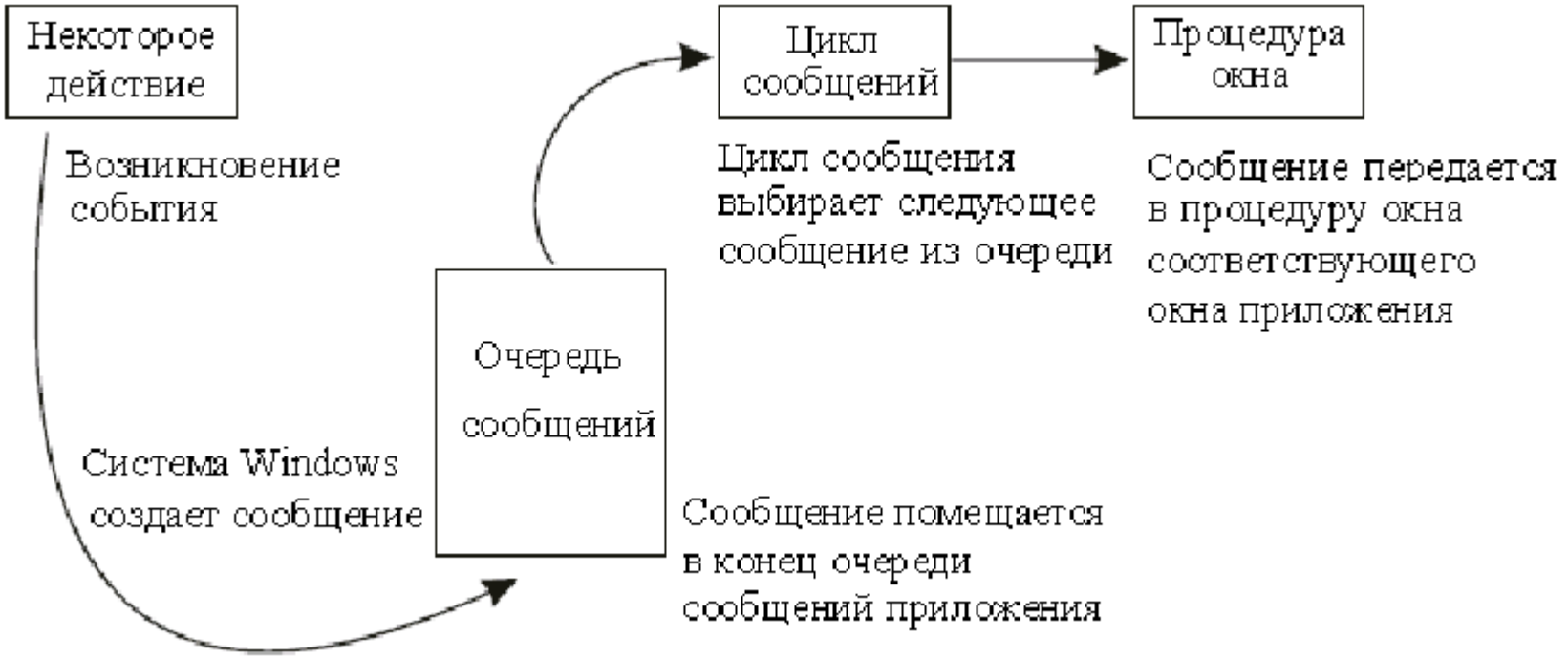
Сами по себе процессы инертны, т.е. они ничего не выполняют. Однако каждый процесс имеет первичный поток (primary thread), в рамках которого и выполняется программный код, присутствующий в контексте данного процесса. Процесс может иметь несколько потоков, однако только один из них является первичным, или главным.

Поток (thread) — это объект операционной системы, который представляет независимую последовательность выполнения программного кода, имеющую место в рамках некоторого процесса. Каждое приложение Win32 имеет минимум один поток, часто именуемый первичным, или главным. Однако при необходимости приложение может создавать и другие потоки, предназначенные для решения отдельных задач.

Windows

Сообщение (message) представляет собой извещение о некотором имевшем место событии, посылаемое системой Windows в адрес приложения. Любые действия пользователя — щелчок мышью, изменение размеров окна приложения, нажатие клавиши на клавиатуре — вынуждают Windows отправить приложению сообщение, извещающее о том, что произошло в системе.

Windows


Некоторое действие
Возникновение события
Система Windows создает сообщение
Очередь сообщений
Сообщение помещается в конец очереди сообщений приложения
Цикл сообщений
Цикл сообщения выбирает следующее сообщение из очереди
Процедура окна
Сообщение передается в процедуру окна соответствующего окна приложения