C++ Craft: #6

Сетевые компоненты TCP/UDP

Асинхронное использование сетевых механизмов

Task-Based Engine, Thread-Based Engine

Язык Программирования C++

Нет такой главы, да и в общем-то это не C++.

OSI

- Physical Layer
- Data Link Layer
- Network Layer
- Transport Layer
- Session Layer
- Presentation Layer
- Application Layer

- Физический
- Канальный
- Сетевой
- Транспортный
- Сеансовый
- Представительский
- Прикладной

http://en.wikipedia.org/wiki/OSI_model

PDU

OSI Model			
	Data unit	Layer	Function
Host layers	Data	7. Application	Network process to application
		6. Presentation	Data representation,encryption and decryption
		5. Session	Interhost communication
	Segments	4. Transport	End-to-end connections and reliability,Flow control
Media layers	Packet	3. Network	Path determination and logical addressing
	Frame	2. Data Link	Physical addressing
	Bit	1. Physical	Media, signal and binary transmission

http://en.wikipedia.org/wiki/Protocol_data_unit

OSI: Physical Layer

- Метод передачи данных в двоичном виде

- IEEE 802.15 (Bluetooth)
- DSL
- 802.11 Wi-Fi

OSI: Data Link Layer

- Обеспечение взаимодействия сетей
- Контроль за ошибками

Два подуровня:

  - MAC
  - LLC

- PPP
- PPPoE
- HDLC

Terms

- MTU (Max Transmission Unit): максимальный размер полезного блока данных без фрагментации (например 1500).
- MAC (Media Access Control): уровень управления доступом к среде (обеспечивает адресацию).
- LLC (Logical Link Control): управление передачей данных, проверка корректности.

---

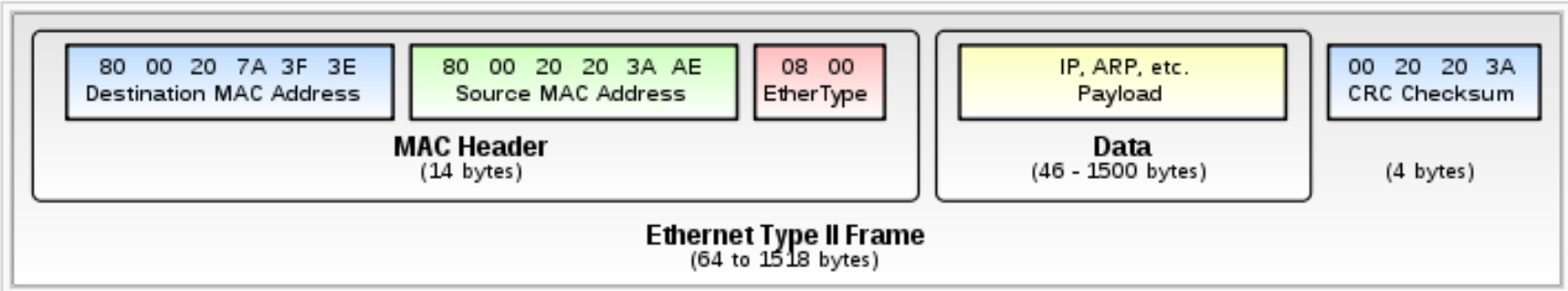
http://ru.wikipedia.org/wiki/Maximum_transmission_unit

LLC: HDLC

Флаг FD	Адрес	Управляющее поле	Информационное поле	FCS	Флаг FD
8 бит	8 бит	8 или 16 бит	0 или более бит, кратно 8	16 бит	8 бит

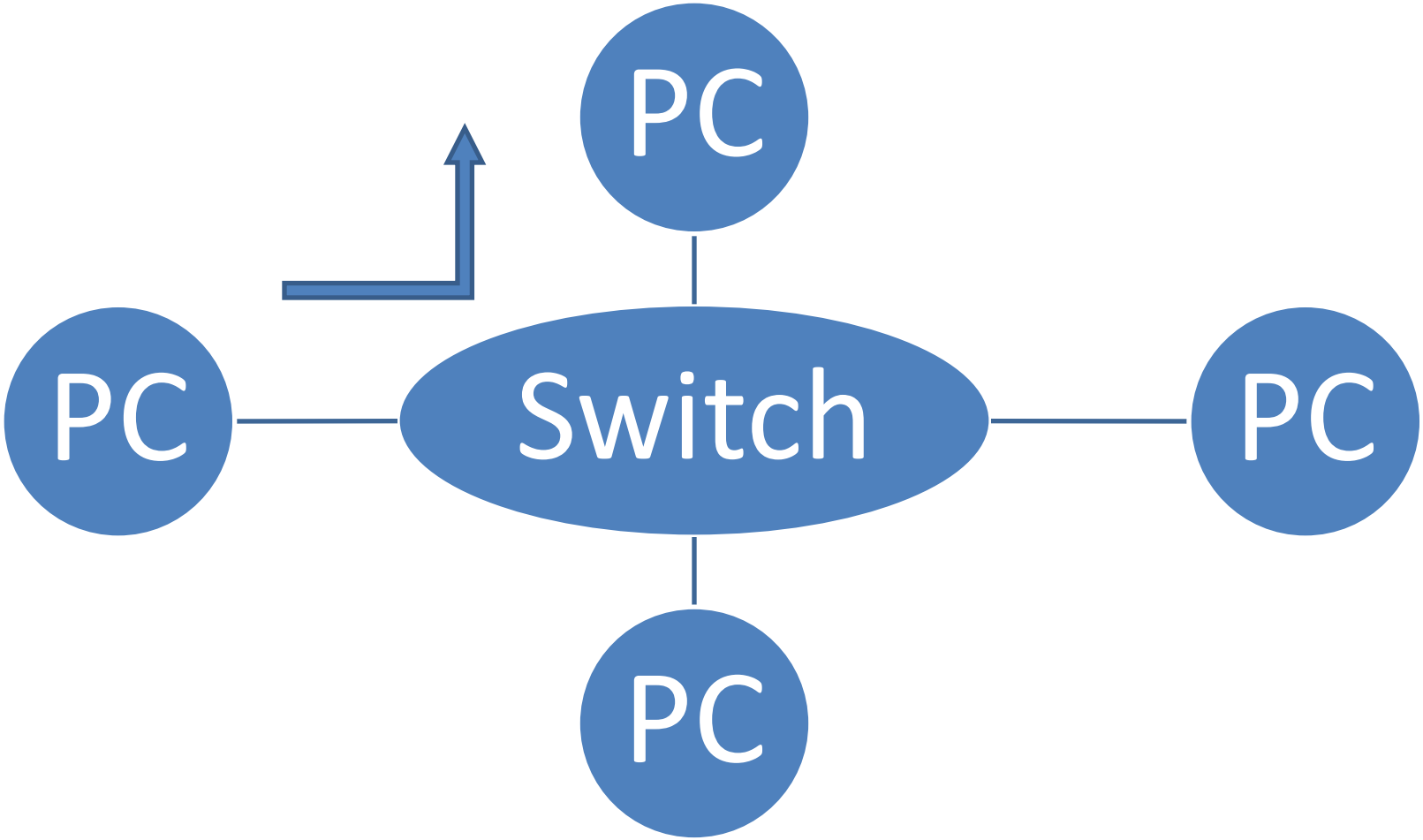
http://ru.wikipedia.org/wiki/Logical_link_control

MAC: Ethernet

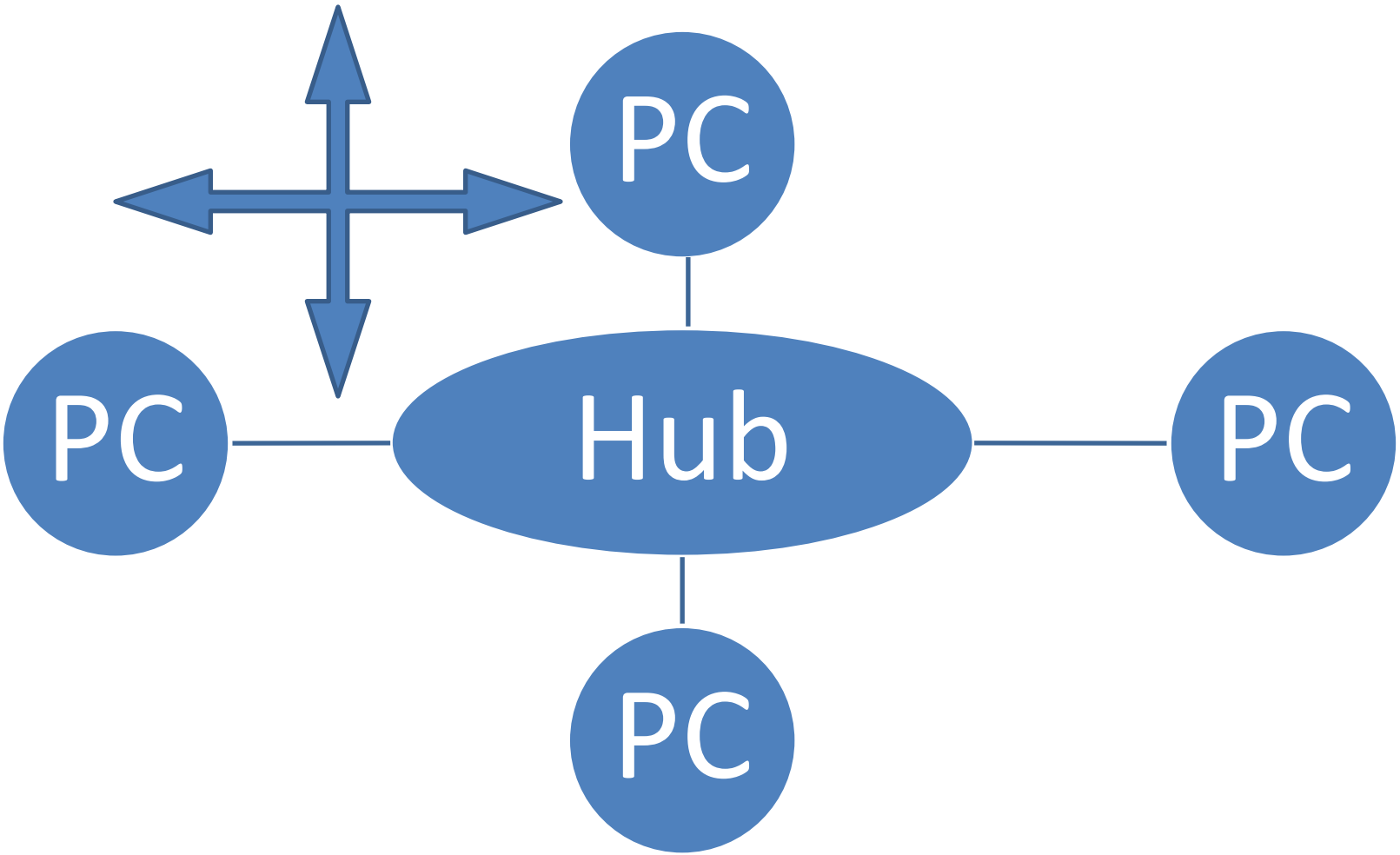


http://ru.wikipedia.org/wiki/Ethernet

Сетевой Коммутатор (Switch)



Сетевой Концентратор (Hub)



OSI: Network Layer

- Определение пути передачи данных
- Трансляция логических адресов в физические
- Поиск кратчайших маршрутов
- Отслеживание неполадок

- IP (v4, v6)
- IPsec
- IPX
- ICMP

IP (Internet Protocol)

Маршрутизируемый протокол сетевого уровня

Октет	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Версия				IHL				Тип обслуживания								Длина пакета															
4	Идентификатор																Флаги			Смещение фрагмента											
8	Время жизни (TTL)								Протокол								Контрольная сумма заголовка															
12	IP-адрес отправителя																															
16	IP-адрес получателя																															
20	Параметры (от 0 до 10-и 32-х битных слов)																															
	Данные																															

http://ru.wikipedia.org/wiki/Internet_Protocol

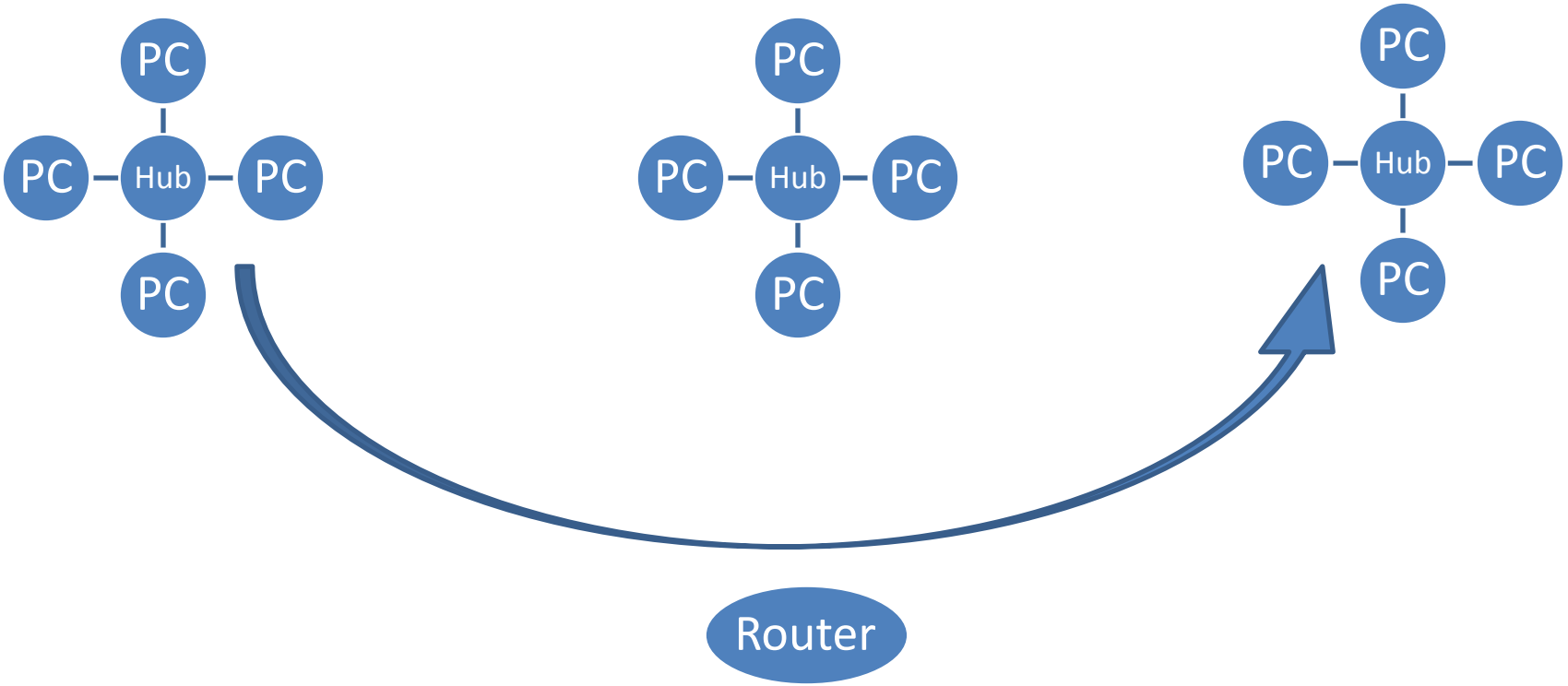
IP Terms

- IHL (Internet Header Length): длина заголовка IP-пакета в 32-битных словах

- TTL (Time To Live): время жизни пакета (число итераций перехода)

Маршрутизатор

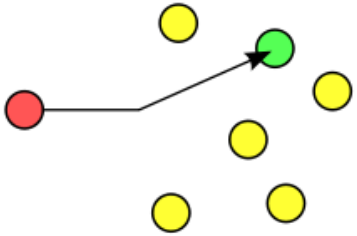
Маршрутизация



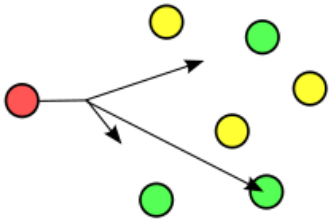
http://ru.wikipedia.org/wiki/Алгоритмы_маршрутизации

Виды маршрутизации

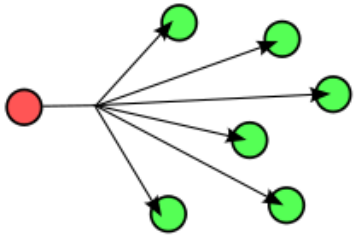
Unicast



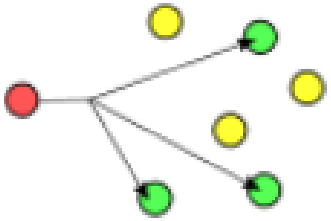
Anycast (DNS)



Broadcast



Multicast



http://ru.wikipedia.org/wiki/Маршрутизация

OCI: Transport Layer

- Для доставки данных (разбитие на блоки, соединение коротких блоков в один)

- TCP
- UDP
- SCTP

TCP (Transmission Control Protocol)

- Надёжная передача данных
- Точка-точка

- TCP-handshake (SYN -> SYN-ACK -> ACK)

- (FTP, SSH, DNS, SMTP, HTTP, IRC, HTTPS, RDP)

http://ru.wikipedia.org/wiki/TCP

TCP

Бит	0 — 3	4 — 9	10 — 15	16 — 31
0	Порт источника			Порт назначения
32	Номер последовательности			
64	Номер подтверждения			
96	Длина заголовка	Зарезервировано	Флаги	Размер Окна
128	Контрольная сумма			Указатель важности
160	Опции (необязательное, но используется практически всегда)			
160/192+	Данные			

UDP (User Datagram Protocol)

- Не гарантирует порядок
- Не гарантирует доставку
- Легковесный

- Системы реального времени

http://ru.wikipedia.org/wiki/UDP

UDP

Биты	0 - 15	16 - 31
0-31	Порт отправителя (Source port)	Порт получателя (Destination port)
32-63	Длина датаграммы (Length)	Контрольная сумма (Checksum)
64-...	Данные (Data)	

?

Boost.Asio C++ Network Programming

- http://habrahabr.ru/post/192284/

Sync, Async

- Синхронность:
  - Просто для разработки
  - Решение простых задач по передаче информации
- Асинхронность:
  - Не тривиально для разработки
  - Решение задач любой сложности

TCP Reference

- boost::asio::io_service
- boost::asio::ip::tcp::acceptor
- boost::asio::ip::tcp::socket
- boost::asio::ip::address::from_string(string)
- boost::asio::ip::tcp::endpoint

Sync TCP Server Side

```
io_service service;

ip::tcp::endpoint ep( ip::tcp::v4() , 50000 );
ip::tcp::acceptor handler(service, ep );

ip::tcp::socket socket(service);
acceptor_.accept( socket );
```

Sync TCP Client Side

```
io_service service;
ip::tcp::socket sock(service);

ip::tcp::endpoint ep(
       ip::address::from_string("127.0.0.1"), 5000);

sock.connect(ep);
```

Async TCP Server Side

```
void start_accept(socket_ptr socket)
{
   acc.async_accept( *socket,
      boost::bind(
          handle_accept,
          socket,
        boost::asio::placeholders::error) );
}
void handle_accept(socket_ptr socket, const
      boost::system::error_code & error)
{
      if ( error ) return;
      // actions with new socket
       socket_ptr new_socket(
             new ip::tcp::socket(service));
       start_accept(new_socket);
}
```

Async TCP Client Side

```
ip::tcp::endpoint ep(from_string("127.0.0.1"), 50000);
sock.async_connect(ep, on_connect);
service.run();

void on_read(const boost::system::error_code &err,
std::size_t bytes)
{
}
void on_connect(const boost::system::error_code &err)
{
      socket.async_read_some(buffer(buff_read),
on_read);
}
```

boost::asio::io_service

```
io_service service;

for ( size_t i = 0; i < N; ++i)
        boost::thread( run_service );

void run_service()
{
        service.run();
}
```

?

Task-Based Engine, Thread-Based Engine

- Thread-Based Engine:
  - Простой код
  - Решение простых задач
  - Для каждой задачи свой поток
- Task-Based Engine
  - Сложный код
  - Решение нетривиальных задач
  - Определённое количество потоков, на каждый тип задач

Thread Based Engine

```
void thread_proc()
{
	// do job
}
for(;dir_it != end_it; ++dir_it)
{
	threads.create_thread( &thread_func );
}
```

Task Based Engine

```
void processor()
{
	while (!stop )
	{
		task t = queue.pop();
		t.do_job();
	}
}
for( size_t i = 0 ; i < number_of_processors ; ++i )
{
	 threads.create_thread( &processor );
}
queue.push( my_task( … ) );
```

Task

```
template< class T >
class task_processor
{
	// next slide
};
class task
{
	friend class task_processor< task >;
	virtual void do_job();
public:
	virtual ~task(){}
};
typedef boost::shared_ptr< task > task_ptr;
```

Task Processor

```
template< class T >
class task_processor
{
    boost::thread_group processors_;
    ts_queue< task_ptr > queue;
    void processing()
    {
        while (!stop)
        {
                task_ptr t = queue.pop(); // magic
                t.do_job();
        }
    }
    task_processor( const size_t N = 4 )
    {
          // create threads
    }
};
```

?