ПО сетевых устройств

Трещановский Павел Александрович, к.т.н.

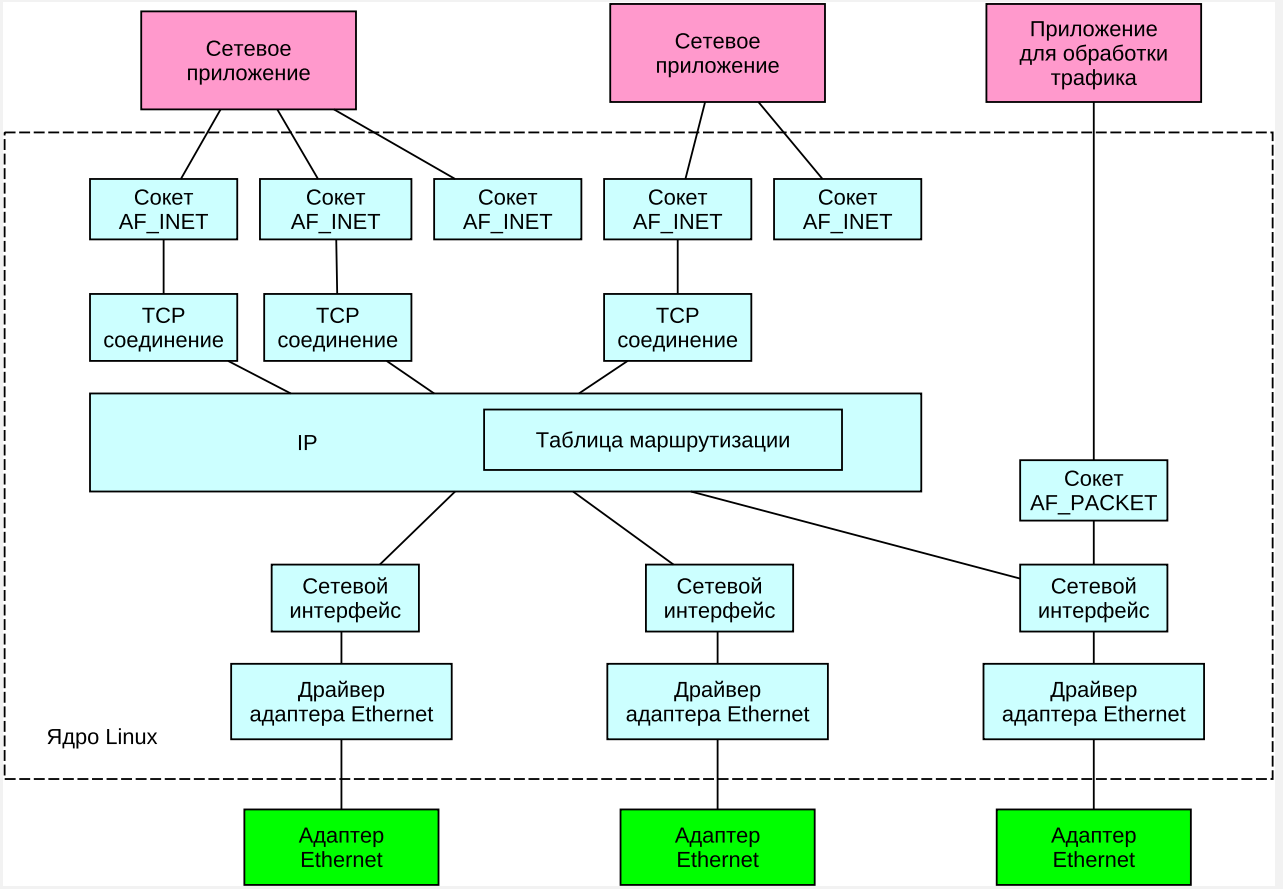
26.05.20

Обработка трафика

Задачи:

- программная реализация коммутатор Ethernet или маршрутизатора IP,
- списки контроля доступа (ACL - access control list),
- глубокий анализ пакетов (DPI - deep packet inspection),
- реализация служебных протоколов (LLDP, STP, UDLD и пр.),
- шифрование трафика
- и др.

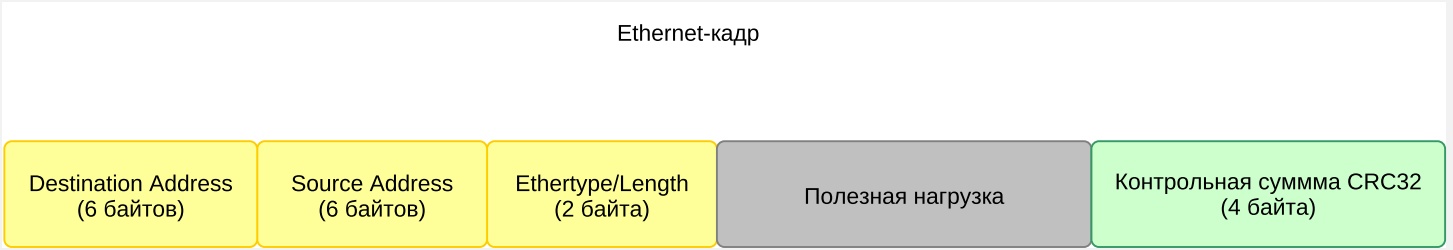
Уровень сетевых интерфейсов



Сокеты семейства AF_PACKET

```
int sock;
struct sockaddr_ll addr;

/* AF_PACKET - семейство адресов канального уровня;
 * SOCK_RAW - тип сокета, при котором приложению передается вместе с
 * данными заголовок канального уровня;
 * EHT_P_ALL - принимать данные любого протокола сетевого уровня. */
sock = socket(AF_PACKET, SOCK_RAW, htons(ETH_P_ALL));

memset(&addr, 0, sizeof(bind_addr));
addr.sll_family   = AF_PACKET;
addr.sll_ifindex  = ifindex; /* Индекс сетевого интерфейса */
/* addr.sll_addr = ... Адрес канального уровня (здесь не используется).
   addr.sll_halen = ...  */
bind(sock, (struct sockaddr*)&addr, sizeof(addr));
```

Структура Ethernet-кадра



- Если значение Ethertype/Length меньше 0x800, то поле содержит длину кадра. В противном случае - тип кадра (IP, ARP и т.д.).
- CRC32 обычно устанавливается и проверяется на аппаратном уровне.
- Размер кадра - от 64 до 1522 байтов.
- ff:ff:ff:ff:ff:ff - широковещательный адрес. Кадр с таким адресом назначения доставляется всем узлам сети.

Стандартная структура `struct ethhdr`

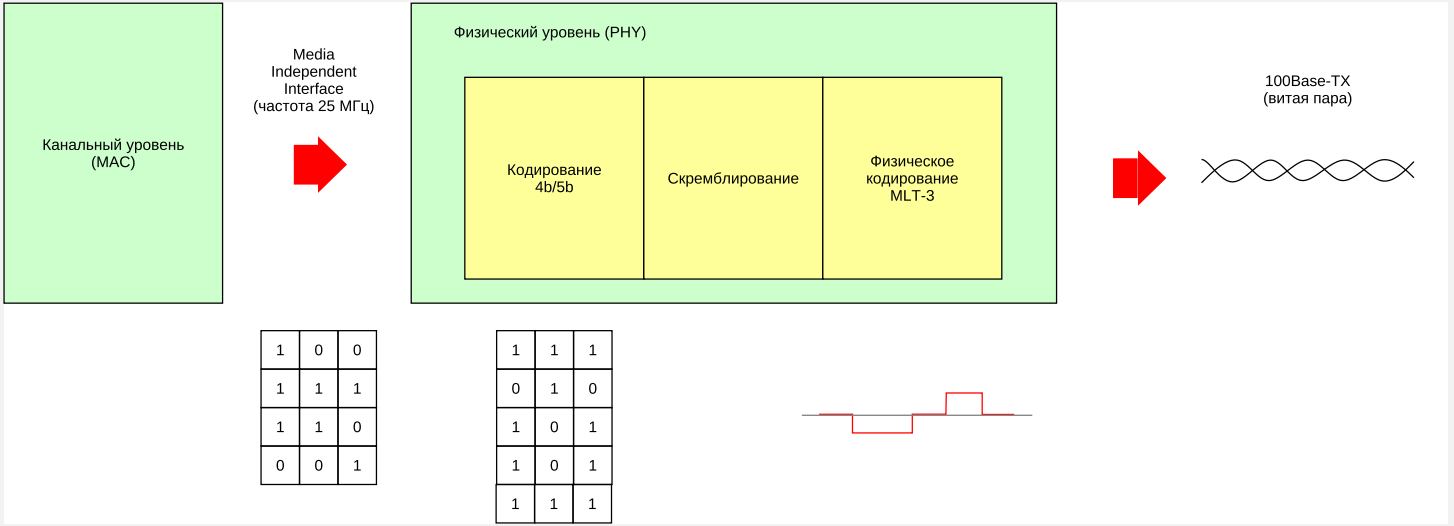
```
#define ETH_ALEN 6

struct ethhdr {
       uint8_t h_dest[ETH_ALEN];
       uint8_t h_source[ETH_ALEN];
       uint16_t h_proto;
};

struct ether_addr {
       uint8_t ether_addr_octet[ETH_ALEN];
};
extern char *ether_ntoa(const struct ether_addr *addr);

void some_function(void) {
       struct ethhdr *eth_header;
       {
              char sa_str[20], da_str[20];
              printf("Source: %s destination %s\n",
                     ether_ntoa_r((struct ether_addr *)&eth_header->h_source, sa_str),
                     ether_ntoa_r((struct ether_addr *)&eth_header->h_dest, da_str));
       }
}
```

Прием кадров из сокета AF_PACKET

```
int sock;
struct sockaddr_ll rx_addr;
socklen_t addrlen = sizeof(src_addr);
char pkt_buf[2048];
struct ethhdr *eth_header;
struct ether_addr da;

sock = socket(AF_PACKET, SOCK_RAW, 0);

...

/* rx_addr - информация об отправителе кадра. */
recvfrom(sock, pkt_buf, sizeof(pkt_buf), 0, &rx_addr, &addrlen);

/* Игнорируем передаваемые кадры. */
if (rx_addr.sll_pkttype == PACKET_OUTGOING)
       return 0;

eth_header = pkt_buf;

memcpy(&da, eth_header->h_dest, ETH_ALEN);
printf("Destination address: %s\n", ether_ntoa(&da));
```

Коммутаторы Ethernet

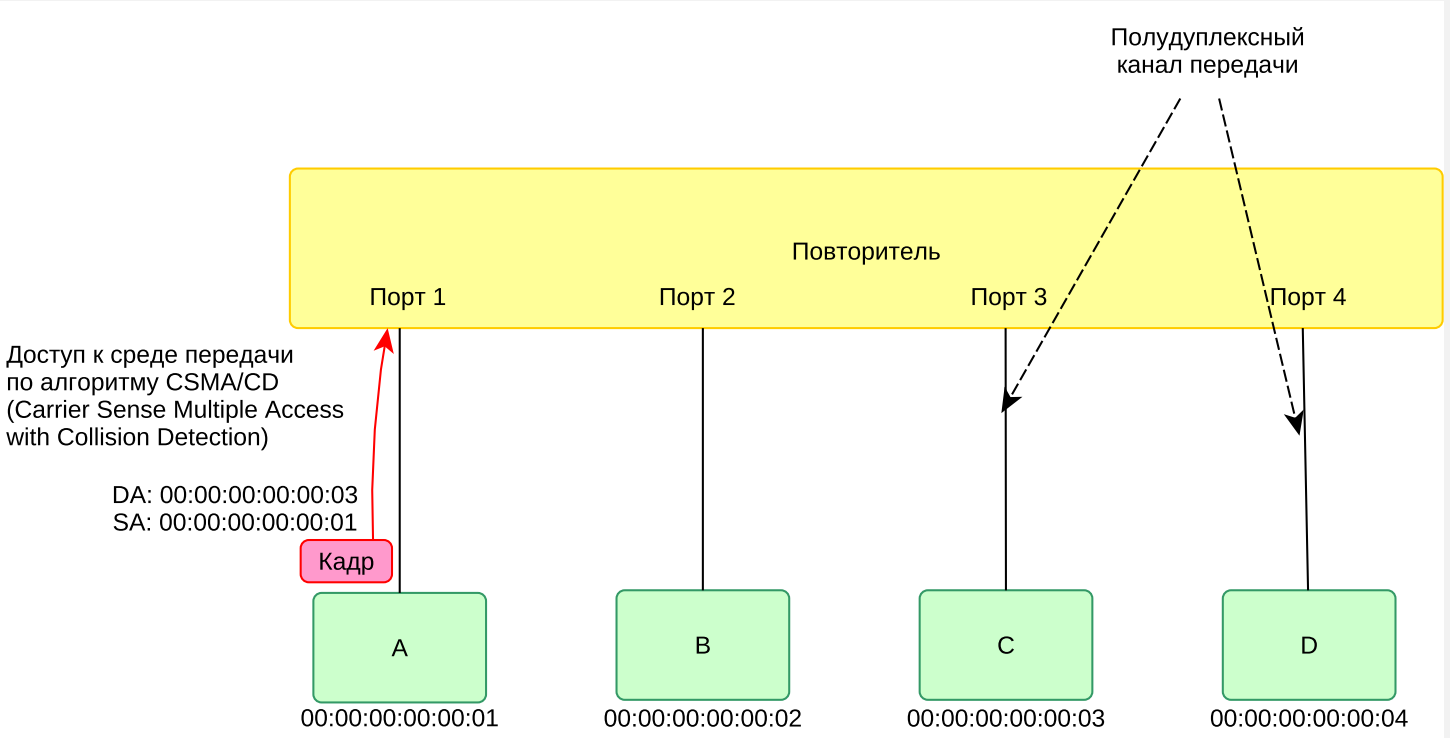
Канальный и физический уровни



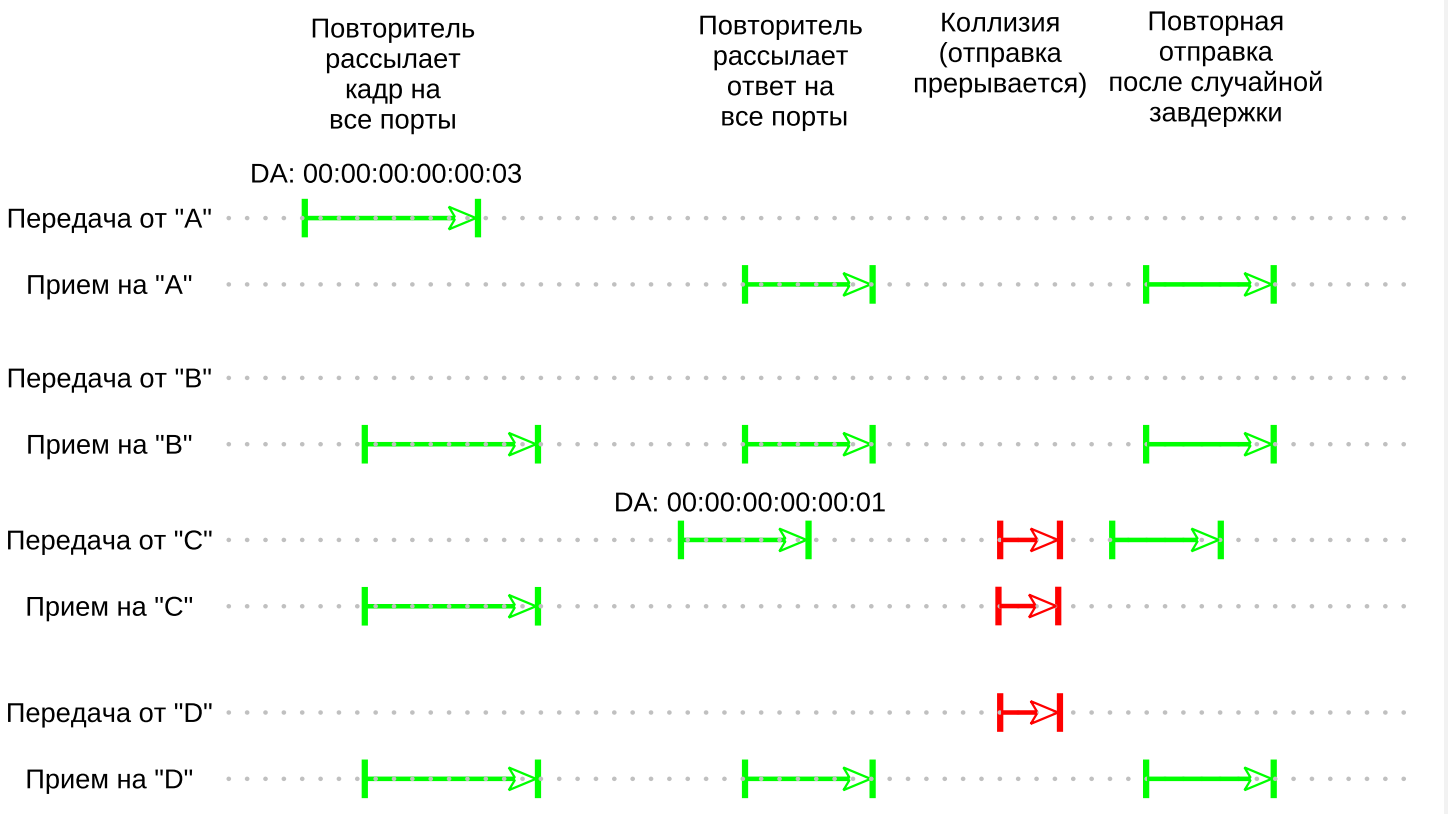
Терминология

- В сети Ethernet передаются *кадры* (англ. frame). В IP - дейтаграммы, TCP - сегменты и др.
- Узлы объединяются в сеть Ethernet с помощью *повторителей* и *коммутаторов* (switch).
- Коммутация производится между *портами*.
- MAC (Media Access Control) - доступ к среде передачи. Название уровня абстракции в стандарте Ethernet. Существуют различные методы доступа в как сетях Ethernet, так и в сетях на основе других технологий.
- MAC-адрес - адрес канального уровня. Обычно термин применяется к Ethernet.
- CSMA/CD (Carrier Sense Multiple Access with Collision Detection) - один из методов доступа в сети Ethernet.

Сеть на основе повторителя



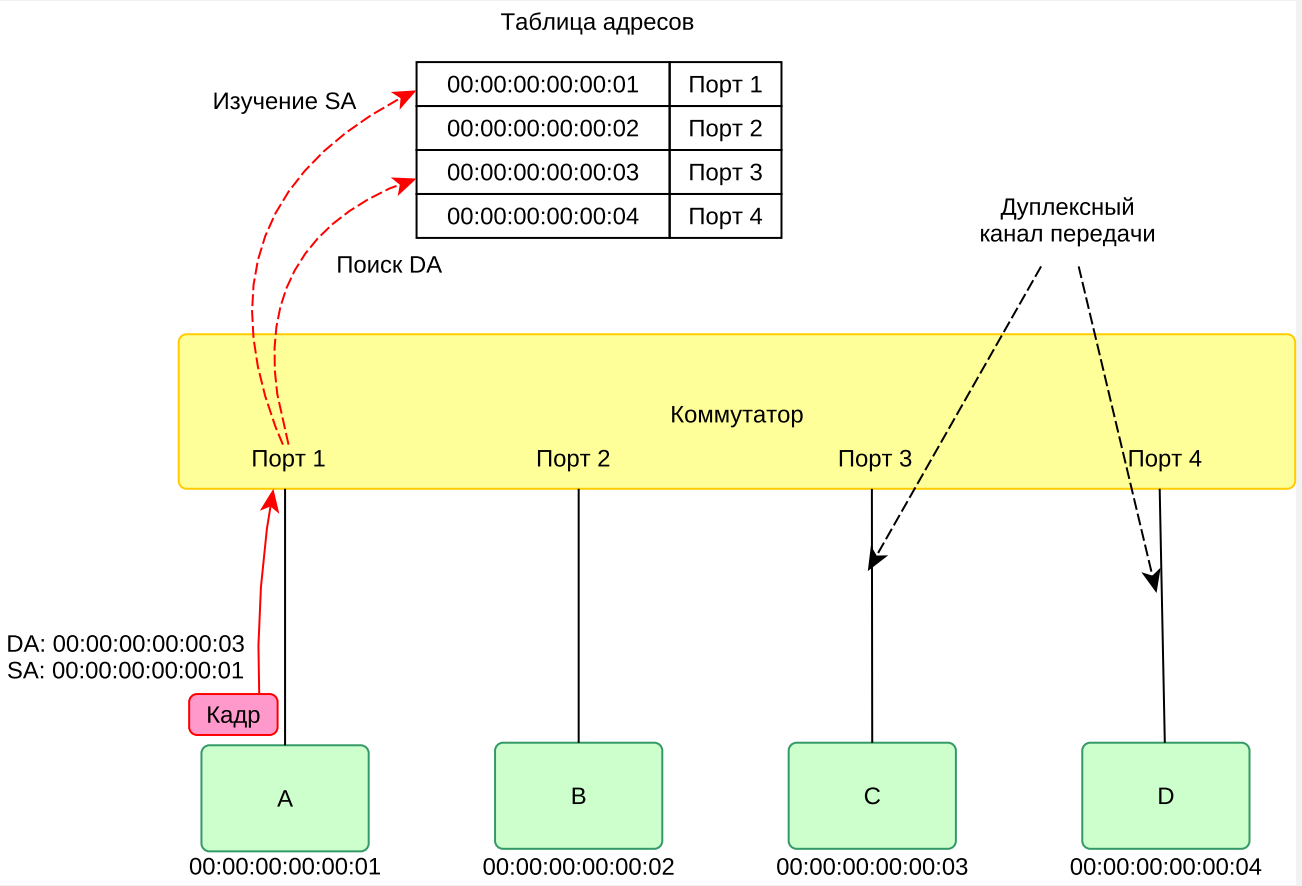
Передача кадров через повторитель



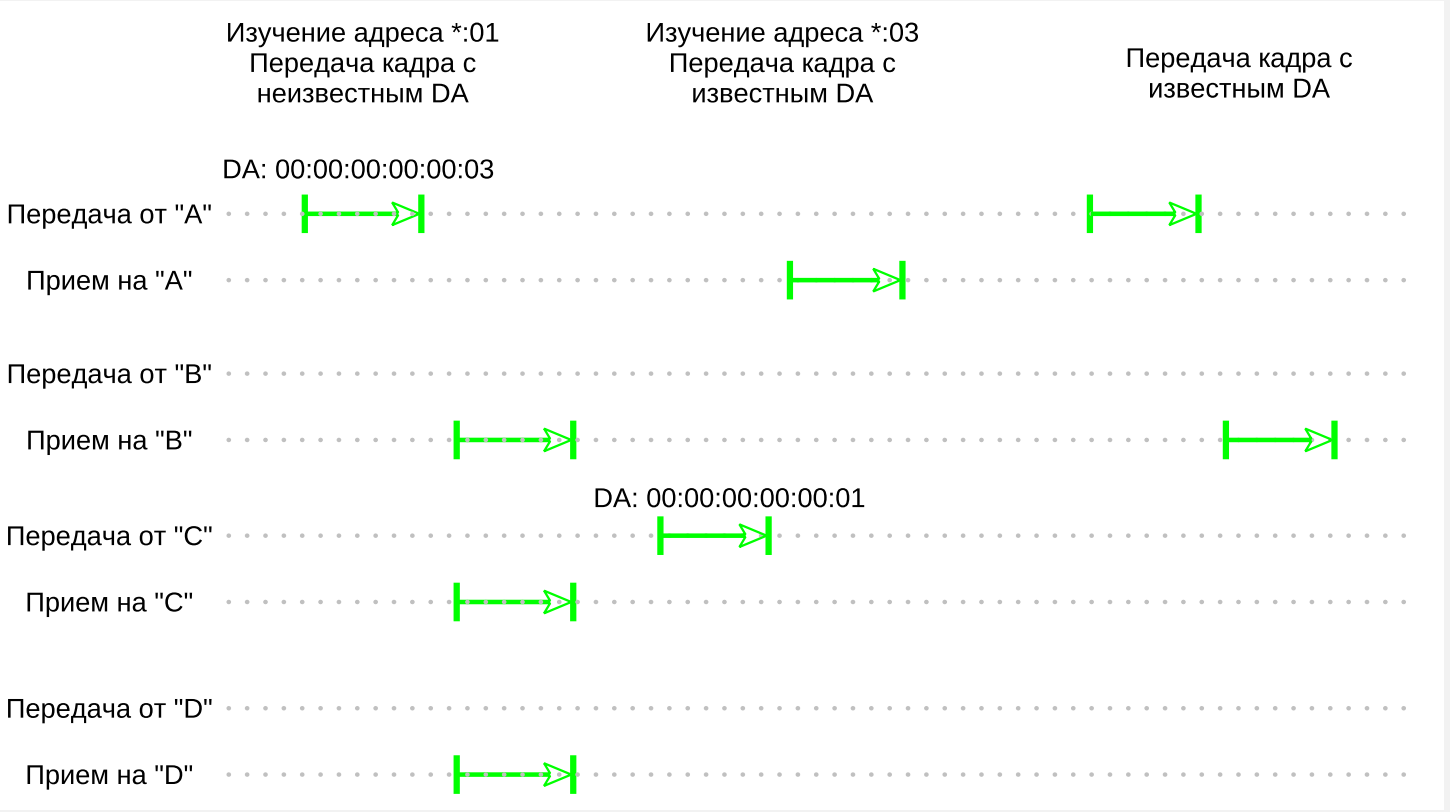
Замечания

- Повторитель не буферизирует и не анализирует принимаемые принимаемые кадры. Принимаемые данные побайтово повторяются на выходе.
- Повторитель всегда передает данные на все выходные порты.
- Одновременно передавать данные может только один узел сети. Если несколько узлов начинают передачу в один момент времени, то наступает *коллизия* и передача прекращается. Повторная попытка передачи производится после случайной паузы.
- Коллизии и широковещательные рассылки не позволяют полностью использовать пропускную способность сети.
- Повторитель не имеет своего MAC-адреса и не „виден” другим устройствам сети.

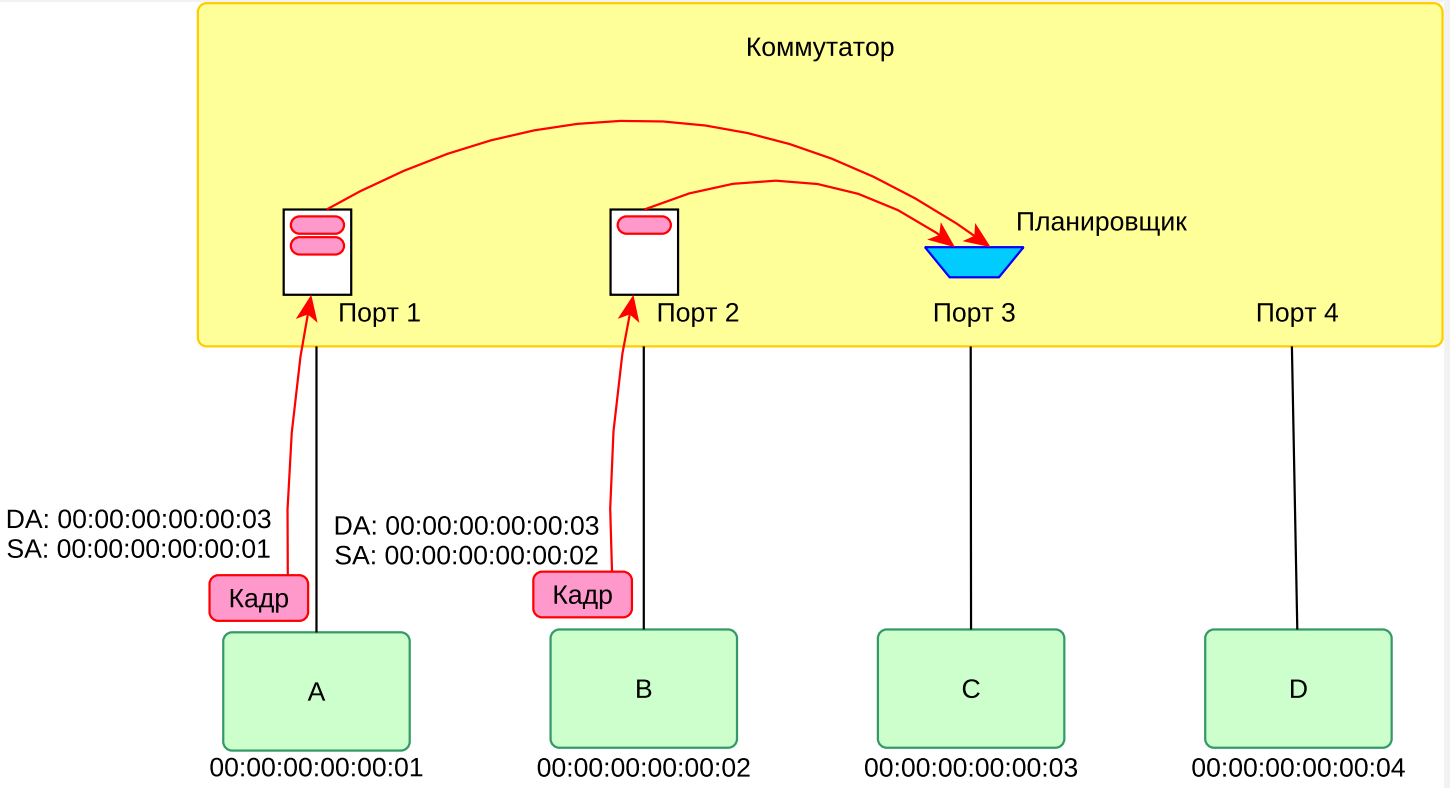
Сеть на основе коммутатора



Передача кадров через коммутатор



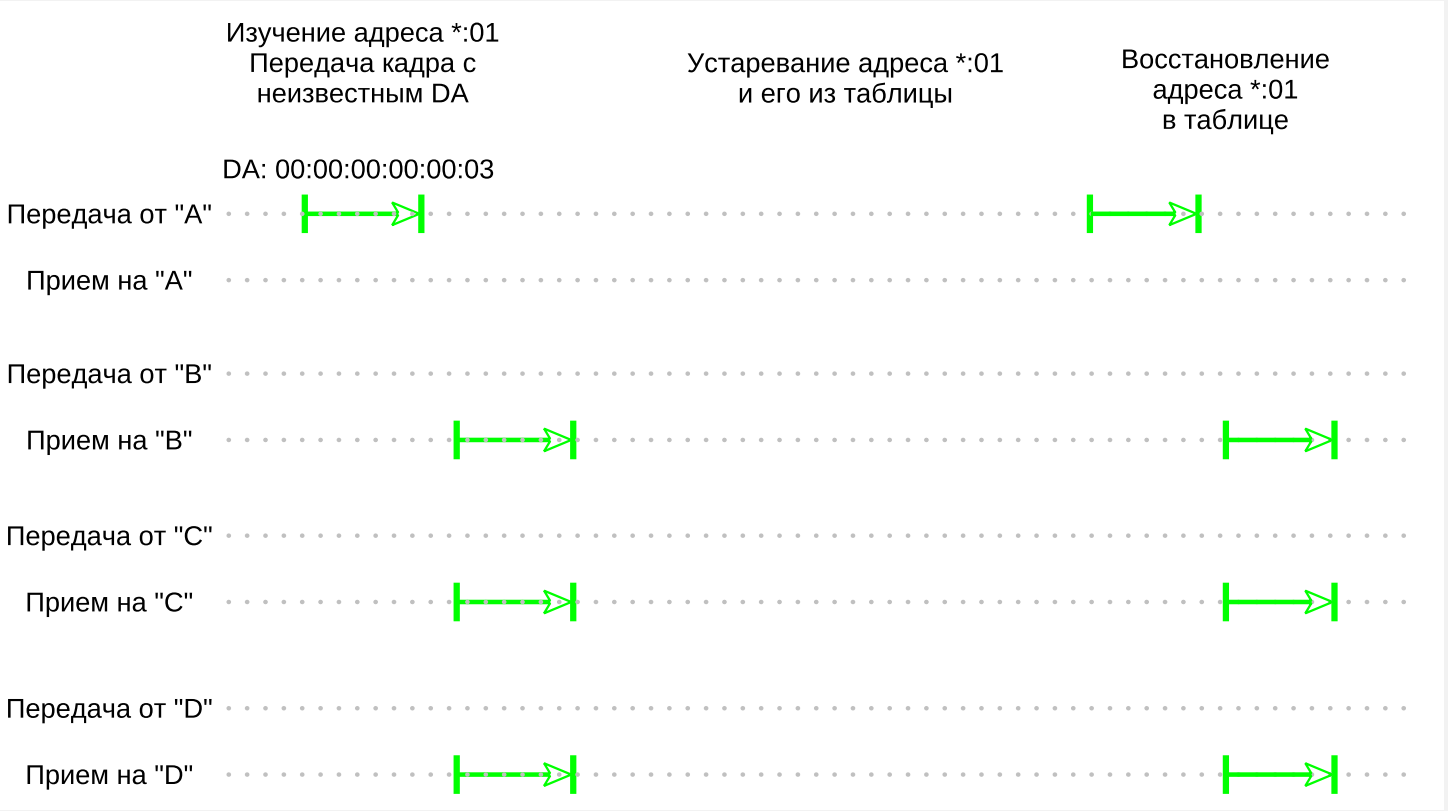
Очереди в коммутаторе



Правила работы таблицы MAC-адресов

- Каждая запись в таблице - кортеж (MAC-адрес, номер порта).
- При приеме каждого кадра производится изучение (learning) его адреса *источника*. Для него создается запись в таблице, куда заносится номер входного порта.
- Если запись уже была, номер порта обновляется.
- Далее по таблице производится определение выходного порта. Для осуществляется поиск записи по адресу *назначения* принятого кадра. Если запись есть, то номер порта в ней является номером выходного порта для кадра.
- Если записи нет, то кадр передается на все порты, кроме входного.
- Поиск выходного порта не производится для широковещательных кадров.

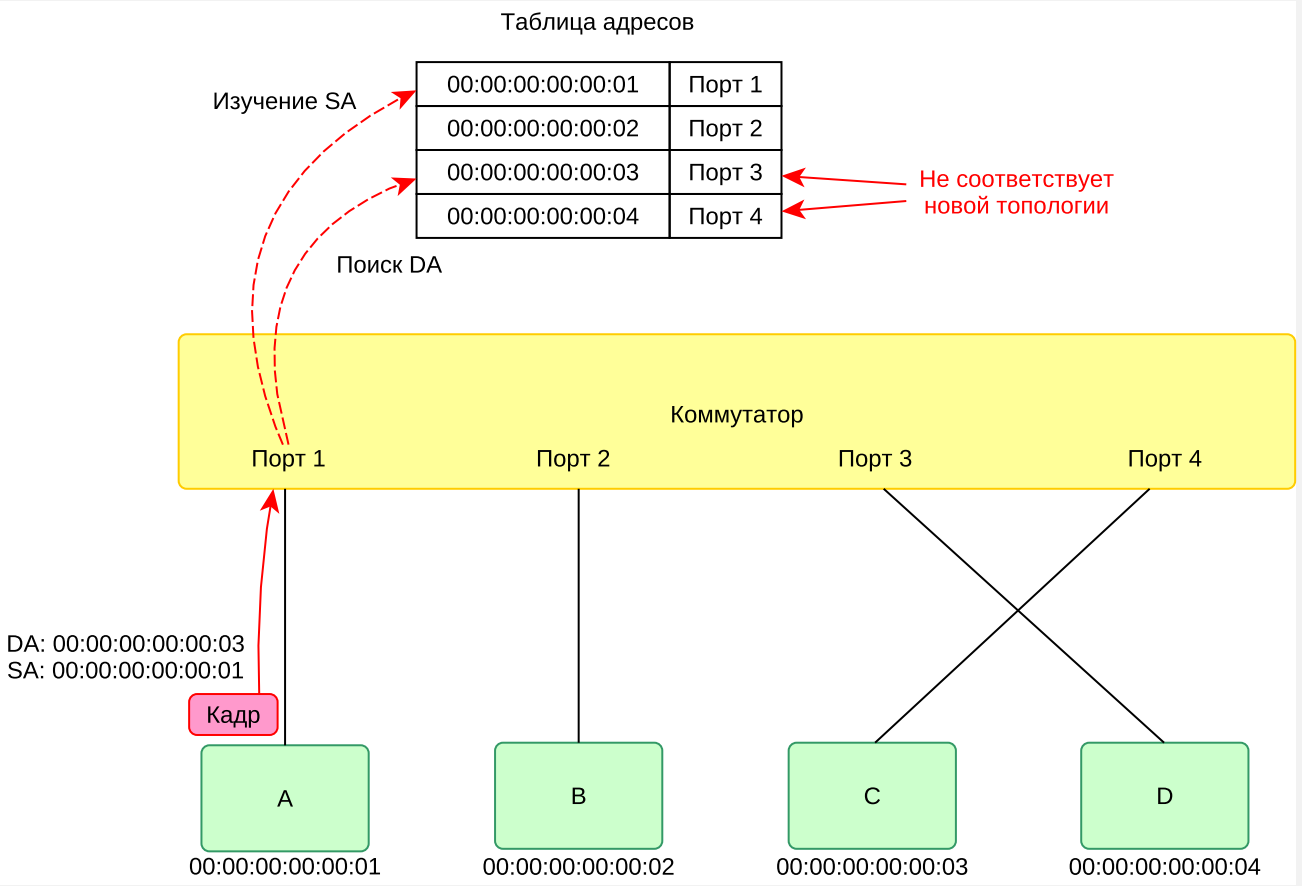
Передача кадров при устаревании адреса



Замечания

- Таблица MAC-адресов позволяет уменьшить количество широковещательных рассылок. Благодаря этому коммутатор эффективнее использует пропускную способность сети.
- Недостаток таблицы MAC-адресов - при изменении топологии сети часть записей становится недействительными. Из-за этого возможна потеря связи между узлами.
- Смысл устаревание записей в таблице - гарантировать удаление недействительных записей и последующее создание новых (действительных) записей.
- Так как коммутатор анализирует заголовок кадра, все принимаемые кадры буферизируются (принцип работы store and forward).

Изменение топологии

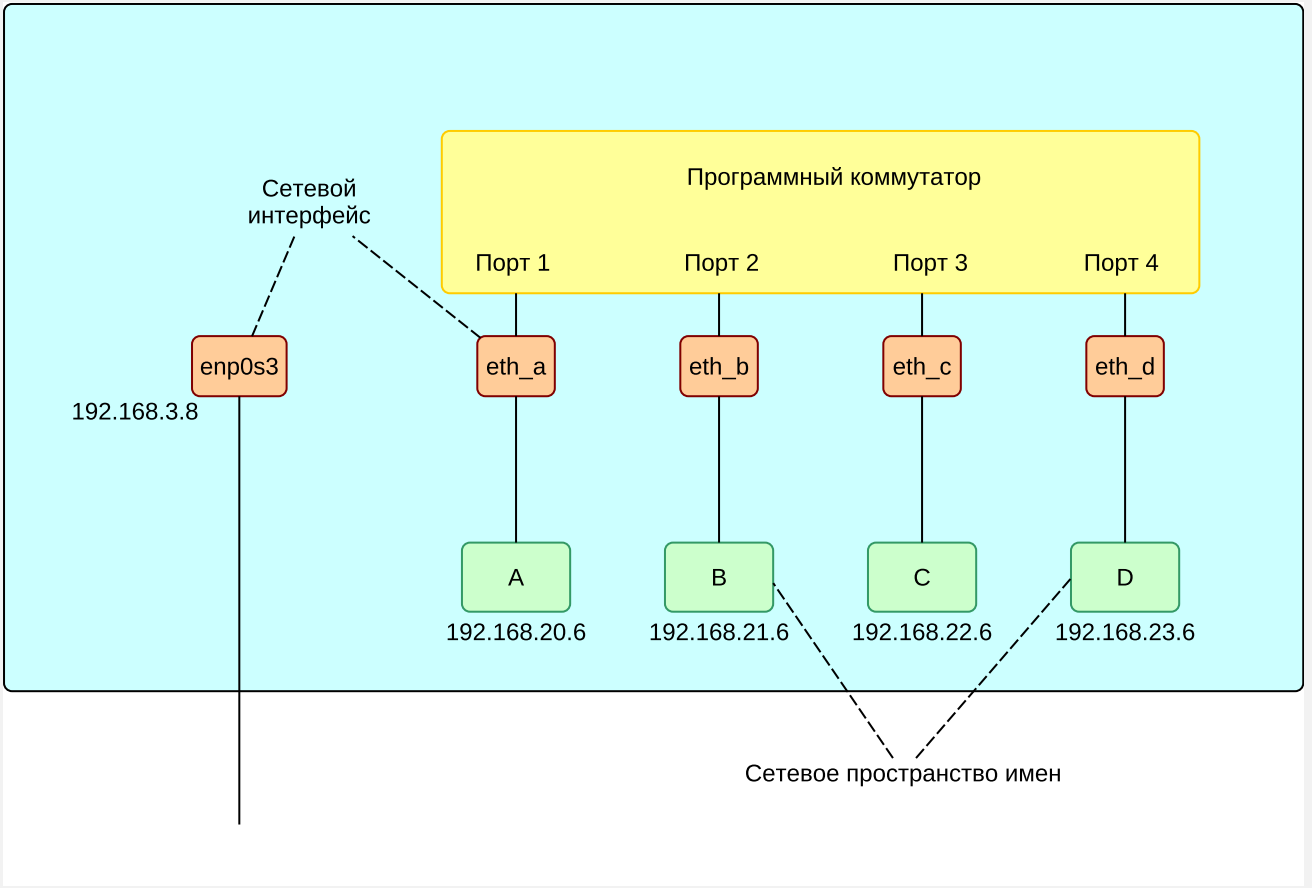


Передача кадров при изменении топологии

Программная реализация

- Коммутатором является приложение Linux.
- Портами коммутатора являются сетевые интерфейсы. Прием и передача осуществляется через сокеты AF_PACKET.
- Входной очередью порта является очередь пакетов сокета.
- Таблица MAC-адресов реализуется с помощью словаря `struct avl_tree`. Ключом является MAC-адрес.
- Прием кадров и устаревание адресов осуществляется с помощью цикла обработки событий.

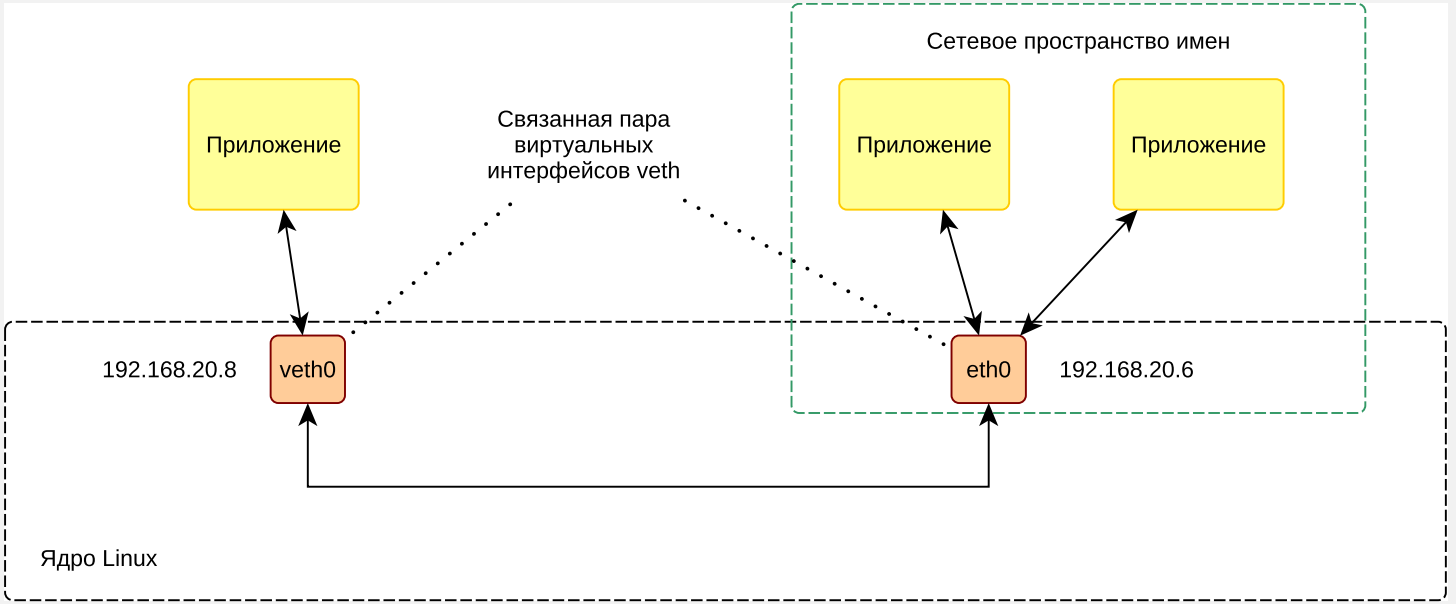
Виртуальная сеть



Сетевые пространства имен

- Пространства имен - механизм виртуализации, разделяющий ресурсы операционной системы на несколько частей. Доступ к каждой из них разрешается только определенной группе процессов. Примеры ресурсов: номера пользователей, файловая система, средства межпроцессного обмена.
- Сетевое пространство имен виртуализирует сетевой стек Linux (таблицу маршрутизации, сокеты и др.). Каждый сетевой интерфейс доступен только в одном сетевом пространстве имен.
- Каждый процесс находится в одном пространстве имен каждого типа.
- Назначения пространства имен для каждого типа происходит независимо. 2 процесса могут находится в разных пространствах типа A, но при этом в одном пространстве типа B.
- Если у группы процессов совпадают все типы пространств имен, то эта группа находится в *контейнере*.
- В отличие от виртуальной машины, которая виртуализирует весь компьютер, контейнер виртуализирует среду исполнения процесса.

Виртуальные интерфейсы



Запуск приложений в определенном пространстве имен

Вывод списка интерфейсов в сетевом пространстве имен node_a:

```
# ip netns exec node_a ip addr
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
4: eth0@if39: <NO-CARRIER,BROADCAST,MULTICAST,UP,M-DOWN> ...
    link/ether 68:eb:c5:00:01:02 brd ff:ff:ff:ff:ff:ff
    inet 192.168.20.6/24 scope global ge2
       valid_lft forever preferred_lft forever
```

Запуск командного интерпретатора в пространстве имен node_a:

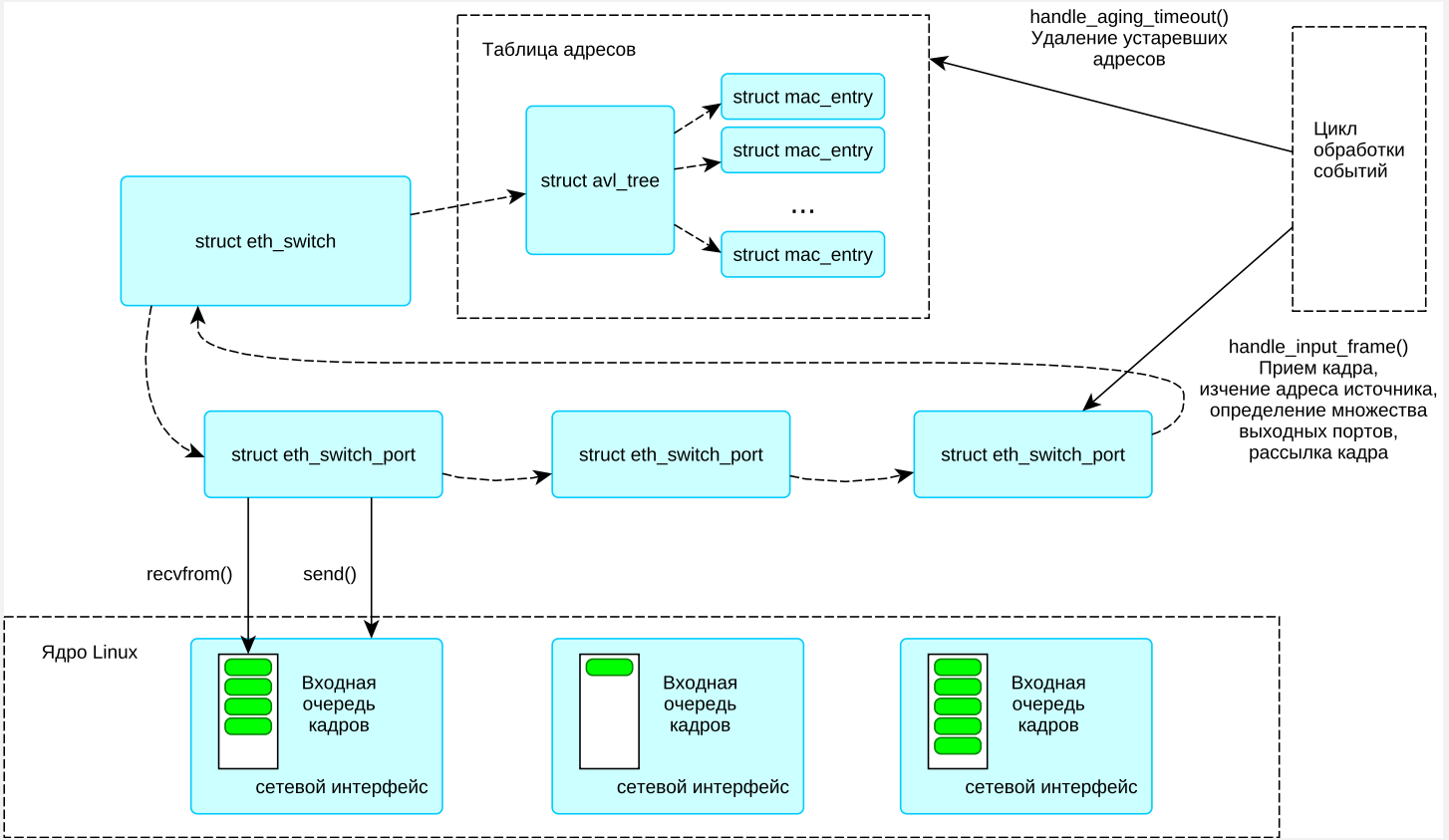
```
# ip netns exec node_a bash
```

Замечание. Имена node_a и eth_a специфичны для AngtelOS. На других системах именование пространств имен будет отличаться.

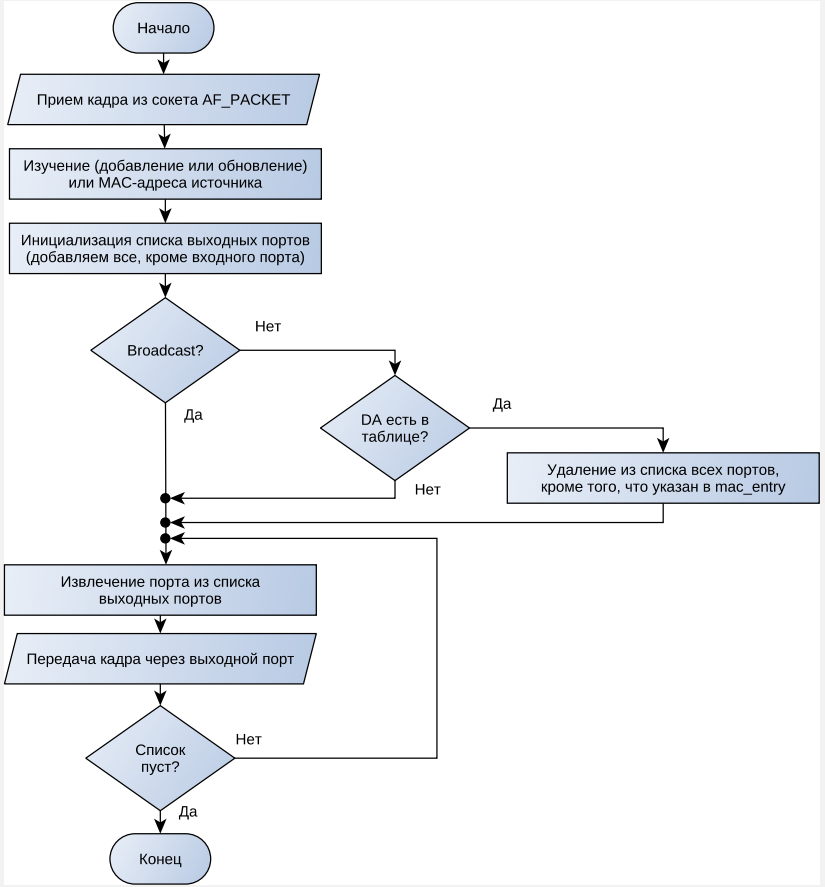
Реализация таблицы адресов с помощью словаря

```
/* Функция сравнения узлов двоичного дерева. */
int avl_maccmp(const void *k1, const void *k2, void *ptr)
{
	const struct ether_addr *mac1 = k1, *mac2 = k2;
	int64_t mac1_int = 0, mac2_int = 0;

	memcpy(&mac1_int, mac1, sizeof(*mac1));
	memcpy(&mac2_int, mac2, sizeof(*mac2));
	if (mac1_int - mac2_int > 0)
		return 1;
	else if (mac1_int - mac2_int < 0)
		return -1;
	else
		return 0;
}

int main(int argc, char *argv[])
{
	struct avl_tree mac_table;
	/* avl_maccmp вместо avl_strcmp! */
	avl_init(&mac_table, avl_maccmp, false, NULL);
}
```

Структура программного коммутатора



Алгоритм обработки кадра



Протоколы ARP и ICMP

- Утилита ping использует протокол ICMP. ping отправляет запрос (echo request) и ожидает получить ответ (echo reply). ICMP пакеты передаются внутри IP-дейтаграмм.
- ping отправляет запрос на определенный IP-адрес.
- Для передачи IP-дейтаграммы ядро должно знать соответствие между IP-адресом адресата и его MAC-адресом. Это соответствие устанавливает протокол ARP.
- Если в какой-то момент IP-адрес назначения неизвестен, ядро автоматически отправляет ARP-запрос. Адресат отправляет ARP-ответ, содержащий его MAC-адрес.
- Полученная информация заносится в ARP-таблицу в ядре.
- Записи в ARP-таблице устаревают через определенное время.

Замечания по задаче

- Порты коммутатора задаются через аргументы командной строки. Для каждого порта указывается ifindex соответствующего интерфейса.
- ifindex интерфейсов можно узнать командой `ip addr`.
- Работа коммутатора проверяется утилитами ping и tcpdump.
- ping запускается в отдельном сетевом пространстве имен. Аргументом является IP-адрес из другого пространства, *не* IP-адрес коммутатора.
- При реализации повторителя возможность коллизий игнорируется.