«Синхроком»

Система протоколов взаимодействия по сети Ethernet

Описание протокола

Синхроком-Сеанс (SCS)

Аннотация

Настоящий документ описывает сетевой протокол стека протоколов «Синхроком» под названием Синхроком-Сеанс (SCS). SCS - протокол уровня L5 модели OSI/ISO. Приложения, неограниченные требованием жесткого реального времени, запущенные в гостевой ОС, могут использовать различные существующие сетевые протоколы, например на базе IP, которые туннелируются в сети «Синхроком».

СОДЕРЖАНИЕ

Введение

1.1 Мотивация

Настоящий документ описывает сетевой протокол стека протоколов «Синхроком» под названием Синхроком-Сеанс (SCS). SCS - протокол уровня L5 модели OSI/ISO. Приложения, неограниченные требованием жесткого реального времени, запущенные в гостевой ОС могут использовать различные существующие сетевые протоколы, например на базе IP, которые туннелируются в сети «Синхроком».

1.2. Сфера действия протокола

Протокол SCS ограничивается доставкой Ethernet пакетов из гостевой ОС до приложения протокола SCTPA, работающего в микроядре, называемого сетевым мостом гипервизора. Для гостевой ОС SCS представляется как Ethernet драйвер, реализующий Ethernet Interface.

Стек протоколов «Синхроком» относится к L3-L6 слоям модели OSI/ISO.

№	Уровни модели OSI/ISO	Уровни стека протоколов «Синхроком»
7	Прикладной	Прикладные службы
6	Представления	Синхроком-Дата (SCD) - протокол представления данных
5	Сеансовый	Синхроком-Сеанс (SCS)- протокол туннелирования. Приложения, неограниченные требованием жесткого реального времени, могут использовать различные существующие сетевые протоколы на базе IP, которые туннелируются в сети «Синхроком», например: UDP, RTP и т.д.
4	Транспортный	Синхроком-Транспорт (SCTP) - двухфазный протокол передачи данных по сети SCA, где: синхронная фаза SCTPS - фаза передачи данных в режиме жесткого реального времени; асинхронная фаза SCTPA - фаза передачи данных в режиме мягкого реального времени.
3	Сетевой	Синхроком-Адрес (SCA)
2	Канальный	IEEE 802.2 на платформе микроядра L4
1	Физический	

Рисунок 1 - Стек протоколов «Синхроком»

1.3. Интерфейсы

Этот протокол вызывается протоколами взаимодействия “хост-хост” гостевой ОС и сам вызывает функции локального сетевого протокола транспортного уровня L4 - SCTPA для передачи по нему Ethernet фреймов.

1.4. Работа протокола

Модули SCS используют адреса из Ethernet фреймов для «оборачивания» их в датаграммы SCTPA и передачи в направлении получателя. Между адресацией уровня гостевых ОС и уровня микроядра существует строгое соответствие. Также модули SCS отвечают за прием входящих Ethernet фреймов в SCTPA пакетах и передачу их сетевому интерфейсу гостевой ОС.

2. Обзор

2.1. Связь с другими протоколами

На Рисунок 3 показаны связи стека протоколов «Синхроком» с другими протоколами и приложениями.

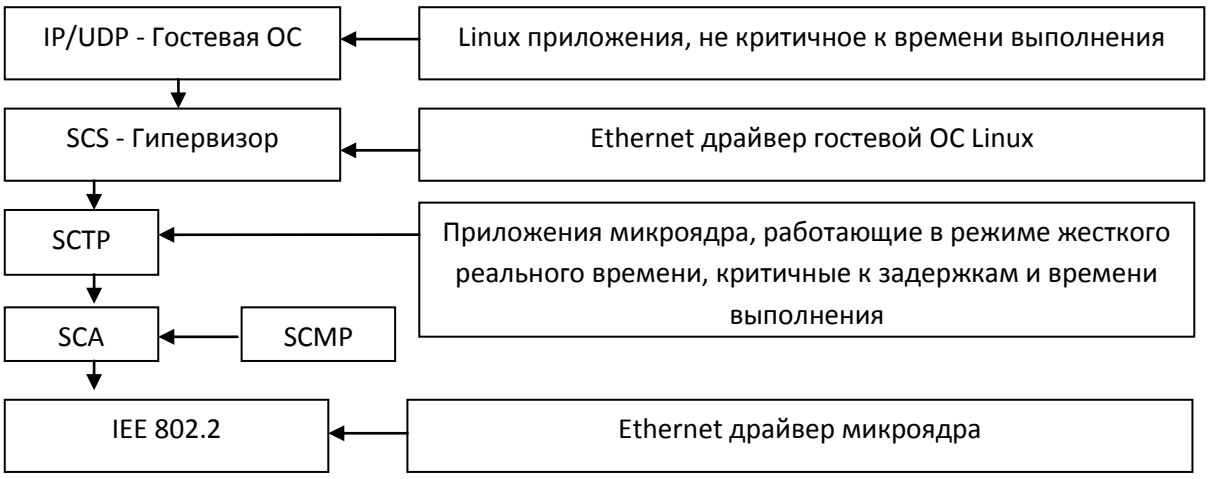

Рисунок 4 - Стек протоколов «Синхроком»

Протокол SCS является драйвером сетевого интерфейса гостевой ОС и взаимодействует с протоколами вышележащего уровня (IP) и с нижележащими протоколами в окружении микроядра: SCA, SCTP.

2.2. Модель работы протокола

Модель передачи Ethernet датаграмм из гостевой ОС в сеть SCA можно проиллюстрировать описанным ниже сценарием.

Сетевой драйвер гостевой ОС отправляет и получает данные Ethernet через драйвер Ethernet гипервизора микроядра, который в свою очередь взаимодействует с SCTPA программой микроядра.

2.3. Функциональное описание

Модули SCS размещаются на хостах бортовой сети. Протокол SCS обеспечивает механизмы туннелирования трафика гостевой ОС.

2.3.1 Адресация

Из IP датаграммы вычленяется адрес получателя и преобразуется в SCA адрес, после чего формируется SCTPA пакет и отправляется получателю. Пользователи протокола являются сетевые службы ядра гостевой операционной системы.

3. Спецификация

3.1 Основная структура SCS протокола ethernetif

```
struct ethernetif {

        struct eth_addr *ethaddr;

        SCTPA *connection;

};
```

Структура для хранения приватных данных, необходимых для работы ethernet интерфейса.

3.2 Функция инициализации SCS для микроядра low_level_init

3.2.1 Аргументы

- netif - lwip структура сетевого интерфейса

3.2.2 Возвращаемый результат

- нет

3.2.3 Псевдокод

```
static void low_level_init(struct netif *netif) {

        struct ethernetif *ethernetif = netif->state;

        /* set MAC hardware address length */

        netif->hwaddr_len = ETHARP_HWADDR_LEN;


        /* Установка MAC адреса сетевого интерфейса гостевой ОС*/

        netif->hwaddr[0] = ;

        netif->hwaddr[5] = ;

        /* Максимально возможный размер пакета */

        netif->mtu = 1500;

        /* Возможности устройства */

        /* Не устанавливать NETIF_FLAG_ETHARP т.к. поддерживаем только IP*/
```

```
        netif->flags = NETIF_FLAG_BROADCAST
        /*                  | NETIF_FLAG_ETHARP */
                          | NETIF_FLAG_LINK_UP;
        /* Инициализация SCTPA*/
        sctpa = SCTPA::open();
}
```

3.3 Интерфейс функции низкоуровневого вывода low_level_output

Функция выполняет фактическую передачу пакета. Пакет содержится в pbuf, который передается в качестве аргумента функции.

3.3.1 Аргументы

- netif - lwip структура сетевого интерфейса
- p пакет на отправку (IP пакет)

3.3.2 Возвращаемый результат

- ERR_OK если пакет будет отправлен
- err_t если возникли ошибки

3.3.3 Псевдокод

```
static err_t low_level_output(struct netif *netif, struct pbuf *p) {
        struct ethernetif *ethernetif = netif->state;
        struct pbuf *q;
        initiate transfer();
#if ETH_PAD_SIZE
        pbuf_header(p, -ETH_PAD_SIZE);
#endif
        for(q = p; q != NULL; q = q->next) {
        /* Отправить данные из pbuf в SCTPA сокет, один pbuf в одно время. Размер данных, содержащихся в pbuf разместить в q ->len*/
        отправить данные в SCTPA сокет(q->payload, q->len);
        }
        получить подтверждение от SCTPA сокета о поставке на отправку методом sent();
#if ETH_PAD_SIZE
```

```
        pbuf_header(p, ETH_PAD_SIZE); /* reclaim the padding word */
#endif


        LINK_STATS_INC(link.xmit);

        return ERR_OK;
}
```

3.4 Интерфейс функции низкоуровневого ввода low_level_input

Для работы с функцией необходимо выделить в памяти pbuf и передать байты входящего пакета в pbuf.

3.4.1 Аргументы

- netif - lwip структура сетевого интерфейса

3.4.2 Возвращаемый результат

- pbuf буфер с данными
- NULL при ошибках в работе с памятью

3.4.3 Псевдокод

```
static struct pbuf* low_level_input(struct netif *netif) {

        struct ethernetif *ethernetif = netif->state;

        struct pbuf *p, *q;

        u16_t len;

        /* Определить размер пакет и сохранить значение в переменной "len". */

        len = ;
#if ETH_PAD_SIZE

        len += ETH_PAD_SIZE;
#endif

        p = pbuf_alloc(PBUF_RAW, len, PBUF_POOL);

        if (p != NULL) {


#if ETH_PAD_SIZE

        pbuf_header(p, -ETH_PAD_SIZE);
```

```
#endif

        /* Итерация по цепочке pbuf до полного прочтения записи пакета в буфере*/

        for(q = p; q != NULL; q = q->next) {

                read data into(q->payload, q->len);

        }

         убедиться что пакет прочитан полностью;

#if ETH_PAD_SIZE

        pbuf_header(p, ETH_PAD_SIZE);

#endif

                LINK_STATS_INC(link.recv);

        } else {

                drop packet();

                LINK_STATS_INC(link.memerr);

                LINK_STATS_INC(link.drop);

        }

        return p;

}
```

3.5 Функция инициализации сетевого интерфейса для гостевой ОС ethernetif_init

Функция вызывается в начале программы для настройки сетевого интерфейса. Предварительно должна быть вызвана функция low_level_init (), чтобы сделать фактическое настройку аппаратного обеспечения. Эта функция должна быть передана в качестве параметра для netif_add ().

3.5.1 Аргументы

- netif - lwip структура сетевого интерфейса

3.5.2 Возвращаемый результат

- ERR_OK если пакет будет отправлен
- err_t если возникли ошибки

3.5.3 Псевдокод

```
err_t ethernetif_init(struct netif *netif) {

        struct ethernetif *ethernetif;
```

```
        LWIP_ASSERT("netif != NULL", (netif != NULL));
        ethernetif = mem_malloc(sizeof(struct ethernetif));
        if (ethernetif == NULL) {
                LWIP_DEBUGF(NETIF_DEBUG, ("ethernetif_init: нет памяти\n"));
                return ERR_MEM;
        }
#if LWIP_NETIF_HOSTNAME
        netif->hostname = "lwip";
#endif /* LWIP_NETIF_HOSTNAME */
        /* Инициализация snmp переменных и счетчиков внутри структуры netif.*/
        NETIF_INIT_SNMP(netif, snmp_ifType_ethernet_csmacd,
                                LINK_SPEED_OF_YOUR_NETIF_IN_BPS);
        netif->state = ethernetif;
        netif->name[0] = IFNAME0;
        netif->name[1] = IFNAME1;
        /* Напрямую использовать etharp_output()чтобы сохранить вызов функции*/
        netif->output = etharp_output;
#if LWIP_IPV6
        netif->output_ip6 = ethip6_output;
#endif /* LWIP_IPV6 */
        netif->linkoutput = low_level_output;
        ethernetif->ethaddr = (struct eth_addr *)&(netif->hwaddr[0]);
        low_level_init(netif);
return ERR_OK;
}
```

3.6 Интерфейс функции ввода сетевого интерфейса для гостевой ОС ethernetif_input

Эта функция должна быть вызвана, когда пакет готов для чтения из интерфейса. Он использует функцию low_level_input (), который должен обрабатывать фактический прием байтов из сетевого

интерфейса. Тогда тип принятого пакета может быть определен и соответствующая функция ввода вызвана.

3.6.1 Аргументы

- netif - lwip структура сетевого интерфейса

3.6.2 Возвращаемый результат

- нет

3.6.3 Псевдокод

```
static void ethernetif_input(struct netif *netif) {

        struct ethernetif *ethernetif;

        struct eth_hdr *ethhdr;

        struct pbuf *p;

        ethernetif = netif->state;

        /* переместить полученный пакет в новый pbuf */

        p = low_level_input(netif);

        /*если нечего читать, то выход */

        if (p == NULL) return;

        /* указатель на данные пакета, начинающегося с заголовка Ethernet */

        ethhdr = p->payload;

        switch (htons(ethhdr->type)) {

        /* Предварительно поддерживаем только IP*/

        case ETHTYPE_IP:

                /* полный пакет отправить в tcpip_thread процесс*/

                if (netif->input(p, netif)!=ERR_OK) {

                LWIP_DEBUGF(NETIF_DEBUG, ("ethernetif_input: IP ошибка\n"));

                pbuf_free(p);

                p = NULL;

        }

        break;

        default:
```

```
                pbuf_free(p);

                p = NULL;

                break;

        }

}
```

3.7 Интерфейс функции вывода сетевого интерфейса для гостевой ОС ethernetif_output

Спецификацией не детализируется.