```
.. SPDX-License-Identifier: GPL-2.0
.. include:: <isonum.txt>
.. _switchdev:

===========================================
Ethernet switch device driver model (switchdev)
===========================================

Copyright |copy| 2014 Jiri Pirko <jiri@resnulli.us>

Copyright |copy| 2014-2015 Scott Feldman <sfeldma@gmail.com>
```

Оригинал

Модель драйвера устройства для коммутатора Ethernet (switchdev)

Оглавление

Модель switchdev является встроенным в ядро драйвером для коммутационных устройств, которые выгружают (offload) плоскость пересылки (данных) из ядра.

На рисунке 1 показана блок-схема компонентов модели switchdev на примере коммутирующей микросхемы (ASIC) коммутатора класса ЦОД. Возможны и другие варианты с SR-IOV или программными коммутаторами, такими как OVS.

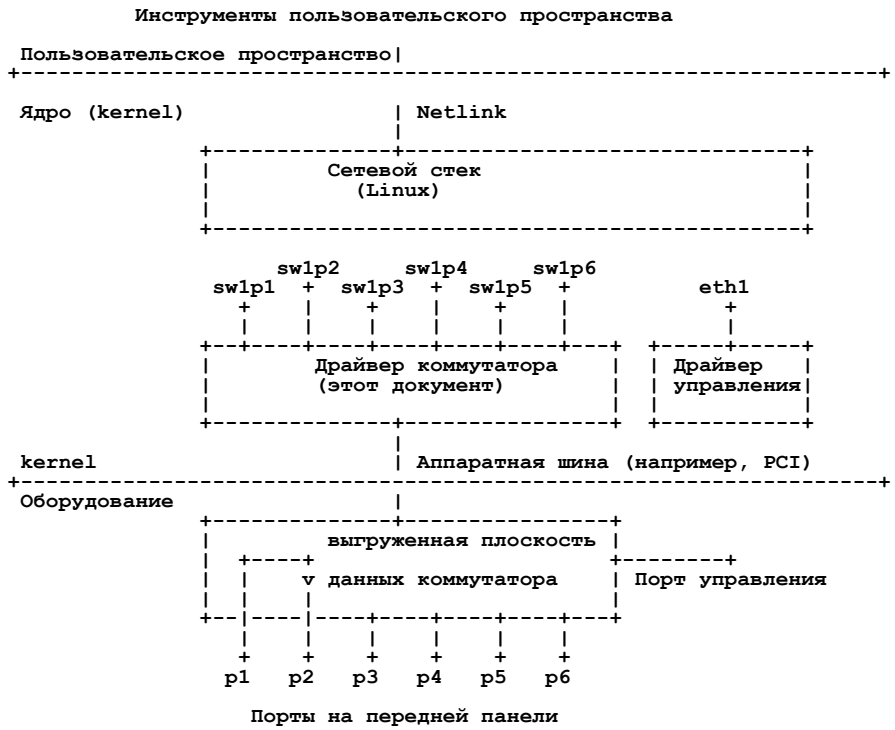

Рисунок 1.

Включаемые файлы

```
#include <linux/netdevice.h>
#include <net/switchdev.h>
```

Настройка конфигурации ядра

Для включения поддержки модели switchdev служит опция depends NET_SWITCHDEV в файле Kconfig драйвера.

Порты коммутатора

При инициализации драйвера switchdev выделяется и регистрируется с помощью register_netdev() структура net_device для перечисляемого (enumerated) физического порта коммутатора, называемого портом netdev. Это программное представление физического порта, обеспечивающее канал для трафика управления между контроллером (ядро) и сетью, а также точку привязки для конструкций вышележащего уровня, таких как мосты, связки каналов (bond), VLAN, туннели, маршрутизаторы L3. С помощью стандартных инструментов netdev (iproute2, ethtool и т. п.) порт netdev может обеспечивать доступ пользователя к физическим свойствам порта коммутатора, таким как состояние канала PHY и статистика ввода-вывода.

В настоящее время в ядре нет объектов верхнего уровня, кроме порта netdev. Все операции драйвера switchdev являются операциями netdev или switchdev.

Порт управления в коммутаторе не включается в модель драйвера switchdev. Обычно этот порт не участвует в выгруженной плоскости данных и загружает свой драйвер, например, драйвер сетевого адаптера NIC на устройстве порта управления.

Идентификатор коммутатора

Драйвер switchdev должен реализовать операцию net_device ndo_get_port_parent_id для каждого порта netdev, возвращающую один и тот же физический идентификатор для каждого порта коммутатора. У разных коммутаторов в одной системе идентификаторы должны быть уникальными, а идентификаторы коммутаторов в разных системах могут совпадать. Идентификатор коммутатора служит для указания портов коммутатора и определения принадлежности агрегированных портов одному коммутатору.

Именование портов netdev

Для именования портов netdev следует использовать правила udev с тем или иным уникальным атрибутом порта в качестве ключа (например, MAC-адрес или физическое имя порта). Жёсткое включение имён kernel netdev в драйвер не рекомендуется, ядро само выберет принятое по умолчанию имя, а udev установит окончательное имя на основе атрибута порта.

Использование физического имени порта (ndo_get_phys_port_name) в качестве ключа особенно полезно при динамическом именовании портов, когда устройство задаёт порта имена на основе внешней конфигурации. Например, если физический порт 40G логически разделен на 4 порта 10G, что даёт 4 порта netdev, устройство может дать каждому порту уникальное имя, используя физическое имя порта. Правил udev будет иметь вид

```
SUBSYSTEM=="net", ACTION=="add", ATTR{phys_switch_id}=="<phys_switch_id>", \
        ATTR{phys_port_name}!="", NAME="swX$attr{phys_port_name}"
```

Предлагается именовать порты в виде swXpYsZ, зде X - имя или идентификатор коммутатора, Y - имя или идентификатор порта, Z - имя или идентификатор субпорта. Например, sw1p1s0 будет указывать субпорт 0 на порту 1 коммутатора 1.

Свойства порта

NETIF_F_NETNS_LOCAL

Если драйвер switchdev (и устройство) поддерживает выгрузку лишь принятого по умолчанию сетевого пространства имён (netns), драйвер следует установить флаг NETIF_F_NETNS_LOCAL, чтобы предотвратить перенос порта netdev в другое пространство имён. Знающее о netns устройство (драйвер) не будет устанавливать этот флаг и отвечать за разделение оборудования для сохранения netns. Это означает, что оборудование не может пересылать трафик из порта в одном пространстве имён в порт из другого пространства имён.

Топология порта

Можно организовать порты netdev, представляющие физические порты коммутатора, в высокоуровневые коммутирующие конструкции. Принятой по умолчанию конструкцией служит автономный порт коммутатора, применяемый для выгрузки пересылки L3. Два и более порта можно связать (bond) в один логический порт для создания LAG. Два или более порта (или LAG) можно связать с мостом для организации сети L2, которую можно сегментировать с помощью VLAN. На портах можно создавать туннели L2-over-L3. Указанные конструкции можно создавать с помощью стандартных инструментов Linux, таких как драйверы моста и агрегирования, инструменты на основе netlink, такие как iproute2.

Драйвер switchdev может узнать расположение конкретного порта в топологии, отслеживая уведомления NETDEV_CHANGEUPPER. Например, порт, перемещённый в связку (bond) будет видеть смену своего ведущего (master). Если связка будет перенесена в мост, её ведущий порт изменится. Драйвер будет отслеживать такие перемещения, чтобы знать позицию порта в общей топологии, регистрируясь на события netdevice и воздействия на NETDEV_CHANGEUPPER.

Выгрузка пересылки L2

Идея заключается в выгрузке данных пути пересылки (коммутации) L2 из ядра в устройство switchdev путём зеркального отображения записей FDB в устройство. Записи FDB имеют форму {port, MAC, VLAN} для адресатов. Для выгрузки функций моста L2 драйверу и устройству switchdev следует поддерживать:

- статические записи FDB, установленные на порту моста;
- уведомления об узнанных и забытых src mac/vlan от устройства;
- смену состояния STP на порту;
- лавинную рассылку в VLAN пакетов для неизвестных адресатов, групповых и широковещательных пакетов.

Статические записи FDB

Драйвер, реализующий операции ndo_fdb_add, ndo_fdb_del, ndo_fdb_dump, способен поддерживать команду добавления статической записи в FDB

```
bridge fdb add dev DEV ADDRESS [vlan VID] [self] static
```

(ключевое слово static не является опцией и при его отсутствии запись по умолчанию будет local, что означает её исключение из процесса пересылки). Необязательный параметр self указывает ядру выполнение операции через реализацию ndo_fdb_add в самого устройства DEV. Если DEV указывает порт моста, команда будет обходить мост и оставлять программную базу данных не синхронизированной с аппаратной. Для предотвращения этого можно использовать ключевое слово master

```
bridge fdb add dev DEV ADDRESS [vlan VID] master static
```

Приведённая выше команда предписывает ядру найти первичный интерфейс DEV и выполнить операцию с помощью метода ndo_fdb_add в нём. В этом случае мост генерирует уведомление SWITCHDEV_FDB_ADD_TO_DEVICE, которое драйвер порта может обработать м использовать для программирования своей аппаратной таблицы. В результате статическая запись будет помещена в программную и аппаратную базу FDB.

Для новых драйверов switchdev, выгружающих мост Linux, настоятельно не рекомендуется реализация методов обхода моста ndo_fdb_add и ndo_fdb_del, все статические записи FDB следует добавлять для порта моста с использованием флага master. Метод ndo_fdb_dump является исключением и может быть реализован для визуализации аппаратных таблиц, если у устройства нет прерывания для уведомления операционной системы о недавно узнанных или забытых адресах динамической таблицы FDB. В этом случае в аппаратной FDB могут оказаться записи, отсутствующие в программной FDB, и реализация ndo_fdb_dump является единственным способом увидеть их.

По умолчанию мост не фильтрует VLAN и пересылает только трафик без меток. Для поддержки VLAN нужно включить фильтрацию

```
echo 1 >/sys/class/net/<bridge>/bridge/vlan_filtering
```

Уведомление об узнанных и забытых Source MAC/VLAN

Коммутатор (device) будет узнавать адреса MAC и VLAN отправителей из входящих пакетов и уведомлять драйвер коммутатора, передавая триплеты mac/vlan/port. Драйвер коммутатора, в свою очередь, будет уведомлять драйвер моста используя вызов switchdev notifier

```
err = call_switchdev_notifiers(val, dev, info, extack);
```

Параметр val имеет значение SWITCHDEV_FDB_ADD при узнавании и SWITCHDEV_FDB_DEL при забывании, а info указывает структуру switchdev_notifier_fdb_info. По значению SWITCHDEV_FDB_ADD драйвер моста будет включать запись в FDB моста и помечать её как NTF_EXT_LEARNED. Команда bridge из пакета iproute2 будет выводить такие записи с меткой offload.

```
$ bridge fdb
52:54:00:12:35:01 dev sw1p1 master br0 permanent
00:02:00:00:02:00 dev sw1p1 master br0 offload
00:02:00:00:02:00 dev sw1p1 self
52:54:00:12:35:02 dev sw1p2 master br0 permanent
00:02:00:00:03:00 dev sw1p2 master br0 offload
00:02:00:00:03:00 dev sw1p2 self
33:33:00:00:00:01 dev eth0 self permanent
01:00:5e:00:00:01 dev eth0 self permanent
33:33:ff:00:00:00 dev eth0 self permanent
01:80:c2:00:00:0e dev eth0 self permanent
33:33:00:00:00:01 dev br0 self permanent
01:00:5e:00:00:01 dev br0 self permanent
33:33:ff:12:35:01 dev br0 self permanent
```

Изучение на порту можно отключить для моста командой

```
bridge link set dev DEV learning off
```

Следует включать обучение на порту устройства, а также синхронизацию обучения learning_sync

```
bridge link set dev DEV learning on self
bridge link set dev DEV learning_sync on self
```

Атрибут learning_sync включает синхронизацию для узнанных и забытых записей FDB с FDB моста. Можно (но не оптимально) включить изучение на порту устройства и моста, отключив learning_sync.

Для поддержки обучения драйвер реализует операцию switchdev switchdev_port_attr_set для SWITCHDEV_ATTR_PORT_ID_{PRE}_BRIDGE_FLAGS.

Старение FDB

Мост будет пропускать устаревшие записи FDB с меткой NTF_EXT_LEARNED, а ответственность за устаревание записей несёт драйвер порта (устройство). Если порт устройства поддерживает устаревание, по завершении срока

действия записи FDB он будет уведомлять драйвер, который, в свою очередь, уведомит мост с помощью SWITCHDEV_FDB_DEL. Если устройство не поддерживает устаревание, драйвер может имитировать старения, используя таймер сбора мусора для отслеживания записей FDB. Старение записей будет указываться мосту с помощью SWITCHDEV_FDB_DEL. Пример использования таймера сбора мусора имеется в драйвере rocker.

Для сохранения записи NTF_EXT_LEARNED живой (alive) драйверу следует обновить запись FDB вызовом call_switchdev_notifiers(SWITCHDEV_FDB_ADD, ...). Уведомление будет устанавливать для момента последнего использования записи FDB (last-used) текущее время. Драйверу следует ограничивать частоту уведомлений об обновлениях, например, не чаще 1 раза в секунду. Посмотреть время последнего использования записи можно с помощью команды bridge -s fdb.

Смена состояния STP на порту

Используя внутреннюю или стороннюю реализацию протокола STP (например, mstpd), драйвер моста поддерживает статус STP для портов и будет уведомлять драйвер коммутатора о смене состояния STP на порту с помощью операции switchdev switchdev_attr_port_set для SWITCHDEV_ATTR_PORT_ID_STP_UPDATE.

Состояния указываются значениями BR_STATE_*. Драйвер коммутатора может использовать обновления статуса STP для изменения списка фильтров входящих пакетов для порта. Например, если порт имел состояние DISABLED, пропускать пакеты не следует, но портам в состоянии BLOCKED можно пропускать STP BPDU и другие групповые пакеты IEEE 01:80:c2:xx:xx:xx link-local. Отметим, что STP BDPU не имеют тегов и состояние STP применяется ко всем VLAN на порту, поэтому фильтры пакетов следует применять согласованно для пакетов с тегами и без тегов VLAN.

Лавинная рассылка в домене L2

Для данного домена L2 VLAN коммутатору следует применять лавинную рассылку групповых и широковещательных пакетов, а также индивидуальных пакетов с неизвестным адресатом во все порты домена, если это позволяет текущий статус STP на порту. Драйвер коммутатора, знающий какие порты относятся к какому домену L2 VLAN, может программировать лавинную рассылку устройством коммутатора. Пакет может передаваться в порт netdev для обработки драйвером моста. Мосту не следует пересылать лавинный пакет в порт, откуда он был получен, поскольку это ведёт к дублированию пакетов в линии. Для предотвращения дублирования пакетов драйверу коммутатора следует помечать уже пересланные пакеты установкой бита skb->offload_fwd_mark bit. Драйвер моста помечает skb с использованием метки входного порта моста и предотвращает пересылку через порт моста с тем же маркером.

Коммутатор может не обслуживать лавинную рассылку и выталкивать пакеты для лавинной рассылки в драйвер моста. Этот подход не идеален, поскольку число портов достаточно велико в домене L2, а устройство выполняет лавинную рассылку пакетов гораздо эффективней, чем программа. Если устройство позволяет, управление лавинной рассылкой следует выгружать в него для предотвращения некоторых netdev от лавинной рассылки индивидуального трафика, для которого нет записи в FDB.

Отслеживание IGMP

Для поддержки отслеживания IGMP (snooping) порту netdev следует перехватывать для драйвера моста все сообщения IGMP о присоединении (join) и выходе (leave). Модуль групповой передачи моста будет уведомлять порты netdev о каждом изменении в multicast-группах, независимо от характера изменения (настроенное или динамическое, присоединение или выход). Аппаратной реализации следует пересылать трафик всех зарегистрированных групп только в настроенные порты.

Выгрузка маршрутизации L3

Для выгрузки маршрутизации L3 устройство должно быть запрограммировано с записями FIB из ядра, по которым оно будет выполнять поиск в FIB и пересылку. Устройство ищет максимальное совпадение префикса (longest prefix match или LPM) с префиксом маршрута в записях FIB и пересылает пакет в выходной порт (порты) для nexthop из записи FIB. Для программирования устройства драйвер регистрирует обработчик уведомлений FIB с помощью register_fib_notifier. Поддерживаемые события указаны в таблице.

FIB_EVENT_ENTRY_ADD	Добавление новой записи FIB в устройство или изменение имеющейся.
FIB_EVENT_ENTRY_DEL	Удаление записи FIB.
FIB_EVENT_RULE_ADD, FIB_EVENT_RULE_DEL	Распространение изменений правил FIB.

События FIB_EVENT_ENTRY_ADD и FIB_EVENT_ENTRY_DEL походят

```
struct fib_entry_notifier_info {
        struct fib_notifier_info info; /* должна быть первой */
        u32 dst;
        int dst_len;
        struct fib_info *fi;
        u8 tos;
        u8 type;
        u32 tb_id;
        u32 nlflags;
};
```

для добавления изменение или удаления префикса IPv4 dst/dest_len в таблице tb_id. Структура *fi`содержит детали маршрута и nexthop, *dev - один из портов netdev, указанных в списке nexthop для маршрута.

Выгруженные в устройство маршруты помечаются как offload в таблице маршрутизации ip.

```
$ ip route show
default via 192.168.0.2 dev eth0
11.0.0.0/30 dev sw1p1  proto kernel  scope link  src 11.0.0.2 offload
11.0.0.4/30 via 11.0.0.1 dev sw1p1  proto zebra  metric 20 offload
11.0.0.8/30 dev sw1p2  proto kernel  scope link  src 11.0.0.10 offload
11.0.0.12/30 via 11.0.0.9 dev sw1p2  proto zebra  metric 20 offload
12.0.0.2  proto zebra  metric 30 offload
```

```
        nexthop via 11.0.0.1  dev sw1p1 weight 1
        nexthop via 11.0.0.9  dev sw1p2 weight 1
12.0.0.3 via 11.0.0.1 dev sw1p1  proto zebra  metric 20 offload
12.0.0.4 via 11.0.0.9 dev sw1p2  proto zebra  metric 20 offload
192.168.0.0/24 dev eth0  proto kernel  scope link  src 192.168.0.15
```

Флаг offload устанавливается в случаях, когда запись FIB выгружена хотя бы в одно устройство.

Распознавание следующего интервала пересылки (nexthop)

Список nexthop в записи FIB содержит пары (gateway, dev), но для пересылки пакета устройством коммутатора по корректному MAC-адресу получателя требуется преобразовать адрес шлюза gateway в его MAC-адрес (resolv). MAC-адреса соседей определяет процесс ARP (или ND) доступный через таблицу соседей arp_tbl. Для распознавания шлюзов nexthop на маршруте драйверу следует инициировать процесс распознавания соседей в ядре. Пример этого можно видеть в процедуре rocker_port_ipv4_resolve() драйвера rocker.

Драйвер может отслеживать обновления arp_tbl с помощью уведомлений netevent NETEVENT_NEIGH_UPDATE. Устройство можно запрограммировать с распознанными адресами соседей для маршрутов при обновлении arp_tbl. Драйвер реализует ndo_neigh_destroy, чтобы знать когда записи arp_tbl для соседей удаляются на порту.

Ожидаемое поведение драйвера устройства

Ниже представлен набор заданных правил поведения, которым должны следовать сетевые устройства switchdev.

Состояние без настройки

После запуска драйвера сетевые устройства должны быть полностью готовы к работе, а резервный драйвер должен настроить сетевое устройство так, чтобы можно было передавать и принимать трафик для него и должным образом отличать устройство от других сетевых устройств и портов (как это часто бывает в коммутаторах на ASIC). Способ достижения этого сильно зависит от оборудования, но простым решением может быть использование идентификаторов VLAN по портам, если нет лучшего механизма (например, фирменных метаданных для порта).

Сетевое устройство должно быть способно поддерживать работу полного стека протоколов IP, включая групповую передачу, DHCP, IPv4/6 и т. п. При необходимости следует запрограммировать подходящие фильтры для VLAN, группового и индивидуального трафика и т. п. базовый драйвер устройства должен быть эффективно настроен как для случая, когда включено отслеживание IGMP для групп IP (multicast) через эти сетевые устройства switchdev, а незапрошенный multicast-трафик должен фильтроваться оборудованием как можно раньше.

При настройке VLAN в сетевом устройстве все VLAN должны работать независимо от состояния других сетевых устройств (например, других портов поддерживающего VLAN моста при проверке VID). Детали этого приведены ниже.

Если устройство реализует, например, фильтрацию VLAN, при переводе интерфейса в неразборчивый (promiscuous) режим следует разрешать восприятие всех тегов VLAN (включая отсутствующие в фильтрах).

Порты моста в коммутаторе

Когда сетевое устройство с поддержкой switchdev добавляется в мост, не должна нарушаться функциональность сетевых устройств, не являющихся мостами и им следует вести себя как обычным устройствам сети. Их поведение зависит от описанных ниже элементов настройки конфигурации моста.

Фильтрация VLAN мостом

Мост Linux позволяет настроить режим фильтрации VLAN (статически при создании устройства или динамически в процессе работы), который должен быть виден базовому устройству (оборудованию) switchdev.

- При выключенной фильтрации VLAN мост ничего не знает о VLAN и его путь данных будет обрабатывать все кадры Ethernet, как будто в них нет тегов VLAN. База VLAN в мосту может обновляться, но эти изменения не будут оказывать какого-либо влияния при выключенной фильтрации VLAN. Входящие кадры с VID, отсутствующим в таблице VLAN моста/коммутатора, должны пересылаться и обрабатываться с использованием устройства VLAN (см. ниже).
- При включённой фильтрации VLAN мост знает о VLAN и входящие кадры с VID, отсутствующим в таблице VLAN моста/коммутатора, должны отбрасываться (строгая проверка VID).

Когда настроено устройство VLAN (например, sw0p1.100) на основе сетевого устройства switchdev, которое является портом моста, поведение программ сетевого стека должно сохраняться или конфигурация должна отклоняться, если она невозможна.

- При выключенной фильтрации VLAN мост будет обрабатывать весь входящий трафик для порта, за исключением трафика с тегами VLAN ID, адресованные в VLAN вышележащего уровня. Интерфейс VLAN вышележащего уровня (который потребляет тег VLAN) может быть даже добавлен ко второму мосту, который включает другие порты коммутатора или программные интерфейсы. Ниже указаны некоторые подходы для обеспечения корректного обслуживания трафика, относящегося к VLAN вышележащих интерфейсов.
    - Если возможно управление адресатами по VLAN, оборудование можно настроить на отображение всего трафика, за исключением VID, относящихся к VLAN вышележащего интерфейса, на внутренний идентификатор VID, соответствующий пакетам без тегов. Этот внутренний идентификатор VID охватывает все порты моста, не знающего о VLAN. Идентификатор VID, соответствующий VLAN вышележащего интерфейса, охватывает физический порт данного интерфейса VLAN и другие порты, которые могут быть соединены с ним мостом.
    - Порты моста с VLAN вышележащих интерфейсов рассматриваются как автономные и разрешается обработка пересылки на программном пути данных.

- При включённой фильтрации VLAN эти устройства VLAN могут создаваться, пока на мосту нет записи VLAN с тем же VID для любого из портов моста. Эти устройства VLAN не могут быть вовлечены (enslaved) в мост, поскольку они дублируют функциональность (вариант использования) обработки данных а пути передачи VLAN в мосту.

Немостовые сетевые порты одного модуля коммутации (switch fabric) недопустимо нарушать каким-либо способом путём включения фильтрации VLAN на мосту. Если установка фильтрации VLAN является глобальной для микросхемы, автономным портам следует указывать сетевому стеку необходимость фильтрации VLAN установкой rx-vlan-filter: on [fixed] с помощью ethtool.

Поскольку фильтрацию VLAN можно включать и отключать в процессе работы, драйвер switchdev должен быть способен перенастраивать базовое оборудование на лету в соответствии с этим переключением и должным образом вести себя. Если это невозможно, драйвер может отклонить динамическое переключение фильтрации VLAN во время работы и требовать уничтожения устройств мостов и создания новых с другим состоянием фильтрации VLAN, чтобы обеспечить передачу оборудованию сведений о VLAN.

Даже при отключённой фильтрации VLAN в мосту оборудование и драйвер базового коммутатора могут настраиваться на поддержку VLAN при условии соблюдения описанного выше поведения.

Протокол VLAN в мосту участвует в принятии решения об учёте тега в пакете - мост, использующий протокол 802.1ad, должен обрабатывать пакеты без тегов VLAN и пакеты с заголовками 802.1Q, как пакеты без тегов.

Пакеты с тегами 802.1p (VID 0) должны обрабатываться как пакеты без тегов, поскольку мост не разрешает манипуляции с VID 0 в своей базе данных.

Когда на мосту включена фильтрация VLAN и на входном порту не задан PVID, пакеты без тега и с тегом 802.1p должны отбрасываться. Если фильтрация VLAN включена и на входном порту есть PVID, пакеты без тегов и с тегами приоритета пересылаются в соответствии с принадлежностью порта в мосту к PVID VLAN. При отключённой на мосту фильтрации VLAN решение о пересылке следует принимать независимо от наличия или отсутствия PVID.

Отслеживание IGMP мостом

Мост Linux позволяет настроить отслеживание IGMP (статически при создании интерфейса или динамически в процессе работы), которое должно быть видимо базовому устройству (оборудованию) switchdev, как описано ниже.

- При выключенном отслеживании IGMP групповой трафик должен рассылаться во все порты моста, для которых установлено mcast_flood=true. В порт управления/CPU в идеальном случае пакеты не передаются (если вхходной интерфейс не имеет флага IFF_ALLMULTI или IFF_PROMISC) и он продолжает изучать групповой трафик через уведомления сетевого стека. Если оборудование не позволяет это, лавинная рассылка в порт управления/CPU должна выполняться с программной фильтрацией группового трафика.
- При включённом отслеживании IGMP групповой трафик должен выборочно направляться в соответствующие сетевые порты (включая CPU/управление). Лавинную рассылку в неизвестные группы следует выполнять лишь в порты, соединённые с групповым маршрутизатором (локальное устройство тоже может играть такую роль).

Коммутатор должен соответствовать требованиям RFC 4541 и соответственно выполнять лавинную рассылку группового трафика, поскольку это делает реализация моста Linux.

Поскольку отслеживание IGMP можно включать и отключать в процессе работы, драйвер switchdev должен быть способен перенастраивать базовое оборудование на лету в соответствии с установленным режимом и обеспечивать должное поведение. Драйвер switchdev может отклонять динамическое переключение отслеживания группового трафика в процессе работы и требовать уничтожения устройства моста и создания нового устройства с другим режимом отслеживания.

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru