Консольный интерфейс simple_switch_CLI

Версия 1.0

24.09.2020

Оглавление

Программы runtime_CLI и simple_switch_CLI служат для управления программными коммутаторами и маршрутизаторами из пакета BMv2, распространяемого в исходных кодах. Эти программы, по сути, представляют собой прототип плоскости управления (control-plane), полнофункциональной реализацией которой в SDN служит контроллер (например, P4Runtime).

Документ подготовлен на основе представленного разработчиками описания.

Большая часть описанных здесь команд работает в runtime_CLI и simple_switch_CLI, несколько команд применимы лишь в simple_switch_CLI и помечены как [simple_switch_CLI].

Управление программными реализациями прототипов устройств BMv2 выполняется через встроенный в эти реализации сервер Thrift. Интерфейс управления в настоящее время имеет достаточно ограниченную функциональность и позволяет выполнять лишь небольшой набор операций управления коммутатором:

- просмотр состояния коммутатора и его портов;
- добавление и удаление портов, без возможности управления их состоянием;
- просмотр, добавление и удаление записей синтаксического анализатора (parser);
- просмотр и изменение имеющихся таблиц сопоставления, без возможности добавления и удаления таблиц;
- просмотр и ограниченное управление действиями в таблицах сопоставления;
- просмотр и ограниченное управление клонированием пакетов для multicast-групп;
- чтение, запись и сброс значений счетчиков и регистров, определенных в программе P4;
- сохранение конфигурации в файл, загрузка новой конфигурации из файла, смена конфигурации.

По умолчанию сервер управления коммутатором доступен через порт 9090. При необходимости номер порта можно изменить с помощью опции --thrift-port в командной строке simple_switch (плоскость данных) и simple_switch_CLI (плоскость управления).

Команды управления портами

Набор команд управления портами в simple_switch достаточно скромен и позволяет лишь добавлять или удалять порты, связанные с физическими или логическими сетевыми интерфейсами аппаратной платформы. Включение (up) или отключение (down) порта из командного интерфейса не поддерживается.

show_ports

Команда без параметров, выводящая номера портов, имена связанных с ними интерфейсов и их состояние (up или down). Для портов коммутатора, связанных с логическими (виртуальными) интерфейсами аппаратной платформы команда всегда выводит состояние DOWN, хотя эти порты реально работают и способны принимать и передавать пакеты.

port_add

Добавляет в коммутатор порт, связанный с физическим или логическим интерфейсом платформы. Результат добавления зависит от используемого менеджера устройств. Команда принимает в качестве параметров имя интерфейса и номер порта. Дополнительно может указываться также путь к программе pcap, служащей для записи или «воспроизведение» сохраненных ранее пакетов.

```
port_add <iface_name> <port_num> [pcap_path]
```

Отметим, что для портов коммутатора simple_switch, связанных с логическими (виртуальными) интерфейсами платформы команда show_ports всегда показывает состояние DOWN, хотя эти порты реально работают и способны принимать и передавать пакеты.

port_remove

Удаляет порт из коммутатора. Результат удаления зависит от используемого менеджера устройств. Команда принимает в качестве параметра номер удаляемого порта.

```
port_remove <port_num>
```

Команды управления профилями действий

Концепция профилей действий (операций) в таблицах сопоставления использовалась в спецификации $P4_{14}$, но была исключена из $P4_{16}$. Тем не менее она сохранена в архитектуре V1Model, используемой simple_switch с программами $P4_{16}$. Ниже приведен фрагмент спецификации $P4_{14}$, связанный с профилями деййствий в таблицах сопоставления.

> В некоторых экземплярах значения параметров действий (action) не специфичны для совпадающей записи и могут совместно использоваться несколькими записями. Это можно выразить в P4 с помощью профилей действий,

которые представляют собой объявляемую структуру, задающую список возможных действий, а также могущую включать иные атрибуты.

Записи размещаются в процессе работы для указания одного действия (из числа указанных в профиле действий), которое будет выполняться при выборе данной записи, а также используемые значения параметров действия.

Вместо статической привязки одной записи из профиля действий к каждой записи таблицы сопоставления, можно связать несколько записей из профиля действий с одной записью таблицы сопоставления и позволить системе (т. е. логике плоскости данных) динамически связывать одну из записей профиля действий с каждым классом пакетов. Такое поведение включается атрибутом dynamic_action_selection. При задании этого атрибута записи профиля действий могут собираться в группы в процессе работы, а запись таблицы сопоставления можно связать с группой записей профиля действий. Для задания конкретного механизма плоскости данных, выбирающего определенную запись профиля действий в группе, нужно предоставить селектор действий. Этот селектор выбирает конкретную запись профиля действий для каждого пакета (псевдо)случайно или предсказуемо на основе полей заголовка и/или метаданных.

Ниже представлено определение профиля действий в формате BNF.

```
action_profile_declaration ::=
       action_profile action_profile_name {
              action_specification
              [ size : const_expr ; ]
              [ dynamic_action_selection : selector_name ; ]
       }
action_specification ::=
       actions { [ action_name ; ] + }
action_selector_declaration ::=
       action_selector selector_name {
              selection_key : field_list_calculation_name ;
}
```

Профили действий задаются и применяются в соответствии с приведенным ниже соглашением.

- Атрибут size указывает число записей, требуемых для профиля действий. Если заданный размер не поддерживается, при обработке объявления будет выдан сигнал об ошибке. При опущенном атрибуте размера нет гарантии создания нужного числа записей в профиле действий в процессе работы.

Далее описаны поддерживаемые в simple_switch_CLI команды для работы с профилями действий в таблицах сопоставления программ P4.

act_prof_add_member_to_group

Добавляет элемент в группу в профиле действия (операции). Параметрами команды служат имя профиля действия, идентификатор элемента и идентификатор группы.

```
act_prof_add_member_to_group <action profile name> <member handle> <group handle>
```

act_prof_create_group

Добавляет группу в профиль действия (операции). Параметром команды служит имя профиля действия.

```
act_prof_create_group <action profile name>
```

act_prof_create_member

Добавляет элемент в профиль действия (операции). Параметрами команды служат имя профиля действия и имя действия, за которым могут следовать параметры действия.

```
act_prof_create_member <action profile name> <action_name> [action parameters]
```

act_prof_delete_group

Удаляет группу из профиля действия (операции). Параметрами команды служат имя профиля действия и идентификатор удаляемой группы.

```
act_prof_delete_group <action profile name> <group handle>
```

act_prof_delete_member

Удаляет элемент из профиля действия (операции). Параметрами команды служат имя профиля действия и идентификатор удаляемого элемента.

```
act_prof_delete_member <action profile name> <member handle>
```

act_prof_dump

Выводит список записей в профиле действия (операции). Параметром команды служит имя профиля действия.

```
act_prof_dump <action profile name>
```

act_prof_dump_group

Выводит информацию о группе из профиля действия (операции). Параметрами команды служат имя профиля действия и идентификатор группы.

```
act_prof_dump_group <action profile name> <group handle>
```

act_prof_dump_member

Выводит информацию об элементе из профиля действия (операции). Параметрами команды служат имя профиля действия и идентификатор элемента.

```
act_prof_dump_member <action profile name> <member handle>
```

act_prof_modify_member

Изменяет элемент в профиле действия (операции). Параметрами команды служат имя профиля действия, имя действия и идентификатор изменяемого элемента, за которым могут следовать параметры действия.

```
act_prof_modify_member <action profile name> <action_name> <member_handle> [action parameters]
```

act_prof_remove_member_from_group

Удаляет элемент из группы в профиле действия (операции). Параметрами команды служат имя профиля действия, идентификатор удаляемого элемента и идентификатор группы.

```
act_prof_remove_member_from_group <action profile name> <member handle> <group handle>
```

show_actions

Команда без параметров. Для каждого действия в загруженной программе P4 выводит имя и список параметров с указанием для каждого параметра имени и размера в битах.

Команды для работы с таблицами

table_add

Добавляет действие в указанную именем таблицу.

```
RuntimeCmd: help table_add
Add entry to a match table: table_add <table name> <action name> <match fields> => <action parameters> [priority]
```

Можно увидеть список имен всех таблиц и действий с помощью команд show_tables и show_actions. Если не используется аннотация @name в программе P4, принятое по умолчанию имя зачастую является иерархическим и начинается с имени элемента управления, в котором определена таблица или действие.

Поля сопоставления должны указываться в том же порядке, в котором заданы поля ключей поиска в программе P4, но без имен. Вывод команды show_tables показывает имена, типы сопоставления (ternary, range, lpm, exact) и размер в битах для каждой таблицы после строки mk=. Поля с типом сопоставления optional представляются как ternary в файле BMv2 JSON, а в simple_switch_CLI всегда идентичны ternary.

Числовые значения можно задавать в десятичном или шестнадцатеричном формате (префикс 0x).

Для удобства представления некоторых числовых значений (адресов) применяются специальные форматы:

- 32-битовые значения можно указывать в формате адресов IPv4 (например, 10.1.2.3);
- 128-битовые значения можно задавать подобно адресам IPv6 в нотации IETF (например, FEDC:BA98:7654:3210:FEDC:BA98:7654:3210 или 1080::8:800:200C:417A) как описано в RFC 2732;
- 48-битовые значения можно задавать подобно адресам Ethernet MAC с парами шестнадцатеричных цифр, разделенными двоеточиями (например, 00:12:34:56:78:9a).

Поля ключей поиска в таблице с типом сопоставления lpm должны иметь значение, за которым указан символ / и размер префикса (например, 0x0a010203/24). Размер префикса должен указываться десятичным числом. Может указываться префикс с размером 0 и такие записи будут соответствовать любому значению искомого поля.

Поля ключей поиска в таблице с типом сопоставления ternary задаются численным значением, за которым следует &&& и численная маска без пробелов между числами и &&&. Биты маски со значением 1 задают совпадение битов, а биты со значением маски 0 не рассматриваются при сопоставлении. Т. е. запись value&&&mask будет соответствовать ключу поиска k, если (k & mask) == (value & mask), где операция & выполняется для каждого бита, как в P4. Для соответствия любому значению при троичном поиске можно задать 0&&&0.

Поля ключей поиска в таблице с типом сопоставления optional в исходном коде P4 задаются так же, как поля с типом сопоставления ternary. Отметим, что CLI при работе не проверяет наличие в маске только нулей или только 1, что позволяет задавать произвольные ternary-маски, даже если это не следует разрешать. Было бы хорошо ограничить задание сопоставления типа optional, но это потребует внесений существенных изменений в код behavioral-model.

Поля ключей поиска в таблице с типом сопоставления range задаются минимальным значением, за которым без пробела следует -> и максимальное значение (без пробела). Для соответствия любому значению из диапазона можно указать 0->255 для 8-битового поля, 0->0xffffffff для 32-битового и т. п.

Если любое поле ключа поиска в таблице имеет тип сопоставления ternary, optional или range, в записи должно быть указано числовое значение приоритета. Для полей с другими типами сопоставления задание приоритета является ошибкой. Если ключу поиска соответствует несколько записей таблицы, из них выбирается запись с меньшим числовым значением приоритета и выполняется действие из этой записи.

Параметры действий должны указываться в том же порядке, как они заданы в программе P4, но без имен. Вывод команды show_actions показывает имена и размеры в битах для всех параметров каждого действия.

table_clear

Удаляет все записи в таблице сопоставления (прямого или опосредованного), указанной именем, оставляя лишь запись default.

```
table_clear <table name>
```

table_delete

Удаляет из указанной именем таблицы заданную идентификатором запись.

```
table_delete <table name> <entry handle>
```

table_dump

Выводит дамп указанной именем таблицы.

```
table_dump MyIngress.ipv4_lpm
==========
TABLE ENTRIES
==========
Dumping default entry
Action entry: MyIngress.drop -
==========
```

table_dump_entry, table_dump_entry_from_key

Выводит дамп записи из указанной по имени таблицы. Запись задается идентификатором или ключом поиска.

```
table_dump_entry <table name> <entry handle>
table_dump_entry_from_key <table name> <match fields> [priority]
```

table_dump_group, table_dump_member

Выводит информацию (дамп) о группе или элементе из указанной именем таблицы. Группа или элемент задаются идентификатором.

```
table_dump_group <table name> <group handle>
table_dump_member <table name> <member handle>
```

table_indirect_add

Добавляет запись в таблицу непрямого сопоставления, указанную именем. Параметрами команды являются поля сопоставления и идентификатор элемента таблицы, может также указываться значение приоритета.

```
table_indirect_add <table name> <match fields> => <member handle> [priority]
```

table_indirect_add_member_to_group

Заменена командой act_prof_add_member_to_group.

table_indirect_add_with_group

Добавляет запись в таблицу непрямого сопоставления, указанную именем. Параметрами команды являются поля сопоставления и идентификатор группы, может также указываться значение приоритета.

```
table_indirect_add <table name> <match fields> => <group handle> [priority]
```

table_indirect_create_group

Заменена командой act_prof_create_group.

table_indirect_create_member

Заменена командой act_prof_create_member.

table_indirect_delete

Удаляет запись из заданной именем таблицы непрямого сопоставления. Запись указывается идентификатором.

```
table_indirect_delete <table name> <entry handle>
```

table_indirect_delete_group

Заменена командой act_prof_delete_group.

table_indirect_delete_member

Заменена командой act_prof_delete_member.

table_indirect_modify_member

Заменена командой act_prof_modify_member.

table_indirect_remove_member_from_group

Заменена командой act_prof_remove_member_from_group.

table_indirect_reset_default, table_indirect_set_default

Сбрасывает или устанавливает принятое по умолчанию действие в таблице непрямого сопоставления указанной именем. Устанавливаемое действие задается идентификатором элемента.

```
table_indirect_reset_default <table name>
table_indirect_set_default <table name> <member handle>
```

table_indirect_set_default_with_group

Устанавливает используемую по умолчанию группу в заданной именем таблице непрямого сопоставления. Группа указывается идентификатором.

```
table_indirect_set_default <table name> <group handle>
```

table_info

Выводит информацию об указанной именем таблице сопоставления.

```
table_info <table name>
```

table_modify

Добавляет запись в указанную именем таблицу сопоставления. Запись задается именем добавляемого действия и идентификатором, а также может включать параметры действия.

```
table_modify <table name> <action name> <entry handle> [action parameters]
```

table_num_entries

Возвращает число записей в таблице прямого или опосредованного сопоставления, указанной именем.

```
table_num_entries <table name>
```

table_reset_default

```
RuntimeCmd: help table_reset_default
Reset default entry for a match table: table_reset_default <table name>
```

Меняет действие, выполняемое в таблице при отсутствии соответствующей ключу поиска записи на исходное значение, заданное в программе P4. Если в программе такое действие не задано, используется «пустая операция» - no-op (иногда ее называют NoAction).

table_set_default

```
RuntimeCmd: help table_set_default
Set default action for a match table: table_set_default <table name> <action name> <action parameters>
```

Назначает действие, выполняемое таблицей при отсутствии записи, соответствующей ключу поиска.

Задание параметров действия рассмотрено в описании команды table_add.

table_set_timeout

Устанавливает время ожидания (тайм-аут) для указанной идентификатором записи в заданной именем таблице. Время ожидания указывается в миллисекундах.

```
table_set_timeout <table_name> <entry handle> <timeout>
```

show_tables

Команда без параметров. Для каждой таблицы в загруженной программе P4 выводит имя, реализацию (часто None, но может выводиться профиль или селектор действия для таблиц, созданных с этими опциями) и список полей поиска в таблице (ключей) с указанием для каждого поля имени, типа соответствия (например, exact, lpm, ternary, range) и размера в битах. Поля с типом соответствия optional в коде P4 представляются в файле BMv2 JSON как ternary.

table_show_actions

Выводит список действий заданной именем таблицы в порядке их представления в программе P4. Например,

```
RuntimeCmd: table_show_actions MyIngress.ipv4_lpm
MyIngress.drop                 []
MyIngress.ipv4_forward         [dstAddr(48),    port(9)]
NoAction
```

Команды управления синтаксическим анализатором

pvs_add, pvs_clear, pvs_get, pvs_remove

Эти команды служат для управления экземпляром набора значений синтаксического анализатора P4. Команда pvs_add добавляет запись в набор значений, pvs_remove удаляет запись, pvs_clear удаляет все записи, а pvs_get служит для просмотра имеющихся записей.

Язык P4 позволяет передавать value_set в качестве параметра структурные типы, как показано ниже.

```
struct vsk_t {
    bit<16> f1;
    bit<7> f2;
}
value_set<vsk_t>(4) pvs;
select (<X>) {
    pvs: accept;
    _: reject;
}
```

При добавлении или удалении записи требуется указать целое число, соответствующее сжатой разрядности набора значений, иначе CLI будет сообщать об ошибке. Когда записи набора значений состоят из множества отдельных полей, как в приведенном выше примере, сжатая разрядность определяется как сумма разрядностей этих полей без учета заполнения. Когда параметр типа для набора значений является целым числом (со знаком или без него) или структурой с одним полем, сжатая разрядность просто совпадает с размером этого поля.

В будущем CLI сможет поддерживать более понятный интуитивно интерфейс и позволит предоставлять целочисленные значения для разных полей, составляющих запись набора значений. Однако в настоящее время размер приходится вычислять самостоятельно.

Отметим, что pvs_add не выдает предупреждений при попытке добавить уже имеющуюся запись, а pvs_remove не предупреждает о попытке удалить отсутствующее значение.

Реализация наборов значений в bmv2 при сопоставлении поддерживает лишь точное совпадение (exact).

show_pvs

Команда без параметров. Для каждого набора значений анализатора в загруженной программе P4 выводит имя и ожидаемый размер записей. При добавлении записей командой pvs_add требуется указывать целочисленное значение, не превышающее этот размер.

Команды для работы с групповыми пакетами

mc_dump, mc_mgrp_create, mc_mgrp_destroy, mc_node_associate, mc_node_create, mc_node_destroy, mc_node_dissociate

При завершении входной обработки в архитектуре v1model входной код P4 может устанавливать значения нескольких полей в standard_metadata, контролирующих отбрасывание пакетов, передаваемых индивидуально (unicast) в один выходной порт, или групповых пакетов, реплицированных в список (возможно пустой) выходных портов. Более подробное описание приведено в разделе «Псевдокод для завершения входной и выходной обработки» описания bmv2.

Буфер пакетов содержит машину репликации PRE[1].

В v1model.p4 standard_metadata поле mcast_grp задает один из 65535 идентификаторов multicast-групп из диапазона [1, 65535]. Значение mcast_grp = 0 задает отсутствие групповой репликации пакетов. Коммутатор simple_switch может использовать большее число идентификаторов групп и указанный диапазон задан определением поля mcast_grp в standard_metadata (файл v1model.p4) с типом bit<16>.

Для реплицируемых пакетов PRE находит в таблице значение mcast_grp и по нему получает (возможно пустой) список пар (egress_port, egress_rid), для которых создаются копии пакета. Для каждой копии позднее выполняется выходная обработка с двумя полями standard_metadata, инициализированными значениями найденной в таблице пары.

Отметим, что таблица групповой репликации не является таблицей, определяемой в программе P4. Здесь термин «таблица» имеет более общий смысл. Таблица репликации похожа на обычную таблицу P4 в том смысле, что поле mcast_grp является единственным ключом поиска и для него всегда применяется точное совпадение. Однако имеется ряд перечисленных ниже различий.

- Таблица не объявляется в программе и присутствует в simple_switch независимо от загруженной программы P4 и применения в программе этой таблицы.
- Программы плоскости управления используют для чтения и записи в таблицу групп иные вызовы API, нежели для таблиц P4.
- Результатом поиска в таблице является список с переменным числом пар, тогда как язык P4 не имеет простого способа представления списков переменного размера в процессе работы.

В управление списками групповой рассылки на деле включены два «уровня» объектов:

- каждая multicast-группа является множеством узлов;
- каждый создаваемый узел групповой рассылки имеет одно значение egress_rid и набор egress_port, в котором не может быть дубликатов.

Предположим, что нужно настроить simple_switch так, чтобы пакеты для группы с mcast_grp = 1113 рассылались в приведенный ниже список пар (egress_port, egress_rid):

- (egress_port=0, egress_rid=5);
- (egress_port=7, egress_rid=10);
- (egress_port=3, egress_rid=5).

Первая и последняя запись имеют общее значение egress_rid = 5, поэтому они могут находится в одном multicast-узле, который мы хотим создать в этом примере. Делается это командой mc_node_create, как показано ниже.

```
RuntimeCmd: help mc_node_create
Create multicast node: mc_node_create <rid> <space-separated port list> [ | <space-separated lag list> ]
```

Команда имеет два обязательных параметра, задающих идентификатор и список выходных портов, а также позволяет указать список агрегированных портов (LAG) для групповой рассылки. Создадим узел для рассылки в порты 0 и 3.

```
RuntimeCmd: mc_node_create 5 0 3
Creating node with rid 5 , port map 1001 and lag map
node was created with handle 0
```

Отметим дескриптор 0 в выводе команды. При каждом создании multicast-узла программа simple_switch выбирает целочисленный идентификатор для его именования, который должен использоваться при указании данного узла в последующих командах. Значения обычно начинаются с 0 и увеличиваются по мере создания новых узлов, но лучше не полагаться на это и смотреть вывод команды.

Создадим другой узел для копирования пакетов с egress_rid = 10, в нашем примере включающий лишь порт 7

```
RuntimeCmd: mc_node_create 10 7
Creating node with rid 10 , port map 10000000 and lag map
node was created with handle 1
```

Вывод показывает, что этому узлу был назначен дескриптор 1.

Далее создадим группу с идентификатором 1113 командой mc_mgrp_create.

```
RuntimeCmd: help mc_mgrp_create
Create multicast group: mc_mgrp_create <group id>
```

[1]Packet Replication "Engine" - «машина» репликации пакетов.

Справка говорит, что команда имеет единственный параметр - идентификатор группы.

```
RuntimeCmd: mc_mgrp_create 1113
Creating multicast group 1113
```

Здесь не используются специальные дескрипторы. Группы всегда именуются по заданным для них идентификаторам (1113 в этом примере).

После этого можно посмотреть созданные группы с помощью команды mc_dump.

```
RuntimeCmd: mc_dump
==========
MC ENTRIES
**********
mgrp(1113)
==========
LAGS
==========
```

Вывод показывает наличие группы 1113, с которой не связано никаких узлов. В последующих примерах будут показаны группы с multicast-узлами. В текущей конфигурации simple_switch, передача пакета для репликации с mcast_grp = 1113 будет создавать 0 копий пакета, что равносильно его отбрасыванию (drop).

Для добавления multicast-узла в группу служит команда mc_node_associate.

```
RuntimeCmd: help mc_node_associate
Associate node to multicast group: mc_node_associate <group handle> <node handle>
```

Команда принимает два параметра, один из которых задает идентификатор группы, другой - узла.

```
RuntimeCmd: mc_node_associate 1113 0
Associating node 0 to multicast group 1113
```

После добавления узла группа выглядит уже иначе.

```
RuntimeCmd: mc_dump
==========
MC ENTRIES
**********
mgrp(1113)
  -> (L1h=0, rid=5) -> (ports=[0, 3], lags=[])
==========
LAGS
==========
```

Видно, что группа 1113 имеет один узел. L1h является сокращением для записей L1 handle, которые можно увидеть в системном журнале simple_switch (это просто идентификатор узла). Показано значение rid = 5 (rid - сокращение для egress_rid), и список портов 0 и 3.

В текущей конфигурации simple_switch передача пакета для репликации с mcast_grp = 1113 будет создавать 2 копии пакета с парами:

- (egress_port=0, egress_rid=5);
- (egress_port=3, egress_rid=5).

Добавим в группу 1113 второй узел, упомянутый выше.

```
RuntimeCmd: mc_node_associate 1113 1
Associating node 1 to multicast group 1113
```

Группа после этого будет иметь вид

```
RuntimeCmd: mc_dump
==========
MC ENTRIES
**********
mgrp(1113)
  -> (L1h=0, rid=5) -> (ports=[0, 3], lags=[])
  -> (L1h=1, rid=10) -> (ports=[7], lags=[])
==========
LAGS
==========
```

В текущей конфигурации simple_switch передача пакета для репликации с mcast_grp = 1113 приведет к созданию трех копий с парами:

- (egress_port=0, egress_rid=5);
- (egress_port=3, egress_rid=5);
- (egress_port=7, egress_rid=10).

Можно исключить узел из группы с помощью команды mc_node_dissociate.

```
RuntimeCmd: help mc_node_dissociate
Dissociate node from multicast group: mc_node_associate <group handle> <node handle>
```

В качестве параметров команды нужно указать группу и дескриптор исключаемого узла.

```
RuntimeCmd: mc_node_dissociate 1113 0
Dissociating node 0 from multicast group 1113
```

Группа после исключения узла будет иметь вид

```
RuntimeCmd: mc_dump
==========
MC ENTRIES
**********
```

```
mgrp(1113)
  -> (L1h=1, rid=10) -> (ports=[7], lags=[])
==========
LAGS
==========
```

В этой конфигурации simple_switch передача пакета для репликации с mcast_grp = 1113 будет создавать 1 копию с (egress_port=7, egress_rid=10).

При исключении узла 1 из группы 1113, она вернется в исходное состояние и групповые пакеты будут отбрасываться.

```
RuntimeCmd: mc_node_dissociate 1113 1
Dissociating node 1 from multicast group 1113

RuntimeCmd: mc_dump
==========
MC ENTRIES
**********
mgrp(1113)
==========
LAGS
==========
```

Можно удалить группу с помощью команды mc_mgrp_destroy.

```
RuntimeCmd: help mc_mgrp_destroy
Destroy multicast group: mc_mgrp_destroy <group id>
```

Единственным параметром команды является идентификатор группы.

```
RuntimeCmd: mc_mgrp_destroy 1113
Destroying multicast group 1113
```

После этого команда просмотра групп будет давать пустой вывод.

```
RuntimeCmd: mc_dump
==========
MC ENTRIES
==========
LAGS
==========
```

Для описанных операций есть ряд ограничений.

- Для привязки узла к группе нужно сначала создать то и другое. Порядок создания не имеет значения.
- Коммутатор simple_switch позволяет связать узел лишь с одной группой. Для привязки узла к другой группе его нужно сначала отсоединить от текущей группы.
- При удалении группы рекомендуется сначала удалить привязки узлов к ней.
- В одном узле значение egress_port не может повторяться. Если нужно отправить в порт несколько копий, следует создать дополнительные узлы. Использование разных egress_rid для этих узлов позволяет коду P4 различать копии и по разному обрабатывать их.

Команда mc_dump выводит информацию об узлах, привязанных к группам. Для просмотра узлов, не связанных с группой, в simple_switch_CLI нет команды.

Отметим, что в справке simple_switch_CLI и выводе команды mc_dump указываются группы lag и их списки, которые еще не документированы. Однако групповую репликацию можно использовать и без них.

mc_node_update

Обновляет групповой узел, принимая в качестве параметра идентификатор узла и список разделенных пробелами номеров портов. Дополнительно может указываться список разделенных пробелами LAG.

```
mc_node_update <node handle> <space-separated port list> [ | <space-separated lag list> ]
```

mc_set_lag_membership

Задает набор портов, включаемых в группу LAG. Параметрами служат индекс LAG и список разделенных пробелами номеров портов коммутатора.

```
mc_set_lag_membership <lag index> <space-separated port list>
```

Программы управления клонированием пакетов

mirroring_add, mirroring_add_mc, mirroring_delete, mirroring_get

[simple_switch_CLI]

При обработке пакетов могут происходить перечисленные ниже события:

- программа с архитектурой $P4_{16}$ v1model вызывает внешнюю функцию clone или clone3 при входной или выходной обработке и на этом соответствующая обработка завершается;
- программа $P4_{14}$ вызывает clone_ingress_pkt_to_egress при входной обработке, которая на этом завершается;
- программа $P4_{14}$ вызывает clone_egress_pkt_to_egress при выходной обработке, которая на этом завершается.

В этих случаях не только «исходный» пакет продолжит прохождение через конвейер, но и пакеты, которые могут быть созданы PRE в составе simple_switch (копии или клоны) из текущего пакета, будут помещены в очередь в буфере пакетов и позднее для каждого из них будет выполнена выходная обработка. Каждый из клонов обрабатывается

независимо от других и от исходного пакета. Дополнительную информацию можно найти поиском по слову «клон» в разделе «Псевдокод для завершения входной и выходной обработки» описания bmv2.

В simple_switch поддерживается до 32768 независимых сессий клонирования с номерами от 0 до 32767, которые называют также сессиями отражения (mirroring session). При клонировании пакета указывается желаемая сессия для вызова из программы P4 операции клонирования.

Целью команд отражения является указание портов, в которые передаются клонированные пакеты. Это настраивается независимо для каждой сессии клонирования по ее идентификатору.

По умолчанию при старте simple_switch сессии клонирования не настроены. Любая операция клонирования для ненастроенной сессии не будет создавать клонов, т. е поведение обработки пакетов пойдет обычным путем, как будто клонирование не запрашивалось.

```
RuntimeCmd: help mirroring_add
Add mirroring session to unicast port: mirroring_add <mirror_id> <egress_port>
```

Например, по команде mirroring_add 5 7 в конфигурации сессии с идентификатором 5 будет задано создание операцией клонирования в сессии 5 одного клона, предназначенного для выходного порта 7.

```
RuntimeCmd: mirroring_add 5 7

RuntimeCmd: mirroring_get 5
MirroringSessionConfig(mgid=None, port=7)
```

Можно задать в конфигурации сессии клонирование пакетов в multicast-группу.

```
RuntimeCmd: help mirroring_add_mc
Add mirroring session to multicast group: mirroring_add_mc <mirror_id> <mgrp>
```

Команда mirroring_add_mc 5 22 изменит конфигурацию сессии 5 так, что последующие операции клонирования для этой сессии будут выполнять множественное клонирование для multicast-группы 22. Команда mc_mgrp_create и другие команды работы с группами описаны выше.

```
RuntimeCmd: mirroring_add_mc 5 22

RuntimeCmd: mirroring_get 5
MirroringSessionConfig(mgid=22, port=None)
```

Можно сбросить состояние настроенной сессии клонирования по ее идентификатору.

```
RuntimeCmd: help mirroring_delete
Delete mirroring session: mirroring_delete <mirror_id>
```

Команда mirroring_delete 5 возвращает сессию 5 в исходное (ненастроенное) состояние.

```
RuntimeCmd: mirroring_delete 5
```

С помощью команды mirroring_get можно увидеть состояние указанной идентификатором сессии клонирования. Например, для сброшенной предыдущей командой сессии 5 вывод будет иметь вид

```
RuntimeCmd: mirroring_get 5
Invalid mirroring operation (SESSION_NOT_FOUND)
```

Команды для работы с измерителями

meter_array_set_rates

Задает скорости для всего массива измерителей, принимая в качестве параметров имя, а также набор значений скоростей и пиков.

```
meter_array_set_rates <name> <rate_1>:<burst_1> <rate_2>:<burst_2> ...
```

meter_get_rates

Возвращает параметры измерителя по имени и индексу.

```
meter_get_rates <name> <index>
```

meter_set_rates

Синтаксис команды в CLI приведен ниже.

```
RuntimeCmd: help meter_set_rates
Configure rates for a meter: meter_set_rates <name> <index> <rate_1>:<burst_1> <rate_2>:<burst_2> ...
```

Поведение измерителя соответствует RFC 2697 и RFC 2698.

Пользователь задает скорость для измерителя в байтах за микросекунду и burst_size в байтах. Для измерителей, учитывающих пакеты, скорость задается числом пакетов за микросекунду, burst_size - числом пакетов.

Команды для работы со счетчиками

counter_read, counter_reset, counter_write

Считывает, сбрасывает или задает значение указанного именем счетчика. При чтении и записи команда включает индекс, а при записи дополнительно указываются записываемые в счетчик значения числа пакетов и байтов.

Команды для работы с регистрами

register_read, register_write

Читает или записывает регистр, указанный именем и индексом. При записи также указывается вносимое в регистр значение.

register_reset
Сбрасывает в 0 все ячейки массива регистров, указанного именем.

```
register_reset <name>
```

Команды управления очередями

set_queue_depth
[simple_switch_CLI]

Задает размер всех или указанной выходной очереди, принимая в качестве параметров число пакетов в очереди и необязательный номер порта. По умолчанию устанавливается размер очередей на всех выходных портах.

```
set_queue_depth <nb_pkts> [<egress_port>]
```

set_queue_rate
[simple_switch_CLI]

Задает скорость (пакет/с) всех или указанной выходной очереди, принимая в качестве параметров скорость для очереди и необязательный номер порта. По умолчанию устанавливается скорость для очередей на всех выходных портах.

```
set_queue_rate <rate_pps> [<egress_port>]
```

Прочие команды

help
При запуске без параметра выводит список поддерживаемых интерфейсом CLI команд. При указании в качестве параметра имени той или иной команды CLI будет выведена краткая справка по работе с командой. Например,

```
RuntimeCmd: help help
List available commands with "help" or detailed help with "help cmd".

RuntimeCmd: help show_tables
List tables defined in the P4 program: show_tables
```

switch_info
Выводит базовую информацию о коммутаторе.

```
switch_info
device_id               : 0
thrift_port             : 9090
notifications_socket    : ipc:///tmp/bmv2-0-notifications.ipc
elogger_socket          : None
debugger_socket         : None
```

serialize_state
Преобразует все состояния коммутатора в последовательную форму и выводит информацию в файл, заданный параметром команды.

reset_state
Сбрасывает все состояния коммутатора (таблицы, регистры и т. п.), сохраняя конфигурацию P4. Команда не имеет параметров.

get_time_elapsed, get_time_since_epoch
[simple_switch_CLI]

Возвращает время, прошедшее с момента запуска коммутатора или с начала «эпохи» в микросекундах. Команда не имеет параметров.

set_crc16_parameters, set_crc32_parameters
Задает параметры вычисления контрольных сумм crc16 или crc32.

```
set_crc16_parameters <name> <polynomial> <initial remainder> <final xor value> <reflect data?> <reflect remainder?>
set_crc32_parameters <name> <polynomial> <initial remainder> <final xor value> <reflect data?> <reflect remainder?>
```

load_new_config_file
Загружает конфигурацию коммутатора из заданного параметром файла JSON. Для активизации этой конфигурации служит команда Ошибка: источник перекрёстной ссылки не найден.

swap_configs
Меняет конфигурацию, используя файл, загруженный ранее командой load_new_config_file.

write_config_to_file
Считывает конфигурацию из используемого в данный момент файла JSON и записывает ее в указанный пользователем файл на станции управления, где запущена программа simple_switch_CLI или runtime_CLI. Если файл задан без абсолютного пути, он будет сохранен в каталоге, из которого была запущена программа.

shell

Позволяет выполнить из консоли коммутатора команду операционной системы, в которой коммутатор работает.

```
shell <command>
```

Николай Малых

nmalykh@protocols.ru