Анонс выпуска P4Runtime v1.0

Опубликован Antonin Bas и Waqar Mohsin 11 марта 2019 г.

Выпущена спецификация P4Runtime v1.0.0, подготовленная рабочей группой P4 API. Выпуск v1.0.0 включает:

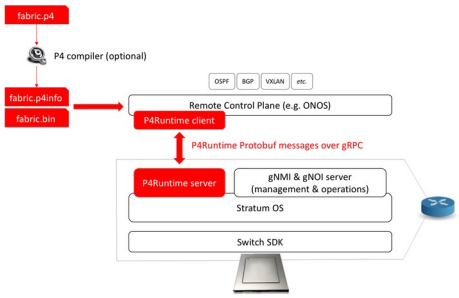
- определение сервиса RPC - файлы Protocol Buffers (Protobuf), которые определяют формат сообщений, используемых для чтения и записи элементов уровня данных, а также набор RPC, которые могут использоваться для передачи этих сообщений между разделёнными клиентом и сервером;
- полная спецификация семантики сообщений Protobuf.

Что такое P4Runtime и зачем это нужно?

Язык программирования P4 стал популярным способом задания и программирования поведения уровня пересылки в сетевых устройствах. Интерфейс P4Runtime API разработан в качестве стандартного способа управления работающими элементами пересылки P4, например для добавления и удаления записей в таблицах пересылки. До выпуска P4Runtime большинство элементов пересылки P4 управлялось с использованием пользовательских или фирменных (proprietary) API уровня управления. Спецификация языка P4 не включает API уровня управления, оставляя программистам и производителям возможность использования своих решений. Это с неизбежностью привело к появлению множества несовместимых API и приложения, использовавшие разные API не могут быть перенесены в другие системы даже если базовые элементы пересылки определены с открытом и независимом от производителя P4.

P4Runtime пытается решить эту проблему путём определения **стандартного**, **открытого** и **аппаратно-независимого** (для устройств с фиксированной и программируемой функциональностью) API, позволяющего контролировать работу произвольных уровней пересылки P4.

P4Runtime **одинаково хорошо подходит для локальных и удалённых уровней управления** (независимо от RPC). На рисунке показан типичный вариант развёртывания SDN с P4Runtime. В примере коммутатор использует операционную системы Stratum на основе Open Networking Linux (ONL).



P4Runtime **не зависит от конвейера обработки**. При добавлении новых функций на уровне пересылки P4 интерфейс P4Runtime API не меняется и не требуется заново компилировать программы уровня управления (и даже перезагружать его).

Для достижения независимости от конвейера P4Runtime отделяет формат обмена сообщениями Protobuf от формата данных, передаваемых в этих сообщениях, так же, как gNMI определяет фиксированный сервис gRPC и фиксированный формат сообщений Protobuf, которые могут применяться для передачи любых структурированных в форме дерева данных, часто моделируемых с помощью YANG. В P4Runtime сама модель данных описывается сообщением Protobuf, называемым P4Info, которое выводится из программы P4. С другой стороны интерфейс является фиксированным и не зависит от программы P4, что способствует расширяемости (т. е. добавлению протоколов и действий) и оптимизирует реализации клиентов и серверов.

P4 program	P4Info message (p4info.proto)	Example P4Runtime message WriteRequest for TableEntry (p4runtime.proto)

`action set_nhop(bit<32> nhop) {`
`  // …`
`}`
 	
`@brief("L3 LPM table")`
`table ipv4_lpm {`
`  key = {`
`  hdr.ipv4.dstAddr : lpm;`
`  hdr.meta.vrf_id : exact;`
`  }`
`  actions = { drop; set_nhop; }`
`}`
	`tables {`
	`  preamble {`
	`  id: <ipv4_lpm_id>`
	`  name: "ipv4_lpm"`
	`  doc { … }`
	`  }`
	`  match_fields {`
	`  id: 1`
	`  name: "hdr.ipv4.dstAddr"`
	`  bitwidth: 32`
	`  match_type: LPM`
	`  }`
	`  match_fields {`
	`  id: 2`
	`  name: "hdr.meta.vrf_id"`
	`  bitwidth: 32`
	`  match_type: EXACT`
	`  }`
	`  action_refs {`
	`  id: <drop_id>`
	`  }`
	`  action_refs {`
	`  id: <set_nhop_id>`
	`  }`
	`} …`
		`device_id: <device_id>`
		`election_id { … }`
		`updates {`
		`  type: INSERT`
		`  entity {`
		`  table_entry {`
		`  table_id: <ipv4_lpm_id>`
		`  match {`
		`  field_id: 1`
		`  lpm {`
		`  value: "\x0a\x00\x00\x00"`
		`  prefix_length: 8`
		`  }`
		`  }`
		`  match {`
		`  field_id: 2`
		`  exact {`
		`  value: "\x00\x00\x00\x01"`
		`  }`
		`  }`
		`  action {`
		`  action {`
		`  action_id: <drop_id>`
		`  }`
		`  }`
		`  }`
		`}`

В приведённой выше таблице сообщение P4Info (средняя колонка) является моделью данных, задающей содержимое сообщений P4Runtime (справа). Например, запись P4Info для ipv4_lpm диктует, что каждая запись сопоставления, добавляемая в таблицу, обеспечивает должным образом форматированные значения для двух полей сопоставления (hdr.ipv4.dstAddr и hdr.meta.vrf_id) и должна выбирать из 2 действий (drop или set_nhop).

P4Runtime v1.0.0 - результат взаимодействия профессионалов

Разработка P4Runtime началась в середине 2016 г. и ускорилась в 2017 после создания рабочей группы P4 API. Параллельно разрабатывалось множество реализаций и с 2017 г. результаты были показаны на разных мероприятиях.

- В ноябре 2017 г Google Cloud, Barefoot Networks и ONF продемонстрировали первую физическую сеть на базе P4Runtime во время конгресса SDN NFV World Congress (видео).
- В феврале 2018 г. на MWC фонд ONF продемонстрировал фабрику коммутации от нескольких производителей, управляемую контроллером ONOS через P4Runtime.

- В марте 2018 г. на саммите OCP компания Google показала видео о реализации P4Runtime на основе коммутатора с ONL, управляемого контроллером P4 SDN (видео).

- В ноябре 2018 г. на ONS Europe и в декабре 2018 г. на ONF Connect фонд ONF показал фабрику коммутации от нескольких производителей, управляемую контроллером ONOS с использованием P4Runtime и OF-DPA. Коммутаторы (white-box), управляемые P4Runtime, работали на основе ОС Stratum.

После двух лет разработок и прототипирования выпуск P4Runtime 1.0.0 достиг готовности. Этот выпуск обеспечивает гарантии совместимости с прежними выпусками - все новые библиотеки будут совместимы с предшественниками, если только не будет увеличен старший номер (с 1 до 2). Предполагается, что такое увеличение будет происходить редко и, следуя рекомендациям Google для версий облачных API, старший номер версии кодируется в имени пакета Protobuf для устранения возможности рассогласования версий между клиентом и сервером, а также обеспечения конечным точкам возможности одновременно поддерживать несколько версий.

Представление чёткого и однозначного интерфейса API с полной спецификацией облегчит разработчикам программ и оборудования реализацию совместимой продукции.

Что интересного в P4Runtime?

- **Арбитраж**. Интерфейс P4Runtime позволяет подключить несколько контроллеров к серверу P4Runtime на устройстве для обеспечения резервирования и отказоустойчивости. Наличие нескольких контроллеров позволяет одному или нескольким работать в режиме резервного с переключением при отказе основного.

- **Разделение ролей**. P4Runtime поддерживает совместное использование уровня данных P4 с разными ведущими (возможно и резервными) контроллерами. Это делает возможными неоднородные системы с комбинацией локального и одного или нескольких удалённых уровней управления (например, локальный для L2 и удалённый для L3).

- **Настраиваемость уровня данных**. Независимая от конвейера природа P4Runtime позволяет «вталкивать» совершенно новый уровень пересылки (с другим P4Info) через сам интерфейс. Это позволяет уровню управления использовать программируемость уровня данных, когда она поддерживается.

- **Ввод-вывод пакетов**. P4Runtime поддерживает потоки пакетов между клиентом и сервером благодаря двухстороннему потоку gRPC. К пакету можно привязать произвольные метаданные (согласно P4Info).

- **Пакетирование**. P4Runtime поддерживает пакетное чтение и запись данных на уровне протокола, что позволяет клиенту и серверу амортизировать стоимость RPC при удалённом управлении и обеспечить высокую производительность. Поддерживаются разные режимы делимости для создания пакетов, что позволяет приложениям использовать расширенные возможности сетевого устройства, такие как транзакции.

- **Отчёты об ошибках**. P4Runtime обеспечивает полнофункциональный механизм информирования клиента об ошибках. Каждое сообщение содержит канонический код ошибки и необязательный специфический для устройства код. Спецификация включает подробные рекомендации об использовании канонических кодов в разных ситуациях, что может быть использовано приложением для подобающего отклика на ошибки и расширения возможностей отладки программ уровня управления.

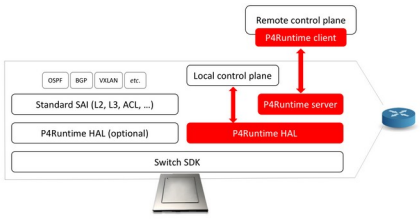
- **Расширяемость архитектуры**. P4Runtime использует любые сообщений protobuf для поддержки произвольных фирменных расширений, которые могут оставаться закрытыми. Новые сообщения Protobuf могут определяться для настройки внешних или потоковых уведомлений между сервером и клиентами.

Могут P4Runtime и SAI дополнять друг друга?

Различие между SAI и P4Runtime описано в статье. Интерфейс SAI был создан для локального управления коммутаторами и, в отличие от P4Runtime, предполагает наличие определённого набора протоколов на базовом уровне пересылки. Основная ценность SAI заключается в наличии унифицированной и независимой от производителя семантики API локальным приложениям уровня управления, что позволяет интегрировать его со стандартными стеками протоколов (например, FRRouting). SAI проще в использовании, чем P4Runtime, но менее расширяем.

Считается, что SAI и P4Runtime хорошо дополняют друг друга. В проекте flexsai уже реализовано использование описания P4 для SAI API с целью создания «соглашения» между уровнями управления и пересылки. Но flexsai базируется на генерации API вместо предоставления приложениям независимого от конвейера интерфейса и не поддерживает функциональность RPC.

Предполагается, что взаимодополняющая природа SAI и P4Runtime может использоваться для расширения P4Runtime в SAI.



Перевод на русский язык

Николай Малых

nmalykh@protokols.ru