Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича

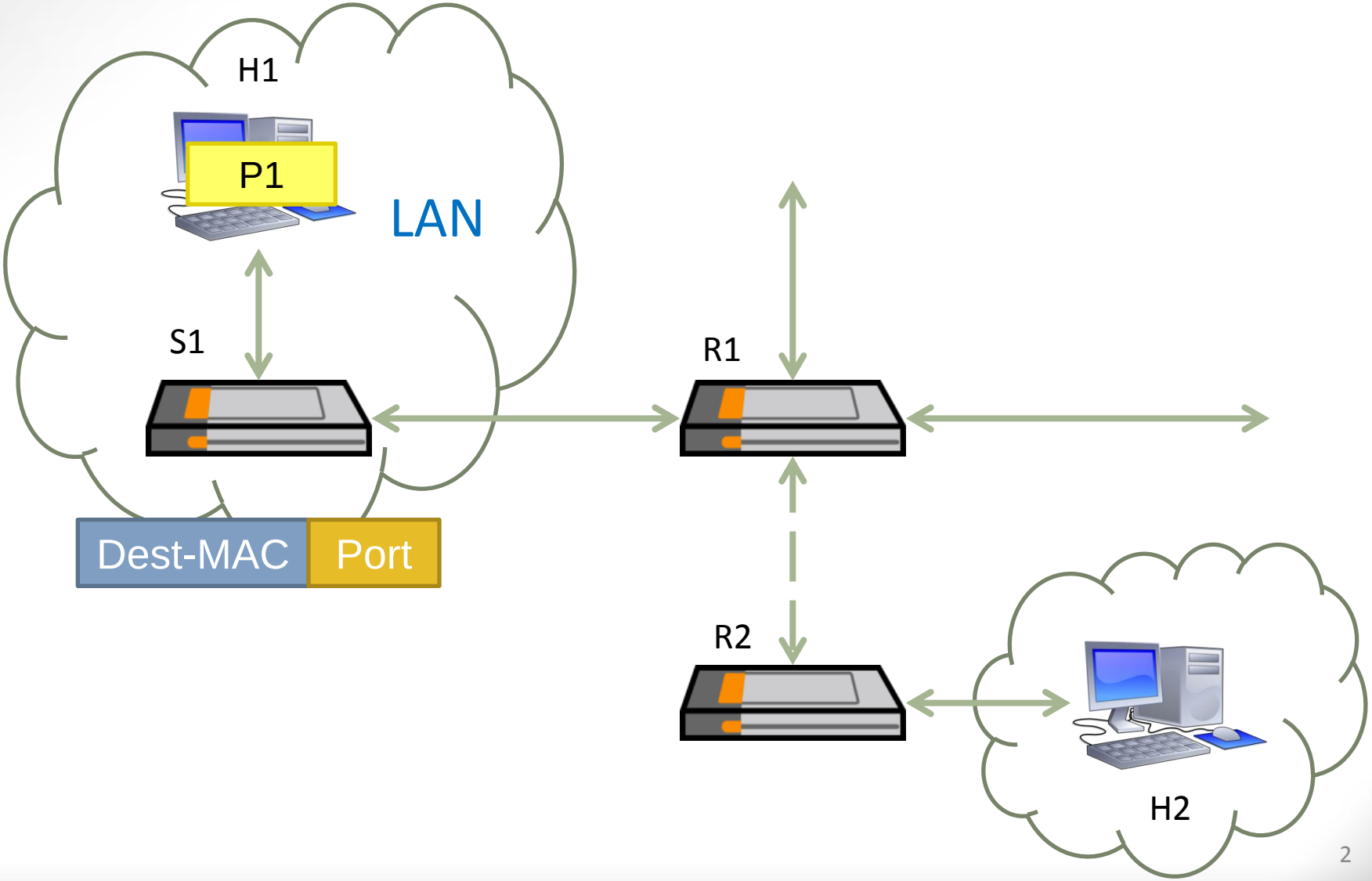
Особенности эксплуатации программно-конфигурируемых сетей

(Лекция 2)

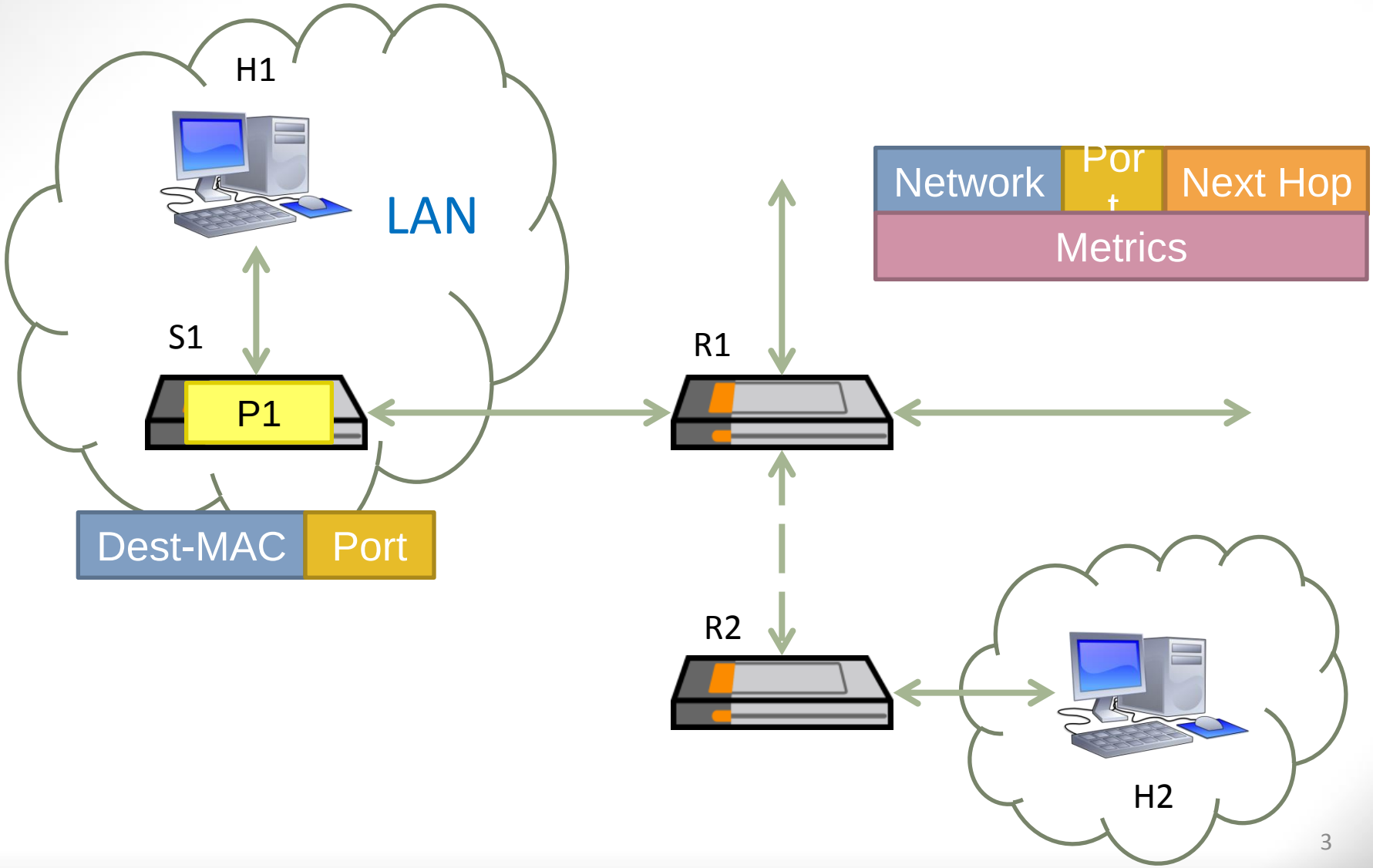
Елагин В.С.
к.т.н. доцент каф. ИКС

СПб ГУТ)))

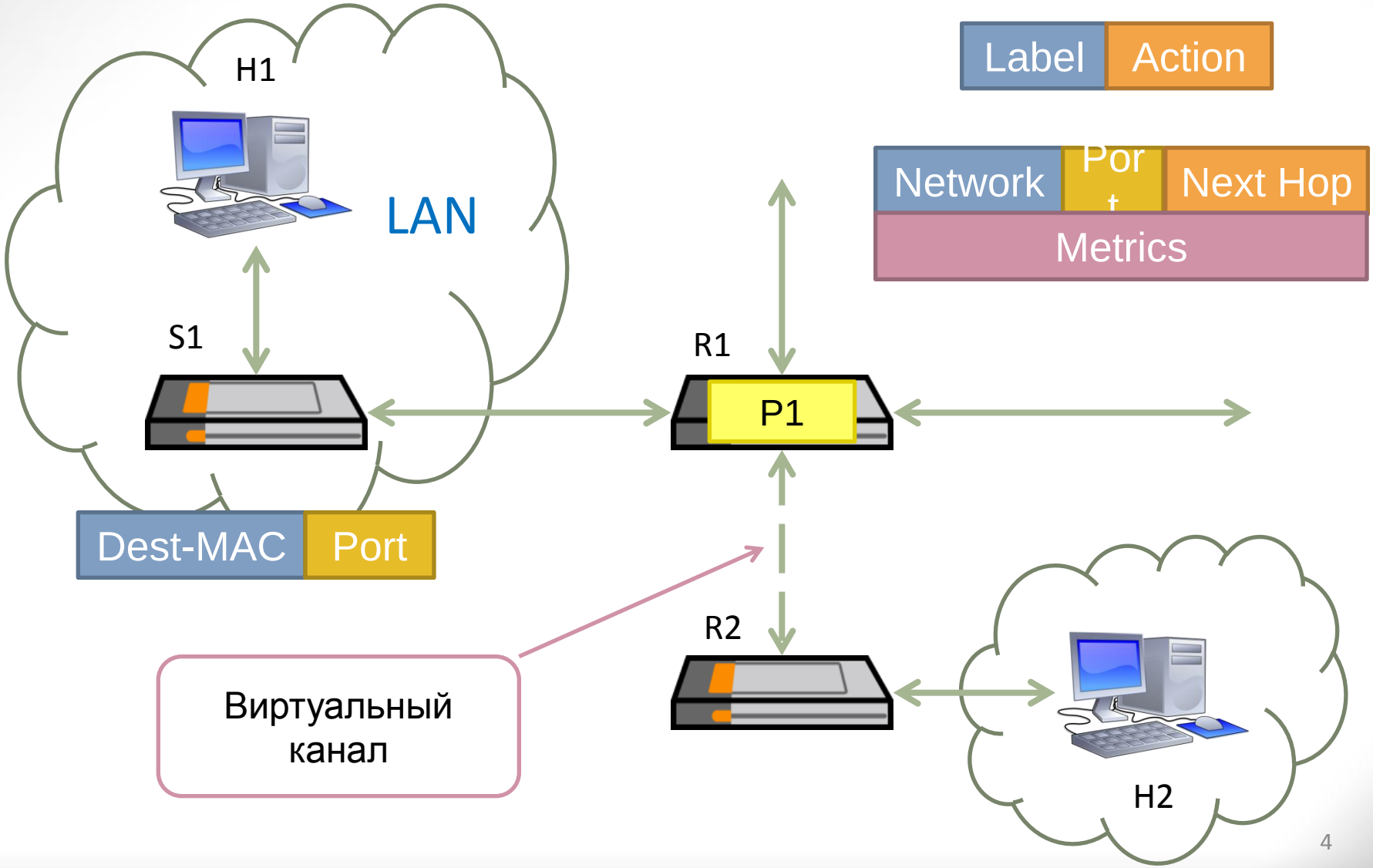
Коммутация пакетов



Коммутация пакетов



Коммутация меток



Программно-Конфигурируемые сети

[разделение передачи и управления]

- Существует фиксированный набор простых инструкций обработки пакетов
- Концепция совместима с различными протоколами
- Работает с разноуровневым оборудованием

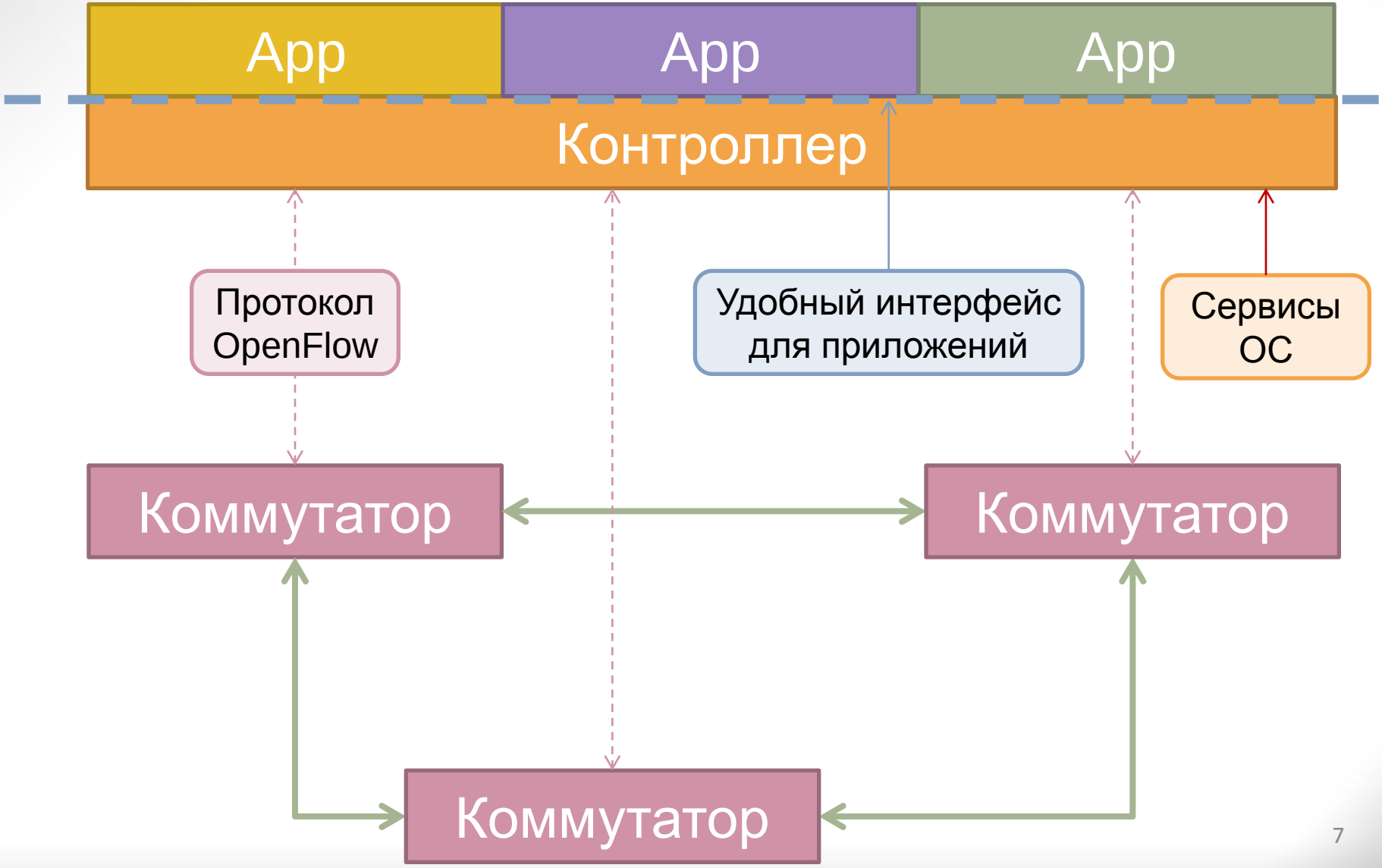
Программно-Конфигурируемые сети

[разделение передачи и управления]

- Существует фиксированный набор простых инструкций обработки пакетов
- Концепция совместима с различными протоколами
- Работает с разноуровневым оборудованием

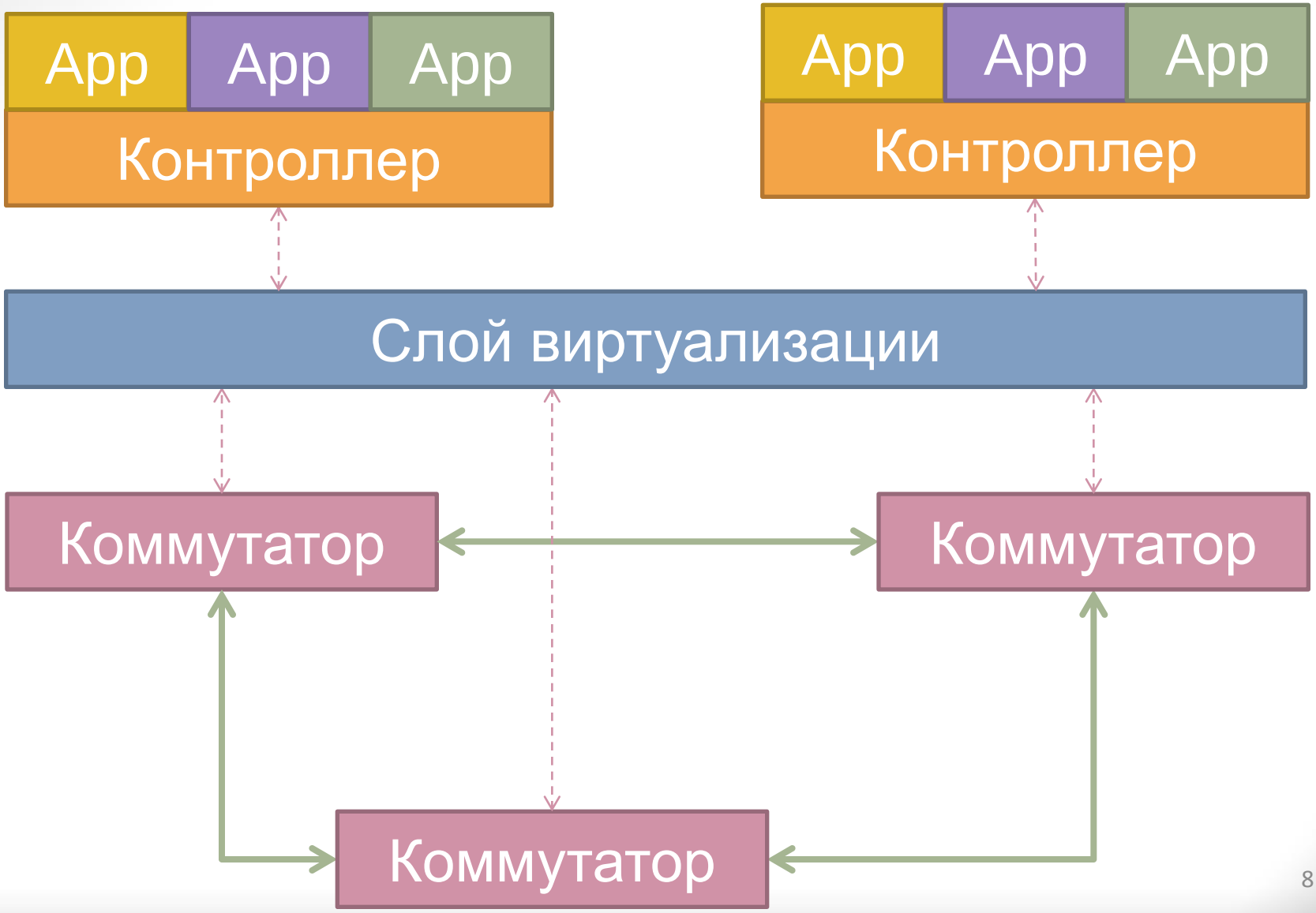
Программно-Конфигурируемые сети

[централизация управления]

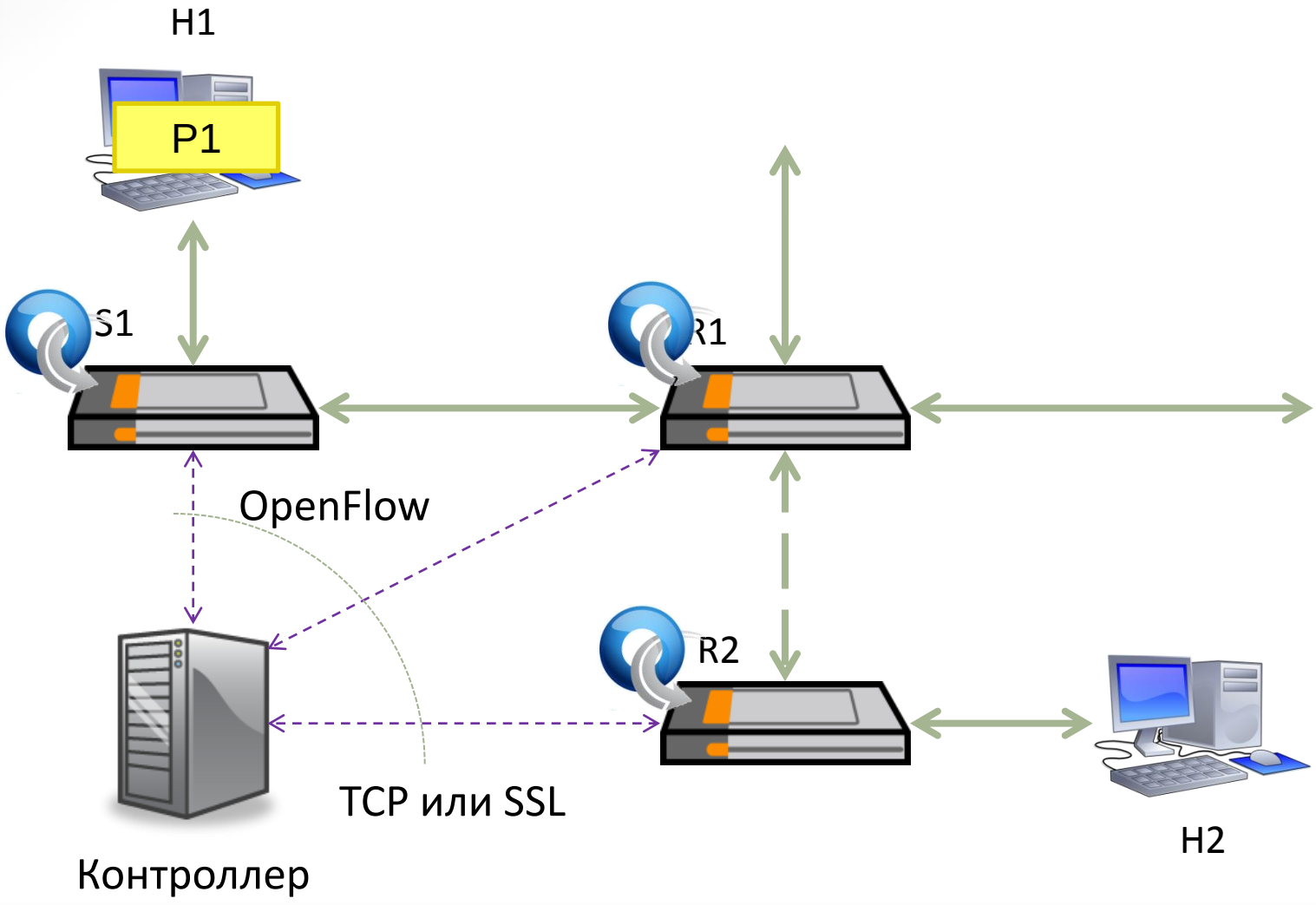


Программно-Конфигурируемые сети

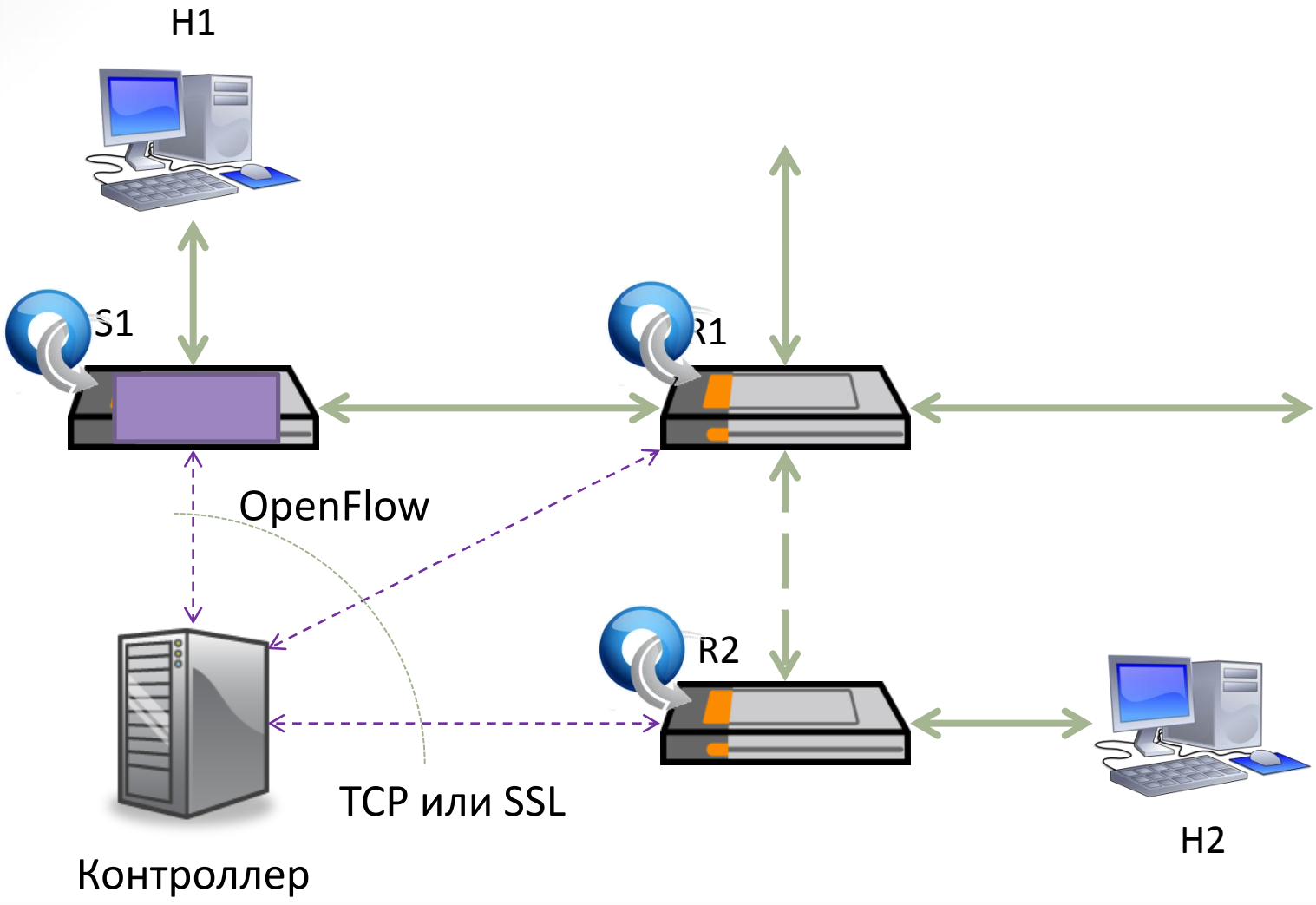
[виртуализация сети]



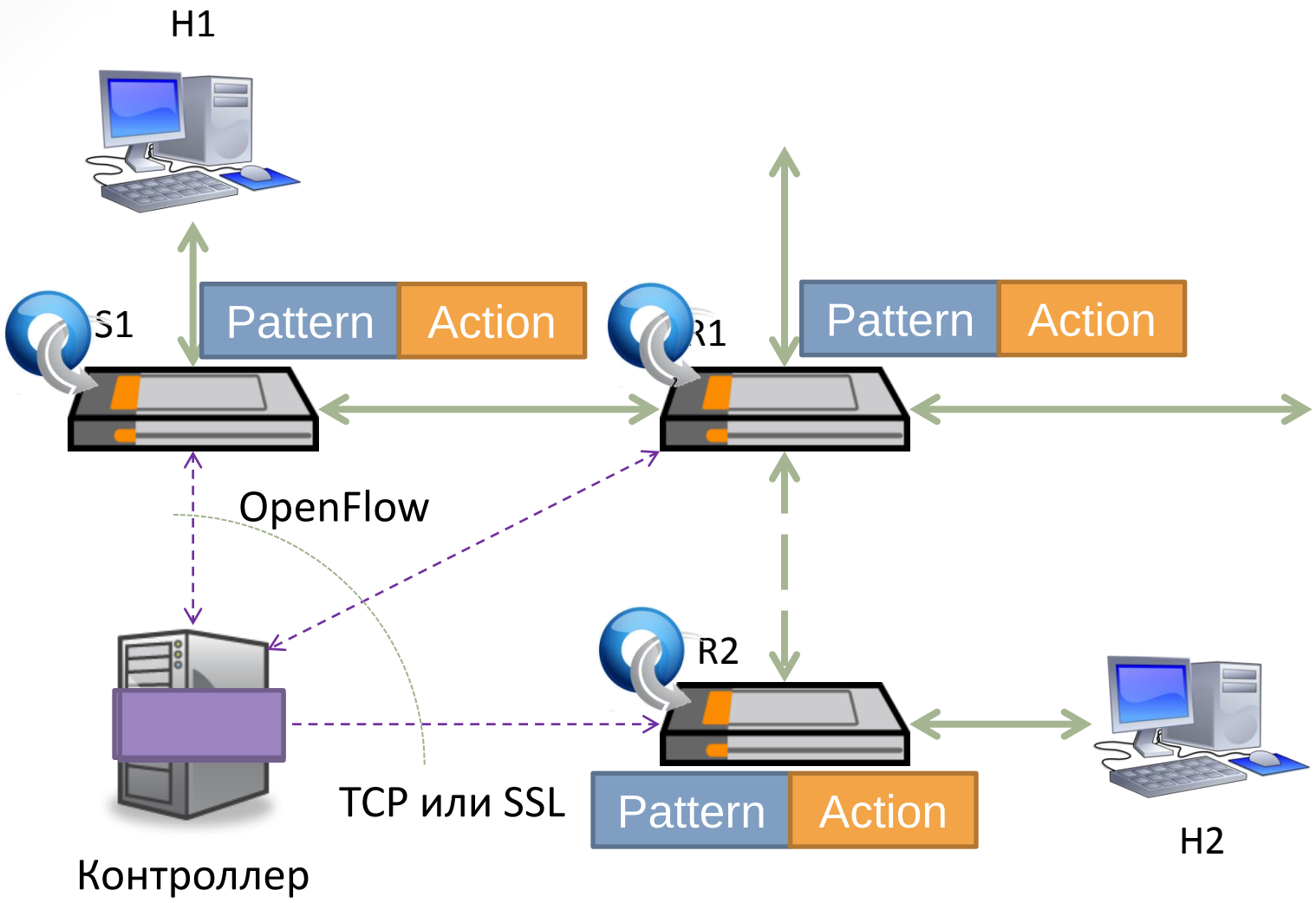
Сегмент ПКС



Сегмент ПКС



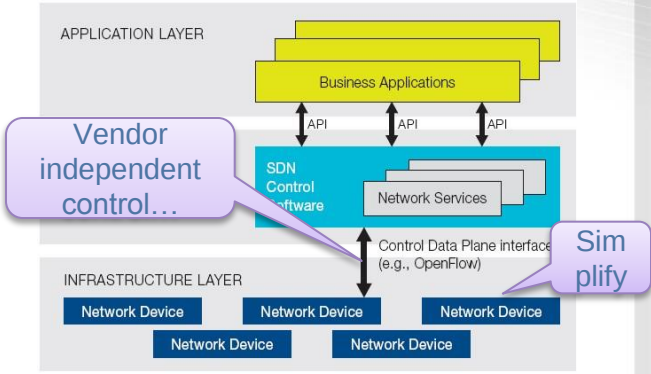
Сегмент ПКС


H1
S1
Pattern
Action
R1
Pattern
Action
OpenFlow
R2
TCP или SSL
Pattern
Action
H2
Контроллер

Протокол OpenFlow

SDN vs. OpenFlow

- ONF определяет
  - SDN выполняет Software Defined Forwarding
    - Управляет пересылкой данных через открытый API
  - SDN обеспечивает управление высокого уровня абстракции
    - Можно сделать приложения любого уровня

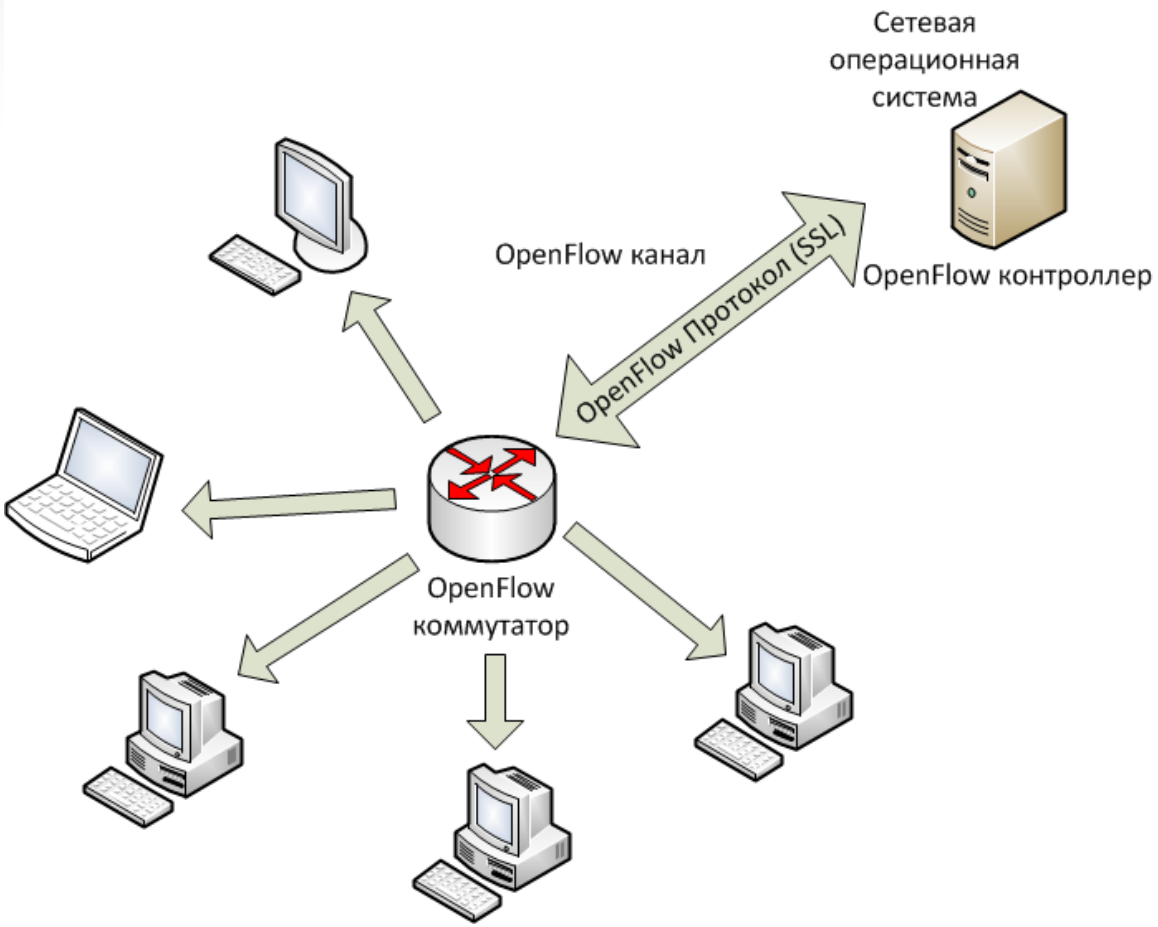


Logical View of SDN architecture

- OpenFlow неверно понимается как эквивалент SDN
  - Нет требований использования OpenFlow в пределах SDN
  - OpenFlow является одним из протоколов SDN, наиболее популярным

Version	Date	Characteristics	Organization
OpenFlow 1.0	2009.12	MAC, IPv4, single flow table	OpenFlow Consortium
OpenFlow 1.1	2011.2	MPLS/tunnel, multiple flow tables, group table	OpenFlow Consortium
OpenFlow 1.2	2011.12	IPv6, Config., поддержка расширений	ONF
OpenFlow 1.3	2012.9	QoS (meter table)…	ONF
OpenFlow 1.4	2013.10	Optical port monitoring and config (frequency, power)	ONF
OpenFlow 1.5.	2016.12	Выходные таблицы, Packet Type Aware Pipeline (поддержка PPP пакетов), flow entry stat trigger	ONF

Технология OpenFlow



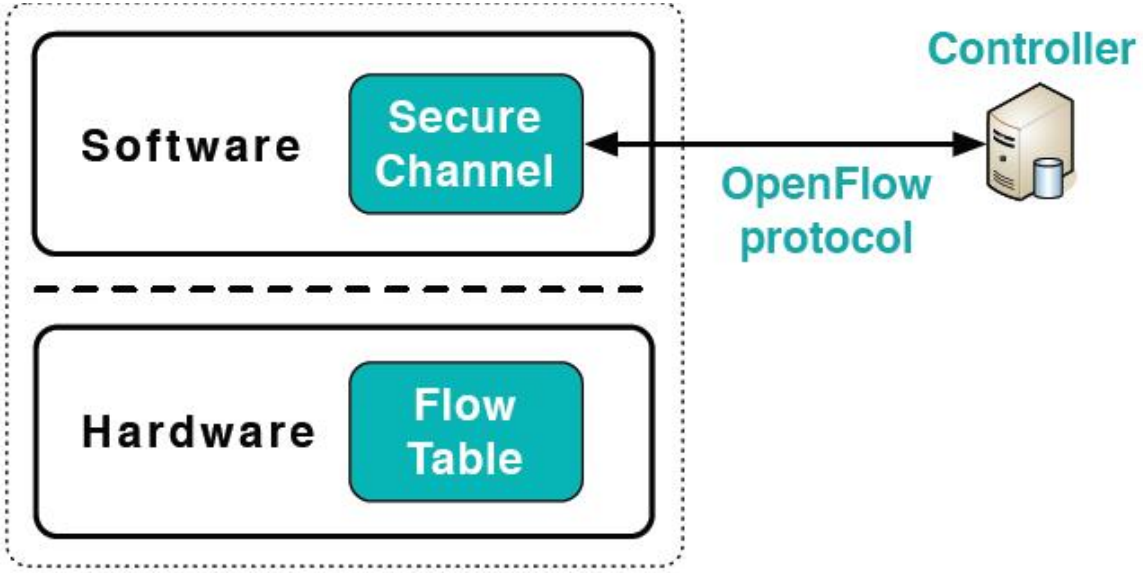
* Разделение уровней управления и передачи данных
* Управление данными с помощью контроллера

OpenFlow History

- USA NSF FIND (Future INternet Design) Program
  - 2006, Stanford and Berkley Univ.
  - SANE(clean-slate Security Architecture for Enterprise Network) project
  - Ethane project
    - MS and Ph.D thesis
- OpenFlow
  - 2007, Stanford Univ.
  - 2008, OpenFlow Consortium
  - 2008, Nicira Networks released NOX platform.
  - 2009, OpenFlow Spec 1.0
  - 2009 MIT Tech. Review → SDN as one of 10 emerging technologies
  - 2011 March, ONF (Open Networking Foundation) was born
  - Facebook, Google, Microsoft, Yahoo → Data Center Operators
  - Expand OpenFlow technologies to SDN
  - 2012 ONF released OpenFlow 1.3
  - 2013 ONF released OpenFlow 1.4
  - Dec. 19th, 2014, ONF released OpenFlow 1.5
  - 2015 April, OpenFlow 1.5.1
  - 2016 Jan, OpenFlow 1.5.2

OpenFlow коммутатор (v1.0)

- Таблица потоков – определяет, как коммутатор будет обрабатывать каждый поток
- Защищенный канал – соединяет коммутатор с удаленным контроллером
- OpenFlow protocol – стандарт для взаимодействия коммутатора с контроллером



Secure Channel

- Отправка сообщение в соответствии с OpenFlow протоколом
- Позволяет контроллеру конфигурировать, управлять и контролировать состояние коммутатора
- TLS сессия устанавливается по инициативе коммутатора, аутентификация осуществляется посредством сертификатов

OpenFlow Protocol Format

- Protocol Layer
  - OpenFlow control message relies on TCP protocol
  - Controllers listen on TCP port 6633/6653 to setup conn. with switch
    - 6633/6653 became the official IANA port since 2013-07-18
  - OpenFlow message structure
    - Version
      - Indicates the version of OpenFlow which this message belongs
    - Type
      - Indicates what type of message is present and how to interpret the payload (version dependent)
    - Message length
      - Indicates where this message will be end, starting from the first byte of header
    - Transaction ID (xid)
      - A unique value used to match requests to response

OpenFlow Message Structure

Bit Offset	0 ~ 7	8 ~ 15	16 ~ 23	24 ~ 31
0 ~ 31	Version	Type	Message Length	
32 ~ 63	Transaction ID			
64 ~ ?	Payload			

OpenFlow протокол

Поддерживает три типа сообщений:

- **Сообщения контроллер-коммутатор**
  - Конфигурирование коммутатора
  - Управление и контроль состояния
  - Управление таблицами (flow tables)
- **Симметричные сообщения**
  - Отправка в обоих направлениях
  - Обнаружение проблем соединения кон[illegible] коммутатором
- **Асинхронные сообщения**
  - Отправка от коммутатора к контроллеру
  - Объявляют об изменении состояния сети, состояния коммутаторов.

OpenFlow Protocol Messages

№	Тип	Направление передачи
Неизменные сообщения		
1	OFPT_HELLO	Симметричное
2	OFPT_ERROR	Симметричное
3	OFPT_ECHO_REQUEST	Симметричное
4	OFPT_ECHO_REPLY	Симметричное
5	OFPT_EXPERIMENTER	Симметричное
Сообщения конфигурирования OpenFlow-коммутатора		
6	OFPT_FEATURES_REQUEST	Контроллер ↔ Коммутатор
7	OFPT_FEATURES_REPLY	Контроллер ↔ Коммутатор
8	OFPT_GET_CONFIG_REQUEST	Контроллер ↔ Коммутатор
9	OFPT_GET_CONFIG_REPLY	Контроллер ↔ Коммутатор
10	OFPT_SET_CONFIG	Контроллер ↔ Коммутатор

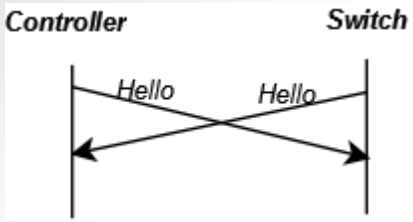
OpenFlow Protocol Messages

Асинхронные сообщения		
11	OFPT_PACKET_IN	Асинхронное
12	OFPT_FLOW_REMOVED	Асинхронное
13	OFPT_PORT_STATUS	Асинхронное
Сообщения-команды контроллера		
14	OFPT_PACKET_OUT	Контроллер ↔ Коммутатор
15	OFPT_FLOW_MOD	Контроллер ↔ Коммутатор
16	OFPT_GROUP_MOD	Контроллер ↔ Коммутатор
17	OFPT_PORT_MOD	Контроллер ↔ Коммутатор
18	OFPT_TABLE_MOD	Контроллер ↔ Коммутатор
Составные сообщения		
19	OFPT_MULTIPART_REQUEST	Контроллер ↔ Коммутатор
20	OFPT_MULTIPART_REPLY	Контроллер ↔ Коммутатор

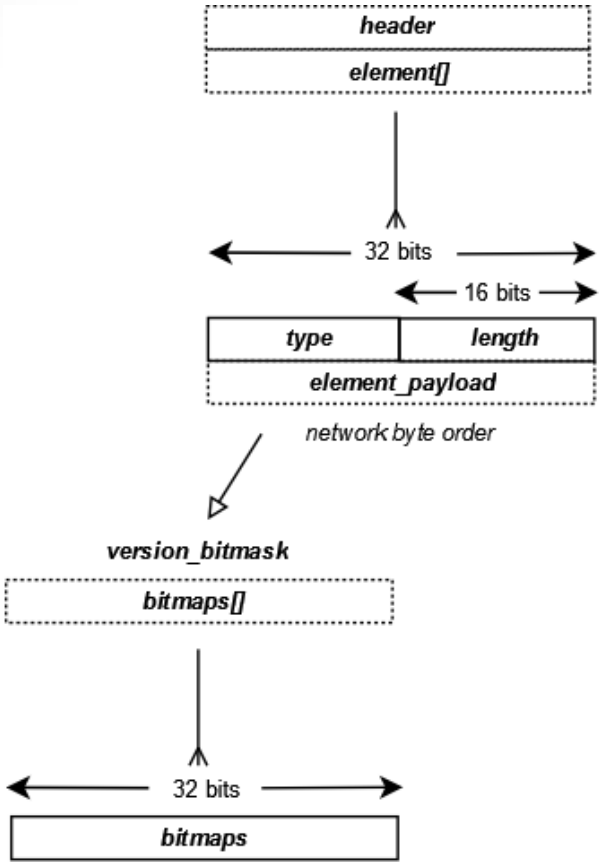
OpenFlow Protocol Messages

Barrier сообщения		
21	OFPT_BARRIER_REQUEST	Контроллер ↔ Коммутатор
22	OFPT_BARRIER_REPLY	Контроллер ↔ Коммутатор
Сообщения конфигурации очередей		
23	OFPT_QUEUE_GET_CONFIG_REQUEST	Контроллер ↔ Коммутатор
24	OFPT_QUEUE_GET_CONFIG_REPLY	Контроллер ↔ Коммутатор
Асинхронные сообщения конфигурации		
25	OFPT_GET_ASYNC_REQUEST	Контроллер ↔ Коммутатор
26	OFPT_GET_ASYNC_REPLY	Контроллер ↔ Коммутатор
27	OFPT_SET_ASYNC	Контроллер ↔ Коммутатор
Сообщения конфигурации таблицы измерений		
28	OFPT_METER_MOD	Контроллер ↔ Коммутатор

OpenFlow сообщения Hello

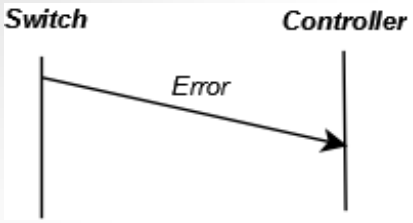


Hello используется либо контроллером или коммутатором для установления соединения.
Оно используется для определения версии протокола.
Когда соединение установлено, каждая сторона должна немедленно отправить сообщение Hello с полем версии установлен в самой высоком значении, поддерживаемом отправителем. Если версия протокола не совпадает, сообщение об ошибке посылается с типом HelloFailed и code Incompatible.

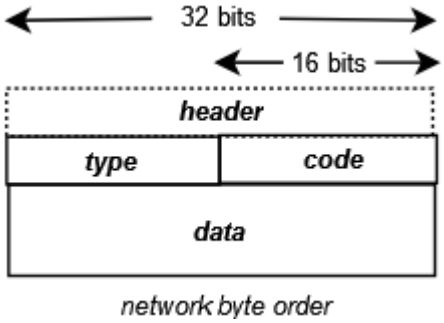


Bitmaps.
Поле заполняется поддерживаемыми версиями.
Например, **1.0.0 (версия 0x01**) **и 1.3 (версия 0x04**) будет иметь bitmap... **10010 (0x00000012**).
Согласованными должны быть самые высокие версии поддерживаемые обеими сторонами.
В противном случае должна быть принята следующая из указанных в версии полей. Если значение версий превышает 32 бита, то будет использоваться вторые 32 бита битовой карты.
Обратите внимание, что младший бит всегда равно 0.

OpenFlow сообщения Error

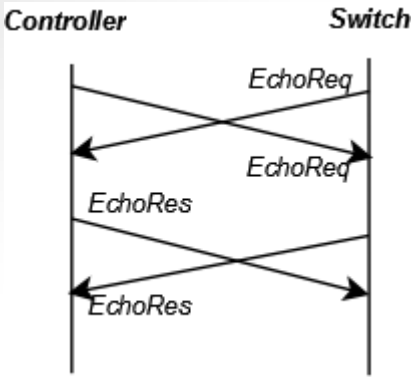


Сообщение об ошибке может быть отправлено либо коммутатором или контроллером и указывает на отказ в операции. Самый простой отказ относится к некорректному сообщению или версии протокола, в то время как более сложные сценарии могут идентифицировать сбой в коммутаторе и т.д. Все сообщения об ошибках начинаются со стандартного заголовка OpenFlow, содержащего версию и тип, за которыми следуют данные о структуре ошибки.

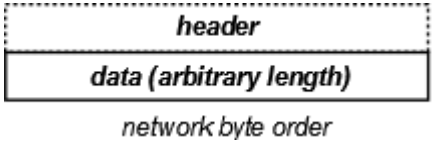


Name	Type	Name	Code
Hello Failed	0x0000	Incompatible	0x0000
		EPerm	0x0001
Bad Request	0x0001	BadVersion	0x0000
		BadType	0x0001
		BadMultipart	0x0002
		BadExperimenter	0x0003
		BadExperimenterType	0x0004
		EPerm	0x0005
		BadLength	0x0006
		BufferEmpty	0x0007
		BufferUnknown	0x0008
		BadTableID	0x0009
		IsSlave	0x000A
		BadPort	0x000B
		BadPacket	0x000C
		MultipartBufferOverflow	0x000D

OpenFlow сообщения EchoRequest и EchoResponse



Echo используется для обмена информацией о времени ожидания, пропускной способности и доступности (latency, bandwidth and liveness). Большой Тайм-аут на **EchoRequest** указывает на недоступность.

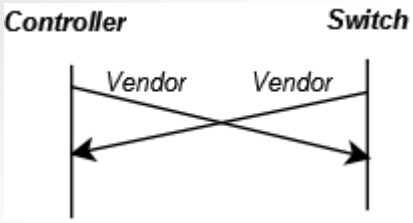


EchoReq имеет заголовок и необязательно поле переменной длины, содержащее данные, произвольную информацию, такую как задержка (метка времени), полоса пропускания, доступность и т.д.

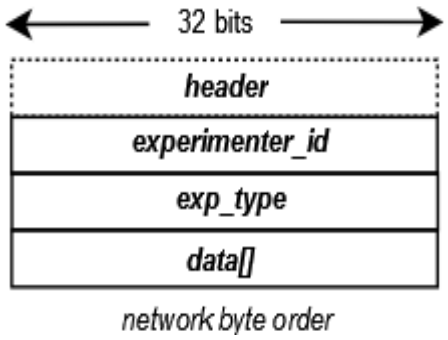
Сообщение EchoRes имеет ту же структуру что и EchoReq.

Данные EchoRes представляет собой точную копию данных, присутствующих из соответствующего EchoReq.

OpenFlow сообщения Vendor - Experimenter



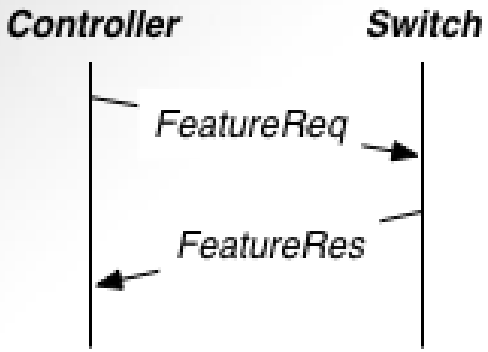
Сообщения **Vendor - Experimenter** представляет собой механизм для создания собственных сообщений в рамках протокола. В версии 1.0 он назывался **Vendor**; Однако в последующих версиях название было изменено на **- Experimenter** .



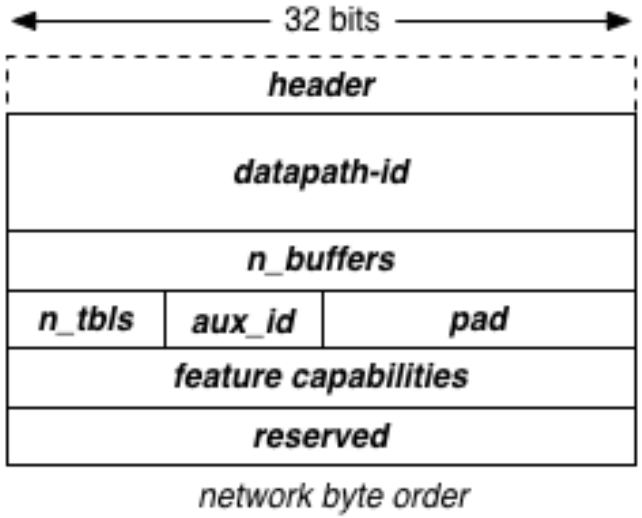
Сообщение в этой версии имеет заголовок, 4 байта experimenter_id, 4 байта exp_type и определяемые пользователем данных произвольной длины.

Name	Bits	Constraints
experimenter _id	32	none
exp_type	32	none
data[]	-	none

OpenFlow сообщения FeatureReq - FeatureRes



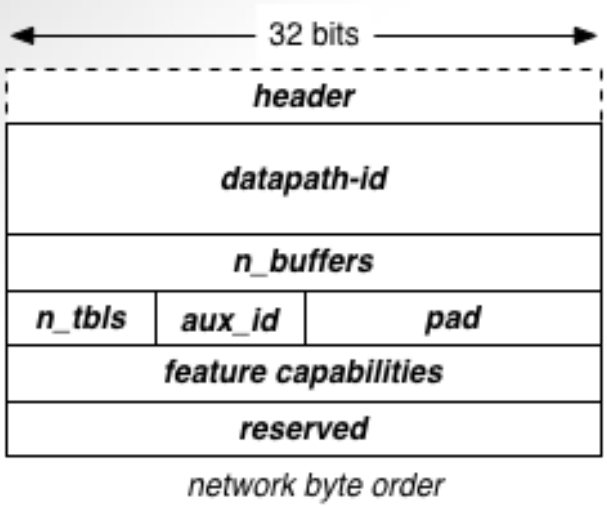
Инициатор: Контроллер
Ответное сообщение: Требуется



После установки транспортного канала (TCP, SCTP, TLS) между контроллером и коммутатором, первоочередным мероприятием является определение функции. Контроллер отправляет **FeatureReq** к коммутатору через транспортный канал. Запрос состоит только из заголовка со значением **FeatureReq** в поле Type. Когда контроллер ответит на этот запрос он пропишет свои параметры. Данный сценарий является базовым для всех версий протокола OpenFlow.

В **FeatureRes** от коммутатора к контроллеру перечисляются его функуиональные возможности.
Поле **datapath-id** представляет собой 64-битовое поле, которое следует рассматривать как уникальный идентификатор для упрвления конкретным конвейером обработки пакетов. Один физический коммутатор может иметь более одного datapath-id (имея ввиду виртуализацию коммутаторов).
Поле **n_buffers** определяет сколько пакетов коммутатор может по очереди передать в PacketIn к контроллеру.
Количество таблиц в контроллере передается в **n_tbls**.
В **auxiliary_id** показан способ подключения openflow канала (master controller или auxiliary).
Pad используются для поддержания байтового выравнивания в случае необходимости.
Capabilities указывают, какие общие функции и действия поддерживаются коммутатором.
И зарезервированное поле **reserved.**

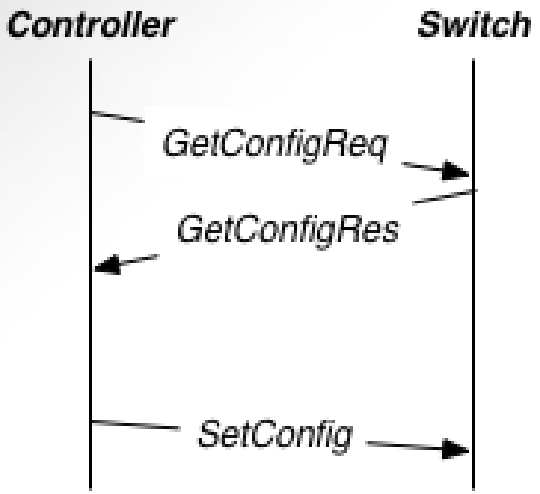
OpenFlow сообщения FeatureReq - FeatureRes



Name	Bits	Constraints
datapath-id	64	none
n_buffers	32	none
n_tables	8	none
auxiliary_id	8	none
pad	16	none
capabilities	32	see below
reserved	32	none

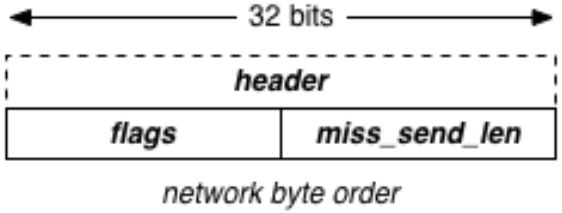
Field	Name	Value
Capabilities	FLOW_STATS	0x00000001
	TABLE_STATS	0x00000002
	PORT_STATS	0x00000004
	GROUP_STATS	0x00000008
	IP_REASM	0x00000020
	QUEUE_STATS	0x00000040
	PORT_BLOCKED	0x00000100

OpenFlow сообщения GetConfigReq - GetConfigRes - SetConfig



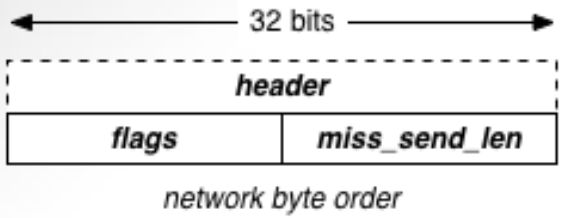
Инициатор: Контроллер
Ответное сообщение: Требуется

Последовательность сообщения используется для запроса и установки свойств обработки фрагментированных пакетов в конвейера обработки пакетов. Должен ли пакеты, которые приходят разбитыми на отдельные куски быть: передан как есть, отброшен, повторно собран и согласован, и т.д. Кроме того, это сообщение помогает определить, сколько пакета будет согласовываться с контроллером.
GetConfigReq является транзакционным сообщение (GetConfigRes) и инициируется только контроллером. Контроллер может изменить состояние обработки пакета с сообщением SetConfig. SetConfig может быть инициировано только контроллером, и является транзакционным.



GetConfigReq состоит только из заголовка OpenFlow.
В GetConfigRes и сообщения SetConfig состоят из заголовка и двух 16 битовых полей**: flags** и **miss_send_length.**
Flags указывают, какие операции обработки фрагмента являются активными (GetConfigRes) или какие операции на установить (SetConfig).
Поле **miss_send_length** указывает количество байтов пакета для захвата и передачи с сообщением **PacketIn** на контроллер. **Значение 0xFFFF** указывает на то, что надо передавать весь пакет.

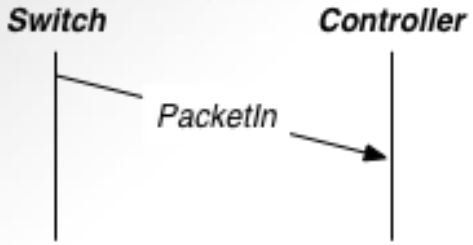
OpenFlow сообщения
GetConfigReq - GetConfigRes - SetConfig



Name	Bits	Constraints
flags	16	See below
miss_send_len	16	none

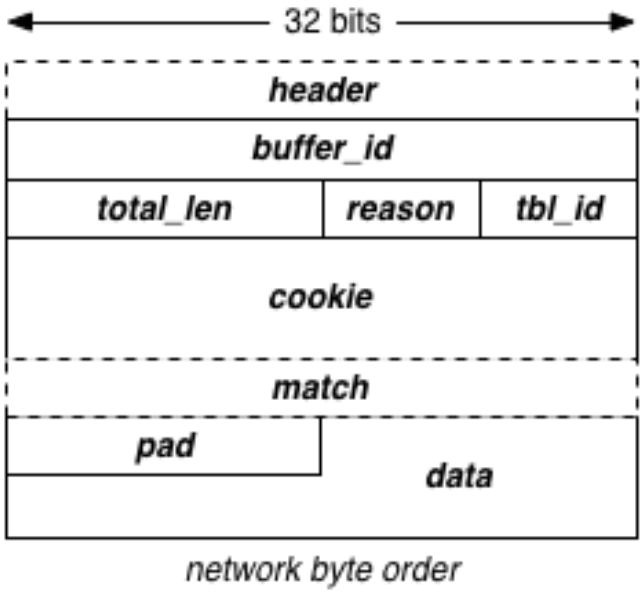
eId	Name	Value	Comments
Flags	FragNormal	0x0000	Match packets, regardless of state
	FragDrop	0x0001	Drop fragmented packets
	FragReasm	0x0002	Reassemble fragmented packets
	FragMask	0x0003	Mask for fragmentation

OpenFlow сообщение PacketIn



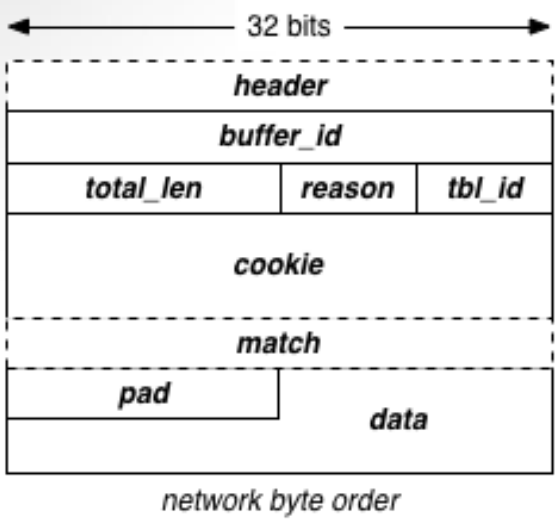
Сообщение **PacketIn** – это способ для отправки полученного пакета к контроллеру. Есть несколько причин, почему это может произойти: может быть прописан конкретный action при совпадении match field, или отсутствие в таблицах соответствия match field или ошибка **ttl**.

Инициатор: Коммутатор
Ответное сообщение: не требуется



Сообщение PacketIn состоит из **заголовка**, за которым следует **buffer_id**.
buffer_id - это уникальное значение, используемое для отслеживания буферизованного пакета.
Длина захваченного пакета указана в **total_len**.
Поле **reason** указывает почему пакет был передан на контроллер.
Tbl_id – идентификатор таблицы, где было совпадение.
Cookie – это параметры потока, которые передаются на контроллер.
Match – метаданные пакета, переменной длины.
Pad используются для поддержания байтового выравнивания в случае необходимости.
Data – Ethernet фрейм.

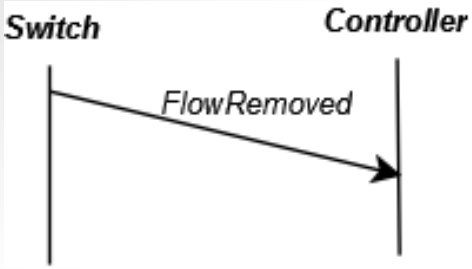
OpenFlow сообщение PacketIn



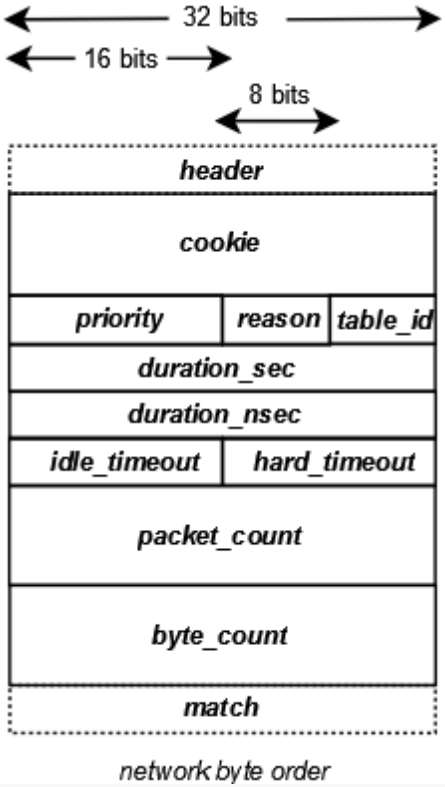
Name	Bits	Constraints
buffer_id	32	none
total_len	16	none
reason	8	See table
tbl_id	8	none
cookie	64	none
match	32	Match restrictions
pad	16	none
data	48	none

Field	Name	Value
Reason	NoMatch	0x00
	Action	0x01
	InvalidTTL	0x02

OpenFlow сообщение FlowRemoved



Инициатор: Коммутатор
Ответное сообщение: не требуется



FlowRemoved посылается к контроллеру, когда запись потока в таблице потоков удаляется. Это происходит при срабатывании тайм-аут, либо из-за бездействия или по причине жесткого тайм-аута. Обычный Тайм-аут срабатывает, когда в течение определенного времени не было пакетов с нужным match field. Жесткий тайм-аут задается изначально независимо от числа совпадающих пакетов. Будет ли коммутатор посылает сообщение FlowRemoved после тайм-аута определяется сообщением FlowMod. Удаление потока при помощи FlowMod от контроллера также может привести к сообщению FlowRemoved.

Cookie – идентификатор выданный контроллером;

Priority – уровень приоритета для потока в таблицах;

Reason – причина удаления записи о потоке;

Table_id – идентификатор таблицы;

Duration_sec – время жизни записи о потоке в секундах;

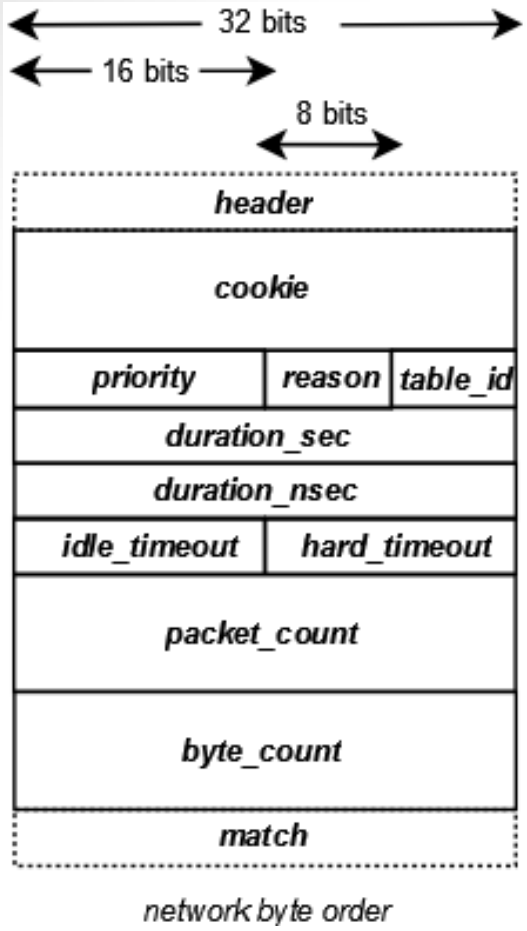
Duration_nsec – время жизни записи о потоке наносекундах помимо duration_sec;

Idle_timeout – тайм аут переданный в исходном сообщении FlowMod;

Hard_timeout – жесткий тайм аут переданный в исходном сообщении FlowMod;

Packet_count, byte_count, match – описание полей, переменной длины;

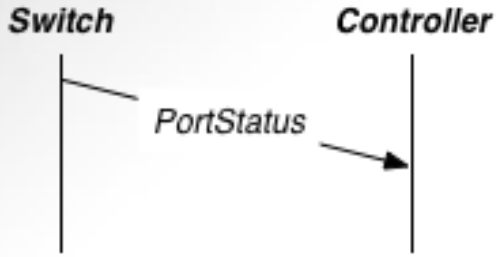
OpenFlow сообщение FlowRemoved



Name	Bits	Constraints
cookie	64	none
priority	16	none
reason	8	see table
table_id	8	none
duration_sec	32	none
duration_nsec	32	none
idle_timeout	16	none
hard_timeout	16	none
packet_count	64	none
byte_count	64	none
match	32	Match restrictions

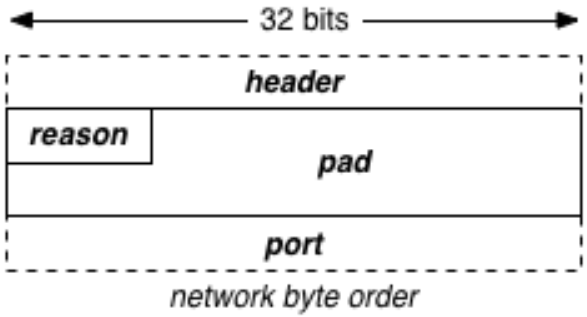
Field	Name	Code
Reason	IdleTimeout	0x00
	HardTimeout	0x01
	Delete	0x02
	GroupDelete	0x03

OpenFlow сообщение Port status



Эти сообщения являются асинхронными и информируют контроллер об изменении статуса указанного порта.

Инициатор: Коммутатор
Ответное сообщение: не требуется



Reason – причина передачи сообщения;

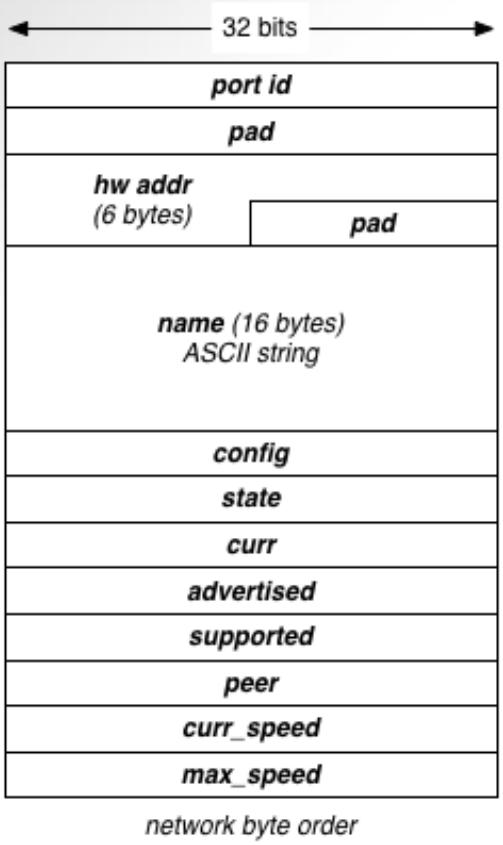
Pad используются для поддержания байтового выравнивания (7 байт).

Port – структурная единица протокола с описанием порта;

Name	Bits	Constraints
reason	8	See below
padding	56	none
port	-	port restrictions

Field	Name	Value
Reason	Add	0x00
	Delete	0x01
	Modify	0x02

OpenFlow Port structure



Структурная единица описывает один порт на openflow коммутаторе.

Port-Id определяет идентификатор порта, с нескольких зарезервированными значениями для специальных портов (Max, Flood, Controller, и т. д.).

Pad используются для поддержания байтового выравнивания (4 байта).

Hw addr - это адрес канального уровня, связанный с портом, например каждый порт коммутатора Ethernet имеет уникальный MAC.

Pad используются для поддержания байтового выравнивания (2 байта).

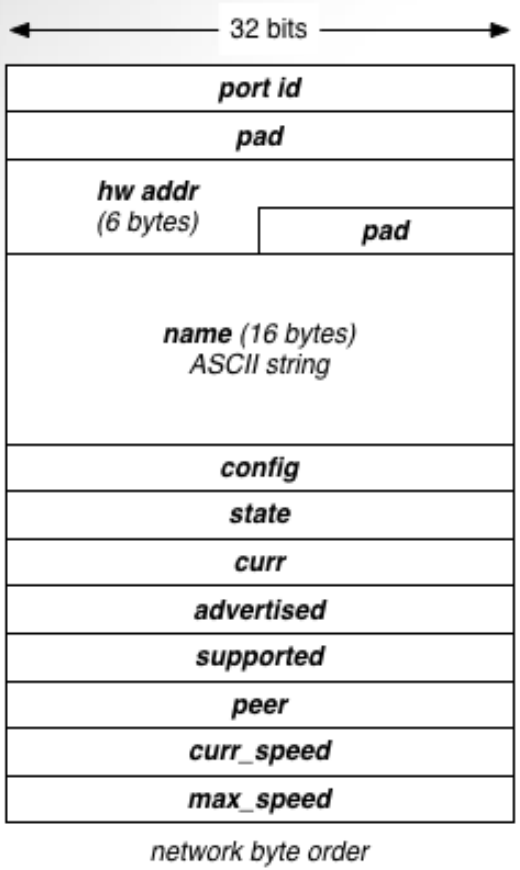
Name – название порта в формате ASCII NULL строки длиной 16 байт.

config - это код, представляющий конфигурацию порта.

state представляет собой текущее состояние порта (link down и т. д.).

Поля **curr, advertised, supported, peer, curr_speed, max_speed** используются для описания характеристик домена и порта.

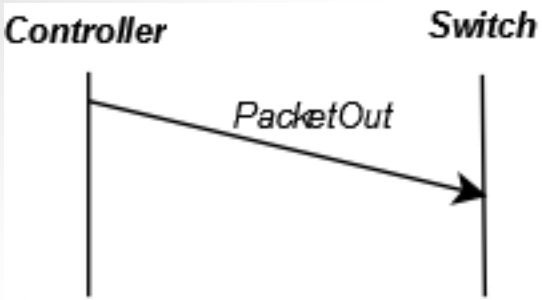
OpenFlow Port structure



Field	Name	Value
Id	Max	0xFFFFFF00
	InPort	0xFFFFFFF8
	Table	0xFFFFFFF9
	Normal	0xFFFFFFFA
	Flood	0xFFFFFFFB
	All	0xFFFFFFFC
	Controller	0xFFFFFFFD
	Local	0xFFFFFFFE
	Any	0xFFFFFFFF
Config	PortDown	0x00000001
	NoRecv	0x00000004
	NoFwd	0x00000020
	NoPacketIn	0x00000040

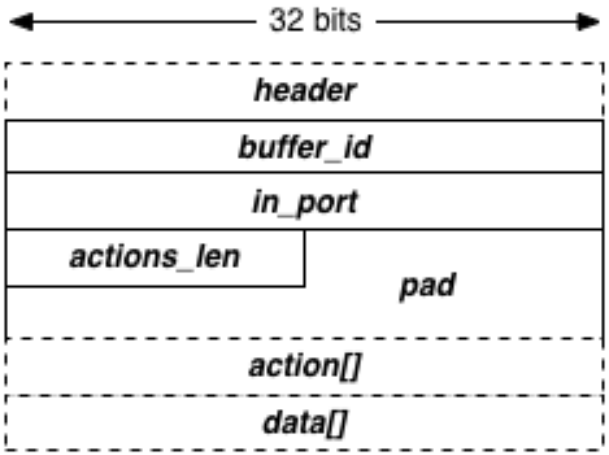
Field	Name	Value
State	LinkDown	0x00000001
	Blocked	0x00000002
	Live	0x00000004
Feature (curr, advertised, supported, peer, curr_speed, max_speed)	10MB_HD	0x00000001
	10MB_FD	0x00000002
	100MB_HD	0x00000004
	100MB_FD	0x00000008
	1GB_HD	0x00000010
	1GB_FD	0x00000020
	10GB_FD	0x00000040
	40GB_FD	0x00000080
	100GB_FD	0x00000100
	1TB_FD	0x00000200
	Other	0x00000400
	Copper	0x00000800
	Fiber	0x00001000
	AutoNeg	0x00002000
	Pause	0x00004000
	PauseAsym	0x00008000

OpenFlow сообщение PacketOut



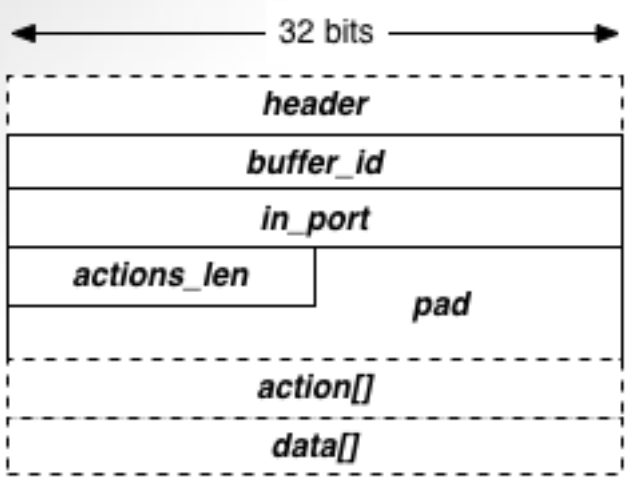
Инициатор: Контроллер
Ответное сообщение: не требуется

Контроллер имеет возможность поместить пакеты в data plane конкретного коммутатора, при помощи сообщения **PacketOut.** Сообщение содержит пакеты для передачи через коммутатор, или указывает локальный буфер на коммутаторе, куда пометить пакет для его передачи в сеть. Пакеты, вводимые в data plane коммутатора, через **PacketOut** обрабатывается отдельно от пакетов, поступающих на стандартные порты. Пакет переходит к действию, на основе стандартной конвейерной обработки пакетов. Если набор действий указывает на обработку в таблицах, то входной port id используется в качестве входящего порта этих пакетов.



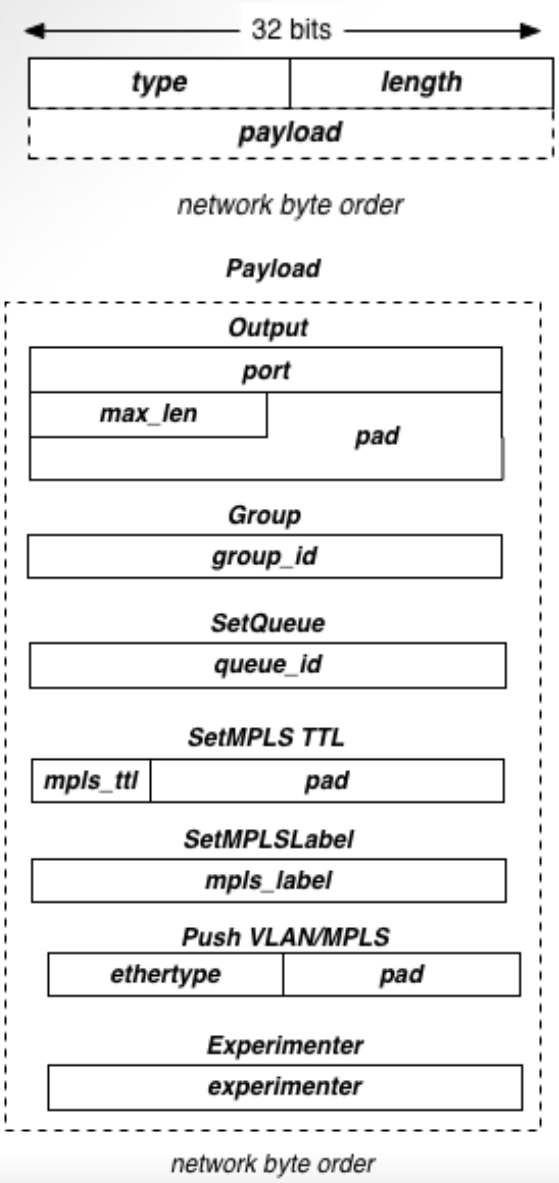
Сообщение PacketOut состоит из **заголовка, buffer_id** - указывает расположение сырых пакетов для ввода в плоскости данных коммутатора. Значение 0xffffffff означает, что исходный пакет содержится в данных байтового массива, в противном случае значение buffer_id указывает на локальный буфер пакетов коммутатора, который содержит пакеты.
In_port – указывает порта прибытия, если исходный пакет должен пройти стандартную обработку таблиц.
Actions_len указывает количество байтов, к которым применяется набор действий **Pad** используются для поддержания байтового выравнивания в случае необходимости. **Action** - это список действий для пакетов. **Data** - это массив байт, содержащих сырой пакет.

OpenFlow сообщение PacketOut

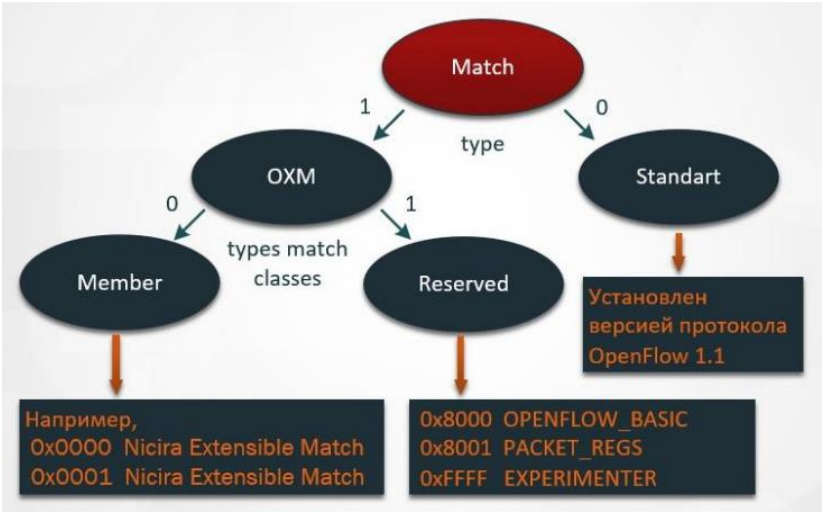


Name	Bits	Constraints
buffer_id	32	none
in_port	32	in 0..0xffffff00
actions_len	16	none
padding	48	none
action[]	32	-
data[]	-	none

OpenFlow Action structure



В версии 1.3 были добавлены: push/pop_PBB.
push_PBB использует такую же структуру, как push_VLAN/MPLS,
pop_PBB просто пустой Заголовок.



Type – определены два типа классификаторов. Первый тип – стандартный (значение поля «тип» равняется нулю), определены версией протокола OpenFlow 1.1 и считаются устаревшими, но базовыми для всех OpenFlow-коммутаторов. Второй тип – расширяемые классификаторыOpenFlow Extensible Match (OXM);

Length – динамически может изменяться в указанных пределах, зависит от количества полей классификаторов

OXM fields – поля классификаторов

Pad – заполнение нулевыми байтами

OpenFlow Action structure (Action Payload)

Type Name	Type Value	Field Name	Bits	Constraints
Output	0x0000	type	16	= 0x0000
		length	16	= 0x000f
		port	32	in 0x00000000..0xffffffef
		max_len	16	none
		pad	48	none
SetVLANVID	0x0001	type	16	= 0x0001
		length	16	= 0x0008
		vlan_vid	16	none
		pad	16	none
SetVLANPCP	0x0002	type	16	= 0x0002
		length	16	= 0x0008
		vlan_pc	8	none
		pad	24	none
SetDLSrc	0x0003	type	16	= 0x0003
		length	16	= 0x000f
		SetDLSrc	48	none
		pad	48	none
SetDLDst	0x0004	type	16	= 0x0004
		length	16	= 0x000f
		SetDLSrc	48	none
		pad	48	none
SetNWSrc	0x0005	type	16	= 0x0005
		length	16	= 0x0008
		nw_addr	32	none
SetNWDst	0x0006	type	16	= 0x0006
		length	16	= 0x0008
		nw_addr	32	none
SetNWTos	0x0007	type	16	= 0x0007
		length	16	= 0x0008
		nw_tos	8	none
		pad	24	none
SetNWECN	0x0008	type	16	= 0x0008
		length	16	= 0x0008
		nw_ecen	8	none
		pad	24	none
SetTPSrc	0x0009	type	16	= 0x0009
		length	16	= 0x0008
		port	16	in 0x000..0xffef
		pad	16	none
SetTPDst	0x000a	type	16	= 0x000a
		length	16	= 0x0008
		port	16	0x000...0xffef
		pad	16	none
CopyTTLOut	0x000b	type	16	= 0x000b
		length	16	= 0x0004
CopyTTLIn	0x000c	type	16	= 0x000c
		length	16	= 0x0004
SetMPLSLabel	0x000d	type	16	= 0x000d
		length	16	= 0x0008
		mpls_label	32	none
SetMPLSTC	0x000e	type	16	= 0x000e
		length	16	= 0x0008
		mpls_tc	8	none
		pad	24	none
SetMPLSTTL	0x000f	type	16	= 0x000f
		length	16	= 0x0008
		mpls_ttl	8	none
		pad	24	none
DecMPLSTTL	0x0010	type	16	= 0x0010
		length	16	= 0x0004

OpenFlow Action structure (Action Payload)

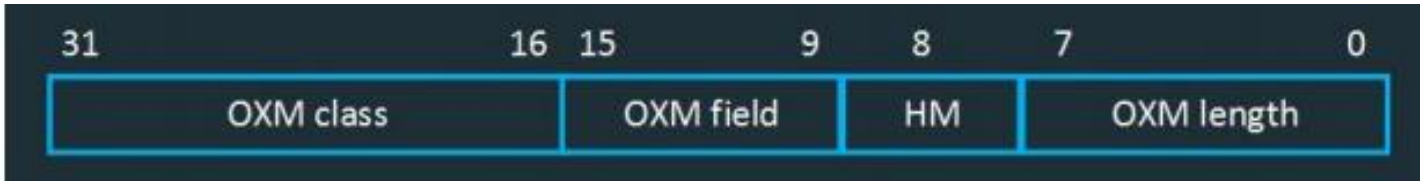
PushVLAN	0x0011	type	16	= 0x0011
		length	16	= 0x0008
		ethertype	16	none
		pad	16	none
PopVLAN	0x0012	type	16	= 0x0012
		length	16	= 0x0004
PushMPLS	0x0013	type	16	= 0x0013
		length	16	= 0x0008
		mpls	16	none
		pad	16	none
PopMPLS	0x0014	type	16	= 0x0014
		length	16	= 0x0004
SetQueue	0x0015	type	16	= 0x0015
		length	16	= 0x0008
		queue_id	32	none
Group	0x0016	type	16	= 0x0016
		length	16	= 0x0008
		group_id	32	none
SetNWTTL	0x0017	type	16	= 0x0017
		length	16	= 0x0008
		ttl	8	none
		pad	24	none
DecNWTTL	0x0018	type	16	= 0x0018
		length	16	= 0x0004
SetField	0x0019	type	16	= 0x0019
		length	16	≥ 8
		OXM TLV	≥ 32	OXM TLV

PushPBB	0x001a	type	16	MSBF	= 0x001a
		length	16	MSBF	= 0x0008
		ethertype	16	MSBF	none
		pad	16	-	none
PopPBB	0x001b	type	16	MSBF	= 0x001b
		length	16	MSBF	= 0x0004
Experimenter	0xffff	type	16	MSBF	= 0xffff
		length	16	MSBF	= 0x0008
		Experimenter	32	MSBF	none

OpenFlow Action structure (OXM fields)

OpenFlow Extensible Match (OXM)

Все поля соответствия классификатора описываются, используя динамически расширяемый формат OXM, которые в совокупности (вместе) образуют компактный type-length-value (TLV) формат. Каждый OXM TLV классификатор может быть от 5 до 259 (включительно) байт длиной. Первые 4 байта каждого OXM TLV классификатора – заголовок.



1. **OXM class** – спецификация OpenFlow разделяет два типа OXM классов:
   ONF reserved – классы, описываемые спецификацией и зарезервированные для последующих стандартизаций;
   ONF member – классы, используемые членами консорциума ONF. Позволяют членам консорциума использовать свои классификаторы и уникально идентифицироваться (напр. Nicira).

2. **OXM field** – поле используется для определения конкретного поля соответствия в каждом типе класса;

3. **OXM hasmask** – поле может принимать значение 0 или 1. Определяет наличие в поле данных OXM битовую маску;

4. **OXM length** – длина полезных данных OXM TLV.

OpenFlow Action structure (OXM fields)

Рассмотрим классы **Reserved** на примере протокола OpenFlow версии 1.5.1. В спецификации протокола определены три класса в данном типе: **basic, register и experimenter.**

1. **experimenter** – класс используется для расширения стандартного набора полей соответствия в OpenFlow-коммутаторе;
2. **register** – класс используется для хранения временных значений и информации, связанной с пакетом в процессе обработки конвейером;
3. **OpenFlow basic** – основной класс классификаторов, который содержит базовый набор полей соответствия

Поля соответствия класса OpenFlow basic (header match fields)

№	Math-поле	Размер (бит)	Маска	Описание
1	OFPXMT_OFB_ETH_DST	48	Да	MAC-адрес получателя
2	OFPXMT_OFB_ETH_SRC	48	Да	MAC-адрес отправителя
3	OFPXMT_OFB_TYPE	16	Нет	Ethernet type
4	OFPXMT_OFB_VLAN_VID	12+1	Да	VLAN-ID по стандарту 802.1Q
5	OFPXMT_OFB_VLAN_PCP	3	Нет	VLAN-PCP по стандарту 802.1Q
6	OFPXMT_OFB_IP_DSCP	6	Нет	Diff Serv Code Point (DSCP)
7	OFPXMT_OFB_IP_ECN	2	Нет	ECN бит в заголовке IP

OpenFlow Action structure (OXM fields)

Поля соответствия класса OpenFlow basic (header match fields). продолжение

8	OFPXMT_OFB_IP_PROTO	8	Нет	IPv4 или IPv6 номер протокола
9	OFPXMT_OFB_IPV4_SRC	32	Да	IPv4 адрес отправителя
10	OFPXMT_OFB_IPV4_DST	32	Да	IPv4 адрес получателя
11	OFPXMT_OFB_TCP_SRC	16	Нет	TCP порт отправителя
12	OFPXMT_OFB_TCP_DST	16	Нет	TCP порт получателя
13	OFPXMT_OFB_TCP_FLAGS	12	Нет	TCP флаги
14	OFPXMT_OFB_UDP_SRC	16	Нет	UDP порт отправителя
15	OFPXMT_OFB_UDP_DST	16	Нет	UDP порт получателя
16	OFPXMT_OFB_SCTP_SRC	16	Нет	SCTP порт отправителя
17	OFPXMT_OFB_SCTP_DST	16	Нет	SCTP порт получателя
18	OFPXMT_OFB_ICMPV4_TYPE	8	Нет	ICMP тип
19	OFPXMT_OFB_ICMPV4_CODE	8	Нет	ICMP код
20	OFPXMT_OFB_ARP_OP	16	Нет	ARP код отправителя/получа-теля
21	OFPXMT_OFB_ARP_SPA	32	Да	IPv4 адрес отправителя в pay-load протокола ARP
22	OFPXMT_OFB_ARP_TPA	32	Да	IPv4 адрес получателя в payload протокола ARP

OpenFlow Action structure (OXM fields)

Поля соответствия класса OpenFlow basic (header match fields). продолжение

23	OFPXMT_OFB_ARP_SHA	48	Да	MAC-адрес отправителя в payload протокола ARP
24	OFPXMT_OFB_ARP_THA	48	Да	MAC-адрес получателя в payload протокола ARP
25	OFPXMT_OFB_IPV6_SRC	128	Да	IPv6 адрес отправителя
26	OFPXMT_OFB_IPV6_DST	128	Да	IPv6 адрес получателя
27	OFPXMT_OFB_IPV6_FLABEL	20	Да	IPv6 метка потока
28	OFPXMT_OFB_ICMPV6_TYPE	8	Нет	ICMPv6 тип
29	OFPXMT_OFB_ICMPV6_CODE	8	Нет	ICMPv6 код
30	OFPXMT_OFB_IPV6_ND_TARGET	128	Нет	IPv6, сообщение NDM, адрес получателя
31	OFPXMT_OFB_IPV6_ND_SLL	48	Нет	IPv6, сообщение NDM, source link-layer address option
32	OFPXMT_OFB_IPV6_ND_TLL	48	Нет	IPv6, сообщение NDM, target link-layer address option
33	OFPXMT_OFB_MPLS_LABEL	20	Нет	MPLS метка
34	OFPXMT_OFB_MPLS_TC	3	Нет	MPLS класс трафика
35	OFPXMT_OFB_MPLS_BOS	1	Нет	MPLS флаг «дно стека»
36	OFPXMT_OFB_PBB_ISID	24	Да	Технология PBB, идентификатор сервиса I-SID
37	OFPXMT_OFB_IPV6_EXTHDR	9	Да	IPv6 Extension Header pseudo-field
38	OFPXMT_OFB _PBB_UCA	1	Нет	Технология PBB, поле UCA

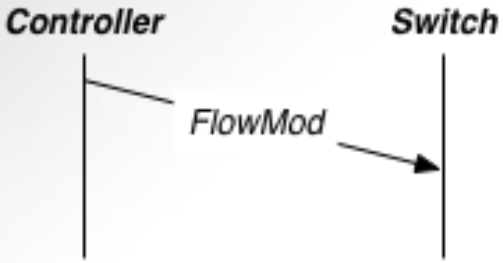
OpenFlow Action structure (OXM fields)

Второй тип полей соответствия содержит поля, которые используются в процессе обработки конвейером (pipeline match fields) и никаким образом не взаимодействуют с заголовками пакета.

Поля соответствия класса OpenFlow basic (pipeline match fields)

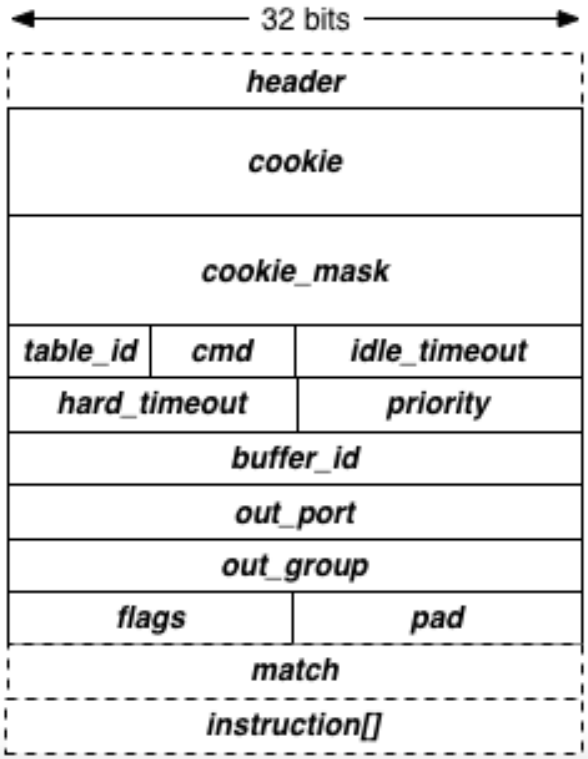
№	Math-поле	Размер (бит)	Маска	Описание
1	OFPXMT_OFB_IN_PORT	32	Нет	Входящий порт. Номер логического или физического входящего порта, начиная с единицы.
2	OFPXMT_OFB_IN_PHY_PORT	32	Нет	Физический порт. Соответствие физического и логического порта, когда сообщение приходит на логический порт.
3	OFPXMT_OFB_METADATA	64	Да	Используется для передачи информации между таблицами.
4	OFPXMT_OFB_TUNNEL_ID	64	Да	Метаданные, ассоциированные с логическим портом.
5	OFPXMT_OFB_ACTSET_OUTPUT	32	Нет	Output port from action set Metadata.
6	OFPXMT_OFB_PACKET_TYPE	16+16	Нет	Packet type - canonical header type of outermost header.

OpenFlow сообщение FlowMod



Это одно из основных сообщений, для изменения состояния коммутатора контроллером. Все FlowMod сообщения начинаются со стандартного заголовка openflow, содержащего соответствующую версию и тип, с дальнейшей передачей структуры FlowMod.

Инициатор: Контроллер
Ответное сообщение: не требуется



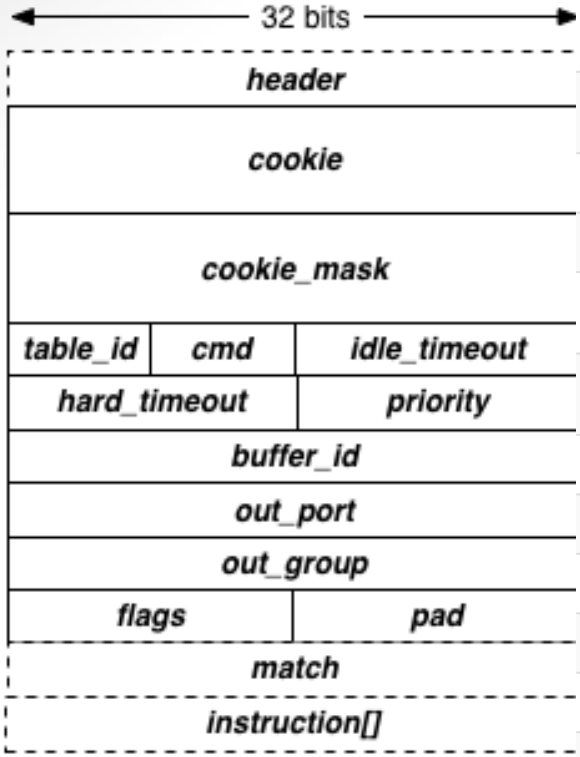
Сообщение начинается со стандартного **Header**, поля **Cookie**, которое является специальным полем, формируемым контроллером, его маски (**Cookie_mask**), идентификаторf таблицы (**Table_id**) и команды (cmd), которая определяет тип таблицы потоков.

Idle_timeout и hard_timeout представляют собой количество секунд с момента неактивности пакетов и с момента создания, соответственно.

Priority подразумевает указание на match, которые пересекаются с другими, более высокими приоритетами.

Buffer_id, out_port, out_group и flag - это ид. буферизированного пакета, который создал packet_in, запросил FlowMod, а затем поступает в обработку.
Match и Insruction – описывают действия и инструкции к пакету.

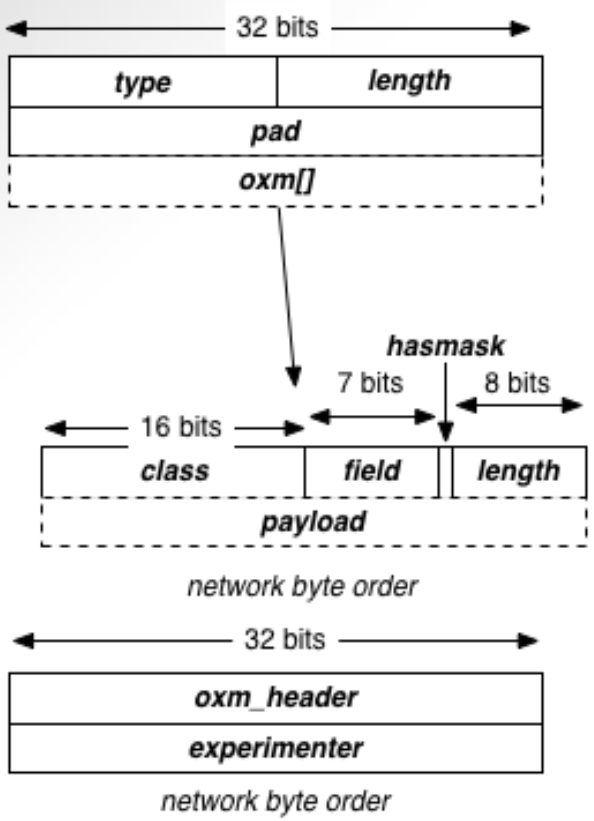
OpenFlow сообщение FlowMod



Name	Bits	Constraints
cookie	64	!= 0xFFFFFFFFFFFFFFFF
cookie_mask	64	none
table_id	8	!= 0xFF
command	8	See below
idle_timeout	16	none
hard_timeout	16	none
priority	16	none
buffer_id	32	none
out_port	32	none
out_group	32	none
flags	16	See below
padding	16	none
match	32	OXM
instruction[]	32	instruction restrictions

Field	Name	Value
Command	Add	0x00
	Modify	0x01
	ModifyStrict	0x02
	Delete	0x03
	DeleteStrict	0x04
Flag	SendFlowRem	0x0001
	CheckOverlap	0x0002
	ResetCounts	0x0004
	NoPacketCounts	0x0010
	NoByteCounts	0x0020

OpenFlow Match structure



С версии OF 1.2. кардинально меняется структуры Match
Вводится OpenFlow Extensible Match (OXM).
Для поддержки старой версии есть только один класс - OpenFlowBasic.

1. **OXM class** – спецификация OpenFlow разделяет два типа OXM классов:

   ONF reserved – классы, описываемые спецификацией и зарезервированные для последующих стандартизаций;

   ONF member – классы, используемые членами консорциума ONF. Позволяют членам консорциума использовать свои классификаторы и уникально идентифицироваться (напр. Nicira).

2. **OXM field** – поле используется для определения конкретного поля соответствия в каждом типе класса;

3. **OXM hasmask** – поле может принимать значение 0 или 1. Определяет наличие в поле данных OXM битовую маску;

4. **OXM length** – длина полезных данных OXM TLV.

OpenFlow Match structure

OXM TLV Constraints

Class Name	Class Value	Field Name	Field Length	Field Value	Hasmask	Dependencies
nxm_0	0x0000	-	-	-	-	-
nxm_1	0x0001	-	-	-	-	-
OpenFlowBasic	0x8000	in_port	32	0x00	0	none
		in_phy_port	32	0x01	0	in_port
		metadata	64	0x02	0	none
			128	0x02	1	none
		eth_dst	48	0x03	0	none
			96	0x03	1	none
		eth_src	48	0x04	0	none
			96	0x04	1	none
		eth_type	16	0x05	0	none
		vlan_vid	13	0x06	0	none
			26	0x06	1	none
		vlan_pcp	3	0x07	0	vlan_vid != none
		ip_dscp	6	0x08	0	eth_type = 0x0800 || 0x86dd
		ip_ecn	2	0x09	0	eth_type = 0x0800 || 0x86dd
		ip_proto	8	0x0a	0	eth_type = 0x0800 || 0x86dd
		ipv4_src	32	0x0b	0	eth_type = 0x0800
			64	0x0b	1	eth_type = 0x0800
		ipv4_dst	32	0x0c	0	eth_type = 0x0800
			64	0x0c	1	eth_type = 0x0800

OpenFlow Match structure

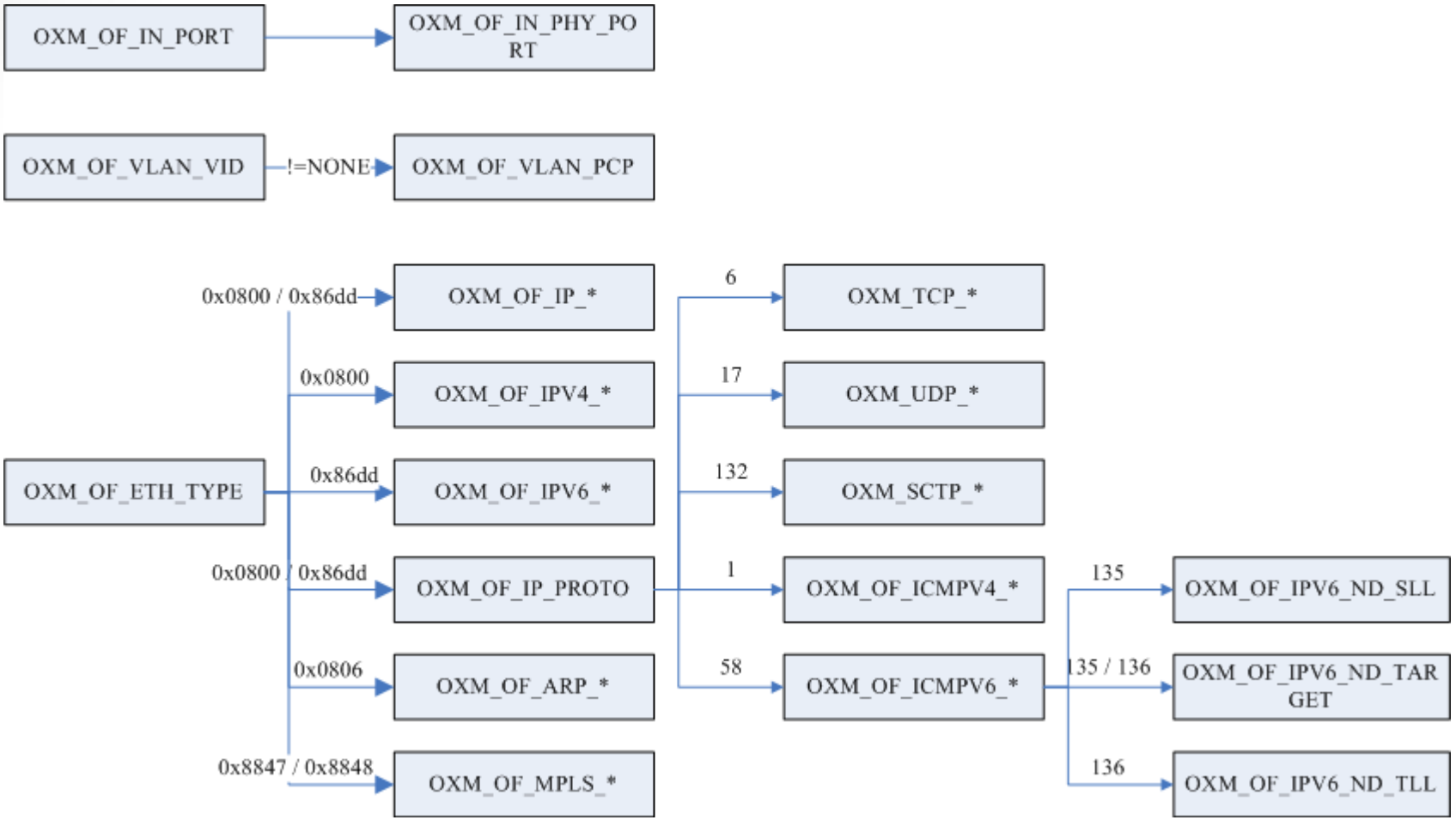
OXM TLV Constraints

Field Name	Field Length	Field Value	Hasmask	Dependencies
tcp_src	16	0x0d	0	ip_proto = 6
tcp_dst	16	0x0e	0	ip_proto = 6
udp_src	16	0x0f	0	ip_proto = 17
udp_dst	16	0x10	0	ip_proto = 17
sctp_src	16	0x11	0	ip_proto = 132
sctp_dst	16	0x12	0	ip_proto = 132
icmpv4_type	8	0x13	0	ip_proto = 1
icmpv4_code	8	0x14	0	ip_proto = 1
arp_op	16	0x15	0	eth_type = 0x0806
arp_spa	32	0x16	0	eth_type = 0x0806
	64	0x16	1	eth_type = 0x0806
arp_tpa	32	0x17	0	eth_type = 0x0806
	64	0x17	1	eth_type = 0x0806
arp_sha	48	0x18	0	eth_type = 0x0806
	96	0x18	1	eth_type = 0x0806
arp_tha	48	0x19	0	eth_type = 0x0806
	96	0x19	1	eth_type = 0x0806
ipv6_src	128	0x1a	0	eth_type = 0x86dd
	256	0x1a	1	eth_type = 0x86dd
ipv6_dst	128	0x1b	0	eth_type = 0x86dd
	256	0x1b	1	eth_type = 0x86dd
ipv6_flabel	20	0x1c	0	eth_type = 0x86dd
	40	0x1c	1	eth_type = 0x86dd

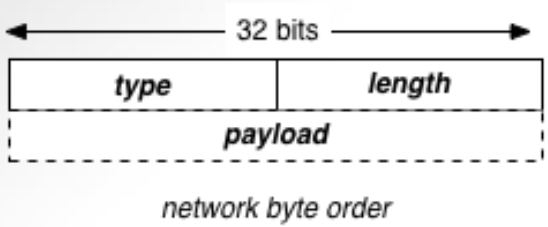
Field Name	Field Length	Field Value	Hasmask	Dependencies
icmpv6_type	8	0x1d	0	ip_proto = 58
icmpv6_code	8	0x1e	0	ip_proto = 58
ipv6_nd_target	128	0x1f	0	icmpv6_type = 135||136
ipv6_nd_sll	48	0x20	0	icmpv6_type = 135
ipv6_nd_tll	48	0x21	0	icmpv6_type = 136
mpls_label	20	0x22	0	eth_type = 0x8847||0x8848
mpls_tc	3	0x23	0	eth_type = 0x8847||0x8848
mpls_bos	1	0x24	0	eth_type = 0x8847||0x8848
pbb_isid	24	0x25	0	eth_type = 0x88E7
	48	0x25	1	eth_type = 0x88E7
tunnel_id	64	0x26	0	none
	128	0x26	1	none
ipv6_hexthdr	9	0x27	0	eth_type = 0x86dd
	18	0x27	1	eth_type = 0x86dd

OpenFlow Match structure

OpenFlowBasic TLV Зависимости

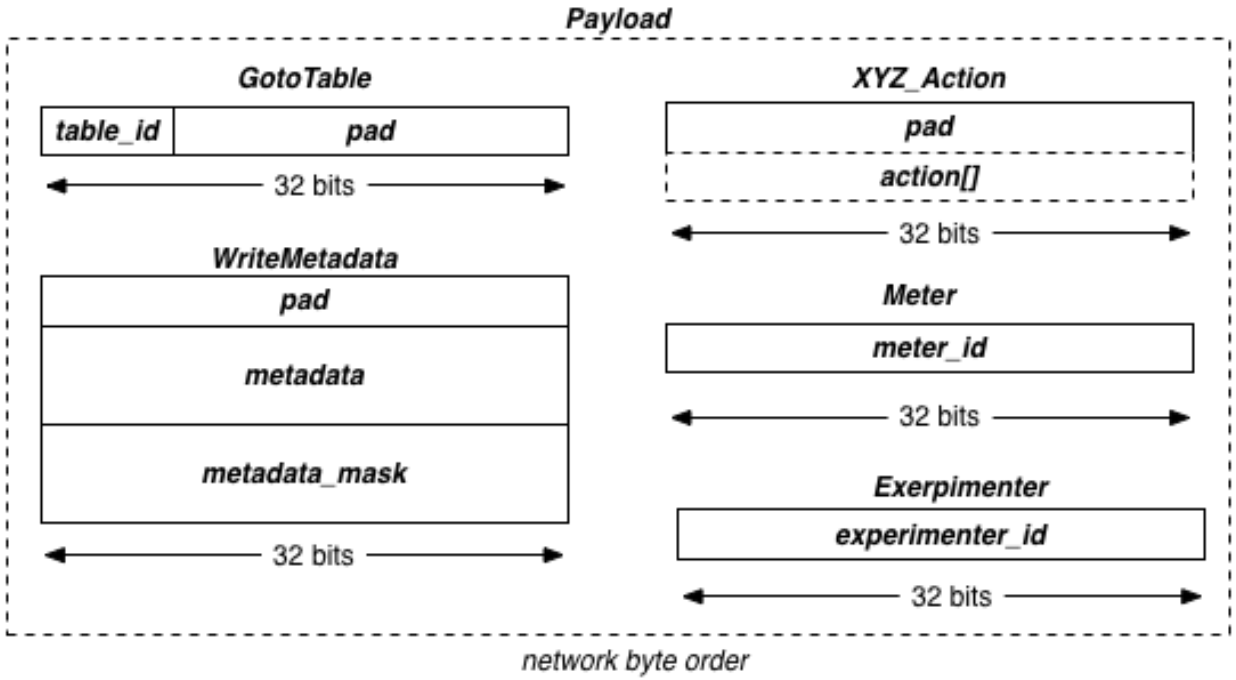


OpenFlow Instruction structure



Instruction – это структура данных, в которой передаются инструкции, необходимые для применения.

С версии OF 1.3. появился новый тип Instruction, которые называется Meter, и нужен для применения определенных правил к потоку.

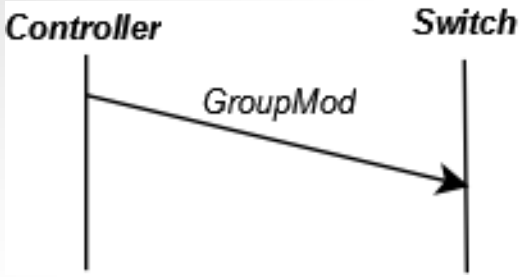


OpenFlow Instruction structure

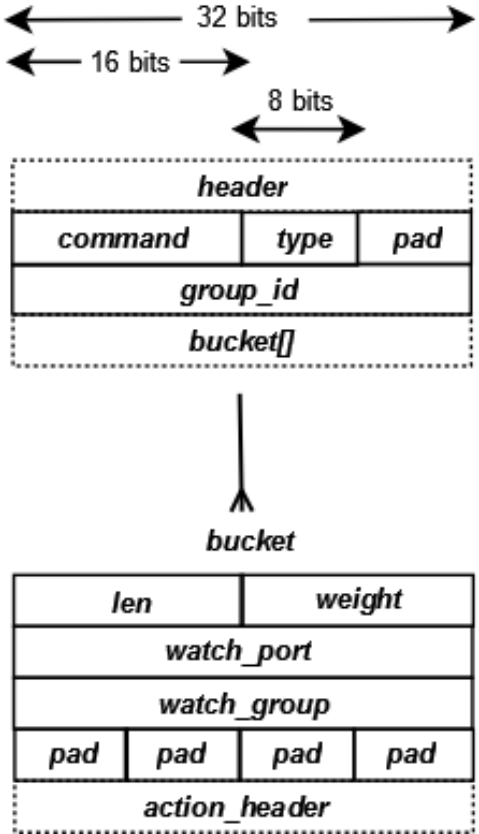
Instruction Payload

Type Name	Type Value	Field Name	Bits	Constraints
GotoTable	0x0001			
		type	16	= 0x0001
		length	16	= 0x0008
		table_id	8	none
		pad	24	none
WriteMetadata	0x0002			
		type	16	= 0x0002
		length	16	= 0x0018
		pad	32	none
		metadata	64	none
		metadata_mask	64	none
WriteActions	0x0003			
		type	16	= 0x0003
		length	16	≥ 8
		pad	32	none
		action	32	action restrictions, only one action of each type
ApplyActions	0x0004			
		type	16	= 0x0004
		length	16	≥ 8
		pad	32	none
		action	32	action restrictions
ClearActions	0x0005			
		type	16	= 0x0005
		length	16	≥ 8
		pad	32	none
		action	32	action restrictions
Meter	0x0006			
		type	16	= 0x0006
		length	16	= 0x0008
		meter_id	32	none
Experimenter	0xFFFF			
		type	16	= 0xFFFF
		length	16	≥ 0x0008
		experimenter_id	32	none

OpenFlow сообщение GroupMod



Инициатор: Контроллер
Ответное сообщение: не требуется



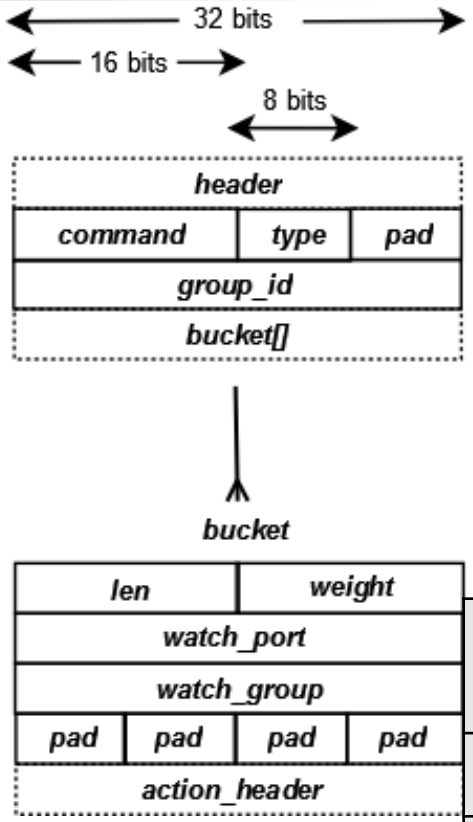
Это одно из основных сообщений, необходимое для изменения таблиц Group Table.

Command – действие, которое необходимо произвести с таблицей;
Type – тип Group Table, к которой направлено сообщение;
Pad –заполнение;
group_id – идентификатор Group Table

Bucket - подмножество с набором действий (action set) для этой Group Table

Len – длина поля bucket в байтах
Weight- относительный вес подмножества (bucket). Определеяется только для групп типа select.
watch_port - порт, действия для которых связаны с временем жизни этого bucket. Необходим только для групп типа fast failover.
watch_group - Group table, действия для которых связаны с временем жизни этого bucket. Необходим только для групп типа fast failover.
pad[4] - заполнение;
action_header – Длина поля action определяется из значения поля length в заголовке

OpenFlow сообщение GroupMod



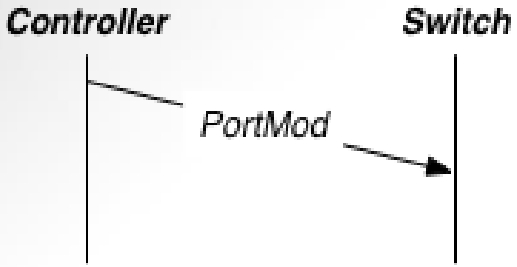
Name	Bits	Constraints
command	16	see table
type	8	see table
pad	8	none
group_id	32	none
bucket[]	-	bucket structure

Field	Name	Code
Command	Add	0x0000
	Modify	0x0001
	Delete	0x0002
Type	All	0x00
	Select	0x01
	Indirect	0x02
	FastFailover	0x03
GroupID	Max	0xFFFFFF00
	All	0xFFFFFFFC
	Any	0xFFFFFFFF

Структура Bucket

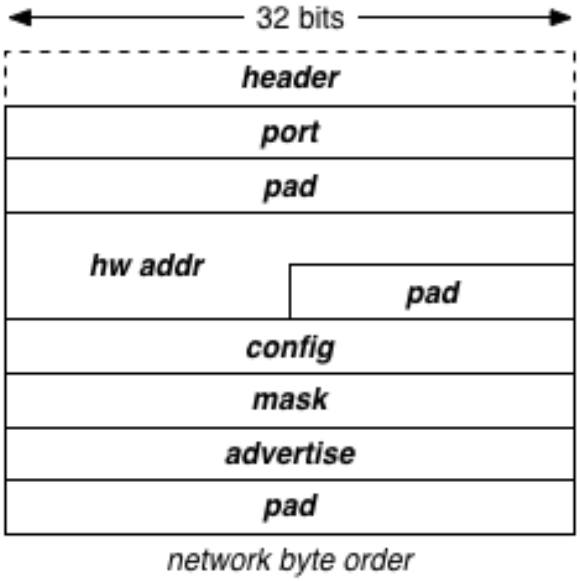
Name	Bits	Constraints
len	16	≥ 16
weight	16	= 1
watch_port	32	none
watch_group	32	none
pad	32	none
action_header	-	action

OpenFlow сообщение PortMod



Это сообщение определяет последовательность в которой контроллер может модифицировать состояние OpenFlow порта..

Инициатор: Контроллер
Ответное сообщение: не требуется



Header – стандартный заголовок длиной 4 байта;

Port – целевой порт для внесения изменений;

Pad –заполнение;

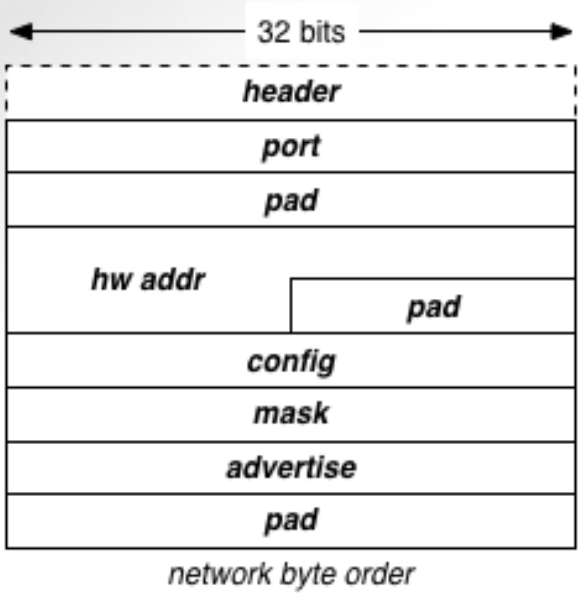
Hw_addr – аппаратный адрес, который снова используется для проверки правильности целевого порта. Аппаратный адрес является неизменным.

Config - набор значений, которые изменяются в аналогичном поле структуры Port.

Mask - поле используется для выбора битов в поле config, которые необходимо изменить.

Advertise – указывает, какие новые функции должен поддерживать порт. Поле не имеет маски, все функции порта изменяются в полном объеме.

OpenFlow сообщение PortMod

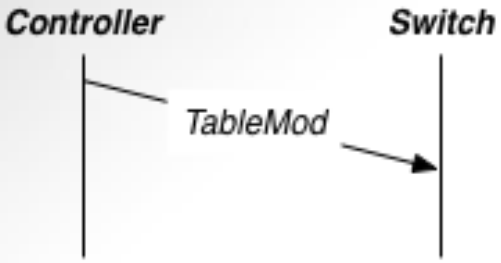


Name	Bits	Constraints
port	16	none
hw_addr	48	none
config	32	See below
mask	32	0..0x0000007F
advertise	32	0..0x00000FFF
padding	32	none

Field	Name	Value
Config	PortDown	0x00000001
	NoSTP	0x00000002
	NoRecv	0x00000004
	NoRecvSTP	0x00000008
	NoFlood	0x00000010
	NoFwd	0x00000020
	NoPacketIn	0x00000040

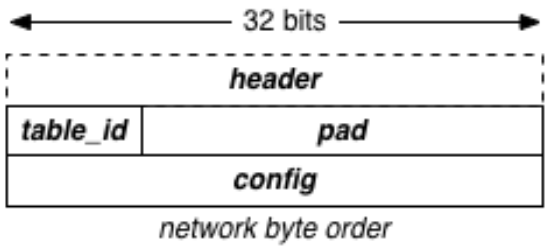
Field	Name	Value
Features (advertise)	10MB_HD	0x00000001
	10MD_FD	0x00000002
	100MB_HD	0x00000004
	100MB_FD	0x00000008
	1GB_HD	0x00000010
	1GB_FD	0x00000020
	10GB_FD	0x00000040
	Copper	0x00000080
	Fiber	0x00000100
	AutoNeg	0x00000200
	Pause	0x00000400
	PauseAsym	0x00000800

OpenFlow сообщение TableMod



Инициатор: Контроллер
Ответное сообщение: не требуется

Это сообщение позволяет определить судьбу пакета, когда правила для него отсутствуют в таблице. Он может быть направлен на контролер, отброшен или отправляются к следующей таблице. С версии 1.3 уже не поддерживается и сообщение оставлено для совместимости с более ранними версиями протокола.

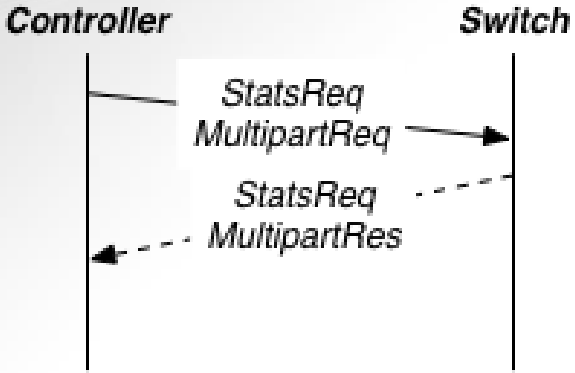


table_id – идентификатор таблицы, OFPTT_ALL = 0xff, обращение ко всем таблицам;
Pad - Заполнение до 32 бит;
Config - поле определяющее действия с пакетом (для версии 1.2 и раньше).

Name	Bits	Constraints
table_id	8	in 0..0xfe
pad	24	none
config	32	see table

Field	Name	Code
config	MissController	0x00000000
	MissContinue	0x00000001
	MissDrop	0x00000002
	MissMask	0x00000003

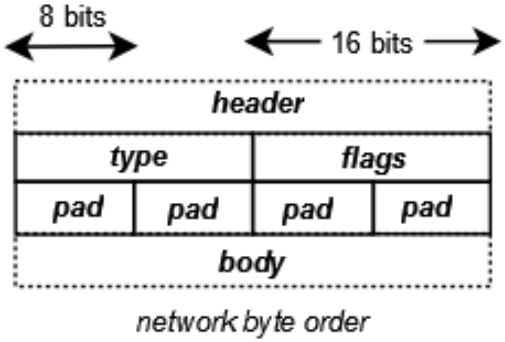
OpenFlow сообщение Multipart request

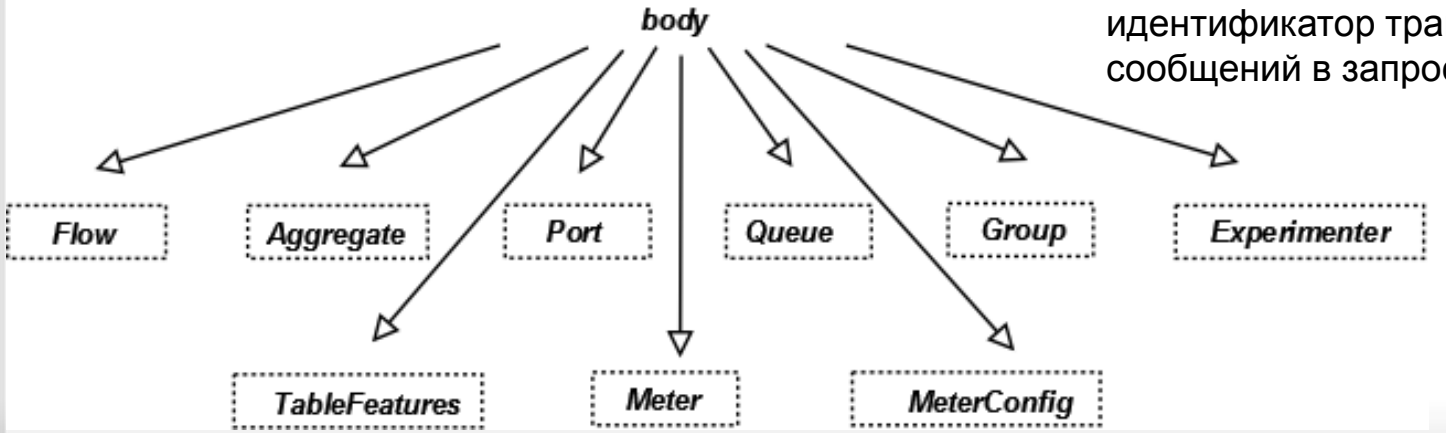


Инициатор: Контроллер
Ответное сообщение: требуется

Multipart Request используется для запроса информации об отдельных потоках и таблицах. (В версиях до 1.3.0 сообщение называлось Stats Request).

Type – тип содержимого сообщения;
Flags – флаг отражает номер сообщения (если их несколько);
Pad - заполнение до 32 бит;
Body - тело сообщения.





Следующее сообщение в потоке всегда должно следовать с флагом больше чем предыдущее. Запрос, который охватывает несколько сообщений (одно или несколько сообщений), должны использовать единый идентификатор транзакции (xid) для всех сообщений в запросе.

Типы передаваемых блоков в теле сообщения Multipart request

В запросе и ответе, поле " Type" указывает Тип информации, которая будет передана в теле сообщения.

«Description»
Type = 0x0000

Запрос описания модели конкретного OpenFlow коммутатора

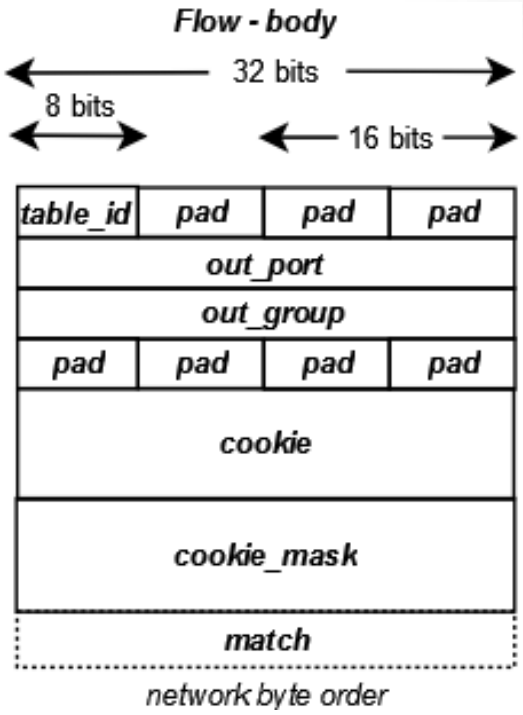
Передается:
В запросе информация не передается (полу Body не заполнено). Ответное сообщение содержит информацию, передаваемую в виде текстовой строки (см. слайд Multipart response).

Type Name	Type Value	Field Name	Bits	Constraints
Description	0x0000	type	16	= 0x0000
		flags	16	none
		pad	32	none

Типы передаваемых блоков в теле сообщения Multipart request

«Flow»

Type = 0x0001



Передается:

В запросе спрашивается информация об отдельных записях потоков (flow entries).

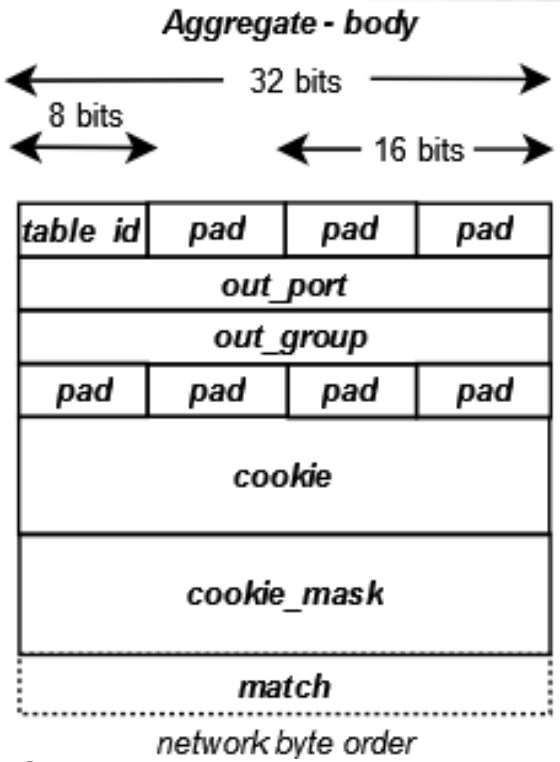
Ответное сообщение содержит данные о запрашиваемой записи (flow entries).

- **table_id** - Идентификатор таблицы потоков, информация о которой требуется (OFPTT_ALL = 0xff для запроса информации по всем таблицам) ;
- **pad** – заполнение до 32 бит;
- **out_port** – Запрос соответствующих правил (match) для данных значений выходных портов;
- **out_group** - Запрос соответствующих правил (match) для данных значений групповых таблиц (group table);
- **pad** – заполнение до 64 бит;
- **cookie** - Запрос соответствующих правил (match), содержащих это значение cookie;
- **cookie_mask** – Маска ограничивает число бит в значении cookie;
- **match** – поле для указания правил (match).

Типы передаваемых блоков в теле сообщения Multipart request

«Aggregate»

Type = 0x0002



Структура и поля соответствуют «Flow»

Передается:

В запросе спрашивается совокупная информация о нескольких записях потоков (flow entries).

Ответное сообщение содержит данные о запрашиваемых записях (flow entries).

- **table_id** - Идентификатор таблицы потоков, информация о которой требуется (OFPTT_ALL = 0xff для запроса информации по всем таблицам) ;
- **pad** – заполнение до 32 бит;
- **out_port** – Запрос соответствующих правил (match) для данных значений выходных портов;
- **out_group** - Запрос соответствующих правил (match) для данных значений групповых таблиц (group table);
- **pad** – заполнение до 64 бит;
- **cookie** - Запрос соответствующих правил (match), содержащих это значение cookie;
- **cookie_mask** – Маска ограничивает число бит в значении cookie;
- **match** – поле для указания правил (match).

Типы передаваемых блоков в теле сообщения Multipart request

«Table»
Type = 0x0003

Запрос информации о таблицах.

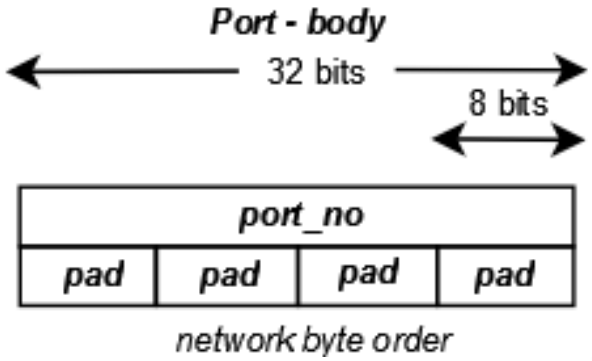
Передается:
В запросе информация не передается (полу Body не заполнено). Ответное сообщение содержит информацию, передаваемую в виде массива данных (см. слайд Multipart response «Table»).

Type Name	Type Value	Field Name	Bits	Constraints
Table	0x0003	type	16	= 0x0003
		flags	16	none
		pad	32	none

Типы передаваемых блоков в теле сообщения Multipart request

«Port»

Type = 0x0004



Передается:
В запросе спрашивается совокупная информация о портах.
Ответное сообщение содержит данные о запрашиваемых портах.

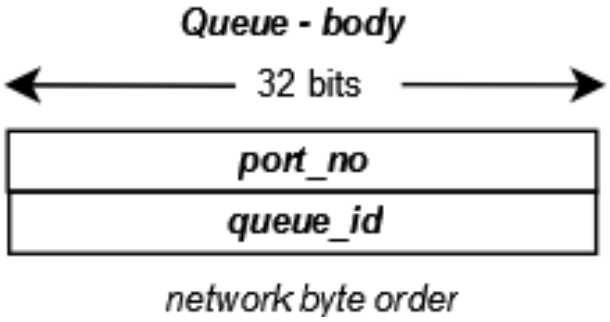
- **port_no** – запросить статистику либо на один порт (указанный в port_no) или для всех портов OFPP_ANY = 0xffffffff;

- **pad** – заполнение до 32 бит.

Типы передаваемых блоков в теле сообщения Multipart request

«Queue»

Type = 0x0005



Передается:

В запросе спрашивается статистическая информация об очереди на одном или нескольких портах, а также для одной или нескольких очередей.
Ответное сообщение содержит данные о запрашиваемых очередях.

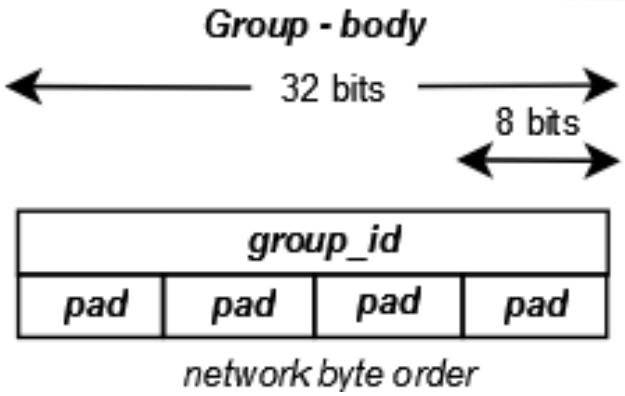
- **port_no** – номер порта OpenFlow коммутатора, для которого запрашивается статистика по очередям или для всех портов OFPP_ANY = 0xffffffff;

- **queue_id** - определяет одну из приоритетных очередей, или OFPQ_ALL для обозначения всех очередей, настроенные на указанный порт, для которого запрашивается статистика OFPQ_ANY = 0xffffffff.

Типы передаваемых блоков в теле сообщения Multipart request

«Group»

Type = 0x0006



Передается:
В запросе спрашивается статистическая информация о группах (group tables) в OpenFlow коммутаторе.
Ответное сообщение содержит данные о запрашиваемых группах.

- **group_id** – идентификатор группы OpenFlow коммутатора, для которой запрашивается статистика или для всех групп OFPG_ALL = 0xfffffffc;

- **pad** – заполнение до 64 бит.

Типы передаваемых блоков в теле сообщения Multipart request

«Group Description»

Type = 0x0007

Передается:
В запросе информация не передается (полу Body не заполнено).

Запрос запрашивает информацию о списке групповых таблиц (group tables) в OpenFlow коммутаторе вместе с набором действий (bucket actions) для каждой из них.
Ответное сообщение содержит данные о запрашиваемых группах.

Type Name	Type Value	Field Name	Bits	Constraints
GroupDesc	0x0007	type	16	= 0x0007
		flags	16	none
		pad	32	none

Типы передаваемых блоков в теле сообщения Multipart request

«Group Features»

Type = 0x0007

Передается:
В запросе информация не передается (полу Body не заполнено).

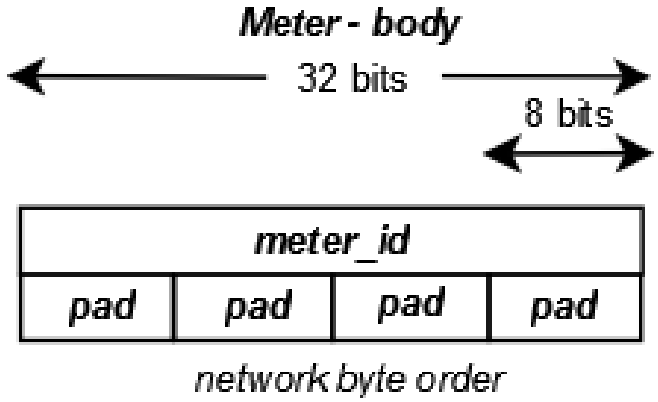
Запрос запрашивает информацию о параметрах списка групповых таблиц (group tables) в OpenFlow коммутаторе.
Ответное сообщение содержит данные о запрашиваемых группах.

Type Name	Type Value	Field Name	Bits	Constraints
GroupFeatures	0x0008	type	16	= 0x0008
		flags	16	none
		pad	32	none

Типы передаваемых блоков в теле сообщения Multipart request

«Meter»

Type = 0x0009



Передается:
В запросе спрашивается статистическая информация о таблицах метрик (meter tables) в OpenFlow коммутаторе.
Ответное сообщение содержит данные о запрашиваемых meter tables.

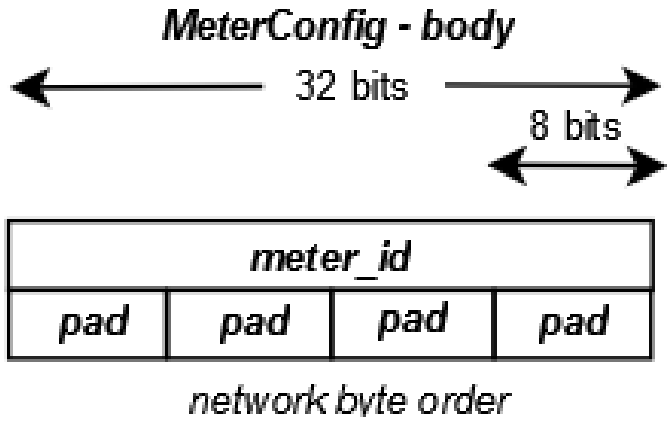
- **meter_id** – идентификатор таблицы метрик (meter tables) OpenFlow коммутатора, для которой запрашивается статистика или для всех таблиц OFPM_ALL = 0xffffffff;

- **pad** – заполнение до 64 бит.

Типы передаваемых блоков в теле сообщения Multipart request

«Meter Config»

Type = 0x000a



Передается:
В запросе спрашивается информация конфигурации таблиц метрик (meter tables) в OpenFlow коммутаторе.
Ответное сообщение содержит данные о запрашиваемых meter tables.

- **meter_id** – идентификатор таблицы метрик (meter tables) OpenFlow коммутатора, для которой запрашивается статистика или для всех таблиц OFPM_ALL = 0xffffffff;

- **pad** – заполнение до 64 бит.

Типы передаваемых блоков в теле сообщения Multipart request

«Meter Features»

Type = 0x000b

Передается:
В запросе информация не передается (полу Body не заполнено).

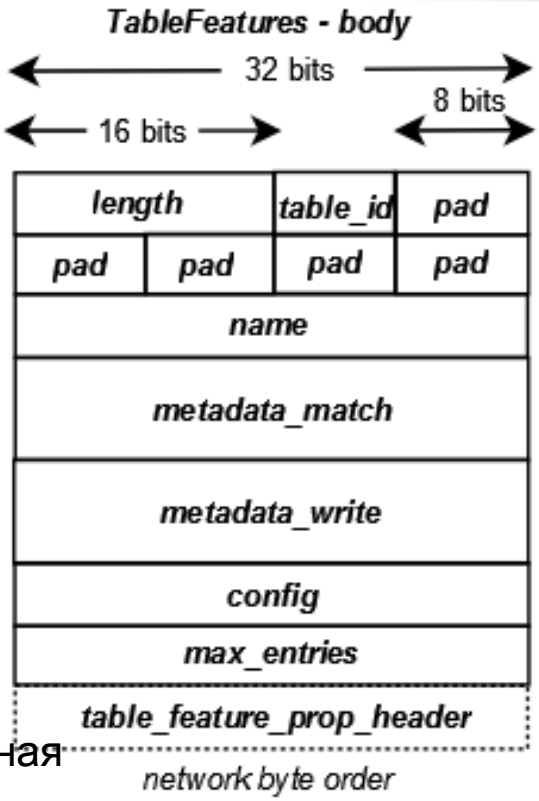
Запрос запрашивает информацию о параметрах и возможностях подсистемы метрик (metering subsystem) в OpenFlow коммутаторе. Ответное сообщение содержит данные о запрашиваемых таблицах.

Type Name	Type Value	Field Name	Bits	Constraints
MeterFeatures	0x000b	type	16	= 0x000b
		flags	16	none
		pad	32	none

Типы передаваемых блоков в теле сообщения Multipart request

«TableFeatures»

Type = 0x000c

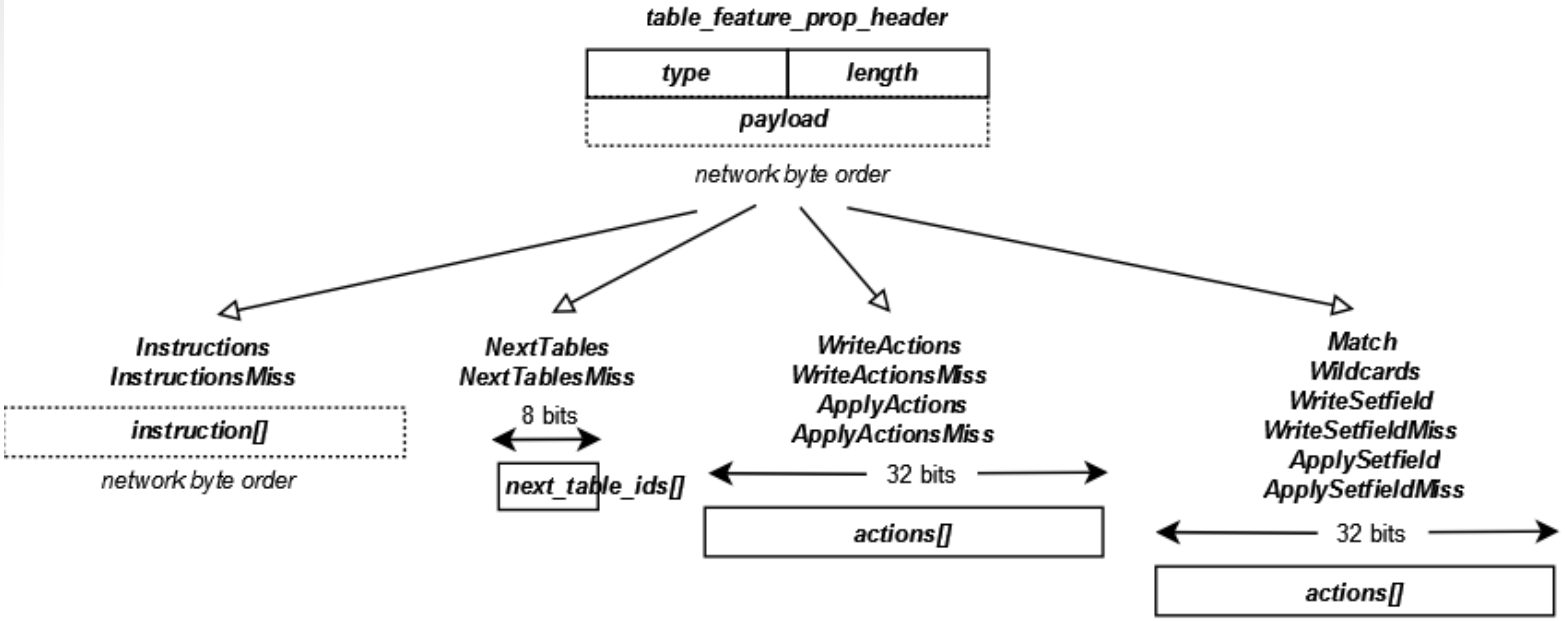


Передается:

Запрос позволяет контроллеру запросить текущие параметры существующих таблиц, а также дополнительнс указать коммутатору на переконфигурацию его таблиц. Ответное сообщение содержит данные о таблицах.

- **Length** – длина данного блока;
- **table_id** – идентификатор таблицы. Таблицы с большими значениями обрабатываются первыми;
- **pad** – заполнение до 64 бит;
- **name** – текстовое наименование таблицы (Нуль-терминированная строка);
- **metadata_match** - биты метаданных пакета, которые сравнивает таблица (match);
- **metadata_write** - биты метаданных пакета, которые могут быть записаны в таблицу, используя OFPIT_WRITE_METADATA инструкцию.
- **config** – установленная конфигурация таблицы;
- **max_entries** - максимальное число записей потоков (flow entries) в таблице;
- **table_feature_prop_header** – список свойств таблицы, описывающих ее возможности

Значения поля table_feature_prop_header (1)



Type Name	Type Value	Описание
Instructions	0x0000	Свойства для инструкций
InstructionMiss	0x0001	Свойства для инструкций к таблицам для потоков, которые не подпадают под правила (match).
NextTables	0x0002	Свойства следующей таблицы
NextTablesMiss	0x0003	Свойства следующей таблицы для потоков, которые не подпадают под правила (match).
WriteActions	0x0004	Запись свойств Действия к таблице (Action)
WriteActionsMiss	0x0005	Запись свойств Действия к таблице для потоков, которые не подпадают под правила (match).

Значения поля table_feature_prop_header (2)

Type Name	Type Value	Описание
ApplyActions	0x0006	Применить свойства Действий (Actions)
ApplyActionsMiss	0x0007	Применить свойства Действия к таблице для потоков, которые не подпадают под правила (match).
Match	0x0008	Свойства правила (match)
Wildcards	0x0009	Свойства правила с любым значением (wildcard)
WriteSetField	0x000a	Запись свойств установленного поля
WriteSetFieldMiss	0x000b	Запись свойств установленного поля в таблице для потоков, которые не подпадают под правила (match).
ApplySetField	0x000c	Применить свойств установленного поля
ApplySetFieldMiss	0x000d	Применить свойства установленного поля в таблице для потоков, которые не подпадают под правила (match).
OFPTFPT_EXPERIMENTER	0xFFFE	Свойства для экспериментальных (любых) значений
OFPTFPT_EXPERIMENTER_MISS	0xFFFF	Свойства для экспериментальных (любых) значений к таблице для потоков, которые не подпадают под правила (match).

Типы передаваемых блоков в теле сообщения Multipart request

«Port Description»

Type = 0x000d

Передается:

В запросе информация не передается (полу Body не заполнено).

Запрос запрашивает информацию с описание всех портов в системе, которая поддерживает OpenFlow.

Ответное сообщение содержит данные о портах.

Type Name	Type Value	Field Name	Bits	Constraints
PortDesc	0x000d	type	16	= 0x000d
		flags	16	none
		pad	32	none

Типы передаваемых блоков в теле сообщения Multipart request

В запросе и ответе, поле " Type" указывает Тип информации, которая будет передана в теле сообщения.

«Description»

Type = 0x0000

Передается:

Описание модели конкретного OpenFlow коммутатора.
В запросе информация не передается . Ответное сообщение содержит информацию, передаваемую в виде текстовой строки, следующей структуры:

- **mfr_desc** - Описание производителя.
- **hw_desc** – Описание аппаратной части (Hardware).
- **sw_desc** - Описание программной части (Software).
- **serial_num** – Серийный номер.
- **dp_desc** - Удобочитаемое описание плоскости коммутации (datapath).

Поле dp_desc является произвольной строкой для описания datapath для отладки, например, “switch3 в комнате 3120”. Это не гарантирует уникальность и эта запись не должна использоваться в качестве основного идентификатора в datapath. Для идентификации используется поле datapath_id коммутатора.

OpenFlow Protocol Messages

Симметричные сообщения

Hello

Сообщение OFPT_HELLO не имеет тела; то есть, оно состоит только из заголовка OpenFlow. Должны быть подготовлены реализации, воспринимающие сообщения hello, которые имеют тело.

Запрос эхо

Сообщение *запроса эхо* (Echo Request) состоит из заголовка OpenFlow плюс поля данных произвольной длины. Поле данных может быть временной меткой сообщения для проверки задержки, или иметь нулевую длину при проверке работоспособности канала переключатель-контроллер.

Эхо отклик

Сообщение *эхо отклик* (Echo Reply) состоит из заголовка OpenFlow и немодифицированного поля данных эхо-запроса.

Сообщение об ошибке

Бывают моменты, когда switch вынужден оповестить контроллер о проблеме или наоборот. Это делается с помощью сообщения OFPT_ERROR.

Экспериментальные сообщения

OFPT_EXPERIMENTER Тип сообщений, которые нужны для различных функций (считывания баз, поддержки расширений и т.д.)

OpenFlow Protocol Messages

Сообщения контроллер-коммутатор

При установлении сессии контроллер посылает сообщение OFPT_FEATURES_REQUEST. Это сообщение не содержит тела после заголовка OpenFlow. Коммутатор откликается сообщением OFPT_FEATURES_REPLY.

Конфигурирование коммутатора

Контроллер способен установить конфигурацию и параметры очереди в коммутаторе с помощью сообщений OFPT_SET_CONFIG и OFPT_GET_CONFIG_REQUEST, соответственно.

Переключатель реагирует на запрос конфигурации сообщением OFPT_GET_CONFIG_REPLY; на запрос конфигурации отклика не посылается.

Асинхронные сообщения

Сообщение packet-In

Когда пакет получен матицей коммутации (datapath) и послан контроллеру, используется сообщение OFPT_PACKET_IN.

OpenFlow Protocol Messages

Сообщение OFPT_FLOW_REMOVED

Если контроллер запросил получение уведомлений, когда поток записей завершен по тайм-ауту или удаляются из таблиц, то DataPath делает это с сообщением OFPT_FLOW_REMOVED.

Сообщение OFPT_PORT_STATUS

Если конфигурационные биты порта изменены через другой интерфейс коммутатора, то он посылает сообщение OFPT_PORT_STATUS, чтобы уведомить контроллер об этом изменении

Сообщение вывода

Когда контроллер хочет послать пакет к определенному порту data path, он использует сообщение OFPT_PACKET_OUT.

Сообщения изменения состояния

Модификации таблицы переадресации со стороны контроллера осуществляется с помощью сообщений OFPT_FLOW_MOD

OpenFlow Protocol Messages

Сообщения изменения состояния

Изменения в таблице групп из контроллера выполняются с сообщением OFPT_GROUP_MOD

Сообщение модификации порта

Контроллер для модификации поведения порта использует сообщение OFPT_PORT_MOD.

Сообщение модификации таблиц

Контроллер для модификации таблиц использует сообщение OFPT_TABLE_MOD

Составные сообщения

Во время работы системы, контроллер может запросить состояние Data Path с помощью OFPT_MULTIPART_REQUEST

Коммутатор отвечает одним или несколькими OFPT_MULTIPART_REPLY

Сообщения Barrier.

Когда контроллер хочет обеспечить выполнение всех зависимостей для сообщения или хочет получать уведомления о завершенных операциях, он может использовать сообщение OFPT_BARRIER_REQUEST. Это сообщение не имеет тела.

При получении, коммутатор должен завершить обработку всех ранее принятых сообщений, в том числе отправка соответствующего ответа или сообщения об ошибках, перед выполнением действий с сообщением на запрос Barrier. Когда такая обработка завершена, коммутатор должен послать сообщение OFPT_BARRIER_REPLY с xID первоначального запроса.

OpenFlow Protocol Messages

Сообщения конфигурации очередей

OFPT_QUEUE_GET_CONFIG_REQUEST и
FPT_QUEUE_GET_CONFIG_REPLY

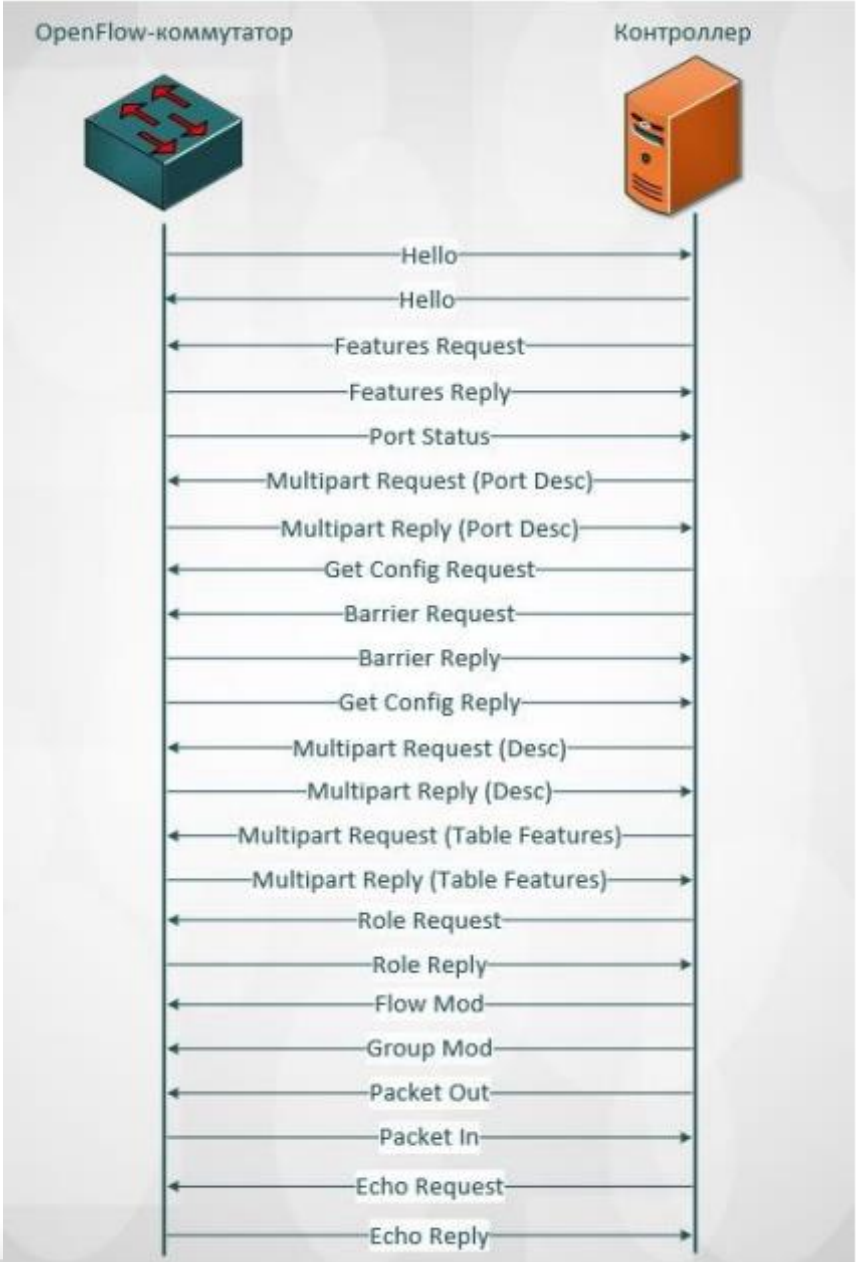
Set Asynchronous Configuration Message

Контроллер может устанавливать и запрашивать асинхронные сообщения, которые он хочет получить (кроме сообщений об ошибках) на данном канале OpenFlow при помощи OFPT_SET_ASYNC и OFPT_GET_ASYNC_REQUEST, соответственно.
Коммутатор отвечает на сообщение OFPT_GET_ASYNC_REQUEST сообщением OFPT_GET_ASYNC_REPLY; он не отвечает на запрос, по установке конфигурации.

Сообщения конфигурации счетчика

Изменения в счетчике от контроллера выполняются при помощи сообщения OFPT_METER_MOD

Пример подключения коммутатора

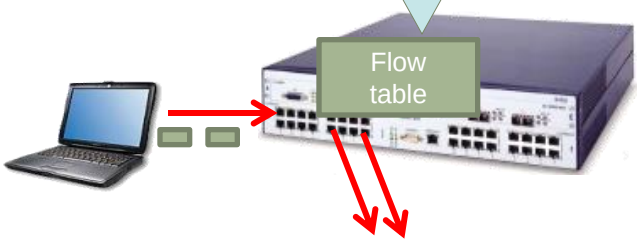


OpenFlow: Flow Table

- Flow Table

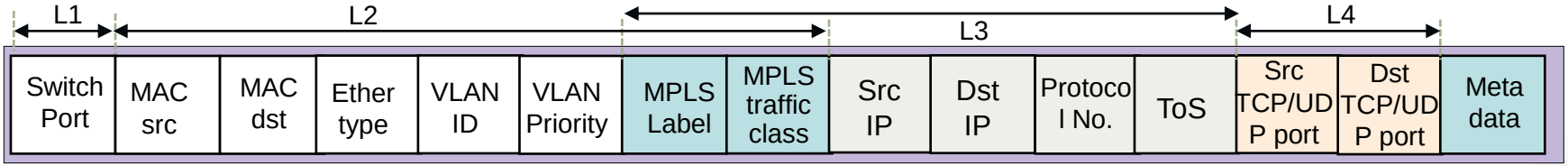
Счетчики используются, когда контроллер вычисляет пути

Flow entry	match field	counter	Action (Instruction)	priority	Timeout	cookie
1						
n	…	…	…			



Actions(Instructions)
1. **Forward packet to port(s)**
2. **Encapsulate and forward to controller**
3. **Drop packet**
4. **Send to normal processing pipeline**
5. **Modify Fields**
6. **Etc.**

- Match field= L1~L4 header information
  - OpenFlow 1.0 → 12 tuples
  - OpenFlow 1.1 → 15 tuples
  - OpenFlow 1.3 → 40 tuples (158 bytes)



Match fields of OpenFlow 1.1

OpenFlow: Flow Table

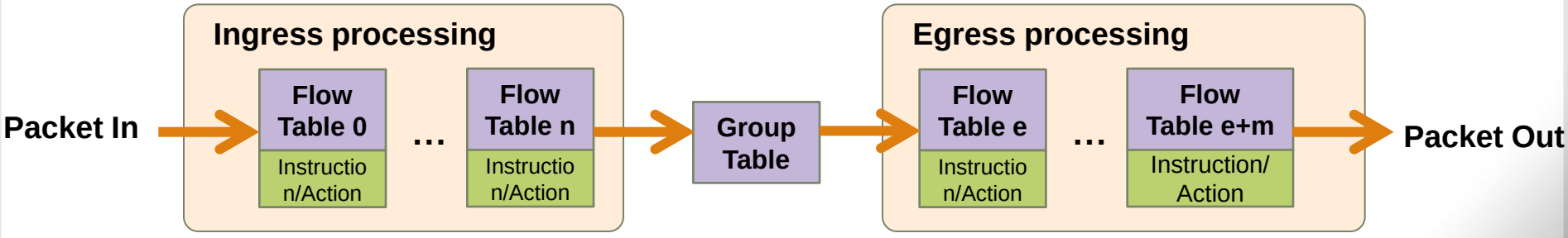
- Flow Table

  (*) – значение в поле не имеет значения

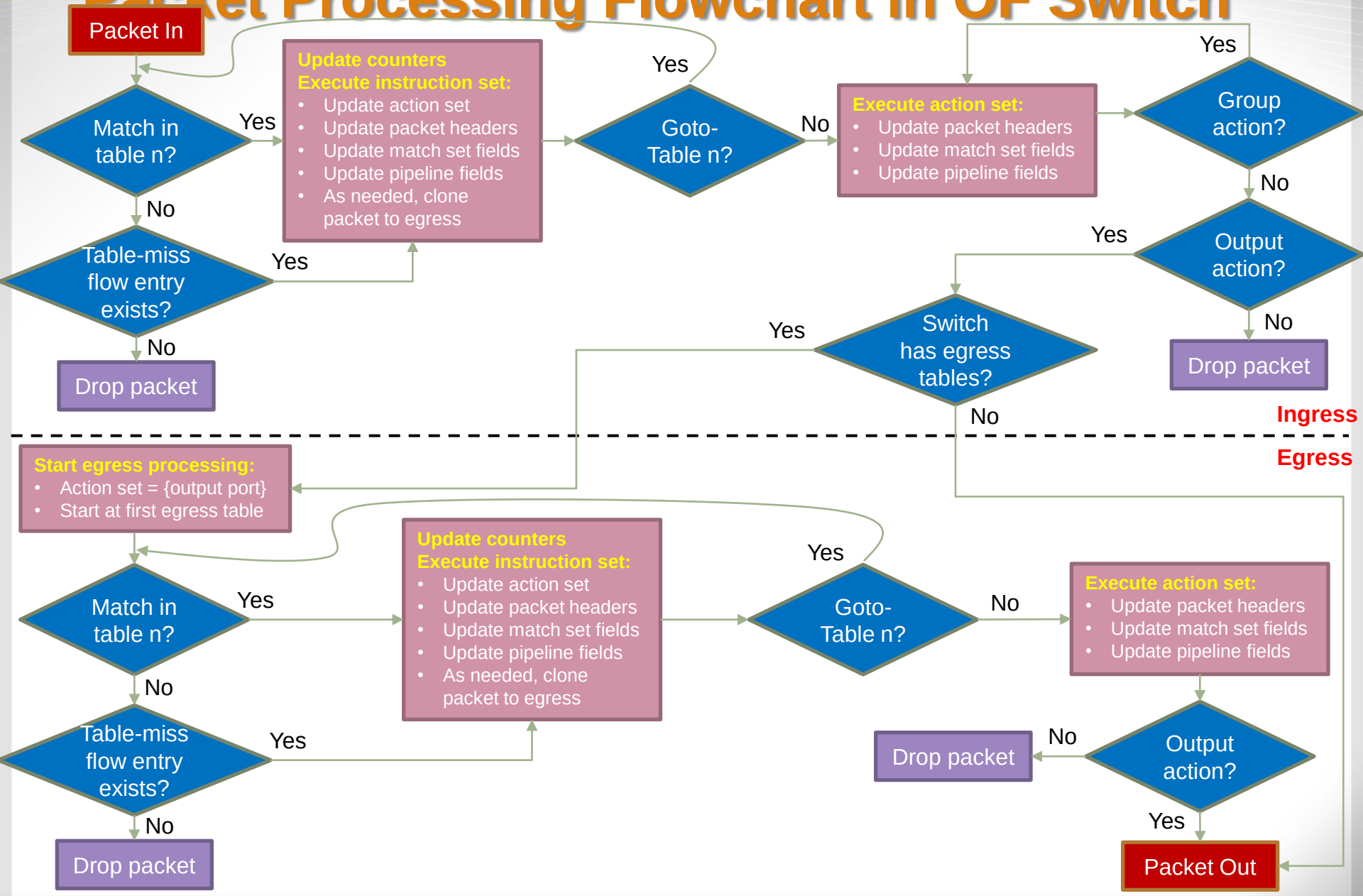
Operation Mode	Switch Port	MAC src	MAC dst	Ether type	VLAN ID	Src IP	Dst IP	Proto No.	TCP S_port	TCP D_port	Action	Counter
Switching	*	*	00:1f..	*	*	*	*	*	*	*	Port1	243
Flow Switching	Port3	00:20..	00:2f..	0800	vlan1	1.2.3.4	1.2.3.9	4	4666	80	Port7	123
Routing	*	*	*	*	*	*	1.2.3.4	*	*	*	Port6	452
VLAN Switching	*	*	00:3f..	*	vlan2	*	*	*	*	*	Port6 Port7 Port8	2341
Firewall	*	*	*	*	*	*	*	*	*	22	Drop	544
Default Route	*	*	*	*	*	*	*	*	*	*	Port1	1364

OpenFlow Pipelining

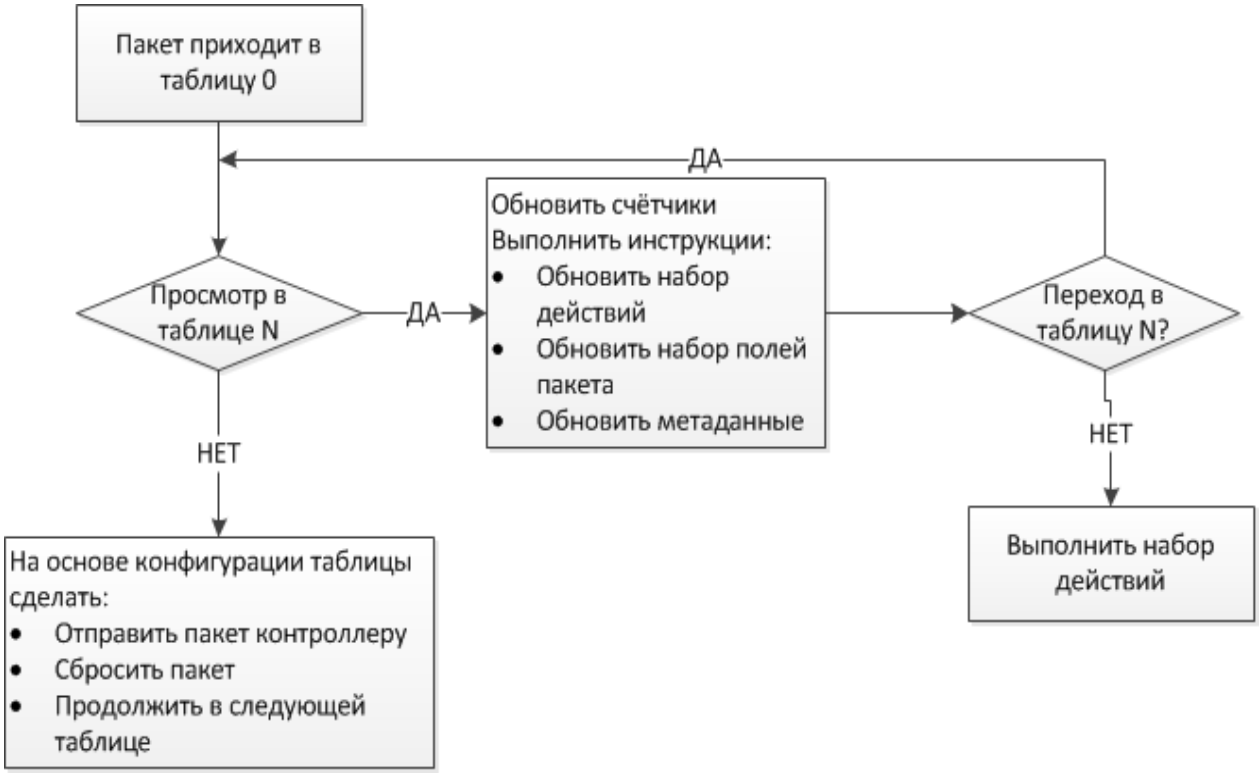
- Конвеиризация
  - Таблицы потоков коммутатора последовательно нумеруются, начиная с 0.
    - Пакет обрабатывается последовательно в нескольких таблицах потоков (версия 1.1)
    - Если запись потока найдена, набор инструкций, включенных в эту запись потока выполняется
    - Инструкции могут явным образом направить пакет на другую таблицу потока ("Goto-table").
    - обработка конвейера может идти только вперед, а не назад.
  - Двух уровневый конвейер (version 1.5)
    - Ingress обработка
      - Обязательный, выполняется до обработки исходящего трафика, использовать правила, указанные в Ingress таблицах
    - Egress обработка
      - Необязательно, выполняется в контексте выходного порта, используются правила, указанные в Egress таблицах
      - Egress таблица может быть настроена на особенности фазы запросов / ответа
  - Полезно для обработки сложных сессий
    - Например, таблица 1 для обработки VLAN, таблица 2 для обработки групповой рассылки.



Packet Processing Flowchart in OF Switch



Обработка пакетов в OpenFlow-коммутаторе



Инструкции в OpenFlow

- Инструкции
  - Инструкции выполняются, когда пакет соответствует записи в таблице
  - Инструкции приводят к изменениям в пакете, набору действий и / или дальнейшей обработке в конвейере

Syntax	Описание
Meter *meter_id*	Направляет пакет к специальной метрике
Apply-Actions *actions*	Применить конкретные действия немедленно. Выполнить несколько действий одного и того же типа.
Clear-Actions	Очистить все действия в установках немедленно
Write-Actions *actions*	Вставить указанные действия в текущий набор действий, если надо заменить действие, в противном случае добавить.
Goto-Table *next-table-id*	Укажите следующую таблицу в конвейере обработки. ID таблицы должен быть больше текущего ID-таблицы.

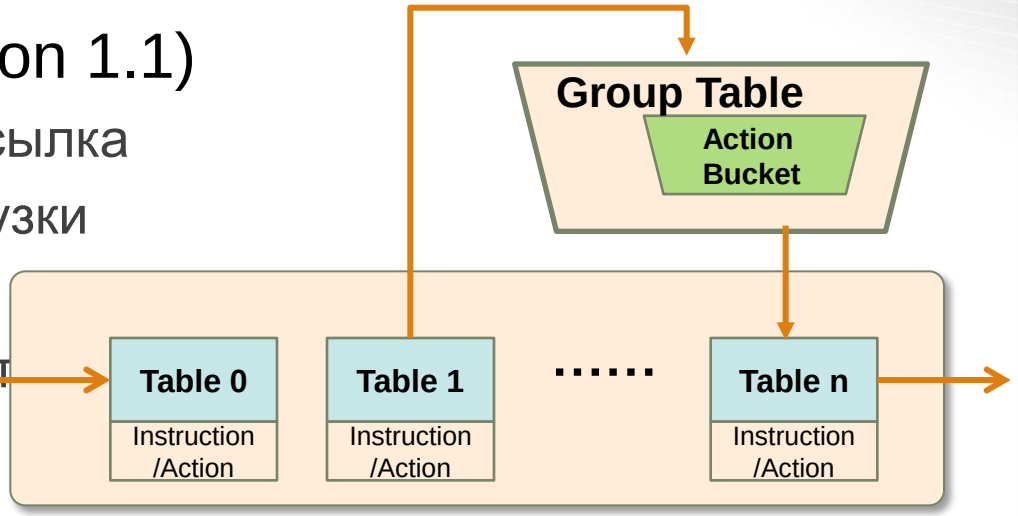
Действия в OpenFlow

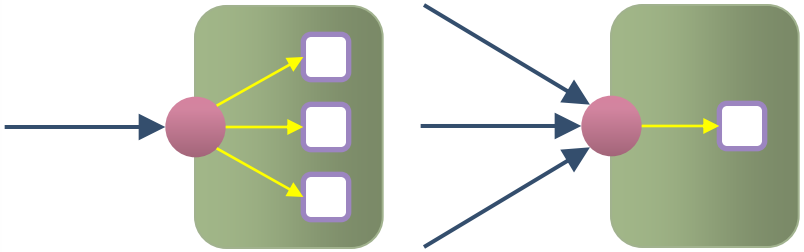
- Действия
  - Действие связано с каждым пакетом
  - Когда набор инструкций не содержит команду Goto-Table, обработка в конвейере прекращается, и действия выполняются.

Syntax	Описание
set	Применить все установочные (set) действия к пакету
qos	Применить все QoS действия, такие как set_queue к пакету
group	Если указана группа действий, применять действия соответствующей группе в порядке, определенном этим списком
output	Если не указано группы действий, пересылать пакет на порт, указанный выходным действием
push_MPLS	Применение действие к MPLS тегу в пакете
push_VLAN	Применение действие к VLAN тегу в пакете
pop	Применить все pop tag действия к пакету.

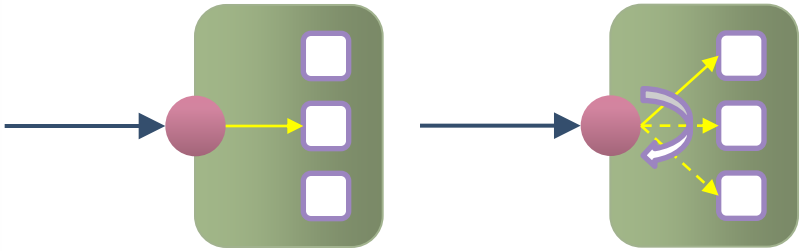
OpenFlow Group Table

- Group Table & Types (version 1.1)
  - All: широковещательная расылка
  - Select: распределение нагрузки
  - Indirect: простая адресация
  - Fast-failover: выбор маршрут

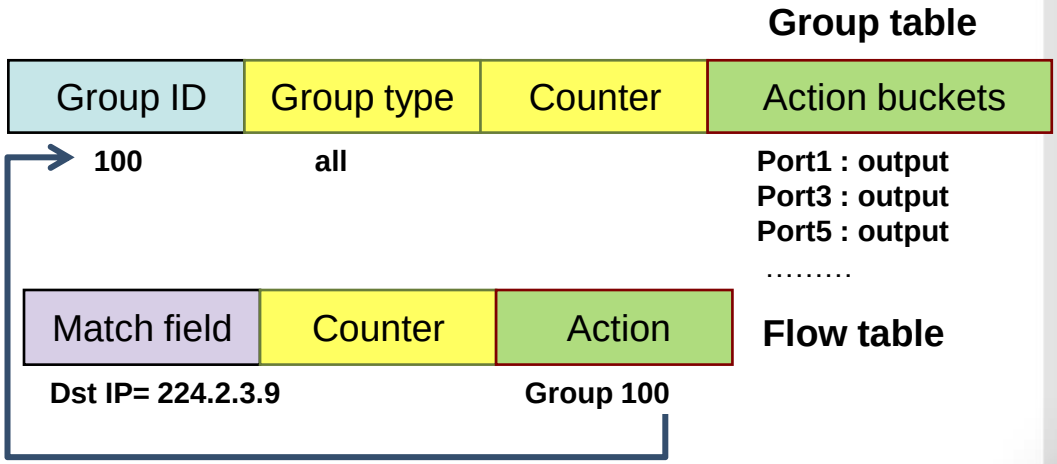




Multicast Load sharing



Indirection Rerouting



- The following group types are defined:
- • all: Execute all buckets in the group. This group is used for multicast or broadcast forwarding. The packet is effectively cloned for each bucket; one packet is processed for each bucket of the group. If a bucket directs a packet explicitly out the ingress port, this packet clone is dropped. If the controller writer wants to forward out the ingress port, the group should include an extra bucket which includes an output action to the OFPP_IN_PORT virtual port.
- • select: Execute one bucket in the group. Packets are sent to a single bucket in the group, based on a switch-computed selection algorithm (e.g. hash on some user-configured tuple or simple round robin). All configuration and state for the selection algorithm is external to OpenFlow. When a port specified in a bucket in a select group goes down, the switch may restrict bucket selection to the remaining set (those with forwarding actions to live ports) instead of dropping packets destined to that port. This behavior may reduce the disruption of a downed link or switch.
- • indirect: Execute the one defined bucket in this group. Allows multiple flows or groups to point to a common group identifier, supporting faster, more efficient convergence (e.g. next hops for IP forwarding). This group type is effectively identical to an all group with one bucket.
- • fast failover: Execute the first live bucket. Each action bucket is associated with a specific port and/or group that controls its liveness. Enables the switch to change forwarding without requiring a round trip to the controller. If no buckets are live, packets are dropped. This group type must implement a liveness mechanism(see 5.8).

OpenFlow Group Table

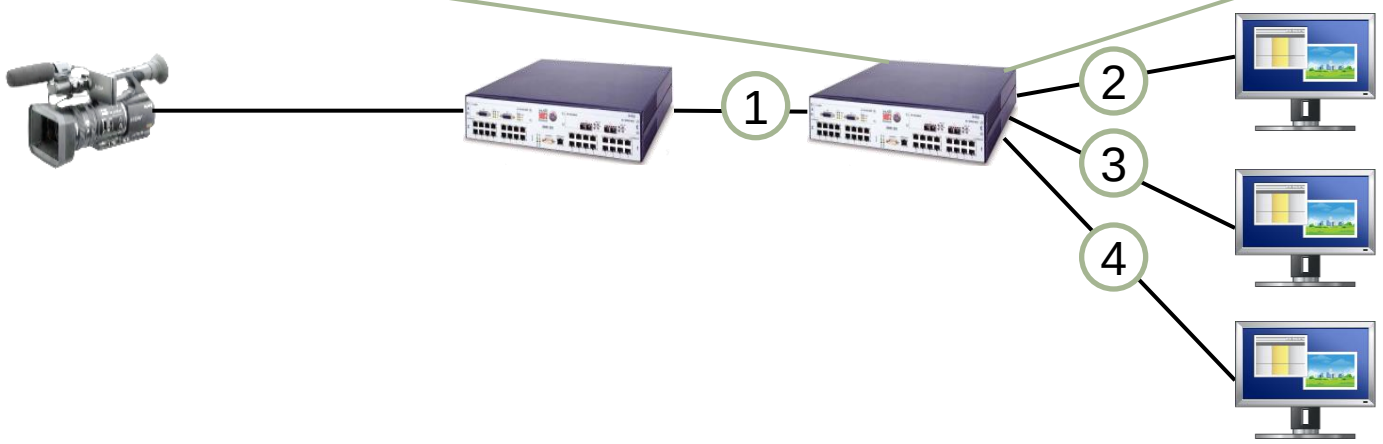
- Multicast
  - Type=all

Group Table

Group ID	Group Type	Counter	Action Buckets
100	All	999	Port2, Port3, Port4

Flow Table

Switch Port	MAC src	MAC dst	Ether Type	VLAN ID	Src IP	Dst IP	Proto No.	TCP S Port	TCP D Port	Action
*	*	00:FF:..	*	*	*	*	*	*	*	Port 1
Port 1	*	*	0800	*	224…	224…	4	4566	6633	Group 100



OpenFlow Group Table

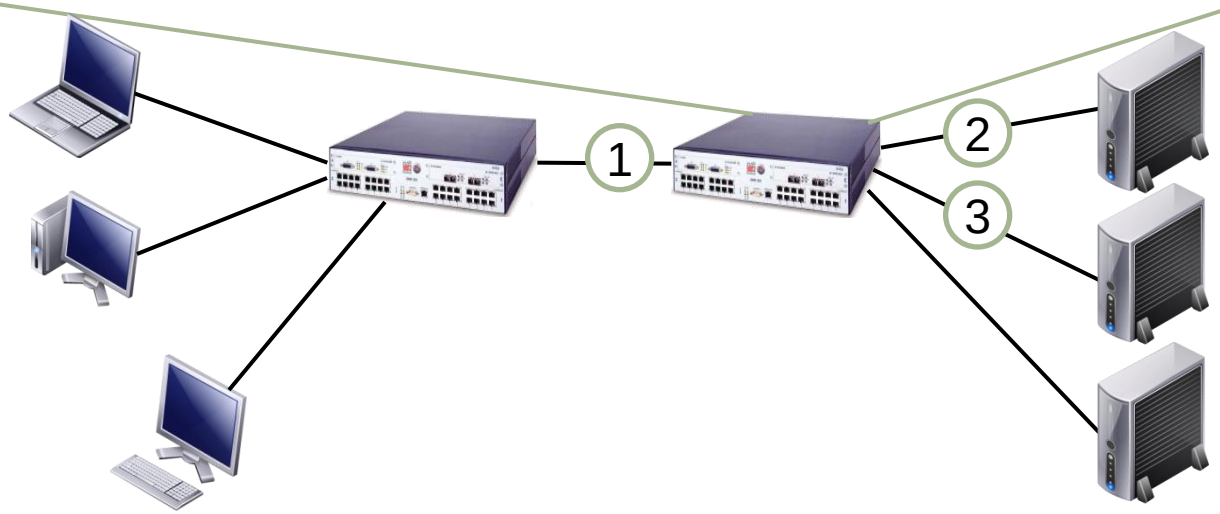
- Load Balancing
  - Type=select

Group Table

Group ID	Group Type	Counter	Action Buckets
100	Select	999	Port2, Port3

Flow Table

Switch Port	MAC src	MAC dst	Ether Type	VLAN ID	Src IP	Dst IP	Proto No.	TCP S Port	TCP D Port	Action
*	*	00:FF:..	*	*	*	*	*	*	*	Port 1
Port 1	*	*	0800	*	1.2.3 …	*	4	*	80	Group 100



OpenFlow Group Table

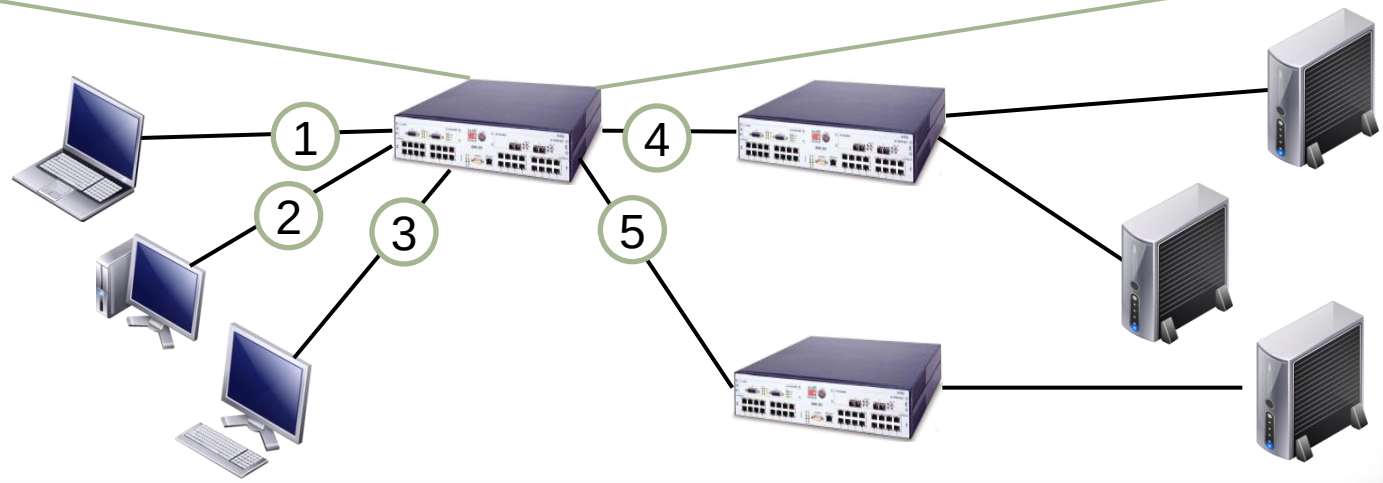
- Indirection
  - Type=indirect

Group Table

Group ID	Group Type	Counter	Action Buckets
100	Indirect	777	Port 5

Flow Table

Switch Port	MAC src	MAC dst	Ether Type	VLAN ID	Src IP	Dst IP	Proto No.	TCP S Port	TCP D Port	Action
*	00:FF …	*	0800	*	1.2.2 …	11.1…	*	*	*	Group 100
*	00:FF...	*	0800	*	1.2.3 …	11.1…	*	*	*	Group 100



OpenFlow Group Table

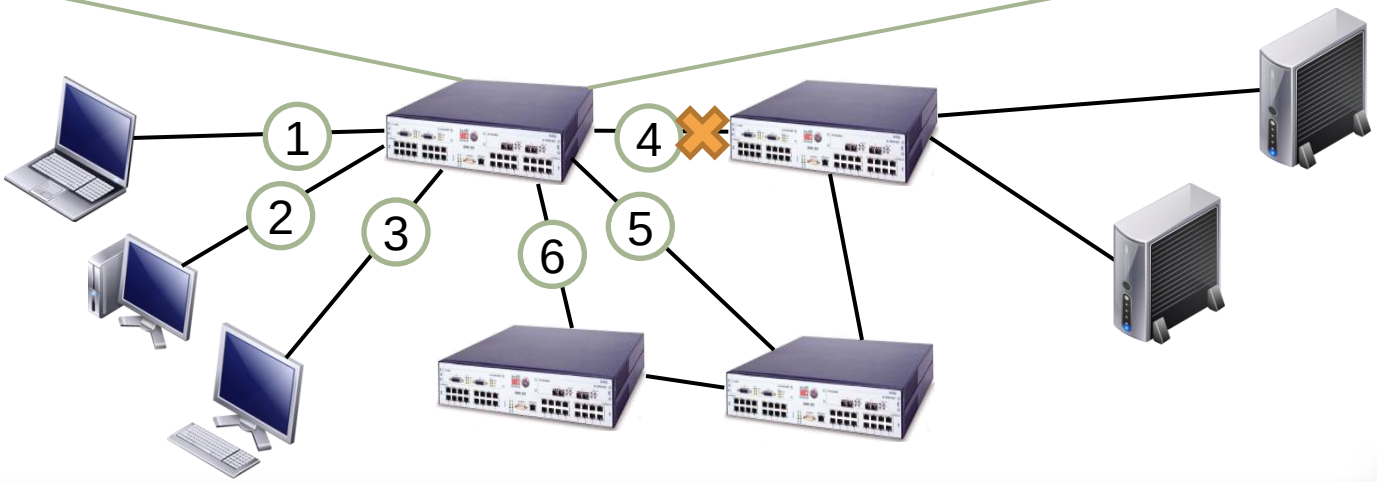
- Fast Failover
  - Type=fast-failover (ff)

Group Table

Group ID	Group Type	Counter	Action Buckets
100	Fast-failover	777	Port4, Port5, Port6

Flow Table

Switch Port	MAC src	MAC dst	Ether Type	VLAN ID	Src IP	Dst IP	Proto No.	TCP S Port	TCP D Port	Action
Port 1	*	*	*	*	1.2.2	*	*	*	*	Port 7
Port 7	00:FF …	*	0800	*	1.2.3 …	11.1…	*	*	*	Group 100



OpenFlow Meter Table

- Meter Table (ver 1.3)
  - Согласовывает пропускную способность для каждого потока
  - QoS control → Rate-limit, DiffServ …

Meter Table

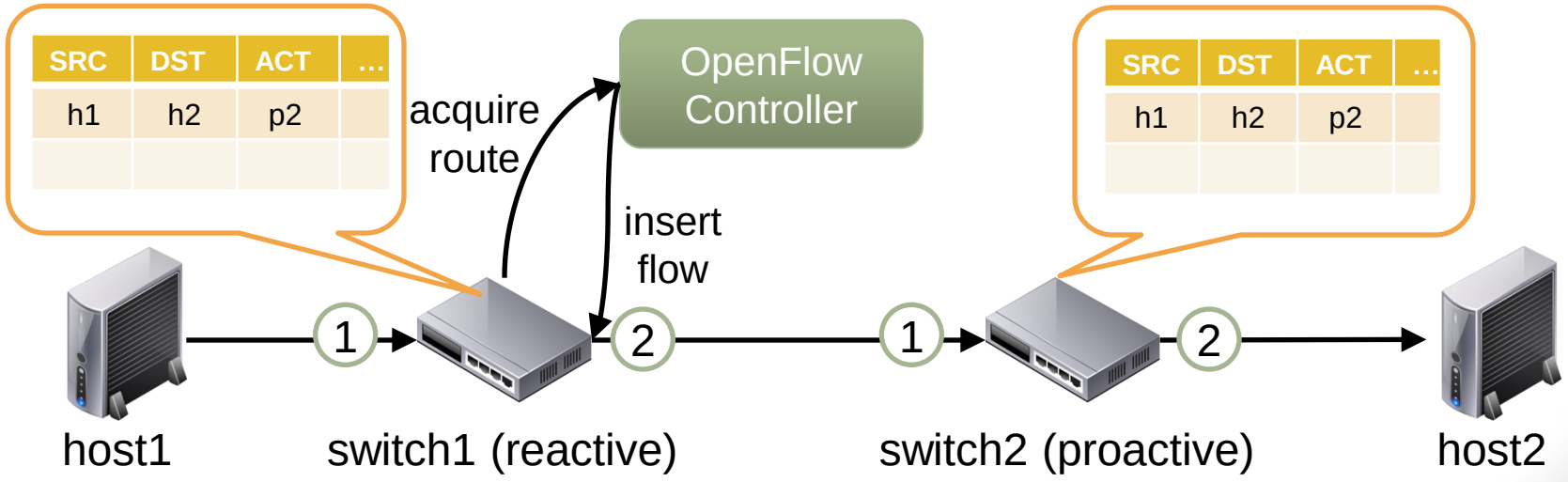
Meter ID	Band Type	Rate	Counter	Argument
100	Drop (remark DSCP)	1000 kbps	1000	xxx

Flow Table

Switch Port	MAC src	MAC dst	Ether Type	Src IP	Dst IP	Proto No.	TCP S Port	TCP D Port	Inst. Meter	Action
Port 1	*	*	*	1.2.2	*	*	*	*	N/A	Port 7
Port 1	00:FF …	*	0800	1.2.3 …	11.1…	*	*	*	Meter 100	Port 2

Packet Forwarding in OpenFlow

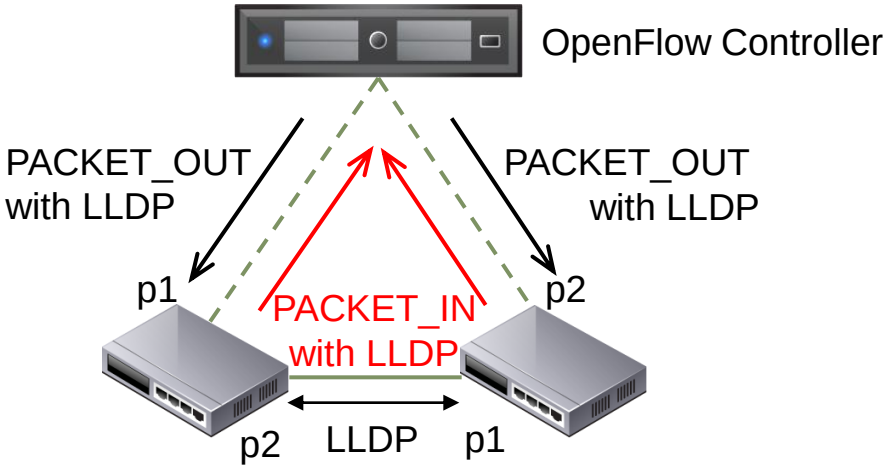
- Packet Forwarding
  - Reactive flow insertion
    - A non-matched packet reaches to OpenFlow switch, it is sent to the controller, based on the info in packet header, an appropriate flow will be inserted
    - Always need to query the path from controller during packet arrival → slow
    - Can reflect the current traffic status
  - Proactive flow insertion
    - Flow can be inserted proactively by the controller to switches before packet arrives
    - No need to communicate during packet arrival → fast packet forwarding
    - Cannot reflect the current traffic status



Исследование топологии сети в OpenFlow

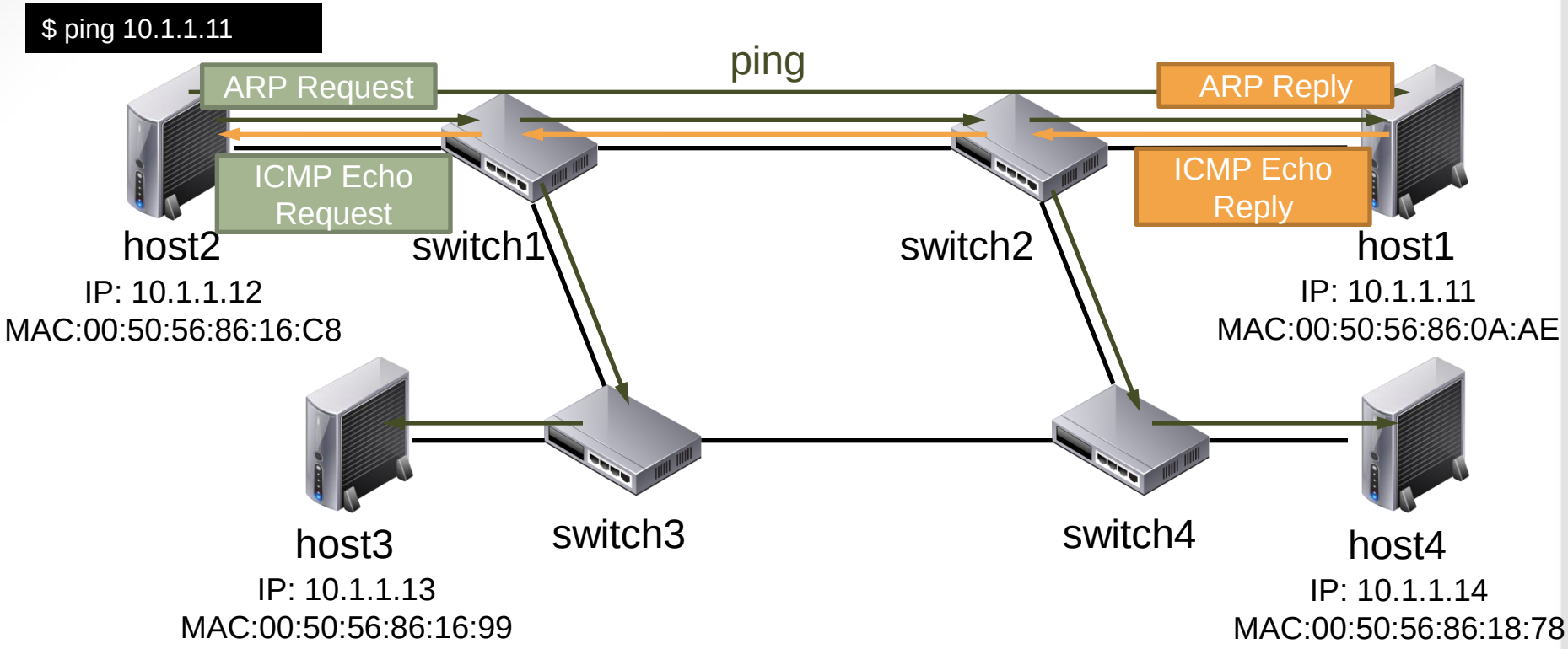
- Цель
  - Для построения архитектуры сети
- Метод
  - Использщзование Link Layer Discovery Protocol (LLDP)

IDX	SRC	DST	SRC PORT	DST PORT
153	sw. A	sw. B	p2	p1
…	…	…	…	…
357	sw. B	sw. A	P1	p2



Взаимодействие в традиционной сети

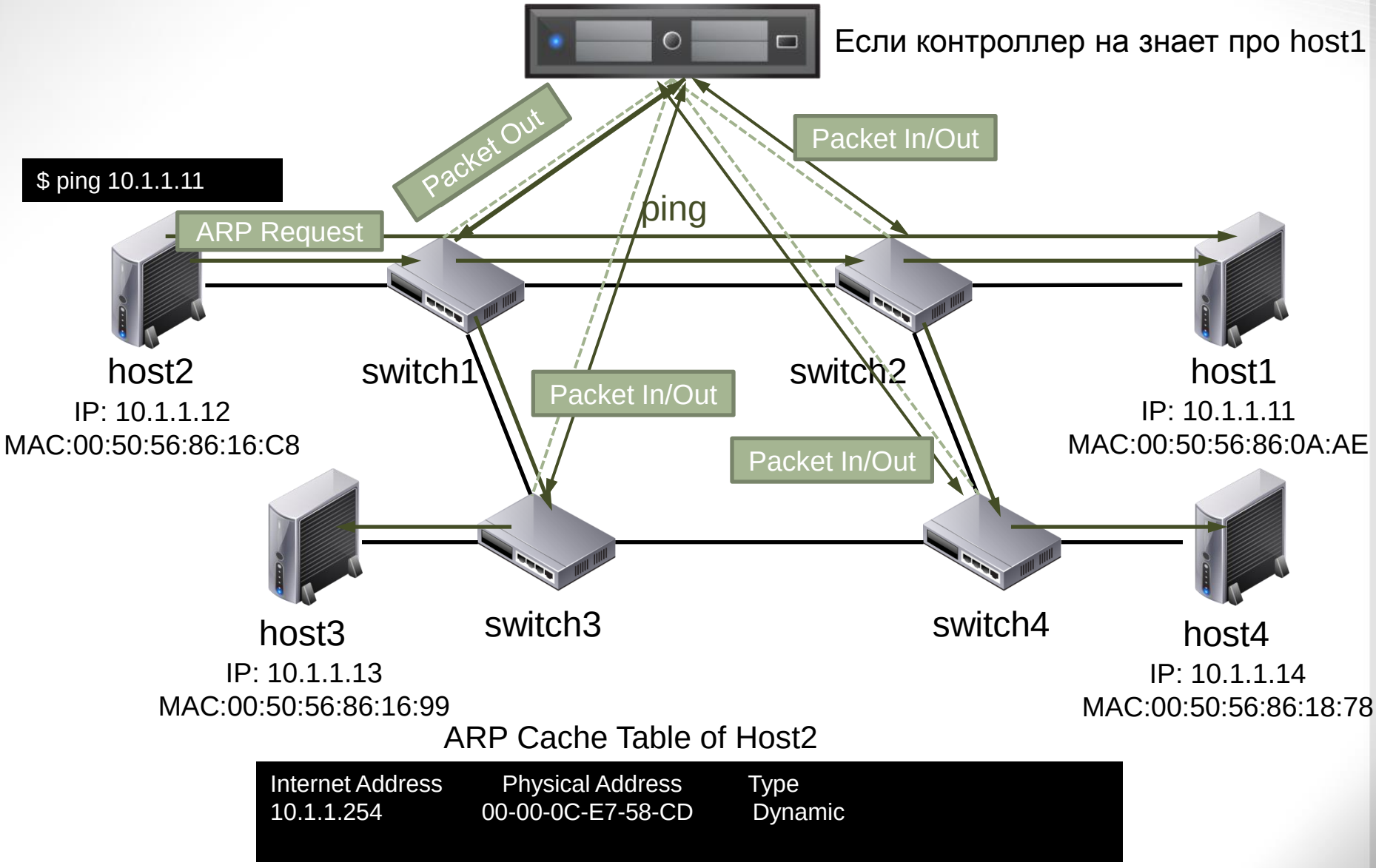
1. host2 tries communication to host1 by sending a ping ICMP packet
2. host2 broadcasts ARP Request packet
3. host1 replies ARP Request with ARP Reply
4. host2 creates entry to ARP Cache Table
5. host2 sends ICMP Echo request packet
6. host1 replies ICMP Echo request with ICMP Echo reply



ARP Cache Table of Host2

Internet Address	Physical Address	Type
10.1.1.254	00-00-0C-E7-58-CD	Dynamic
10.1.1.11	00-50-56-86-0A-AE	Dynamic

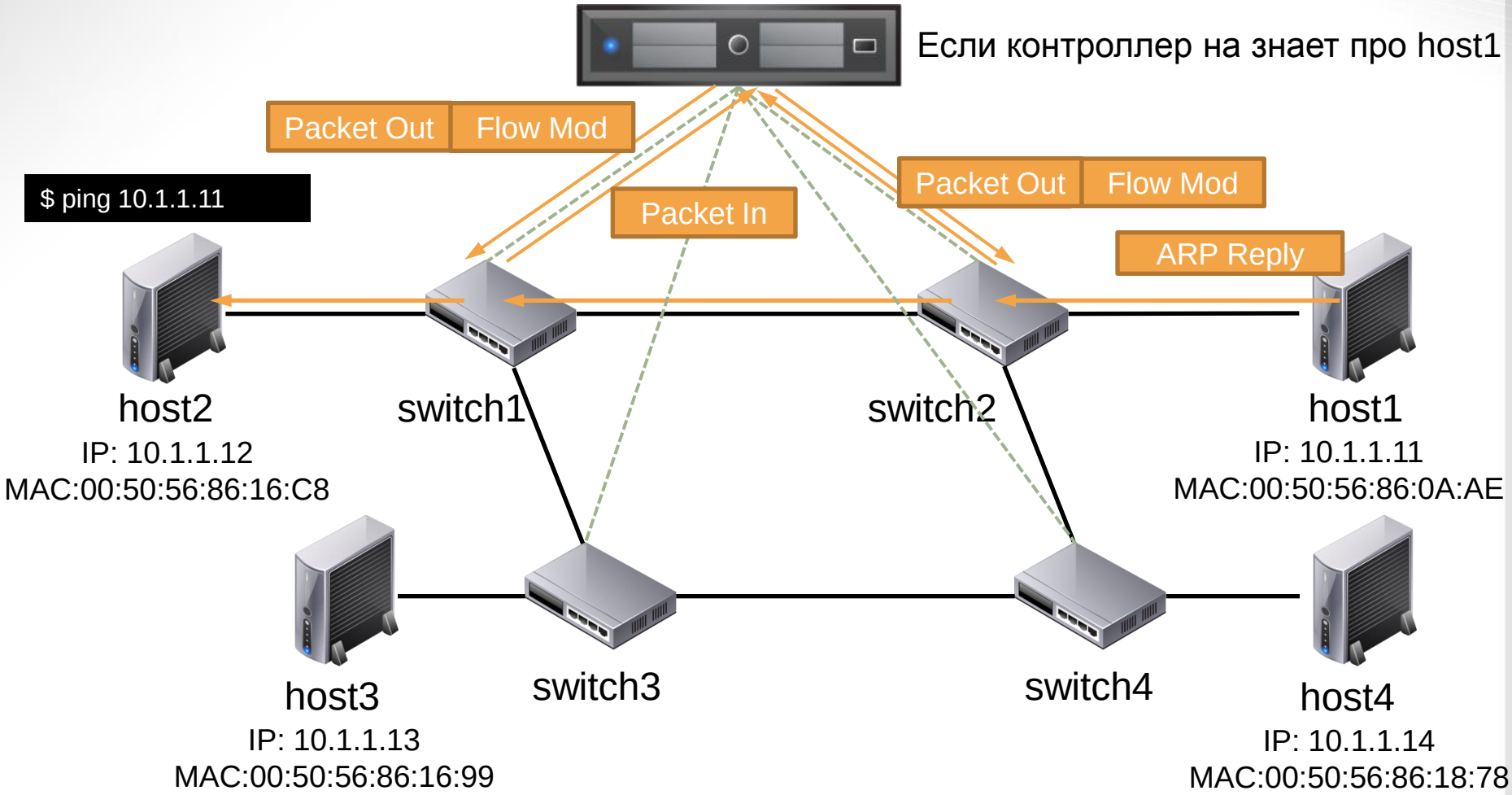
Взаимодействие в OpenFlow



ARP Cache Table of Host2

Internet Address	Physical Address	Type
10.1.1.254	00-00-0C-E7-58-CD	Dynamic

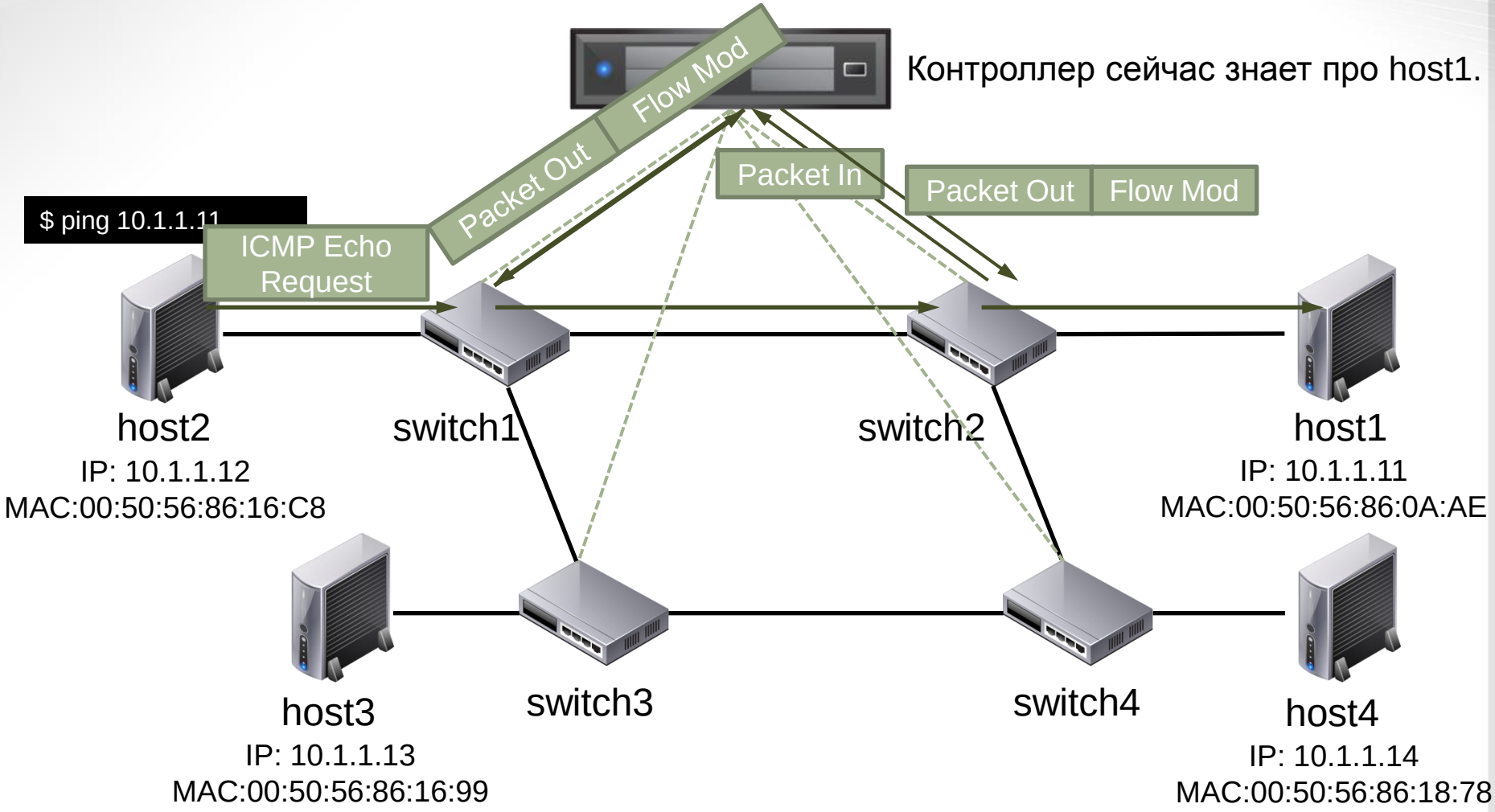
Взаимодействие в OpenFlow



ARP Cache Table of Host2

Internet Address	Physical Address	Type
10.1.1.254	00-00-0C-E7-58-CD	Dynamic
10.1.1.11	00-50-56-86-0A-AE	Dynamic

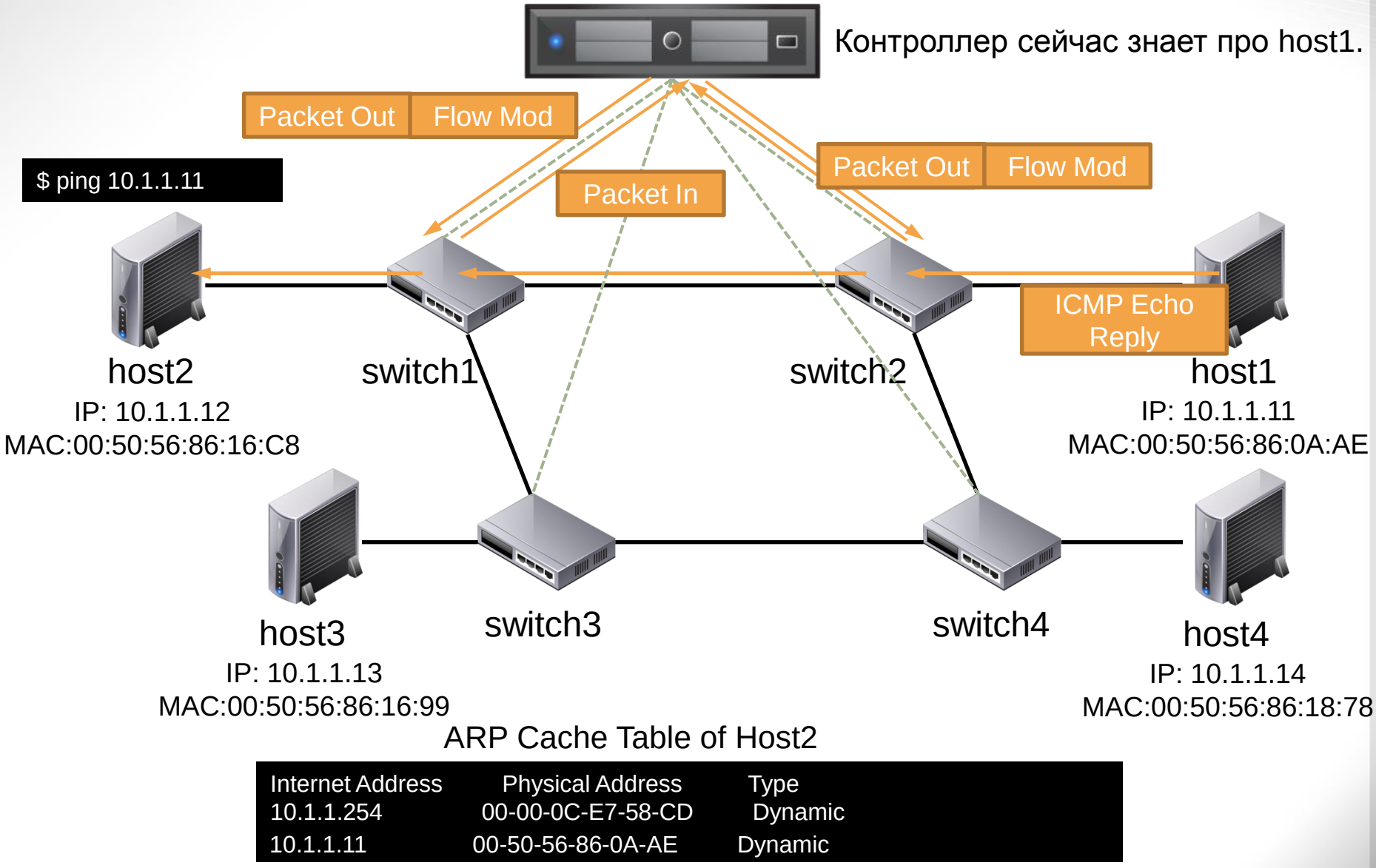
Заимодействие в OpenFlow



ARP Cache Table of Host2

Internet Address	Physical Address	Type
10.1.1.254	00-00-0C-E7-58-CD	Dynamic
10.1.1.11	00-50-56-86-0A-AE	Dynamic

Взаимодействие в OpenFlow



ARP Cache Table of Host2

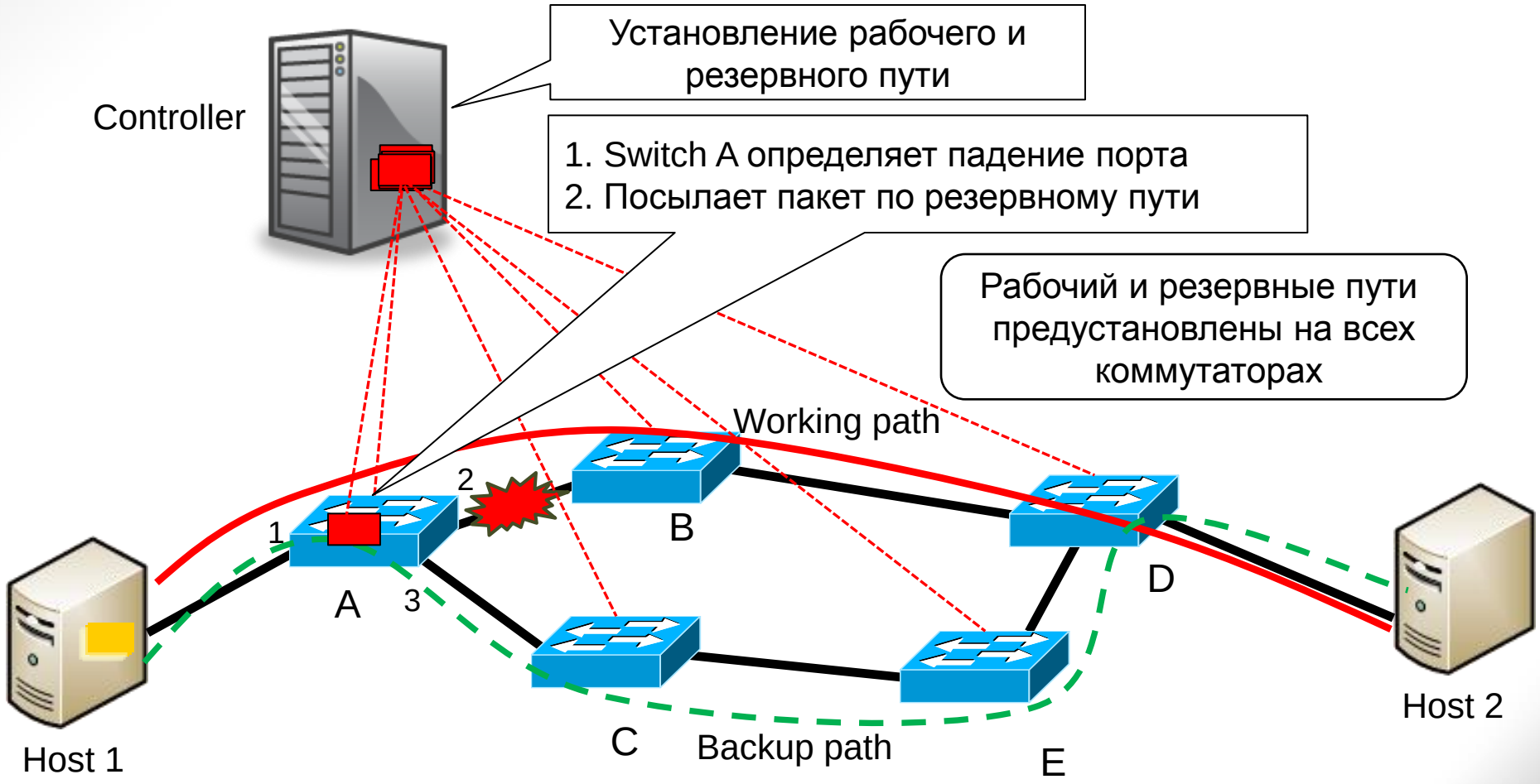
Internet Address	Physical Address	Type
10.1.1.254	00-00-0C-E7-58-CD	Dynamic
10.1.1.11	00-50-56-86-0A-AE	Dynamic

OpenFlow Отказоустойчивость

- OpenFlow Failover
  - Защита

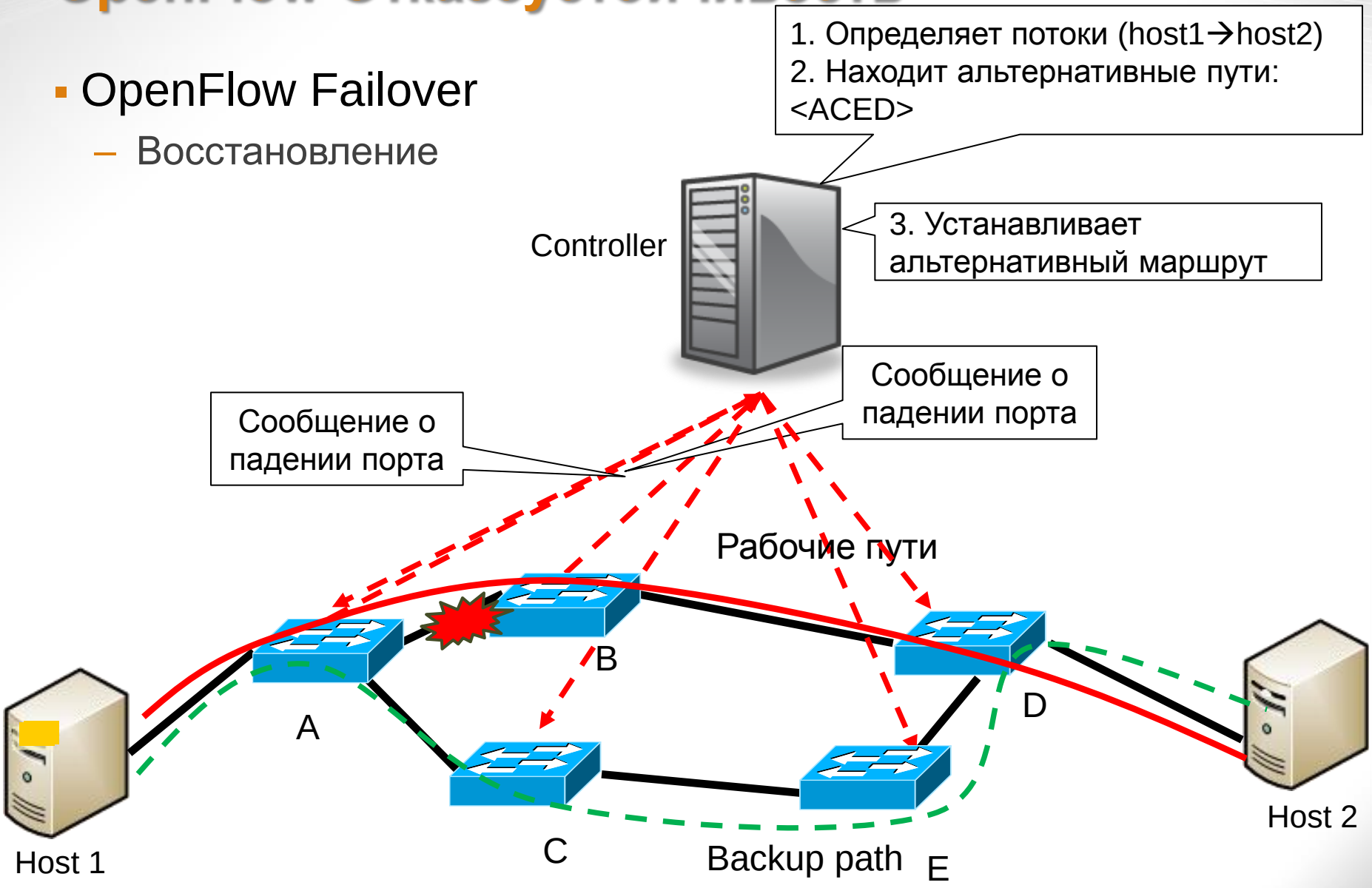
Flow table of Switch A (group table combined)

src	dst	Out port	Failover port
h1	h2	2	3



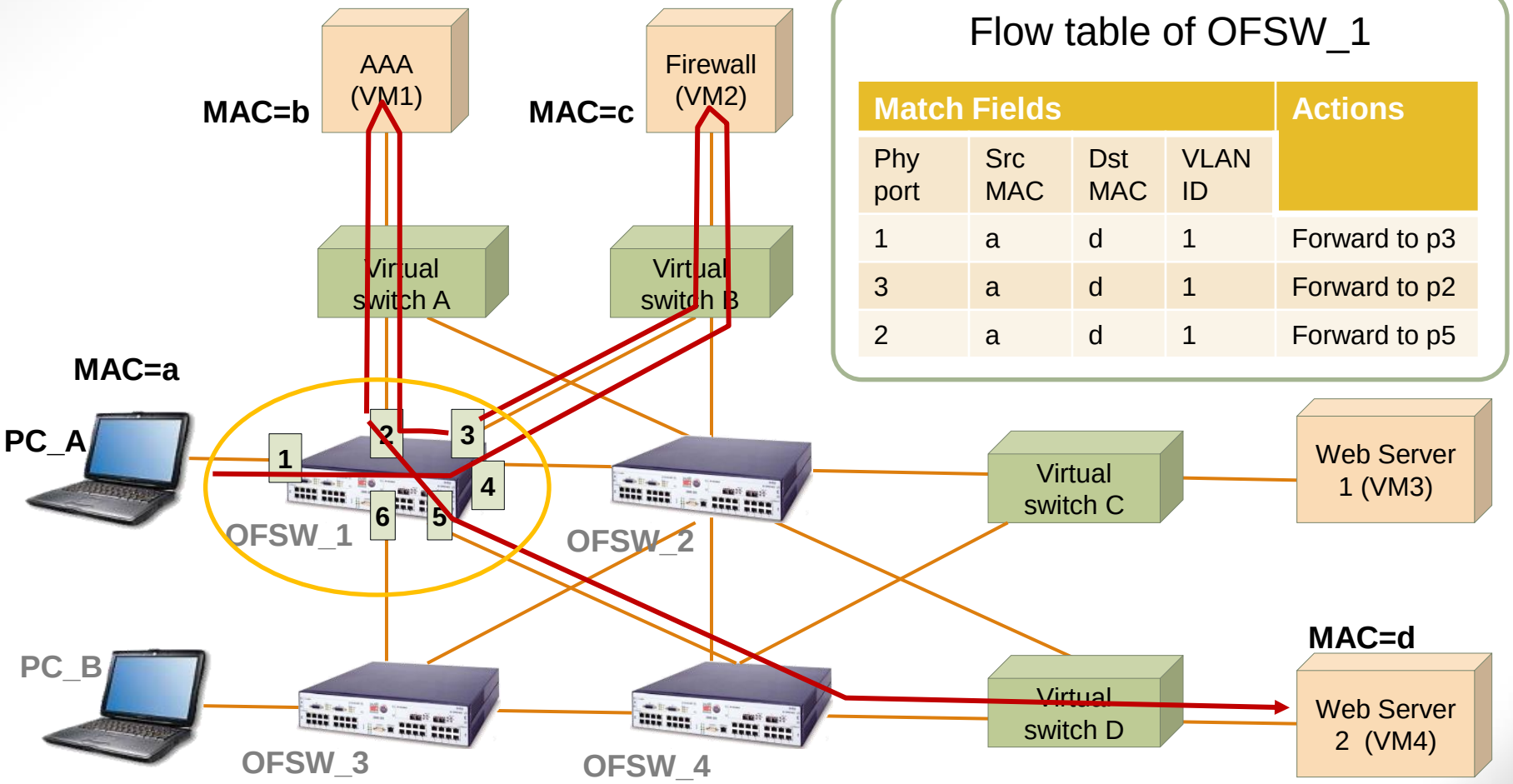
OpenFlow Отказоустойчивость

- OpenFlow Failover
  - Восстановление



OpenFlow Пример

- Пример управления маршрутизации(hop-by-hop routing)

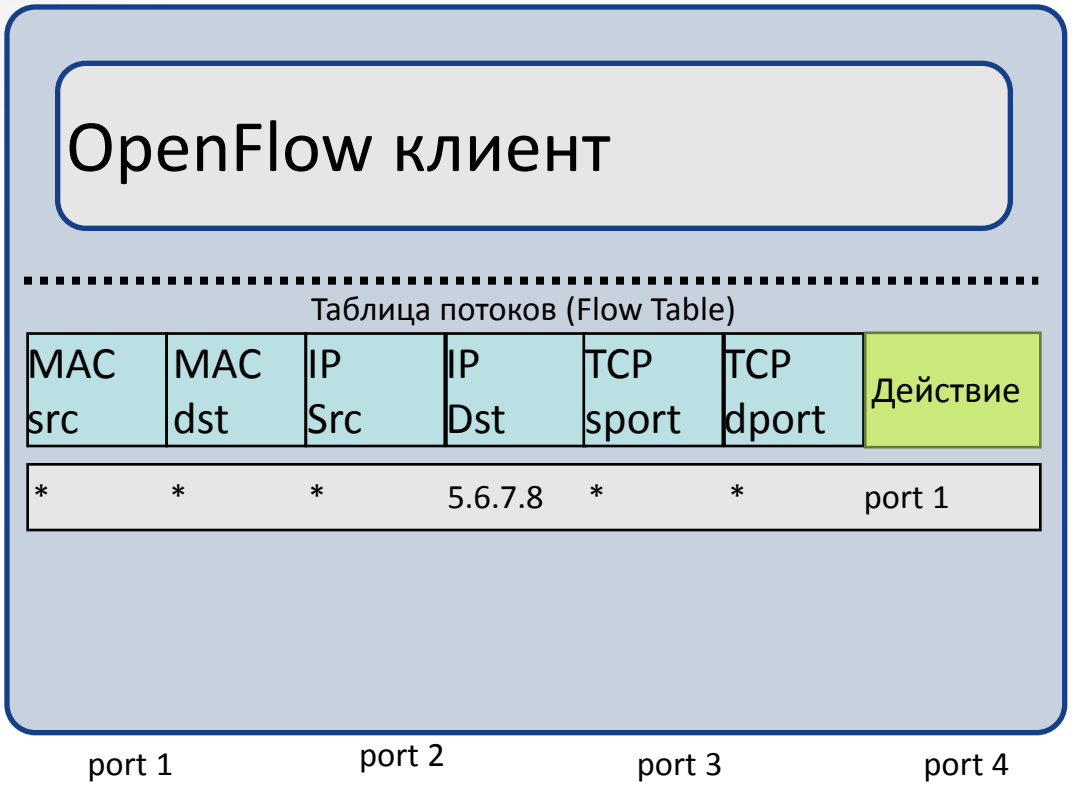


Flow table of OFSW_1

Match Fields				Actions
Phy port	Src MAC	Dst MAC	VLAN ID	
1	a	d	1	Forward to p3
3	a	d	1	Forward to p2
2	a	d	1	Forward to p5

Троичная ассоциативная память (TCAM)

Устройство OpenFlow коммутатора



- Таблица потоков содержит шаблоны заголовков и ассоциированные действия
- Заголовки имеют фиксированную длину
- Количество записей в таблице – единицы тысяч
- Пакеты анализируются и обрабатываются с канальной скоростью

Троичная ассоциативная память (TCAM)

В отличие от обычной машинной памяти (памяти произвольного доступа, или RAM), в которой пользователь задает адрес памяти и ОЗУ возвращает слово данных, хранящееся по этому адресу, АП разработана таким образом, чтобы пользователь задавал слово данных, и АП ищет его во всей памяти, чтобы выяснить, хранится ли оно где-нибудь в нем.

Если слово данных найдено, АП возвращает список одного или более адресов хранения, где слово было найдено (и в некоторых архитектурах, также возвращает само слово данных, или другие связанные части данных). Таким образом, АП — аппаратная реализация того, что в терминах программирования назвали бы ассоциативным массивом.

Троичная ассоциативная память (TCAM)

Из-за того, что АП разработана, чтобы искать во всей памяти одной операцией, это получается намного быстрее, чем поиск в RAM фактически во всех приложениях поиска.

Минус:

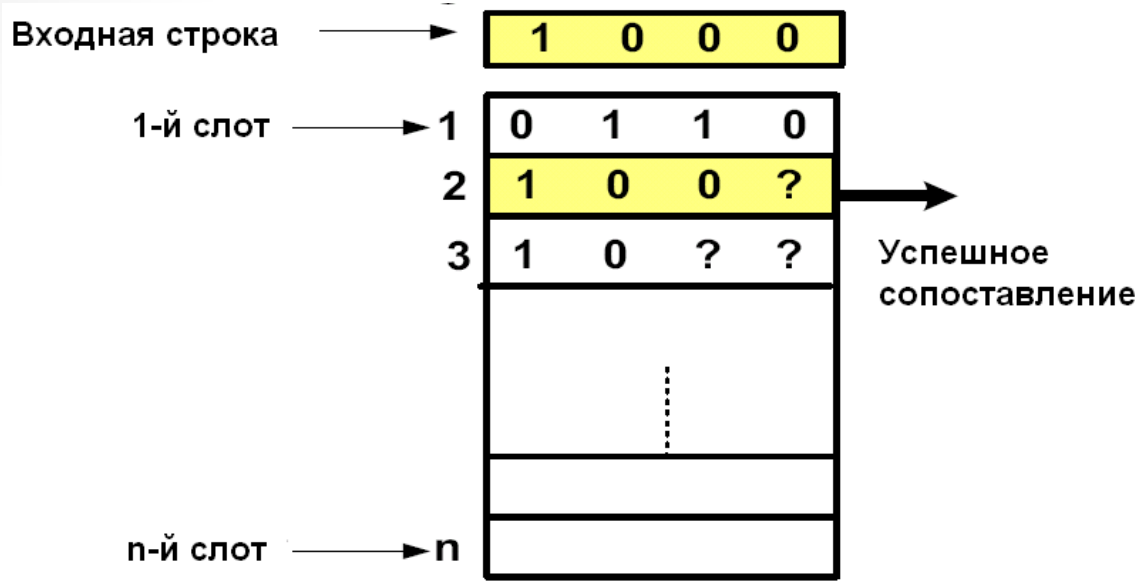
1. Большая стоимость АП. В отличие от чипа RAM, у которого хранилища простые, у каждого отдельного бита памяти в полностью параллельной АП должна быть собственная присоединенная схема сравнения, чтобы обнаружить совпадение между сохраненным битом и входным битом. К тому же, выходы сравнений от каждой ячейки в слове данных должны быть объединены, чтобы привести к полному результату сравнения слова данных.
2. Дополнительная схема увеличивает физический размер чипа АП, что увеличивает стоимость производства.
3. Дополнительная схема также увеличивает рассеиваемую мощность, так как все схемы сравнений активны на каждом такте. Как следствие, АП используется только в специализированных приложениях, где скорость поиска не может быть достигнута, используя другие, менее дорогостоящие, методы.

Троичная ассоциативная память (TCAM)

Двоичная АП — простейший тип ассоциативной памяти, который использует слова поиска данных, состоявшие полностью из единиц и нулей.

В троичной АП (Ternary Content Addressable Memory, TCAM) добавляется третье значение для сравнения «?» или «не важно», для одного или более битов в сохраненном слове данных, добавляя дополнительную гибкость поиску.
Например, в троичной АП могло бы быть сохранено слово «10??0», которое выдаст совпадение на любое из четырех слов поиска «10000», «10010», «10100», или «10110». Добавление гибкости к поиску приходит за счет увеличения сложности памяти, поскольку внутренние ячейки теперь должны кодировать три возможных состояния вместо двух. Это дополнительное состояние обычно осуществляется добавлением бита маски «важности» («важно»/«не важно») к каждой ячейке памяти.

Троичная ассоциативная память (TCAM)



- Множество нумерованных слотов
- Три возможных значения каждого бита: “0”, “1” и “?”
- Ширина TCAM (длина слота) – настраиваемый параметр
- На вход подается битовая строка
- TCAM выдает номер первого слота с успешным сопоставлением
- Фиксированное время каждого такта работы TCAM

Можно ли использовать TCAM для поиска произвольных шаблонов?

- Шаблоны относительно сложного вида
- Поиск шаблонов в телах пакетов
- Обработка трафика на гигабитном канале

Область применения: DPI для средств IDS

```
content: “|04|”; depth:1;
content: “|81 F1 03 01 04
9B 81 F1 01|”;
content: “sock”;
content: “send”
```

a. MS-SQL Worm detection

```
content: “USER”; nocase;
content: !”|0a|”; within: 50;
```

b. POP3 User Overflow Attempt

Примеры шаблонов в сигнатурах IDS SNORT

Постановка задачи поиска образцов в пакете

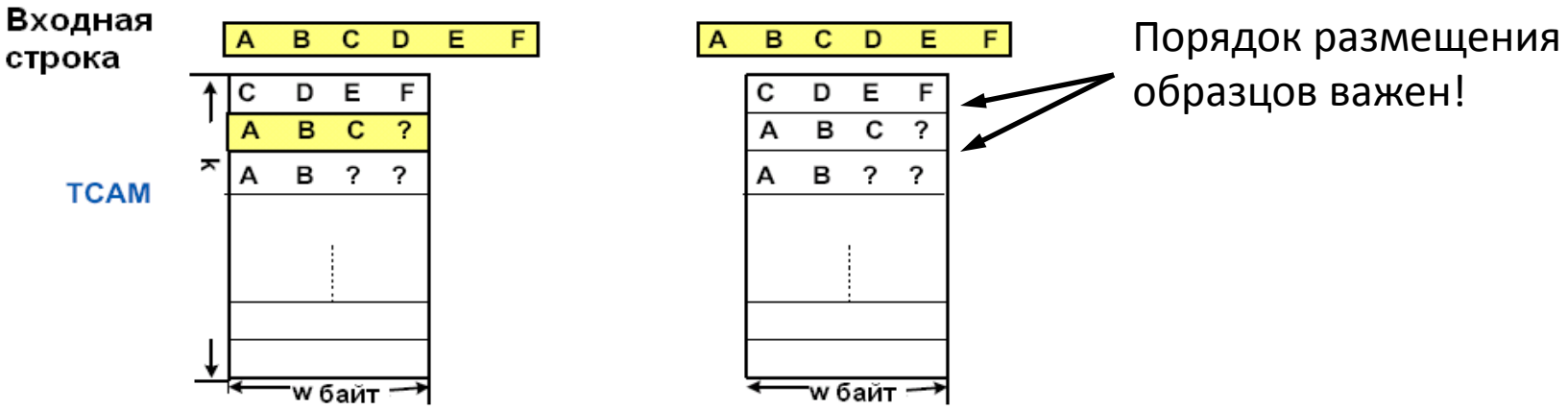
"Gigabit Rate Packet Pattern-Matching Using TCAM"

- Дано k образцов {P1,P2,…,Pk} и входная строка длины n
- Требуется для каждого образца определить, входит ли он в строку

- Виды образцов:
  - Простые – байтовая строка длины m
    - Детерминированные (единственное значение каждого байта)
    - Недетерминированные (возможны wildcard и case insensitive)
  - Составные
    - Связанные P = P1*P2, где * - произвольная строка ограниченной длины
    - Отрицание !P

Поиск простых детерминированных образцов

- Дано k простых детерминированных образцов длины меньше или равной w (ширине TCAM)
- В такой ситуации, все образцы можно сразу разместить в слотах TCAM
- Короткие образцы дополнить справа символами “?”

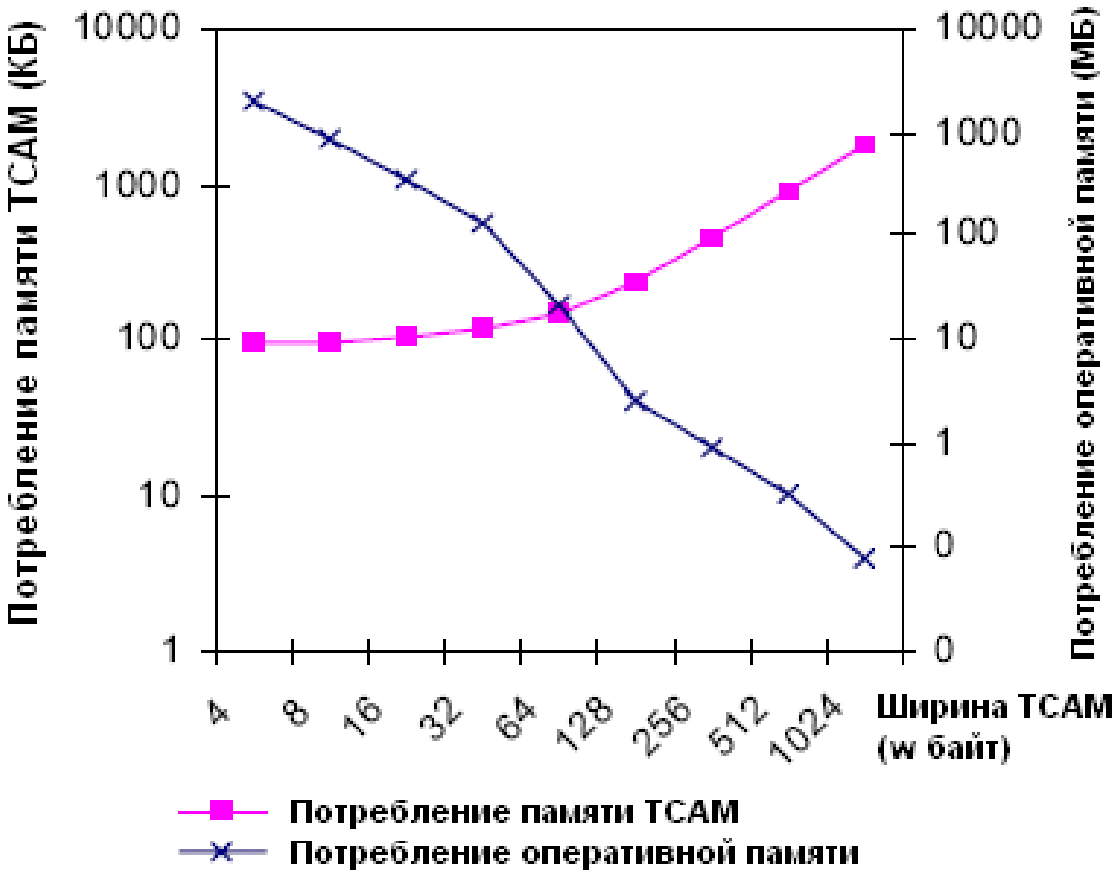


Скорость обработки трафика полностью определяется скоростью работы TCAM!

Методы поиска образцов (шаблонов)

- Программные
  - Алгоритмы Кнута-Морриса-Пратта и Бойера-Мура
    - Рассчитаны на поиск одного образца
    - Сложность O(n+m)
    - Сложность для k образцов O(k*(n+m))
  - Алгоритмы Ахо-Карасик и Комменц-Вальтера
    - Одновременный поиск множества образцов
    - Сложность O(n) и не зависит от числа образцов
    - Экспоненциальный рост потребляемой памяти
- Аппаратно-ускоренные
  - ПЛИСы
    - Сложность O(n)
    - Высокие затраты памяти (квадратичные от длины образца)
    - Плохо масштабируются для множества образцов

Зависимость потребления памяти TCAM и оперативной памяти от ширины w TCAM



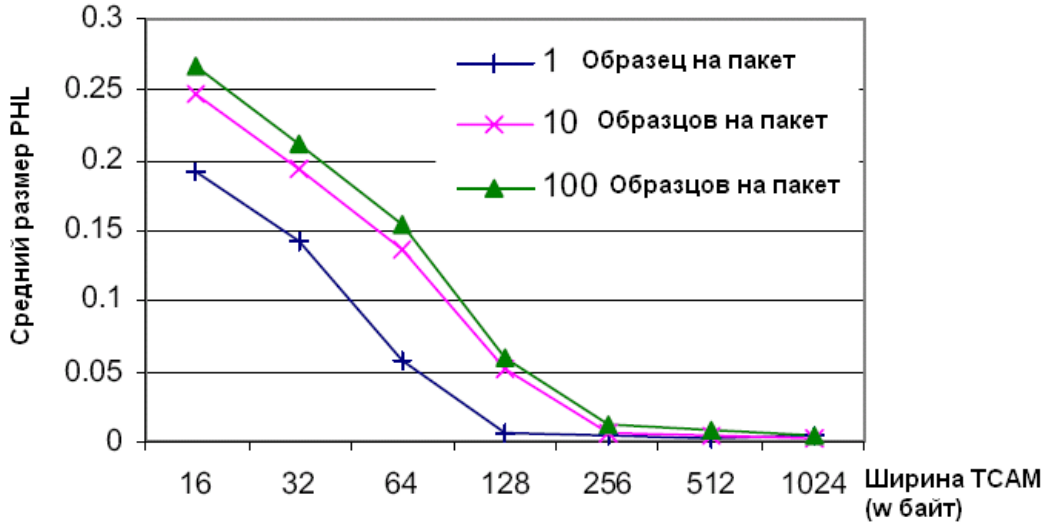
Набор образцов взят из базы сигнатур ClamAV 0.15

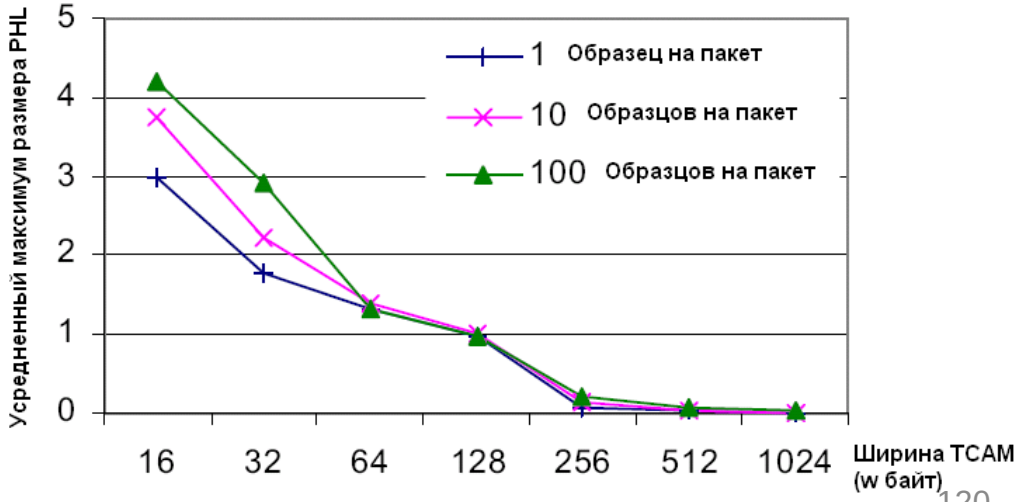
1768 образцов

Длины: от 6 до 2189 байт

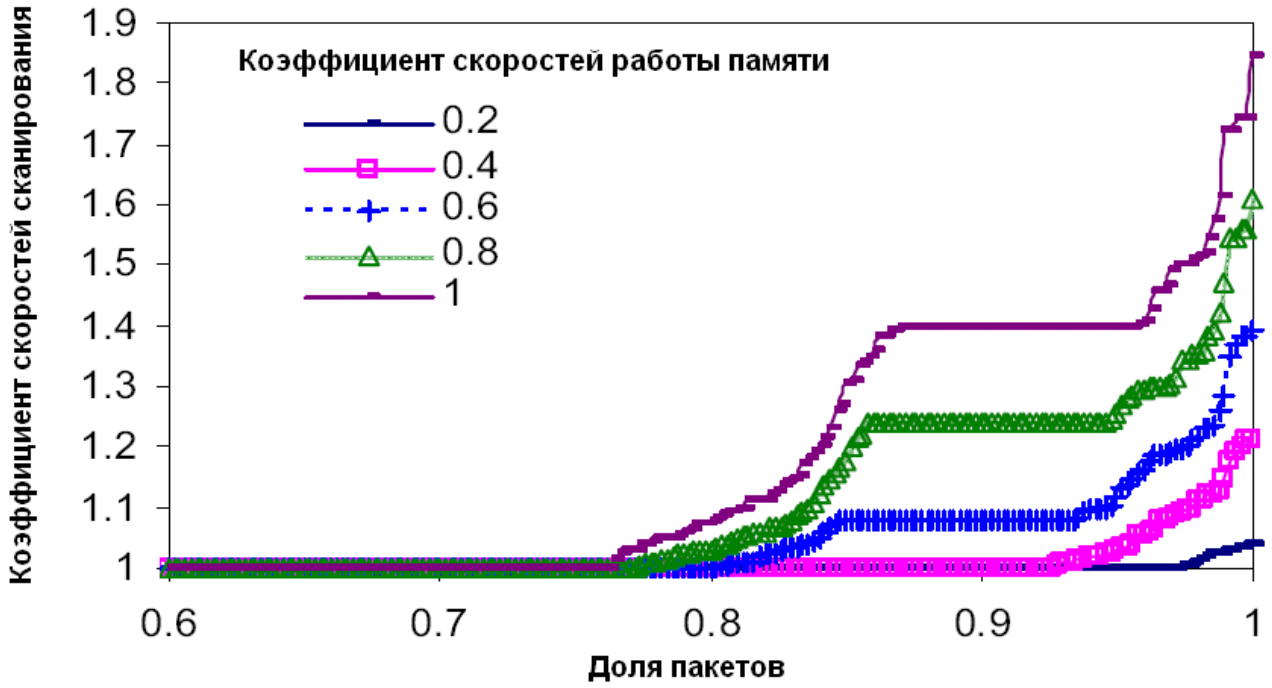
Средняя длина: 55 байт

Размер списка частичных совпадений для синтетического трафика





Зависимость времени обработки пакетов от соотношения скоростей работы RAM и TCAM

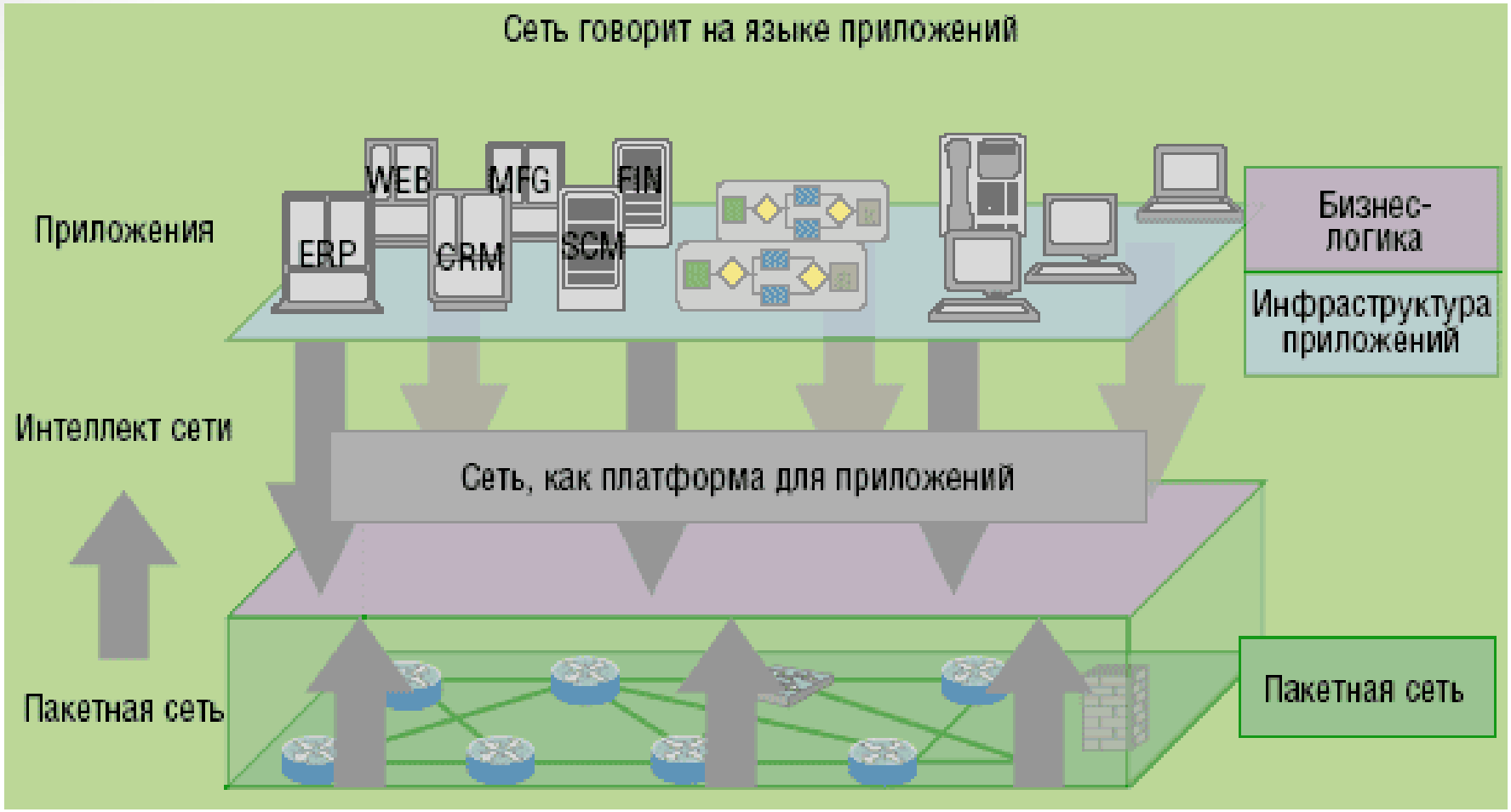


Коэффициент скоростей сканирования – отношение общего времени обработки пакета ко времени обработки на TCAM

Коэффициент скоростей работы памяти – отношение затрат времени на одно обращение к оперативной памяти к длительности такта TCAM

Образцы взяты из базы сигнатур IDS SNORT

Сеть как платформа сервисов



Спасибо за внимание!

www.sut.ru

СПб ГУТ)))