第一章 OpenFlow1.1

注：Wire 协议版本号 0x02。

相比较 of1.0 版本，of1.1 最大的变化在于：改变了交换机流表结构（引入多层级流表序列和组流表）并扩充了匹配和执行操作，让交换机对流进行的操作更加灵活。本部分介绍 of1.1 的内容，并重点突出重要的修改部分。

第1.1节 交换机组件

交换机包括一个或多个普通流表（flow table），以及一个组流表（group table）和一个 of 通道（openflow channel）连接到外部控制器。其中流表负责网包查找和转发，控制器通过 of 协议来管理交换机。控制器的操作包括添加、更新或删除流表表项，支持（被动）响应（reactively）或主动（proactive）模式。

每一个流表项分为三个域，包括匹配域（包括包头、ingress 端口、metadata 值）、计数器、操作指令集（Instructions set，of1.0 中只有一个操作行为）。

网包的流匹配首先从第一个流表开始。在每一个流表中按照表项的优先级顺序进行匹配，找到第一个匹配（最高优先级）则执行响应的指令集。如果没有发现匹配项，根据交换机的配置不同，网包可能经 of 通道发送到控制器、丢弃或继续在下一个流表中进行查找（管道处理）。

操作指令集描述包转发、修改、组流表处理或管道处理（pipeline processing）。流水处理指令允许网包发送到后续流表中进行进一步的处理，并在流表之间传递相关信息（以 metadata 的形式）。直到在某个表中的匹配表项没有继续指定后续流表，管道处理停止，网包被修改或转发。

转发操作的端口可能是实际的物理端口，也可能是虚拟端口（交换机定义或标准预留）。交换机定义虚拟端口可能指定链路聚合组（link aggregation group）、隧道（tunnel）或回送端口（loopback interface）。标准预留虚拟端口可能指定通用的转发操作（发往控制器、洪泛、正常交换机转发等）。

流表项可以指向一个组表项，指定额外的处理，来实现复杂的操作。

组流表中存储组表项。每条组表项包括一系列的操作动作。

第1.2节 术语说明

比特（Byte）：八位，一字节。

网包（Packet）：以太帧，包括包头和载荷/内容。

管道（Pipeline）：连接在一起的流表的集合，提供匹配、转发或修改等操作。

端口（Port）：网包进入或离开的接口。可能是物理端口、虚拟端口等。

匹配域（Match Field）：匹配网包的域，包括包头、ingress 端口、metadata 值。

元数据（Metadata）：可掩码寄存器值，用于在流表间传递信息。

指令（Instruction）：一种操作，或者是包括一系列行为的集合，或者是修改管道处理。

行动（Action）：一种操作，转发网包到某个端口或者修改网包（例如减少 TTL 等）。
行动集（Action Set）：行动的集合，绑定到需要进行管道处理的网包。
组（Group）：一个行动桶的列表和一些选择方法，来选择一个或多个桶来执行。
行动桶（Aciton bucket）：一个行动集合，包括行动和对应参数。为组而定义。
标签（Tag）：一个头部。可以通过 push 或 pop 操作插入网包或删除。
最外层标签（Outermost Tag）：离包头开头最近的标签，也就是最外层的标签。

第1.3节 表

包括流表（Flow Table）和组表（Group Table）。

1.3.1 流表

包括若干条流表项。每条流表项包括匹配域、计数器和指令集。

表格 一-1 of1.1 流表项结构

Head Fileds	Counter	Instructions

- 匹配域：匹配网包，包括 ingress 端口、包头和可选的 metadata 信息（由前一个表来指定）。
- 计数器：更新记录匹配包数。
- 指令集：在管道处理中修改行动集合。

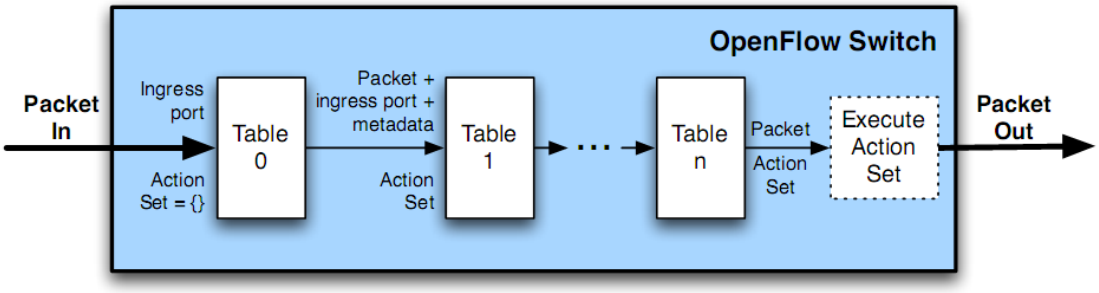
1.3.1.1 管道处理

OpenFlow 兼容交换机分为两种：纯 OpenFlow 交换机（OpenFLow-only）和混合 OpenFlow 交换机（OpenFlow-hybrid）。

纯 OpenFlow 交换机：仅支持 OpenFlow 操作。所有网包执行 of 管道处理，不支持其他操作。

混合 OpenFlow 交换机：支持 OpenFlow 操作和正常的以太交换机操作（传统的二层以太交换、Vlan 隔离，三层的路由、ACL、QoS 处理等）。此类交换机需要有一个额外的分类操作（规范以外）来指定网包执行 OpenFlow 处理还是正常的交换处理。此外，也允许网包通过 NORMAL 和 FLOOD 虚拟口从 OpenFlow 处理转到正常的交换处理。

of 管道处理包括若干张流表。如图表 一-1 所示。所有的查找都从表 0（第一张表）开始，后续表由前一张表匹配输出来指定。查找的结果可以指定一张序号比自己大的表，如果没有指定后续表，则管道处理结束，对应的行动集被执行。



图表 一-1 of 管道处理

如果网包查找没有匹配表项，则发生 miss。执行操作由表的配置来指定，默认是发送到控制器或者丢弃。也可以指定后续表来处理。

1.3.2 组表

包括若干条组表项。每条组表项包括组标号、组类型、计数器和行动桶列表。

表格 一-2 of1.1 组表项结构

Group Identifier	Group Type	Counters	Action Buckets

- 组标号：32 位唯一无符号整数，标识组。
- 组类型：确定组的类型。
- 计数器：更新记录处理包数。
- 行动桶列表：有序的列表，每个行动桶中包括执行行动和对应的参数。

1.3.2.1 组类型

all

执行组中的所有行动桶。用于多播或广播转发。网包被复制为多份，分别在对应桶上执行。如果某个行动桶让网包从 ingress 发出，则默认丢弃对该包的复制。控制器可以通过设置额外的桶（包含发送到 OFPP_IN_PORT 虚拟口行动）来强制该操作。

select

执行组中的某个行动桶。网包通过某种选择算法（哈希、轮询等）来执行某个桶。当某个端口故障时，选择被自动限制在剩余可用端口上进行，用以减少故障引发的影响。

indirect

执行组内一个定义的桶。让多个流或组指向同一个组标志，进行高效快速的收敛操作。

fast failover

执行第一个激活的桶。

1.3.3 匹配

匹配可以在包括包头、ingress 和 metadata 等域上进行，如表格 一-3 所示。

表格 一-3 of1.1 表项的匹配域

Ingress Port	Metadata	Ether src	Ether Dst	Ether Type	VLAN id	VLAN Pri	MPLS Label	MPLS traffic class	IPv4 src	Ipv4 dst	Ipv4 proto/ ARP opcode	IP ToS bits	TCP/UDP/ SCTP Src Port ICMP type	TCP/UDP/ SCTP Dst Port ICMP Code

各个匹配域的说明如表格 一-4 所示。

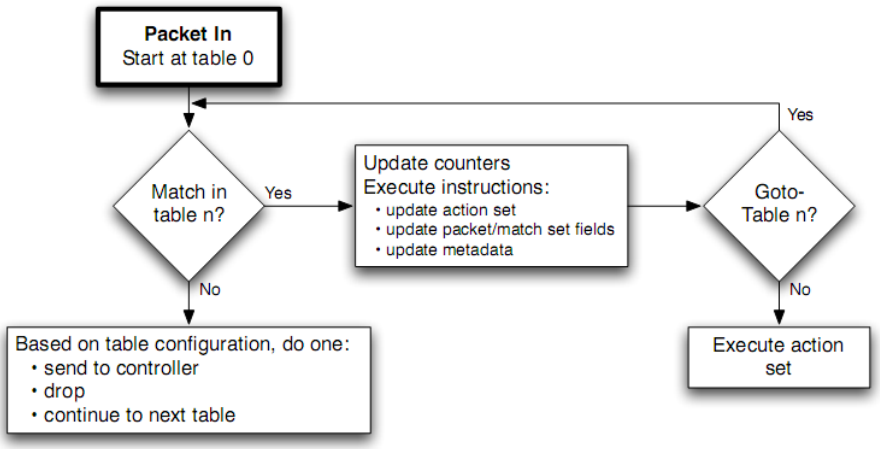
表格 一-4 匹配域说明

域	位	使用范围	备注
Ingress Port	32	所有网包	标识进口，从 1 开始编号。可以是物理端口或虚拟端口。
Metadata	64	Table 1 以及以上	
Ethernet source address	48	激活端口的所有网包	任意掩码
Ethernet destination address	48	激活端口的所有网包	任意掩码
Ethernet type	16	激活端口的所有网包	VLAN 标签后的以太类型。802.3 帧特殊处理。
VLAN id	12	所有 VLAN 网包	最外层 VLAN 头的标识号。
VLAN priority	3	所有 VLAN 网包	最外层 VLAN 头的 PCP 域。
MPLS label	20	所有 MPLS 网包	最外层 MPLS 头标签
MPLS traffic class	3	所有 MPLS 网包	最外层 MPLS 头标签
IP source address	32	所有 IP 和 ARP 网包	部分掩码或任意掩码
IP destination address	32	所有 IP 和 ARP 网包	部分掩码或任意掩码
IP protocol/ARP opcode	8	所有 IP 和 ARP 网包	ARP opcode 只有最低 8 位使用
IP ToS bits	6	所有 Ipv4 网包	按照 8-bit 值使用，设置 ToS 到高 6 位
Transport sourceport / ICMP Type	16	所有 TCP、UDP、 ICMP 网包	ICMP 类型只有低 8 位使用
Transport destination port / ICMP Code	16	所有 TCP、UDP、 ICMP 网包	ICMP 代码只有低 8 位使用

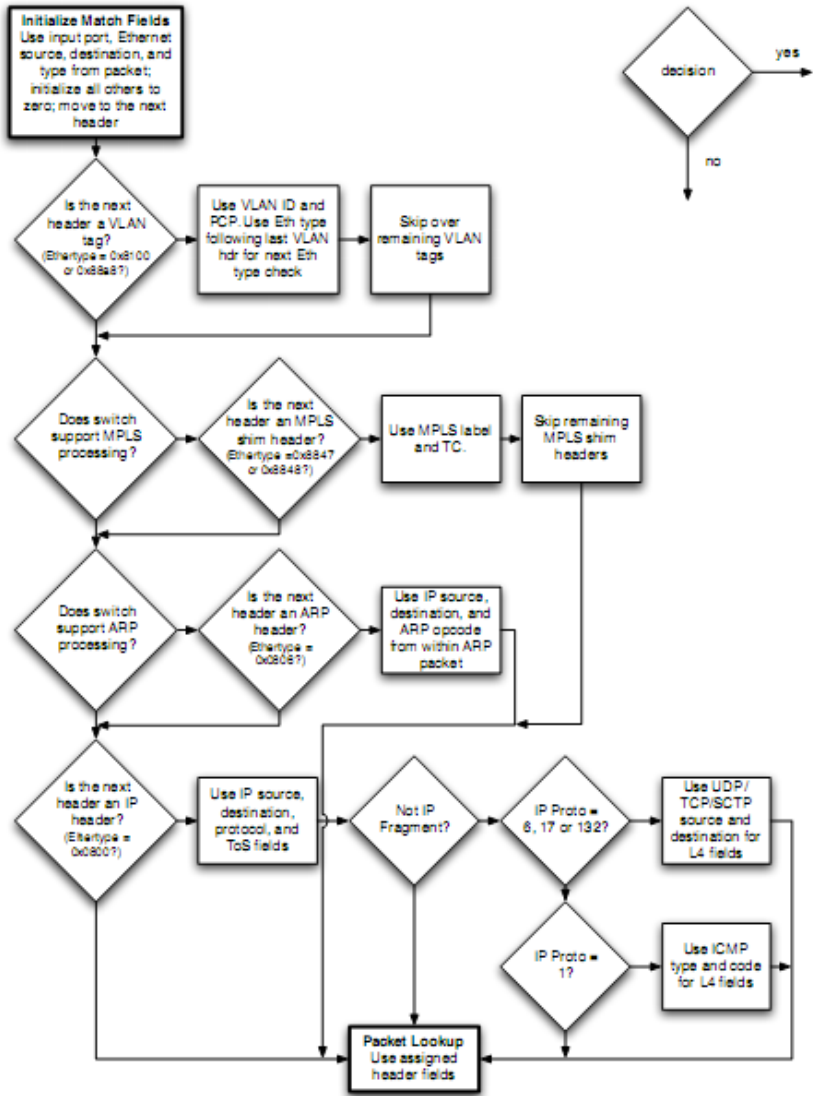
当收到网包后，首先在第一张表上进行匹配操作，可能的后续表由匹配结果指定。整个匹配流程如图表 一-2 所示。具体的匹配过程如图表 一-3 所示。进行匹配时，只有最高优先级的匹配表项执行。如果有多条匹配项具备同等优先级，此种情况未作定义。

如果交换机配置包括 OFPC_FRAG_REASM 标签，则 IP 碎片在进行管道处理之前必须先进行重组。

of1.1 中未定义收到非法格式的网包时的操作。



图表 一-2 整体流匹配过程



图表 一-3 具体流匹配过程

1.3.4 计数器

计数器可以针对每个表、流、端口、队列、组或桶来指定。计数器循环使用，可能溢出，不可用的计数器默认被置为-1。

表格 一-5 统计信息需要的计数器

Counter	Bits
Per Table	
Active Entries	32
Packet Lookups	64
Packet Matches	64
Per Flow	
Received Packets	64
Received Bytes	64
Duration (seconds)	32

Duration (nanoseconds)	32
Per Port	
Received Packets	64
Transmitted Packets	64
Received Bytes	64
Transmitted Bytes	64
Receive Drops	64
Transmit Drops	64
Receive Errors	64
Transmit Errors	64
Receive Frame Alignment Errors	64
Receive Overrun Errors	64
Receive CRC Errors	64
Collisions	64
Per Queue	
Transmit Packets	64
Transmit Bytes	64
Transmit Overrun Errors	64

1.3.5 指令

当网包被对应流表项匹配后，表项关联的指令集将被执行，目前包括如下指令类型。

应用行动（Apply-Action）

立刻执行指定的行动。可以用于修改表之间的包或者执行同一类型的多个行动。这些行动被一张行动列表所指定。

清除行动（Clear-Action）

清除在行动集合中的所有行动。

写行动（Write-Action）

将指定的行动合并到当前的行动集合中。

写元素据（Write-Metadata）

将给定数据写到 metadata 域。同时给定数据中的掩码用于标记寄存器中修改的位。

跳转到表（Goto-Table）

指定管道处理中的下一张表，其序号必须比当前表序号大。最后一张表不存在该操作。

一般的，对应到一条表项的指令集合最多包括上述类型最多各一条指令，并被顺序执行。实际上，在写行动前执行清除行动，跳转到表则最后被执行。

当交换机无法执行某个指令的时候，可以拒绝，并必须返回一个不支持的流错误（unsupported flow error）。

1.3.6 行动集合

行动集合绑定到每个网包，默认为空。流表项可以通过写行动或者清除行动来修改

对应匹配表项的行动集合。当指令集不包括跳转到表指令时，管道操作停止，行动集合被执行。

一个行动集合包含的行动各个类型最多只有一个。当需要在同一类型的多个行动时，可以采取应用行动。

行动集合中的行动按照如下顺序执行，即便并非按照该顺序添加。如果行动集合中包括组行动，也会按照如下顺序执行。要实现任意顺序的执行，则需要使用应用行动。

顺序为：复制进入的 TTL 到网包、执行所有的标签 pop 操作、执行所有的标签 push 操作、复制向外的 TTL 到网包、减小 TTL、执行所有的设置域操作、执行所有的 QoS 操作、执行组行动、发送出网包。

在行动集合中，发送行动在最后被执行。如果组行动和发送行动同时被指定，则发送行动被忽略。如果没有发送行动，同时没有组行动，则网包被丢弃。组行动可以继续指定其他组行动。

1.3.7 行动列表

应用行动（Apply-Action）指令和网包发出（Packet-out）消息包括了一个行动列表。行动列表中的多个行动被依次立即执行。所有的行动都是累加执行。如果有两个添加 VLAN 头的操作，则网包会被添加上两个 VLAN 头。如果遇到发送行动，则当前状态的网包的一份复制将被发送。如果碰到组操作，则当前状态的网包的一份复制将被相关的组处理。

应用行动操作中的行动列表执行后，管道处理会在修改后的网包上继续执行。行动集合并不因行动列表的执行而改变。

1.3.8 行动

同 1.0 中一样，交换机并非需要支持所有行动，但最少需要支持必需行动（Required Action）。交换机需要在跟控制器建立连接的时候协商支持的可选行动。

必需行动之发包（output）

指将包转发到指定的物理端口或交换机定义的部分虚拟端口。支持的保留虚拟端口包括

ALL：所有标准端口，但不包括进口和配置为 OFPPC_NO_FWD 的端口。

CONTROLLER：添加包头然后将包送往控制器。

TABLE：将包发给流表，进行管道处理。仅当在 packet-out 消息对应的行为集合时候启用。

IN_PORT：将包从入口发出。

可选行动之发包（output）

将网包发送到如下的虚拟端口。

LOCAL：发送到交换机本地网络栈。本地端口让远端的表项与本地交换机通过 OpenFlow 网络进行交互，而不是隔离的控制网络。通过合适的默认规则，可以实现同域内的控制连接。

NORMAL：通过传统的非 OpenFlow 管道交换机进行处理。如果交换机不支持将网包从 OpenFlow 管道发送到普通管道，则它必须指明这一点。

FLOOD：使用正常的管道将网包进行洪泛。一般的，将网包从不包括入口或 OFP

PS_BLOCKED 状态的其他所有端口发出。交换机也可以通过 VLAN 来指定洪泛到某些特定端口。

纯 of 交换机不支持发送到 NORMAL 端口或 FLOOD 端口，而 of 混合交换机可能支持。将包转发到 FLOOD 端口将取决于交换机的实现和配置，而通过组转发到 all，将可以利用控制器实现更灵活的洪泛机制。

可选行动之设置队列（set queue）

设置队列行动给一个网包设置一个队列标号。当网包通过发包行动发送到某个端口时，队列标号将来决定发送队列。发送行为受队列配置限制，可以实现简单的 QoS 操作。

必备行动之丢包（drop）

没有特定的行动来指定丢包操作。相反的，如果网包没有发包行动，则默认丢弃。这一结果可以由管道处理中的空指令或空行动桶引发，或在执行清除行动后引发。

必备行动之组（group）

将网包通过指定的组进行处理，具体行为取决于组类型。

可选行动之标签头处理（push or pop tag）

将网包打上或去掉 Ethernet、VLAN、MPLS、ARP/IP、TCP/UDP/SCTP 等标签头。后添加的标签头必须在包的对应域的最外层（outermost）。

可选行动之设置域（set field）

设置域操作将修改网包中对应域的值。一般的，对同一域的多个包头标签，只修改最外层的。

一般的，当执行 push 操作的时候，VLAN 和 MPLS 头将被复制到新的包头。没有对应域的新域值被设置为 0。通过设置域无法修改的域将被初始化为合适的协议值。push 操作后，新包头中的域可能被设置域操作重写。

第1.4节 OpenFlow 通道

OpenFlow 通道（OpenFlow Channel，即 of1.0 中的安全通道）是用来连接交换机和控制器的接口通道。通过这一接口，控制器可以配置、管理交换机、接收交换机的事件信息，或者将网包发出交换机。

在数据平面（datapath）和 OpenFlow 通道之间，接口可能根据实现而略有不同。但所有的 OpenFlow 通道消息必须遵守 OpenFlow 协议。一般的，该通道可以用 TLS 进行加密，也可以直接运行在 TCP 之上。

关于对多个控制器的支持，目前还没有规定。

1.4.1 OpenFlow 协议概述

OpenFlow 协议支持三种消息类型：*controller-to-switch*（控制器到交换机），*asynchronous*（异步）和 *symmetric*（对称），每一类消息又有多个子消息类型。controller-to-switch 消息由控制器发起，用来直接管理或获取 switch 状态；asynchronous 消息由 switch 发起，用来将网络事件或交换机的状态更新通知控制器；symmetric 消息可由交换机或控制器发起，并且无需请求。

1.4.1.1 controller-to-switch 消息

由控制器（controller）发起，可能需要或不需要来自交换机的应答消息。包括 Features、Configuration、Modify-state、Read-state、Send-packet、Barrier 等。

Features

控制器发送 feature 请求消息给交换机，交换机需要应答自身支持的功能。

Configuration

控制器设置或查询交换机上的配置信息。交换机仅需要应答查询消息。

Modify-state

控制器管理交换机流、组表项和端口状态等。

Read-state

控制器向交换机请求一些诸如流、网包等统计信息。

Packet-out

控制器通过交换机指定端口发出网包。必须包括完整网包或者存储网包的 buffer 的 id。信息必须包括被执行的行动列表。空的行动列表将导致丢弃操作。

Barrier

控制器用以确保消息依赖满足，或接收完成操作的通知。

1.4.1.2 asynchronous 消息

asynchronous 消息不需要控制器请求发起，主要用于交换机向控制器通知包到达，交换机状态变化、错误等事件信息。主要消息包括 Packet-in、Flow-removed、Port-status、Error 等。

Packet-in

交换机收到一个网包，在流表中没有匹配项，则发送 Packet-in 消息给控制器。如果交换机缓存足够多，网包被临时放在缓存中，网包的部分内容（默认 128 字节）和在交换机缓存中的的序号也一同发给控制器；如果交换机缓存不足以存储网包，则将整个网包作为消息的附带内容发给控制器。被缓存的网包一般被控制器的 packet-out 消息处理，或过段时间后自动超时。

Flow-removed

交换机中的流表项通过流修改（flow modify）消息被添加的时候需要制定一个空闲超时（多久无活动则被删除）和硬超时（到期强行删除）。流修改消息同时指定当流超时的时候，交换机是否需要发送删除流信息到控制器。此外，当通过流删除消息来删除流表时，若该流表项设置了 OFPFF_SEND_FLOW_REM 标签，则需要发送流删除消息给控制器。

Port-status

交换机端口配置状态发生变化时（例如 down 掉），交换机发送 Port-status 消息给控制器。

Error

交换机通过 Error 消息来通知控制器发生的错误。

1.4.1.3 symmetric 消息

symmetric 消息也不必通过请求建立，包括 Hello、Echo、Experimenter 等。

Hello

交换机和控制器用来建立连接。

Echo

交换机和控制器均可以向对方发出 Echo 消息，接收者则需要回复 Echo reply。该消息用来测量延迟、带宽、是否连接保持、是否活跃等等。

Experimenter

该消息为交换机提供额外的附加信息功能。为未来版本预留。

1.4.2 连接建立

交换机通过用户指定的地址和端口来通过 TLS 或者 TCP 连接到控制器。所有的 OpenFlow 通道流量不经过 OpenFlow 的管道处理。因此，交换机需要将 OpenFlow 通道中的流量视为本地流量。后续版本将描述动态确认控制器地址和端口的协议。

当 of 连接建立起来后，两边必须先发送 OFPT_HELLO 消息给对方，该消息携带支持的最高协议版本号，接受方将采用双方都支持的最低协议版本进行通信。

经过协商，一旦发现双方共同支持的协议版本，则连接建立；否则发送 OFPT_ERROR 消息（类型域为 OFPET_HELLO_FAILED，代码域为 OFPHFC_COMPATIBLE，并且可以在数据域附加一段 ASCII 串来描述失败原因)，，并终止连接。

1.4.3 连接中断

当连接发生异常时（超时、连接中断等），交换机应尝试连接备份的控制器（尝试顺序在协议中并未规定）。

当尝试失败后，根据实现和配置的不同，交换机将立刻进入失败安全模式（fail secure mode）或失败独立模式（fail standalone mode）。在失败安全模式，所有发往控制器的网包和消息将被丢弃。而流表项则继续正常行为直到超时。在失败独立模式，交换机使用 OFPP_NORMAL 端口来处理所有的网包，即按照传统的交换机或路由器模式运行。

一旦重新连接到控制器，仍存在的流表项将保留，但控制器可以选择选项来选择删除所有流表项。

当交换机刚启动时，默认进入失败安全模式或失败独立模式。规范中对于默认的表项并无规定。

1.4.4 加密

安全通道可以采用 TLS（Transport Layer Security）连接加密。当交换机启动时，尝试连接到控制器的 6633 TCP 端口。双方通过交换证书进行认证。因此，每个交换机至少需配置两个证书，一个是用来认证控制器，一个用来向控制器发出认证（认证交换

机）。

1.4.5 消息处理

OpenFlow 协议提供了可靠的消息发送和处理，但不能自动保证按照顺序的消息处理。

消息发送（Message Delivery）

除非连接完全中断，消息被确保送达。在中断状态下，控制器无法假设关于交换机的任何状态。

消息处理（Message Processing）

交换机必须将从控制器收到的每个消息都进行完整的处理，并可能进行回复。如果无法完整的执行消息，则必须返回一个错误信息。对于 packet-out 消息，完整的消息处理无法保证网包完全离开交换机。交换机拥塞、QoS 限制、发送到失败端口等，可能导致网包可能被默认无声丢弃。

另外，交换机必须向控制器发出所有因内部状态变化导致的异步消息，包括流删除和包到达等。然而，转发到控制器的网包可能因为拥塞、QoS 策略等原因丢弃，并不引发任何包到达消息。

控制器可以任意丢弃消息，但应当通过应答 hello 和 echo 消息来确保交换机不丢弃通道连接。

消息顺序（Message Ordering）

顺序可以通过使用 barrier 消息来确保。当没有给定 barrier 消息时，交换机可以任意重排消息来优化性能，此时，控制器无法控制交换机的某种特定处理顺序。特别的，流被添加到流表的顺序可能跟流修改消息的收到顺序并不一致。然而，对于 barrier 消息，则必须严格按照顺序执行。barrier 之前的消息必须先执行完毕，并发送完毕可能的回复消息或错误消息；barrier 被执行，并且发出回复消息；barrier 后的消息再被继续处理。如果两个消息依赖彼此的先后顺序，需要在它们之间添加 barrier 消息来保序。

1.4.6 流表修改消息

流表修改消息（Flow Table Modification Message）可以有以下类型：

```
enum ofp_flow_mod_command {
OFPFC_ADD, /* New flow. */
OFPFC_MODIFY, /* Modify all matching flows. */
OFPFC_MODIFY_STRICT, /* Modify entry strictly matching wildcards and priority. */
OFPFC_DELETE, /* Delete all matching flows. */
OFPFC_DELETE_STRICT /* Delete entry strictly match wildcards and priority. */
};
```

1.4.6.1 ADD

对于带有 OFPFF_CHECK_OVERLAP 标志的添加（ADD）消息，交换机将先检查

新表项是否跟现有表项冲突（包头范围 overlap，且有相同的优先级），如果发现冲突，将拒绝添加，并返回 ofp_error_msg，并且指明 OFPET_FLOW_MOD_FAILED 类型和 OFPFMFC_OVERLAP 代码。

对于合法无冲突的添加，或不带 OFPFF_CHECK_OVERLAP 标志的添加，交换机必须将表项添加到要求的表中。如果该表中已经存在与新表项相同头部域和优先级的旧表项，则该项，包括计数器和时间（duration）都被清除，然后新的表项被添加。此时，ADD 指令结束，并无须产生流删除消息。如果控制器需要流删除指令，则在添加新表项之前必须对被清除表项发送 DELETE_STRICT。

1.4.6.2 MODIFY

对于修改（OFPFC_MODIFY 或者 OFPFC_MODIFY_STRICT），如果所有已有表中没有与要修改表项同样头部域的表项，并且 cookie-mask 域包含 0，则等同于 ADD 消息，计数器置 0；否则更新现有表项的指令域，同时保持计数器、标志、空闲时间等均不变。

1.4.6.3 DELETE

对于删除（OFPFC_DELETE 或者 OFPFC_DELETE_STRICT），如果没有找到要删除表项，不发出任何消息；如果存在，则进行删除操作。如果被删除的表项带有 OFPFF_SEND_FLOW_REM 标志，则触发一条流删除的消息。

1.4.6.4 其他说明

修改和删除均包括 STRICT 和 non STRICT 版本。对于非_STRICT 版本，通配流表项是激活的，因此，所有匹配消息描述的流表项均受影响（包括包头范围被包含在消息表项中的流表项）。例如，一条指定目标端口为全部的非 STRICT 删除消息会删除某条指定目标端口为 80 的表项。

在_STRICT 版本情况下，表项头跟优先级等都必须严格匹配才执行，即只有严格相同的那条表项会受影响。例如，一条所有域都是通配符的 DELETE_STRICT 消息仅删除给定优先级的某条规则。

此外删除消息还可以支持指定额外的目标组（destination group）或发出端口（out_port）。如果发出端口域包括一个特定的值（非 OFPP_ANY），则在匹配时候将引入限制。这一限制是每条匹配的规则必须直接包括一个 output 行动，并且该行动发送到对应的端口。同样的，目标组（out_group）如果不是 OFPG_ANY，也将引入类似的限制。这两个域在 OFPFC_ADD、OFPFC_MODIFY 或 OFPFC_MODIFY_STRICT 等消息中被忽略。

修改和删除命令还可以通过 cookie 值来进行过滤。如果 cookie_mask 域包含一个非 0 的值，则引发匹配限制。流消息中的 cookie 域值经 cookie_mask 掩码后的值必须跟表项中的 cookie 域值经 cookie_mask 掩码后的值相等，即(flow.cookie &flow mod.cookie mask) == (flow mod.cookie & flow mod.cookie mask)。

1.4.6.4.1 错误消息

部分错误消息总结如表格 一-6。

表格 一-6 流修改错误消息

错误状况	ofp_error_msg 类型	ofp_error_msg 代码
任何未知类型错误	OFPET_FLOW_MOD_FAILED	OFPFMC_UNKNOWN
流修改消息指定了非法的流表或者 0xFF		OFPFMFC_BAD_TABLE_ID
流添加消息，但无足够空间添加新的表项		OFPFMFC_TABLE_FULL
不支持收到的流修改消息	OFPET_BAD_INSTRUCTION	OFPBIC_UNSUP_INST
包括一个跳转表指令（Goto-Table），而下一个表 id 非法		OFPBIC_BAD_TABLE_ID
指令包括写元数据（Write-Metadata），但交换机不支持元数据或其掩码		OFPBIC_UNSUP_METADATA 或 OFPBIC_UNSUP_METADATA_MASK
包括 Experimenter 指令，但交换机不支持		OFPBIC_UNSUP_EXP_INST
指定了某个不支持的域	OFPET_BAD_MATCH	OFPBMC_BAD_FIELD
指定了某个不支持的通配域		OFPBMC_BAD_WILDCARDS
指定了某个任意掩码，但数据链路或者网络地址不支持		OFPBMC_BAD_DL_ADDR_MASK 或 OFPBMC_BAD_NW_ADDR_MASK，都不支持则用前者
指定了无法匹配的值（如大于 4095 的 VLAN ID）		OFPBMC_BAD_VALUE
行动指定了非法端口	OFPET_BAD_ACTION	丢弃或返回 OFPBAC_BAD_OUT_PORT
行动指定了非法组		OFPBAC_BAD_OUT_GROUP
行动指定了非法值		OFPBAC_BAD_ARGUMENT
行动进行了不一致的匹配操作（如去掉 VLAN 头而未找到匹配）		OFPBAC_MATCH_INCONSISTENT

1.4.7 流删除

所有的流表项均包含一个空闲超时（idle_timeout）和硬超时（hard_timeout）。前者非 0，则当该流在给定期限内未被匹配时自动删除。后者非 0，则当流表项被添加后到达期限时被自动删除。此外，控制器可以通过删除命令（OFPFC_DELETE 或 OFPFC_DELETE——STRICT）来使交换机主动删除对应表项。

当流表项被删除时，交换机必须检查表项的 OFPFF_SEND_FLOW_REM 标志，如

果该标志被指定，则交换机必须发送删除消息（removed message）给控制器。每条删除消息包括了一个完整的描述，包括流表项、删除原因（超时还是主动删除）、表项的存活时间和统计信息等。

1.4.8 组表修改消息

组表（Group Table）消息修改包括如下类型。

```
/* Group commands */
enum ofp_group_mod_command {
OFPGC_ADD, /* New group. */
OFPGC_MODIFY, /* Modify all matching groups. */
OFPGC_DELETE, /* Delete all matching groups. */
};
```

每个桶（bucket）的行动集合需要跟流修改规则的要求一致，需要对收到的修改消息进行组检查。如果某个桶中的行动是非法的或不支持的，则返回 ofp_error_msg（类型为 OFPET_BAD_ACTION，代码则同流修改消息，根据错误不同而不同）。

组中可能包括 0 个或者多个桶。没有行动桶的组将不修改网包关联的行动集合。组中的行动桶自身也可以转发到其他的组。例如，一个快速的重路由组可能有两个桶，分别指向某个组。添加修改、删除（无需指定类型，删除所有组需要指定组值为 OFPG_ALL）操作的处理均跟流修改消息类似。一些错误消息格式参见表格 一-7。

表格 一-7 组修改错误消息

错误状况	ofp_error_msg 类型	ofp_error_msg 代码
交换机不支持组的组	OFPET_GROUP_MOD_FAILED	OFPGMFC_CHAINING_UNSUPPORTED
收到的组修改消息可能导致发送循环		OFPGMFC_LOOP
添加请求，对应组 id 已经存在，拒绝添加		OFPGMFC_GROUP_EXISTS
修改请求，对应组不存在		OFPGMFC_UNKNOWN_GROUP
指定组类型非法，拒绝添加		OFPGMFC_INVALID_GROUP
对指定的多个组，交换机不支持非均等的流量分配		OFPGMFC_WEIGHT_UNSUPPORTED
空间不足，无法添加新组项		OFPGMFC_OUT_OF_GROUPS
组的桶数目被限制，无法添加		OFPGMFC_OUT_OF_BUCKETS
不支持在线的配置，无法添加（包括 watch_port 或 watch_group）		OFPGMFC_WATCH_UNSUPPORTED

快速的故障转移组（failover group）支持要求活跃监测（liveness monitoring），来

决策执行特定的行动桶。其他的组类型不要求必须实现该功能。如果交换机不支持活跃检查，则必须拒绝组修改消息并且返回错误。认为活跃的规则包括：

一个端口如果在端口状态中具有 OFPPS_LIVE 标签，则是活跃的。如果端口配置位 OFPPC_PORT_DOWN 指定了端口已经下线，或者 OFPPC_LINK_DOWN 指定链路下线，则该端口必须被指定为非活跃的。

一个桶是活跃的有两种情况。一是 watch_port 不是 OFPP_ANY，且监测的端口是活跃的，或者 watch_group 不是 OFPG_ANY，且所监测的组是活跃的。

一个组被认为是活跃的，只要它至少一个桶是活跃的。

1.4.9 协议实现

包括一些数据结构等。