OpenFlow 交换机规范(概要)

Version 1.3.0 （June 25, 2012）

SDNAP 开放的SDN联合发布博客

N.J.C.H

1 介绍

www.sdnap.com 独家提供。本文档介绍的OpenFlow交换机的要求。规范包括交换机的组件和基本功能，和OpenFlow的协议，通过一个远程控制器来管理一个OpenFlow的交换机。

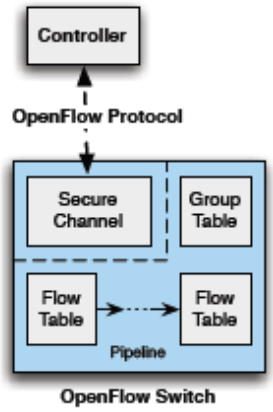

Figure 1: Main components of an OpenFlow switch.

2 交换机组成

OpenFlow 的交换机包括一个或多个流表和一组表，执行分组查找和转发，和到一个外部控制器 OpenFlow 的信道（图 1）。该交换机与控制器进行通信，并通过 OpenFlow 的协议控制器管理的交换机。

控制器使用 OpenFlow 的协议，它可以添加、更新和删除流流表中的表项，既主动或者被动响应数据包。 在交换机中的每个流表中包含的一组流 表项;每个流表项包含匹配字段，计数器和一组指令，用来匹配数据包（见 5.2）。

匹配开始于第一个流程表，并可能会继续额外的流表 （见 5.1）。 流表项匹配数据包按照优先级的顺序，从每个表的第一个匹配表项开始（见 5.3）。如果找到匹配的项，那么具体流表项按照指令去执行。 如果在流表中未找到 匹配项 ，结果取决于漏表的流表项配置:（例如， 数据包可被转发到 OpenFlow 的信道控制器、丢弃、或者可以继续到下一个的流表，见 5.4）。

指令与每个包含行动或修改流水线处理的流表项相联系（见 5.9）。 行动描述了数据包转发，数据包的修改和组表 处理。 流水线处理的指令允许数据包被发送到后面的表进行进一步的 处理 ， 并允许信息以元数据的形式在表之间进行通信。当与一个匹配的流表项相

关联的指令集没有指向下一个表的时候，表流水线处理停止，这时该数据包通常被修改和转发（见 5.10）。

流表项可能包含数据包转发到某个端口。 这通常是一个物理端口，但它也可能是由交换机定义的一个逻辑端口或通过本规范中定义的一个保留的端口（见 4.1）。 保留端口可以指定通用的转发行为，如发送到控制器、泛洪、或使用非 OpenFlow 的方法转发。如 “ 普通”交换机转发处理（见 4.5）；而交换机定义的逻辑端口，可以指定链路汇聚组，隧道或环回接口（见 4.4）。

流表项相关的行动，也可直接把数据包发送到组，进行额外的处理（见 5.6）。 组表示一组泛洪的指令集，以及更复杂的转发（如多路径，快速重路由，链路聚合）。作为间接的通用层，组也 使多个流表项转发到一个单一的标识符（例如一个共同的下一跳的 IP 转发 ）。这种抽象的行为使相同的输出行动非常有效。

组表包含组表项，每个组表项包含了一系列依赖于组类型的特定规范的行动存储段（见 5.6.1）。一个或多个操作的行动用来使数据包发送到该组。

假如将正确的匹配和指令规范保护起来，交换机设计者可以任意的实现内部结构。 例如，如果需要使用一个流表项将所有的组转发到多个端口，交换机设计师可以在硬件转发表中用一个单一的位掩码去实现。 另一个例子是匹配; 如果 OpenFlow 交换机使用用不同数量的硬件表物理实现，那么流水线就会被暴露出来。

3 名词解释

本节介绍了关键 OpenFlow 的规范条款：

• 字节：一个 8 位字节。

• 数据包：以太网帧，包括报头和有效载荷。

• 端口：数据包进入和退出 OpenFlow 的流水线地方（见 4.1）。可以是一个物理端口，由交换机定义一个逻辑端口，或由 OpenFlow 的协议定义一个保留端口。

• 流水线：在一个 openflow 交换机中提供匹配、转发和数据包修改功能的流表连接集合。

• 流表：流水线的一个阶段，包含若干流表项。

• 流表项：在流表中用于匹配和处理数据包的一个元素。它包含用于匹配数据包的匹配字段、匹配次序的优先级，跟踪数据包的计数器，以及对应的的指令集 。

• 匹配字段：用来匹配数据包的字段，包括包头，进入端口，元数据值。匹配字段可能会进行通配符匹配（匹配任何值）或者在某些情况下通过位掩码进行匹配。

• 元数据：一个可屏蔽寄存器的值，用于携带信息从一个表到下一个。

• 指令：指令存在于流表项中，描述报文匹配流表项时 OpenFlow 的处理方式。指令可以修改流水线处理，如指导包匹配另一个流表，也可以包含一系列添加到行动集的行动，还可

以包含一系列立即应用到数据包的行动。

• 行动：将数据包转发到一个端口或修改数据包，如 TTL 字段减 1 操作。行动可能是与流表项相关联的指令集或者与组表项相关联的行动存储段的一部分。我们可以将行动积累在数据包的行动集，也可以立即将行动应用到该数据包。

行动集：与数据包相关的行动集合，在报文被每个表处理的时候这些行动可以累加，在指令集指导报文退出处理流水线的时候这些行动会被执行。

• 组：一系列的行动存储段和一些选择一个或者多个存储段应用到数据包单元的手段。

• 行动存储段：一组行动和相关参数，定义组。

• 标记：一个头，可以插入到数据包或者通过压入和弹出行动进行移除。

• 最外层的标签：一个数据包最开始出现的标签。

• 控制器：一个实体与 OpenFlow 交换机使用 OpenFlow 协议交互的实体。

• 计量： 一个交换机元件，可以测量和控制数据包的速度。当数据包速率或通过计量的字节速率超过预定义的阈值时，计量触发计量带。如果计量带丢弃该数据包，它则被称为一个速率限制器。

4 OpenFlow 端口

本节介绍了 OpenFlow 的端口的抽象概念和 OpenFlow 支持的各类端口。

4.1 OpenFlow 端口

OpenFlow 的端口是 OpenFlow 处理进程和网络的其余部分之间传递数据包的网络接口。OpenFlow 交换机之间通过 OpenFlow 端口在逻辑上相互连接。

OpenFlow 交换机使一些 OpenFlow 的端口，可用于 OpenFlow 的处理。 OpenFlow 的端口组可能与交换机硬件中提供的网络端口不完全相同，因为有些硬件网络接口可能被 OpenFlow 禁用，OpenFlow 交换机也可以定义额外的端口。

OpenFlow 的数据包从入口端口接收，经过 OpenFlow 的流水线处理（见 5.1），可将它们转发到一个输出端口。入端口是数据包的属性，它贯穿了整个 OpenFlow 流水线，并代表数据包是从哪个 OpenFlow 交换机的端口上接收的。匹配报文的时候会用到入端口（见 5.3）。OpenFlow 流水线可以决定数据包通过输出行动发送到输出端口（见 5.12），它定义了数据包怎样传回到网络中。

OpenFlow 交换机必须支持三种类型的 OpenFlow 的端口：物理端口，逻辑端口和保留端口。

4.2 标准端口

OpenFlow 的标准端口为物理端口，逻辑端口，本地保留端口（其他保留的端口除外）。

标准端口可以被用作入口和出端口，它们可用于在组（见 5.6），都有端口计数器（见 5.8）。

4.3 物理端口

OpenFlow 的物理端口为交换机定义的端口，对应于一个交换机的硬件接口。例如，以太网交换机上的物理端口与以太网接口一一对应。

在某些部署中，OpenFlow 交换机可以实现交换机的硬件虚拟化。在这些情况下，一个 OpenFlow 物理端口可以代表一个与交换机硬件接口对应的虚拟切片。

4.4 逻辑端口

OpenFlow 的逻辑端口为交换机定义的端口，并不直接对应一个交换机的硬件接口。逻辑端口是更高层次的抽象概念，可能是交换机中不使用 OpenFlow 的端口（如链路汇聚组，隧道，环回接口）。

逻辑端口可能包括报文封装，可以映射到不同的物理端口。这些逻辑端口的处理动作相对于 openflow 处理来说必须是透明的，而且这些端口必须通过 openflow 处理起作用，像硬件接口一样。

物理端口和逻辑端口之间的唯一区别是：一个逻辑端口的数据包可能有一个叫做隧道 ID 的额外的元数据字段与它相关联；而当一个逻辑端口上接收到的分组被发送到控制器时，其逻辑端口和底层的物理端口都要报告给控制器。

4.5 保留端口

本规范所定义的 OpenFlow 的保留端口。它们指定通用的转发动作，如发送到控制器，泛洪，或使用非 OpenFlow 的方法转发，如“正常”交换机处理。

某个交换机只支持那些标记为“Required”的保留端口，至于“Optional”的端口可以根据需要可选。

• Required: ALL: 表示交换机转发特定数据包到所有端口，它仅可用于为输出端口。在这种情况下，数据包被复制后发送到所有的标准端口，包括数据包的入端口，这些端口被配置 OFPPC_NO_FWD。

• Required: CONTROLLER: 表示的 OpenFlow 控制器的控制通道，它可以用作一个入端口或作为一个出端口。当用作一个出端口，封装数据包中为数据包消息，并使用的 OpenFlow 协议发送（见 A.4.1）。当用作一个入口端口，确认来自控制器的数据包。

• Required: TABLE: 表示 openflow 流水线的开始。这个端口仅在输出行为的时候有效，此时交换机提交报文给第一流表使数据包可以通过定期通过 OpenFlow 流水线处理。

• Required: IN PORT:代表数据包进入端口。用于输出端口时，只允许入端口发送的数据包通过。

• Required: ANY: 特别值，用在未指定端口的 OpenFlow 指令（端口通配符）。不能使用的入口端口，也不作为一个输出端口。

• Optional: LOCAL: 表示交换机的本地网络堆栈和管理堆栈。可以用作一个入口端口或作为一个输出端口。本地端口使远程实体通过 OpenFlow 网络和交换机以及其网络服务互通，而不是通过一个单独的控制网络进行互通。使用一组合适的默认流表项，本地端口可以用来实现一个带内控制器的连接。

• Optional: NORMAL: 代表传统的非 OpenFlow 流水线（见 5.1）。仅可用于为一个输出端口，使用普通的流水线处理数据包。如果交换机不能转发数据包从 OpenFlow 流水线到普通流水线，它必须表明它不支持这一行动。

• Optional: FLOOD: 表示使用普通流水线处理进行泛洪（见 5.1）。可用于作为一个输出端口，一般可以讲数据包发往所有标准端口，但不能发往入端口或 OFPPS_BLOCKED 状态的端口。交换机也可以通过数据包的 VLAN ID 选择哪些端口泛洪。

只有 OpenFlow-only 交换机不支持 NORMAL 端口和 FLOOD 端口，而 OpenFlow-hybrid 交换机均支持上述端口（见 5.1）。转发数据包到 FLOOD 端口依赖交换机上的实现和配置，而使用一组类型进行转发可以使控制器能够更灵活地实现泛洪（见 5.6.1）。

5 OpenFlow 表

本节描述流表和组表的组件，以及与匹配和行动处理的技术。

5.1 流水线处理

OpenFlow 兼容的交换机有两种类型：OpenFlow-only 和 OpenFlow-hybrid。 OpenFlow-only 交换机只支持 OpenFlow 操作，在这些交换机中的所有数据包都由 OpenFlow 流水线处理，否则不能被处理。

OpenFlow-hybrid 交换机支持 OpenFlow 的操作和普通的以太网交换操作，即传统的 L2 以太网交换，VLAN 隔离，L3 路由（IPv4 的路由，IPv6 路由），ACL 和 QoS 处理。这些交换机提供一个交换机外的分类机制，使流量路由到 OpenFlow 流水线或普通流水线。例如，某个交换机可以使用 VLAN 标签或数据包的输入端口，来决定是否使用一个流水线或其他流水线，或者它可指导所有数据包都到 OpenFlow 流水线进行处理。

这种分类机制是本规范的范围之外。 一个 OpenFlow-hybrid 交换机也允许数据包通过 NORMAL 或者 FLOOD 的保留端口从 OpenFlow 流水线到普通流水线处理（见 4.5）。

每个 OpenFlow 交换机的流水线包含多个流表，每个流表包含多个流表项。 OpenFlow 的流水线处理定义了数据包如何与那些流表进行交互（参见图 2）。 OpenFlow 交换机需要具有流表中的至少一个，并可以有更多的可选择的流表。只有一个单一的流表的 OpenFlow 交换机是有效的，而且在这种情况下流水线处理进程可以大大简化。

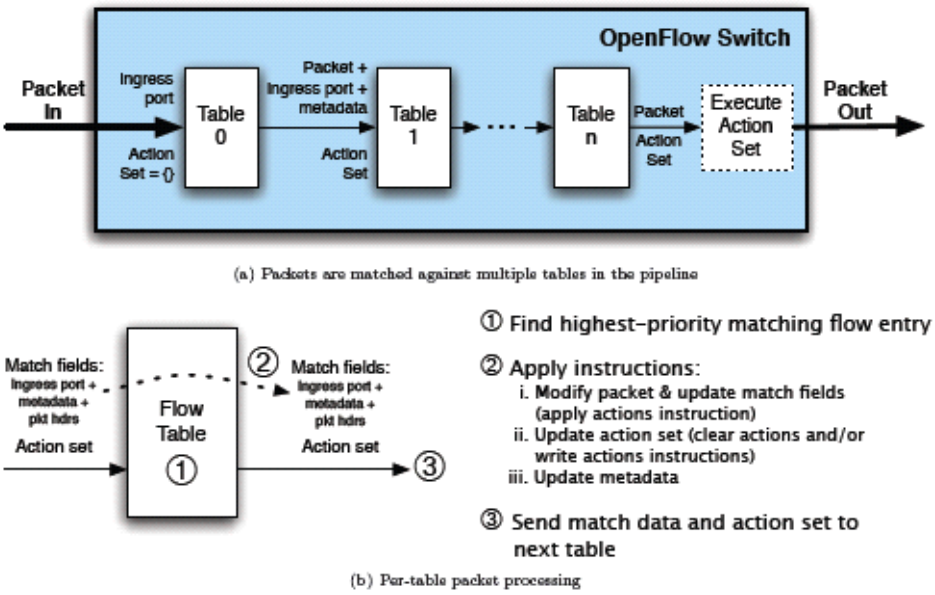

Figure 2: 通过处理了流水线的数据包流

OpenFlow 交换机的流表按顺序编号的，从 0 开始。流水线处理总是从第一流表开始：数据包第一个与流表 0 的流表项匹配。其他流表根据第一个表的匹配结果来调用。

根据某个流表进行处理时，将数据包与流表中的流表项进行匹配，从而选择流表项（见 5.3）。如果匹配到了流表项，那么包括在该流表项的指令集被执行时，这些指令可能明确指导数据包传递到另一个流表（使用 Goto 指令，见 5.9），在那里同样的处理被重复执行。表项只能指导数据包到大于自己表号的流表，换句话说流水线处理，只能前进，而不能后退。显然，流水线的最后一个表项可以不包括 GOTO 指令。如果匹配的流表项并没有指导数据包到另一个流表，流水线处理将停止在该表中。当流水线处理停止，数据包被与之相关的行动集处理并通常被转发（见 5.10）。...

如果数据包在流表中没有匹配到流表项，这是一个 table-miss 的行为。table-miss 行为依赖于表的配置（见 5.4）。一个 table-miss 的流表中的表项可以指定如何处理无法匹配的数据包：包括丢弃，传递到另一个表中，或凭借数据包中的信息通过控制通道发送到控制器（见 6.1.2）。

5.2 流表

一个流表中包含多个流表项。

每个流表项包含：

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie

Table 1: Main components of a flow entry in a flow table.

•匹配字段：对数据包匹配。包括入口端口和数据包报头，以及由前一个表指定的可选的元数据。

•优先级：流表项的匹配次序

•计数器：更新匹配数据包的计数

•指令：修改行动集或流水线处理

•超时：最大时间计数或流有效时间

•cookie：由控制器选择的不透明数据值。控制器用来过滤流统计数据、流改变和流删除。但处理数据包时不能使用。

流表项通过匹配字段和优先级决定，在一个流表中匹配字段和优先级共同确定唯一的流表项。所有字段通配（所有字段省略）和优先级等于 0 的流表项被称为 table-miss 流表项（见 5.4）。

5.3 匹配

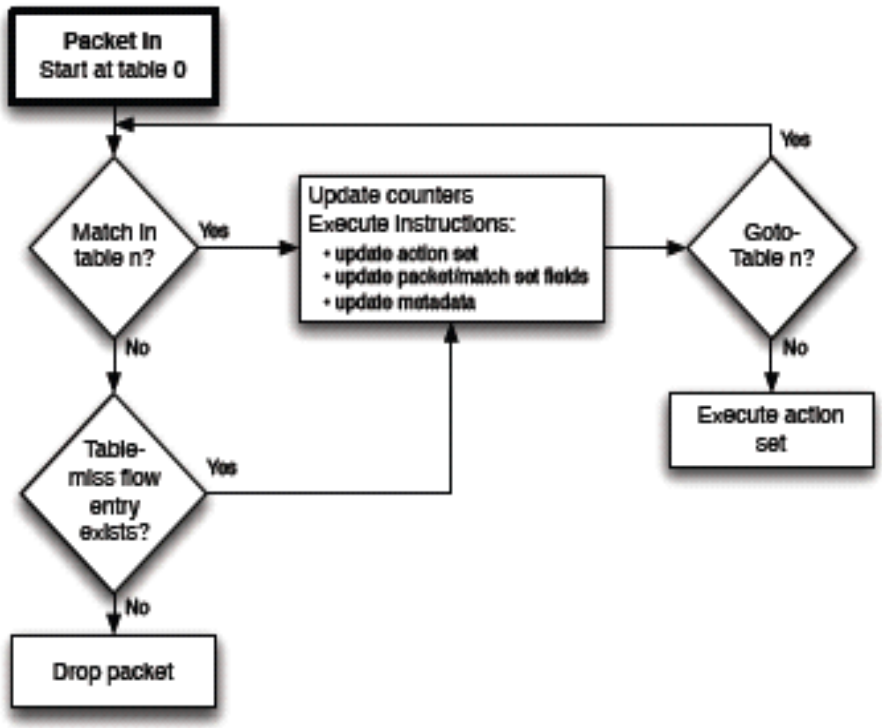

Figure 3:流程图详细描述了数据包流通过一个 OpenFlow 交换机。

OpenFlow 交换机在接收一个数据包时，执行在图 3 中所示的功能。交换机开始执行一个查找表中的第 1 流表，并基于流水线处理，也可能在其它流表中（见 5.1）执行表查找。

数据匹配字段从数据包中提取。用于表查找的数据包匹配字段依赖与数据包类型，这些类型通常包括各种数据包的报头字段，如以太网源地址或 IPv4 目的地址（见 A.2.3）。除了通过数据包报头中进行匹配，也可以通过入口端口和元数据字段进行匹配。元数据可以用来在一个交换机的不同表里面传递信息。报文匹配字段表示报文的当前状态，如果在前一个表中使用 Apply-Actions 改变了数据包的报头，那么这些变化也会在数据包匹配字段中反映。

数据包匹配字段中的值用于查找匹配的流表项。如果流表项字段具有值的 ANY（字段省略），它就可以匹配包头中的所有可能的值。如果交换机支持任意的的位掩码对特定的匹配字段，这些掩码可以更精确地进行匹配。

数据包与表进行匹配，优先级最高的表项必须被选择，此时与选择流表项相关的计数器也会被更新，选定流表项的指令集也被执行。如果有多个匹配的流表项具有相同的最高的优先级的，所选择的流表项被确定为未定义表项。只有控制器记录器在传统的流信息中没有设置 OFPFF_CHECK_OVERLAP 位并且增加了重复的表项的时候，这种情况才能出现。

如果在流水线处理前，交换机配置包含 OFPC_FRAG_REASM 标志（见 A.3.2），IP 碎片必须被重新组装。

当交换机接收到一个格式不正确或损坏的数据包，此版本的规范没有定义预期的行为。

5.4 Table-miss

每一个流表必须支持能处理 table-miss 的流表项。table-miss 表项指定在流表中如何处理与其他流表项未匹配的数据包（见 5.1）。比如数据包发送到控制器，丢弃数据包或直接将包扔到后续的表。

table-miss 的流表项也有它的匹配字段和优先级（见 5.2），它通配所有匹配字段（所有领域省略），并具有最低的优先级（0）。table-miss 流表项的匹配可能不属于正常范围内流表支持的匹配，例如精确匹配表可能不支持在其他流表项中使用通配符，但必须支持 table-miss 的通配符流表项。table-miss 流表项可能没有与正常流表项（见 A.3.5.5）相同的能力。交换机最好支持和 OpenFlow 的以前版本的处理能力相同的 table-miss 流表项：将数据包发送到控制器，丢弃数据包或直接包到后续的表。

table-miss 表项的行为在许多方面像任何其他流表项：默认情况下，在流表中不存在 table-miss 表项。控制器可以在任何时候添加或删除它（见 6.4),而 且它可能会超时失效（见 5.5）。table-miss 流表项可以匹配流表中其他表项中不能匹配的数据。当数据包与 table-miss 表项匹配时，table-miss 表项指令就会执行（见 5.9）。如果该 table-miss 表项直接将数据包通过 CONTROLLER 端口发送到控制器（见 4.5），那么报文中的信息必须与一个 table-miss 表项匹配（见 A.4.1）。

如果该 table-miss 表项不存在，默认情况下，流表项无法的数据包将被丢弃（丢弃）。一个交换机的配置，例如使用 OpenFlow 的配置协议，可以覆盖此默认值，指定其他行为。

5.5 流表项删除

流表项可以通过两种方式在流表中删除，控制器的请求或交换机流超时机制。

交换流超时机制运行基于相关的控制器和流表项的状态和配置。每个流的表项具有一个和它相关的 idle_timeout 和 hard_timeout 值。如果两个值中有一个不为零，交换机必须注意的流表项的老化时间，因为交换机可能删除该项。如果给定非零 hard_timeout 的值，那么一段时间后，可以导致流表项被删除，无论有多少数据包与之匹配。如果给定非零 idle_timeout 的值，那么如果在一段时间没有报文与之匹配，可以导致流表项被删除。交换机必须实现流表项超时和删除功能。

该控制器可积极的从流表中通过发送流表修改信息（OFPFC_DELETE，或 OFPFC_DELETE_STRICT - 见 6.4）删除流表项。流表项被删除时，无论是控制器控制或流表项超时机制，交换机必须检查流表项的 OFPFF_SEND_FLOW_REM 标志。如果该标志被设置，该交换机必须将流删除消息发送到控制器。每个流清除消息中包含的流表项的完整的描述、清除的原因（超时或删除），在清除时的流表项的持续时间，在清除时的流的统计数据。

5.6 组表

组表包括若干组表项。这是另外的 openflow 转发方法，就是若干流表项指向一组（例如选择部分和所有）。

Group Identifier	Group Type	Counters	Action Buckets

Table 2: Main components of a group entry in the group table.

（见表 2）每个组表项由组编号确定，具体内容包含：

•组编号：一个 32 位的无符号整数，唯一标识该组

•组类型：确定组语义（参见 5.6.1 节）

•计数器：更新数据当报文被组表项处理时

•行动存储段：一系列有序的行动存储段，其中的每个动作存储段包含了一组要执行的动作和相关参数

5.6.1 Group Types

某个交换机只支持那些标记为“Required”的组类型，至于“Optional”的组类型可以根据需要可选。

• Required: all: 执行组中的所有存储段。这个组用于多播或广播的转发。数据包为每个存储段都有效地复制一份，然后被每个存储段处理。如果某个存储段中明确地指导数据包发往入端口，那么这个复制的包被丢弃。如果控制器希望数据包转发到入端口，那么这个组需要包含一个额外的存储段，这个存储段含有到 OFPP_IN_PORT 保留端口的输出行动。

• Optional: select: 执行组中的一个存储段。报文基于一个计算交换机选择算法（如一些用户配置数组的哈希算法或简单的循环算法）被组中的一个存储段处理。所有的配置和状态的选择算法都在 OpenFlow 外部运行。选择算法实现可以使用等负荷分配，页可以有选择地根据存储段的权重进行。当存储段中指定的端口出现故障时，交换机可以选择剩余的存储段（生存端口的转发行为），而不是丢弃数据包。此行为可能会减少报文在链路或交换机的中断。

• Required: indirect: 执行此组中定义的一个存储段。这个组只支持单一的存储段。允许多个流表项或者组指向一个共同的组编号，这样可以使转发更快，更高效的聚集（例如：下一跳 IP 转发）。对于所有组的这个存储段，组类型应该是相同的。

• Optional: fast failover: 执行第一个活跃的存储段。每一个动作存储段与一个特定的端口和/或组，这些端口或者组控制其活跃性。存储段的顺序由组定义，并且第一个活跃的存储段对应的端口或者组也是活跃的。这个组类型可以直接使交换机改变转发，而无需和控制器联系。如果没有存储段是活的，数据包将被丢弃。这组的类型必须实行活跃机制（见 6.5）。

5.7 计量表

一个计量表包含若干计量表项，确定每个流量的计数。单位流量的计量可以使 OpenFlow 实现各种简单的 QoS 业务，如限速，并且可以结合每个端口队列（见 5.12）来实现复杂的 QoS 框架，如 DiffServ。

计量可以测试数据包的速率，使这些数据包可以实现速率控制。计量直接连接到流表项（而不是被连接到端口的队列）。任意的流表项可以在它的指令集中定义一个计量（见 5.9），计量测量和控制和它相连的所有流的速率。在同一个表中可以使用多个计量，但必须使用独有的方式（分离的流表项）。多个计量也可在用在连续的流表中，针对数据包集合。

Meter Identifier	Meter Bands	Counters

Table 3: Main components of a meter entry in the meter table.

每计量表项（见表 3）由其计量标识符标识，并且包含：

•计量的标识符：一个 32 位的无符号整数唯一识别仪

•计量带：计量带的无序列表，其中每个计量带定义带的速度和处理数据包的方式

•计数器：报文被计量表项处理时，更新计数

5.7.1 Meter Bands

每个计量可能有一个或多个计量带。每个带指定带适用的速率和并数据被处理的方式。数据包基于当前的计量速率被单个计量带处理，计量带适用于当前速率超过配置的速率的情况，如果目前的速度比任何指定的计量带率较低，那么计量带就不工作。
每个计量带（见表 4）用速率识别，包括：

Band Type	Rate	Counters	Type specific arguments

Table 4: Main components of a meter band in a meter entry.

•带类型：定义了数据包怎样被处理

•按计量率：用于选择计量带，定义了带可以运行的最低速率

•计数器：当数据报文被计量带处理时时，更新计数

•类型的具体参数：带类型的可选参数

本规范没有“Required”的带类型。控制器可以查询交换机支持的“Optional”计量带型

• Optional: drop:丢弃数据包。可以用来定义的速率限制带。

• Optional: dscp remark:减少数据包的 IP 头中的 DSCP 字段丢弃的优先级。可用于定义一个简单的 DiffServ 策略。

5.8 计数器

维护计数器可以有如下计数：每一个流程表，流量入口，端口，队列，组，组存储段，计量和计量带。 用来统计流量的一些信息，例如活动表项、查找次数、发送包数等。也可以OpenFlow-compliant计数器可以在软件中实现，而且可以通过硬件计数器获取计数进行维护。表5中包含了openflow规范中定义的计数器集。交换机不一定支持所有的计数器，只有那些标记为“Required”是必须支持的。

Counter	**Bits**	
Per Flow Table		
Reference count (active entries)	32	*Required*
Packet Lookups	64	*Optional*
Packet Matches	64	*Optional*
Per Flow Entry		
Received Packets	64	*Optional*
Received Bytes	64	*Optional*
Duration (seconds)	32	*Required*
Duration (nanoseconds)	32	*Optional*
Per Port		
Received Packets	64	*Required*
Transmitted Packets	64	*Required*
Received Bytes	64	*Optional*
Transmitted Bytes	64	*Optional*
Receive Drops	64	*Optional*
Transmit Drops	64	*Optional*
Receive Errors	64	*Optional*
Transmit Errors	64	*Optional*
Receive Frame Alignment Errors	64	*Optional*
Receive Overrun Errors	64	*Optional*
Receive CRC Errors	64	*Optional*
Collisions	64	*Optional*
Duration (seconds)	32	*Required*
Duration (nanoseconds)	32	*Optional*
Per Queue		
Transmit Packets	64	*Required*
Transmit Bytes	64	*Optional*
Transmit Overrun Errors	64	*Optional*
Duration (seconds)	32	*Required*
Duration (nanoseconds)	32	*Optional*
Per Group		
Reference Count (flow entries)	32	*Optional*
Packet Count	64	*Optional*
Byte Count	64	*Optional*
Duration (seconds)	32	*Required*
Duration (nanoseconds)	32	*Optional*
Per Group Bucket		
Packet Count	64	*Optional*
Byte Count	64	*Optional*
Per Meter		
Flow Count	32	*Optional*
Input Packet Count	64	*Optional*
Input Byte Count	64	*Optional*
Duration (seconds)	32	*Required*
Duration (nanoseconds)	32	*Optional*
Per Meter Band		
In Band Packet Count	64	*Optional*
In Band Byte Count	64	*Optional*

Table 5: List of counters

5.9 指令

每个流表项中包含一组的指令，当一个数据包匹配表项时指令会被执行。这些指令可以更改数据包，行动组和/或流水线处理。

• Optional Instruction: Meter meter id : 直接将包计量后丢弃。

• Optional Instruction: Apply-Actions action(s): 立即进行指定的行动，而不改变行动集。这个指令经常用来修改数据包，在两个表之间或者执行同类型的多个行动的时候。

• Optional Instruction: Clear-Actions: 在行动集中立即清除所有的行动。

• Required Instruction: Write-Actions action(s): 将指定的行动添加到正在运行的行动集中。

• Optional Instruction: Write-Metadata metadata / mask : 在元数据区域记录元数据。

• Required Instruction: Goto-Table next-table-id: 指定流水线处理进程中的下一张表的 ID。

5.10 行动集

行动集与每个报文相关，默认情况下是空的。一个流表项可以使用 Write-Action 指令或者 Clear-Action 指令修改行动集。行动集在表间被累加。当一个表项的指令集没有包含 Goto-Table 指令时，流水线处理就停止了，然后报文的行动集就被执行。

行动集包含所有的行动，无论他们以什么顺序加入到行动集中，行动的顺序均按照下列顺序执行。如果行动集包含组行动，那么组行动存储段中的行动也按照下列顺序执行。当然，交换机也可以支持通过 Apply-Actions 指令修改行动执行顺序。

1. copy TTL inwards: apply copy TTL inward actions to the packet

2. pop: apply all tag pop actions to the packet

3. push-MPLS: apply MPLS tag push action to the packet

4. push-PBB: apply PBB tag push action to the packet

5. push-VLAN: apply VLAN tag push action to the packet

6. copy TTL outwards: apply copy TTL outwards action to the packet

7. decrement TTL: apply decrement TTL action to the packet

8. set: apply all set-field actions to the packet

9. qos: apply all QoS actions, such as set queue to the packet

10. group: 如果指定了组行动，那么按照这个序列中的顺序执行组行动存储段里的行动。

11. output: 如果没有指定组行动，报文就会按照 output 行动中指定的端口转发。

Output 行动最后执行。如果组行动和输出行动均存在，那么组行动优先级高。如果两者均不存在，那么报文被丢弃。

5.11 行动列表

Apply-Actions 指令和 Packet-out 消息中存在行动列表。行动的效果是被累积的，也就是这些行动全部都会执行的。比如行动列表中有两个行动都是 push VLAN，那么行动后，报文会被加上两个 VLAN tag。

5.12 行动

Required Action: Output. 报文输出到指定端口

Optional Action: Set-Queue.设置报文的队列 ID，为了 Qos 的需要。

Required Action: Drop. 丢弃。

Required Action: Group. 用组表处理报文.

Optional Action: Push-Tag/Pop-Tag.

Optional Action: Set-Field. 设置报文包头的类型和修改包头的值。

Optional Action: Change-TTL.修改 TTL 值。

6 OpenFlow 通道

Openflow 通道是交换机连接控制器的接口。通过这个接口，控制器可以对交换机进行管理和配置，接收到交换机的事件，并且想交换机发送数据包。

消息被封装为 openflow 协议中规定的格式在控制器和交换机之间传输，这个控制器—交换机协议运行在安全传输层协议(TLS)或无保护 TCP 连接之上。

6.1 OpenFlow 协议介绍

OF 协议支持三种消息类型：controller-to-switch，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状态变化更新到控制器；symmetric 消息可由交换机或控制器发起。

controller-to-switch

Features 在建立传输层安全会话（Transport Layer Security Session）的时候，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。

Barrier 控制器确保消息依赖满足，或接收完成操作的通知

asynchronous

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。

Flow-removed 交换机中的流表项因为超时或修改等原因被删除掉，会触发 Flow-removed 消息。

Port-status 交换机端口状态发生变化时（例如 down 掉），触发 Port-status 消息。

Error 交换机发生问题时触发消息。

Symmetric

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、是否连接保持等。

Vendor 交换机提供额外的附加信息功能。为未来版本预留。

6.2 连接建立

通过安全通道建立连接，所有流量都不经过交换机流表检查。因此交换机必须将安全通道认为是本地链接。今后版本中将介绍动态发现控制器的协议。

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。一旦发现共同支持的协议版本，则连接建立，否则发送 OFPT_ERROR 消息，描述失败原因，并中断连接。

6.3 连接中断

当连接发生异常时，交换机应尝试连接备份的控制器。当多次尝试均失败后，交换机将进入紧急模式，并重置所有的 TCP 连接。此时，所有包将匹配指定的紧急模式表项，其他所有正常表项将从流表中删除。此外，当交换机刚启动时，默认进入紧急模式。

6.4 加密

安全通道采用 TLS 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。

6.5 多控制器

交换机可能与一台控制器建立通信，或者与多个控制器建立通信。避免单台控制器和交换机中断连接，我们可以使用多台控制器增加可靠性。多台控制器通过规范外的机制协调控制交换机。

附录：版本间差别(1.0 与 1.1 1.2 1.3 之间)

1. OpenFlow version 1.1

Release date : February 28, 2011

1 Multiple Tables

Prior versions of the OpenFlow specification did expose to the controller the abstraction of a single table. The OpenFlow pipeline could internally be mapped to multiple tables, such as having a separate wildcard and exact match table, but those tables would always act logically as a single table.

OpenFlow 1.1 introduces a more flexible pipeline with multiple tables. Exposing multiple tables has many advantages. The first advantage is that many hardware have multiple tables internally (for example L2 table, L3 table, multiple TCAM lookups), and the multiple table support of OpenFlow may enable to expose this hardware with greater efficiency and flexibility. The second advantage is that many network deployments combine orthogonal processing of packets (for example ACL, QoS and routing), forcing all those processing in a single table creates huge ruleset due to the cross product of individual rules, multiple table may decouple properly those processing.

The new OpenFlow pipeline with multiple table is quite different from the simple pipeline of prior OpenFlow versions. The new OpenFlow pipeline expose a set of completely generic tables, supporting the full match and full set of actions. It's difficult to build a pipeline abstraction that represent accurately all possible hardware, therefore OpenFlow 1.1 is based on a generic and flexible pipeline that may be mapped to the hardware. Some limited table capabilities are available to denote what each table is capable of supporting.

Packet are processed through the pipeline, they are matched and processed in the first table, and may be matched and processed in other tables. As it goes through the pipeline, a packet is associated with an action set, accumulating action, and a generic metadata register. The action set is resolved at the end of the pipeline and applied to the packet. The metadata can be matched and written at each table and enables to carry state between tables.

OpenFlow introduces a new protocol object called instruction to control pipeline processing. Actions which were directly attached to flows in previous versions are now encapsulated in instructions, instructions may apply those actions between tables or accumulate them in the packet action set. Instructions can also change the metadata, or direct packet to another table.

• The switch now expose a pipeline with multiple tables.

• Flow entry have instruction to control pipeline processing
• Controller can choose packet traversal of tables via goto instruction
• Metadata field (64 bits) can be set and match in tables
• Packet actions can be merged in packet action set
• Packet action set is executed at the end of pipeline
• Packet actions can be applied between table stages
• Table miss can send to controller, continue to next table or drop
• Rudimentary table capability and configuration

2 Groups

The new group abstraction enables OpenFlow to represent a set of ports as a single entity for forwarding packets. Different types of groups are provided, to represent different abstractions such as multicasting or multipathing. Each group is composed of a set group buckets, each group bucket contains the set of actions to be applied before forwarding to the port. Groups buckets can also forward to other groups, enabling to chain groups together.

• Group indirection to represent a set of ports
• Group table with 4 types of groups :
- All - used for multicast and flooding
- Select - used for multipath
- Indirect - simple indirection
- Fast Failover - use first live port
• Group action to direct a flow to a group
• Group buckets contains actions related to the individual port

3 Tags : MPLS & VLAN

Prior versions of the OpenFlow specification had limited VLAN support, it only supported a single level of VLAN tagging with ambiguous semantic. The new tagging support has explicit actions to add, modify and remove VLAN tags, and can support multiple level of VLAN tagging. It also adds similar support the MPLS shim headers.

• Support for VLAN and QinQ, adding, modifying and removing VLAN headers
• Support for MPLS, adding, modifying and removing MPLS shim headers

4 Virtual ports

Prior versions of the OpenFlow specification assumed that all the ports of the OpenFlow switch were physical ports. This version of the specification add support for virtual ports, which can represent complex forwarding abstractions such as LAGs or tunnels.

• Make port number 32 bits, enable larger number of ports
• Enable switch to provide virtual port as OpenFlow ports

• Augment packet-in to report both virtual and physical ports

5 Controller connection failure

Prior versions of the OpenFlow specification introduced the emergency flow cache as a way to deal with the loss of connectivity with the controller. The emergency flow cache feature was removed in this version of the specification, due to the lack of adoption, the complexity to implement it and other issues with the feature semantic.

This version of the specification add two simpler modes to deal with the loss of connectivity with the controller. In fail secure mode, the switch continues operating in OpenFlow mode, until it reconnects to a controller. In fail standalone mode, the switch revert to using normal processing (Ethernet switching).

• Remove Emergency Flow Cache from spec
• Connection interruption trigger fail secure or fail standalone mode

6 Other changes

• Remove 802.1d-specific text from the specification
• Cookie Enhancements Proposal - cookie mask for filtering
• Set queue action (unbundled from output port action)
• Maskable DL and NW address match fields
• Add TTL decrement, set and copy actions for IPv4 and MPLS
• SCTP header matching and rewriting support
• Set ECN action
• Define message handling : no loss, may reorder if no barrier
• Rename VENDOR APIs to EXPERIMENTER APIs
• Many other bug fixes, rewording and clarifications

2. OpenFlow version 1.2

Release date : December 5, 2011

Please refers to the bug tracking ID for more details on each change

1 Extensible match support

Prior versions of the OpenFlow specification used a static fixed length structure to specify ofp_match, which prevents flexible expression of matches and prevents inclusion of new match fields. The ofp_match has been changed to a TLV structure, called OpenFlow Extensible Match (OXM), which dramatically increases flexibility.

The match fields themselves have been reorganised. In the previous static

structure, many fields were overloaded ; for example tcp.src_port, udp.src_port, and icmp.code were using the same field entry. Now, every logical field has its own unique type.

List of features for OpenFlow Extensible Match :

- Flexible and compact TLV structure called OXM (EXT-1)
- Enable flexible expression of match, and flexible bitmasking (EXT-1)
- Pre-requisite system to insure consistency of match (EXT-1)
- Give every match field a unique type, remove overloading (EXT-1)
- Modify VLAN matching to be more flexible (EXT-26)
- Add vendor classes and experimenter matches (EXT-42)
- Allow switches to override match requirements (EXT-56, EXT-33)

2 Extensible ' set field' packet rewriting support

Prior versions of the OpenFlow specification were using hand-crafted actions to rewrite header fields. The Extensible set_field action reuses the OXM encoding defined for matches, and enables to rewrite any header field in a single action (EXT-13). This allows any new match field, including experimenter fields, to be available for rewrite. This makes the specification cleaner and eases cost of introducing new fields.

- Deprecate most header rewrite actions
- Introduce generic set-field action (EXT-13)
- Reuse match TLV structure (OXM) in set-field action

3 Extensible context expression in ' packet-in'

The packet-in message did include some of the packet context (ingress port), but not all (metadata), preventing the controller from figuring how match did happen in the table and which flow entries would match or not match. Rather than introduce a hard coded field in the packet-in message, the flexible OXM encoding is used to carry packet context.

- Reuse match TLV structure (OXM) to describe metadata in packet-in (EXT-6)
- Include the ' metadata' field in packet-in
- Move ingress port and physical port from static field to OXM encoding
- Allow to optionally include packet header fields in TLV structure

4 Extensible Error messages via experimenter error type

An experimenter error code has been added, enabling experimenter functionality to generate custom error messages (EXT-2). The format is identical to other experimenter APIs.

5 IPv6 support added

Basic support for IPv6 match and header rewrite has been added, via the Flexible match support.

• Added support for matching on IPv6 source address, destination address, protocol number, traffic class, ICMPv6 type, ICMPv6 code and IPv6 neighbor discovery header fields (EXT-1)
• Added support for matching on IPv6 flow label (EXT-36)

6 Simplified behaviour of flow-mod request

The behaviour of flow-mod request has been simplified (EXT-30).

• MODIFY and MODIFY STRICT commands never insert new flows in the table
• New flag OFPFF RESET COUNTS to control counter reset
• Remove quirky behaviour for cookie field.

7 Removed packet parsing specification

The OpenFlow specification no longer attempts to define how to parse packets (EXT-3). The match fields are only defined logically.

• OpenFlow does not mandate how to parse packets
• Parsing consistency acheived via OXM pre-requisite

8 Controller role change mechanism

The controller role change mechanism is a simple mechanism to support multiple controllers for failover (EXT-39). This scheme is entirely driven by the controllers ; the switch only need to remember the role of each controller to help the controller election mechanism.

• Simple mechanism to support multiple controllers for failover
• Switches may now connect to multiple controllers in parallel
• Enable each controller to change its roles to equal, master or slave

9 Other changes

• Per-table metadata bitmask capabilities (EXT-34)
• Rudimentary group capabilities (EXT-61)
• Add hard timeout info in flow-removed messages (OFP-283)
• Add ability for controller to detect STP support(OFP-285)
• Turn off packet buffering with OFPCML NO BUFFER (EXT-45)
• Added ability to query all queues (EXT-15)
• Added experimenter queue property (EXT-16)
• Added max-rate queue property (EXT-21)
• Enable deleting flow in all tables (EXT-10)
• Enable switch to check chaining when deleting groups (EXT-12)
• Enable controller to disable buffering (EXT-45)

• Virtual ports renamed logical ports (EXT-78)
• New error messages (EXT-1, EXT-2, EXT-12, EXT-13, EXT-39, EXT-74 and EXT-82)
• Include release notes into the specification document
• Many other bug fixes, rewording and clarifications

3. OpenFlow version 1.3

Release date : April 13, 2012

Please refers to the bug tracking ID for more details on each change

1 Refactor capabilities negotiation
Prior versions of the OpenFlow specification included limited expression of the capabilities of an OpenFlow switch. OpenFlow 1.3 include a more flexible framework to express capabilities (EXT-123).

The main change is the improved description of table capabilities. Those capabilities have been moved out of the table statistics structure in its own request/reply message, and encoded using a flexible TLV format. This enables the additions of next-table capabilities, table-miss flow entry capabilities and experimenter capabilities.

Other changes include renaming the 'stats' framework into the 'multipart' framework to reflect the fact that it is now used for both statistics and capabilities, and the move of port descriptions into its own multipart message to enable support of a greater number of ports.

List of features for Refactor capabilities negotiation :

• Rename 'stats' framework into the 'multipart' framework.
• Enable 'multipart' requests (requests spanning multiple messages).
• Move port list description to its own multipart request/reply.
• Move table capabilities to its own multipart request/reply.
• Create flexible property structure to express table capabilities.
• Enable to express experimenter capabilities.
• Add capabilities for table-miss flow entries.
• Add next-table (i.e. goto) capabilities

2 More flexible table miss support
Prior versions of the OpenFlow specification included table configuration flags to select one of three 3 behaviour for handling table-misses (packet not matching any flows in the table). OpenFlow 1.3 replace those limited flags with the table-miss flow entry, a special flow entry describing the behaviour on table miss (EXT-108).

The table-miss flow entry uses standard OpenFlow instructions and actions to process table-miss packets, this enables to use the full flexibility of OpenFlow in processing those packets. All previous behaviour expressed by the table-miss config flags can be expressed using the table-miss flow entry. Many new way of handling table-miss, such as processing table-miss with normal, can now trivially be described by the OpenFlow protocol.

- Remove table-miss config flags (EXT-108).
- Define table-miss flow entry as the all wildcard, lowest priority flow entry (EXT-108).
- Mandate support of the table-miss flow entry in every table to process table-miss packets (EXT-108).
- Add capabilities to describe the table-miss flow entry (EXT-123).
- Change table-miss default to drop packets (EXT-119).

3 IPv6 Extension Header handling support

Add the ability of match the presence of common IPv6 extension headers, and some anomalous conditions in IPv6 extension headers (EXT-38). A new OXM pseudo header field OXM_OF_IPV6_EXTHDR enables to match the following conditions :

- Hop-by-hop IPv6 extension header is present.
- Router IPv6 extension header is present.
- Fragmentation IPv6 extension header is present.
- Destination options IPv6 extension headers is present.
- Authentication IPv6 extension header is present.
- Encrypted Security Payload IPv6 extension header is present.
- No Next Header IPv6 extension header is present.
- IPv6 extension headers out of preferred order.
- Unexpected IPv6 extension header encountered.

4 Per flow meters

Add support for per-flow meters (EXT-14). Per-flow meters can be attached to flow entries and can measure and control the rate of packets. One of the main applications of per-flow meters is to rate limit packets sent to the controller. The per-flow meter feature is based on a new flexible meter framework, which includes the ability to describe complex meters through the use of multiple metering bands, metering statistics and capabilities.
Currently, only simple rate-limiter meters are defined over this framework. Support for color-aware meters, which support Diff-Serv style operation and are tightly integrated in the pipeline, was postponed to a later release.

- Flexible meter framework based on per-flow meters and meter bands.
- Meter statistics, including per band statistics.

• Enable to attach meters flexibly to flow entries.
• Simple rate-limiter support (drop packets).

5 Per connection event filtering

Previous version of the specification introduced the ability for a switch to connect to multiple controllers for fault tolerance and load balancing. Per connection event filtering improves the multi-controller support by enabling each controller to filter events from the switch it does not want (EXT-120).

A new set of OpenFlow messages enables a controller to configure an event filter on its own connection to the switch. Asynchronous messages can be filtered by type and reason. This event filter comes in addition to other existing mechanisms that enable or disable asynchronous messages, for example the generation of flow-removed events can be configured per flow. Each controller can have a separate filter for the slave role and the master/equal role.

• Add asynchronous message filter for each controller connection.
• Controller message to set/get the asynchronous message filter.
• Set default filter value to match OpenFlow 1.2 behaviour.
• Remove OFPC_INVALID_TTL_TO_CONTROLLER config flag.

6 Auxiliary connections

In previous version of the specification, the channel between the switch and the controller is exclusively made of a single TCP connection, which does not allow to exploit the parallelism available in most switch implementations. OpenFlow 1.3 enables a switch to create auxiliary connections to supplement the main connection between the switch and the controller (EXT-114). Auxiliary connections are mostly useful to carry packet-in and packet-out messages.

• Enable switch to create auxiliary connections to the controller.
• Mandate that auxiliary connection can not exist when main connection is not alive.
• Add auxiliary-id to the protocol to disambiguate the type of connection.
• Enable auxiliary connection over UDP and DTLS.

7 MPLS BoS matching

A new OXM field OXM_OF_MPLS_BOS has been added to match the Bottom of Stack bit (BoS) from the MPLS header (EXT-85). The BoS bit indicates if other MPLS shim header are in the payload of the present MPLS packet, and matching this bit can help to disambiguate case where the MPLS label is reused across levels of MPLS encapsulation.

8 Provider Backbone Bridging tagging

Add support for tagging packet using Provider Backbone Bridging (PBB) encapsulation (EXT-105). This support enables OpenFlow to support various network deployment based on PBB, such as regular PBB and PBB-TE.

- Push and Pop operation to add PBB header as a tag.
- New OXM field to match I-SID for the PBB header.

9 Rework tag order

In previous version of the specification, the final order of tags in a packet was statically specified. For example, a MPLS shim header was always inserted after all VLAN tags in the packet. OpenFlow 1.3 removes this restriction, the final order of tags in a packet is dictated by the order of the tagging operations, each tagging operation adds its tag in the outermost position (EXT-121).

- Remove defined order of tags in packet from the specification.
- Tags are now always added in the outermost possible position.
- Action-list can add tags in arbitrary order.
- Tag order is predefined for tagging in the action-set.

10 Tunnel-ID metadata

The logical port abstraction enables OpenFlow to support a wide variety of encapsulations. The tunnel-id metadata OXM_OF_TUNNEL_ID is a new OXM field that expose to the OpenFlow pipeline metadata associated with the logical port, most commonly the demultiplexing field from the encapsulation header (EXT-107).
For example, if the logical port perform GRE encapsulation, the tunnel-id field would map to the GRE key field from the GRE header. After decapsulation, OpenFlow would be able to match the GRE key in the tunnel-id match field. Similarly, by setting the tunnel-id, OpenFlow would be able to set the GRE key in an encapsulated packet.

11 Cookies in packet-in

A cookie field was added to the packet-in message (EXT-7). This field takes its value from the flow the sends the packet to the controller. If the packet was not sent by a flow, this field is set to 0xffffffffffffffff.
Having the cookie in the packet-in enables the controller to more efficiently classify packet-in, rather than having to match the packet against the full flow table.

12 Duration for stats

A duration field was added to most statistics, including port statistics, group statistics, queue statistics and meter statistics (EXT-102). The duration field enables to more accurately calculate packet and byte rate from the counters included in those statistics.

13 On demand flow counters

New flow-mod flags have been added to disable packet and byte counters on a per-flow basis. Disabling such counters may improve flow handling performance in the switch.

14 Other changes

- Fix a bug describing VLAN matching (EXT-145).
- Flow entry description now mention priority (EXT-115).
- Flow entry description now mention timeout and cookies (EXT-147).
- Unavailable counters must now be set to all 1 (EXT-130).
- Correctly refer to flow entry instead of rule (EXT-132).
- Many other bug fixes, rewording and clarifications.