OpenFlow 1.3流表项

- 流表项字段
  - 1 基本字段
    - cookie — 流表标识
    - duration=seconds — 流表项创建持续的时间
    - n_packets=number
    - n_bytes=number
    - priority=number
    - table=number
    - in_port=port number
    - idle_age — 此流表项从最后一个匹配到的报文到现在空闲的时间
    - hard_age — 此流表项从最后一次被创建或修改到现在持续的时间
    - idle_timeout
    - hard_timeout
  - 2 匹配字段
    - 基本匹配
      - dl_vlan=vlan tag — 匹配vlan tag值，范围为0-4095（2^12），其中vlan为0xffff表示为无vlan标识的数据包
      - dl_vlan_pcp=priority — 匹配数据包的Priority Code point Priority，取值为0-7（包括0和7，数越大优先级越高）
      - dl_src=MAC src/dl_dst=MAC dst — 匹配数据包的源、目MAC地址，匹配01:00:00:00:00:00/01:00:00:00:00:00为多播地址 0:0:0:0:0:0/0:0:0:0:0:0匹配00:00:00:00:00:00为保留地址，可用通配符
      - dl_typ=ethertype — 匹配值为ethertype，例如0x0800为IP Packet 0x0806为ARP Packet，完整的以太网类型协议列表参照下面链接（0-65535）
        - https://en.wikipedia.org/wiki/EtherType
      - nw_proto=protocol number — 匹配不合协议类型的数据包
        - 匹配值为protocol number,当dl_type字段共用使用,完整的类型协议列表参照下面链接
          - https://en.wikipedia.org/wiki/List_of_IP_protocol_numbers
        - 当指定ip或dl_type=0x0800时，匹配IP协议类型，指定0-255间的十进制数（如只匹配ICMP Packet, 6匹配TCP Packet）
        - 当指定ipv6或dl_type=0x86dd时，匹配IPv6第头条类proto,指定0-255之间的十进制数（如58匹配ICMPv6 Packet，6匹配TCP Packet）
        - 当指定arp或dl_type=0x0806时，匹配ARP操作码的低8位，指定arp或dl_type=0x8035时，匹配ARP操作码的低8位
      - nw_src=ip src/nw_dst=ip dst — 根据包匹配ip源地址/目标地址，匹配值为ip/netmask
      - tp_src=port/tp_dst=port — 当dl_type和nw_proto指定为TCP或UDP时，匹配数据包tcp/udp源端口和目的端口,port取值为[0,65535]
      - nw_tos=tos — 匹配IP Tos/DSCP指定IPv4流量类字段的tos,取值0-255的十进制数
      - nw_ecn=ecn — 匹配IP Tos指定IPv4流量类字段中的ecn位，取值为0-3十进制数
      - nw_ttl=ttl — 匹配IP TTL或IPv6跳数限制值HL取值为0-255的十进制数
      - icmp_type/icmp_code — dl_type和nw_proto表明匹配为ICMP或ICMPv6时，type匹配ICMP type,code匹配ICMP code
      - arp_sha/arp_tha — dl_type表明匹配ARP或RARP时，arp_sha匹配源硬件地址，arp_tha匹配目标硬件地址
      - arp_spa/arp_tpa — arp源ip地址，arp目标ip地址
      - table — 指定的流表编号（0-253），未指定时默认为0
      - tunnel_id=tunnel-id — 匹配隧道标识符tunnel-id
      - reg<idx>=value[/mask] — 交换机中的寄存器阵列，当一个数据包进入交换机时，所有的寄存器都被清零，用户可以通过Action 的指令修改寄存器中的值
    - 地址匹配（部分）
      - L1层
        - in_port — port为端口号或LOCAL
      - L2 层
        - dl_vlan
        - dl_vlan_pcp
        - dl_type
        - dl_src/dl_dst
          - dl_src/dl_dst时可加通配符掩码
            - 1. 01:00:00:00:00:00 匹配多播
              位
              dl_dst=01:00:00:00:00:00/01:00:00:00:00 匹配所有的组播数据包，
              dl_dst=00:00:00:00:00:00/01:00:00:00:00匹配所有的单播数据包
            - 2. ff:ff:ff:ff:ff:ff 掩码匹配，相当于没有掩码
            - 3. 00:00:00:00:00:00 匹配所有的位，相当于 dl_dst=*
            - 4. feff:ff:ff:ff:ff:ff 匹配组播位外的所有位
      - L3 层 → 注意：需要加入二层流表参数后，三层流表参数才起作用
        - nw_proto
          - ip：dl_type=0x0800
          - arp: dl_type=0x0806
          - 8qc: dl_type=0x88cc
          - icmp: dl_type=0x0800 nw_proto=1
          - tcp:dl_type=0x0800 nw_proto=6
          - udp: dl_type=0x0800 nw_proto=17
        - nw_tos — 当dl_type为通配符或0x800,0x086dd时才能增加匹配（ip/ipv6）
        - nw_ecn — 当dl_type为通配符或0x800,0x086dd时才能增加匹配(ip/ipv6)
        - nw_ttl — 当dl_type为通配符或0x800,0x086dd时才能增加匹配(ip/ipv6)
        - nw_src/nw_dst — 当dl_type为通配符或0x800,0x0806,0x8035时才能增加匹配(ip/arp/rarp)
          - nw_src=ip[/netmask] — netmask不填，默认32位，例如：192.168.1.0/24
          - nw_dst=ip[/netmask]
      - L4 层
        - icmp_type
        - icmp_code
        - tp_src/tp_dst — 需指定dl_type和nw_proto为tcp/udp时，此项生效
  - 3 动作字段
    - 转发
      - output:port
      - in_port
      - normal — 指定数据包不再openflow处理，用6/2/L3组转发即可
      - all — 除入口之外，数据包流来到其他物理端口
      - flood — 除入口和禁止flood的端口外，数据包流来到其他的物理端口
      - CONTROLLER（key:value...）
        - 上传数据包到控制器打包packet in信息，可选的key,value如下：
          - max_len=nbytes 传送给控制器的最大字节数 65535,默认是发送整个数据包
          - reason=reason 发送packet in 消息的原因，reason可取值为ofpr_action常量，no_match,invalid_ttl
          - id=controller-id，默认为0
      - → 一个流规则中可能有多个动作，按照指定的先后顺序执行，actions=[target[],target...]
    - 丢弃
      - drop
    - 修改字段
      - mod_vlan_vid: vlan id
      - mod_vlan_pcp:priority
      - strip_vlan — 移除数据包的VLAN tag
      - push_vlan:ethertype
      - mod_dl_src/mod_dl_dst:mac address
      - mod_nw_src/mod_nw_dst : ip
      - mod_tp_src/mod_tp_dst:port
      - mod_nw_tos:tos
      - mod_nw_ecn
      - set_tunnel:id / set_tunnel64:id
      - set_queue:queue
      - pop_equeue
      - dec_ttl/dec_ttl[1],[id2]
      - set_mpls_ttl:ttl / dec_mpls_ttl
      - resubmit[port],[table]] — 按照流表编号来匹配流表项，知指定端口，in_port字段替换为port字段 resubmit(,20)：重新执行table 20的rule
      - set_field：value-> dst — 将值写入字段dst，dst应此为匹配标准，例如set_field:fe80:0123:4567:890a:dbff:fefe:59fa-> ipv6_src
      - load：value-> dst [start,end] load：55-> NXM_NX_REG2 [0..5]将值55（位图式110111）加载到寄存器2中的位0至5中
      - push：src [start,end] push：NXM_NX_REG2 [0..5]将存储在寄存器2初始0位至第5位（含）的值压入内部堆栈
      - pop：dst[start,end] pop：NXM_NX_REG2[0..5]将存储在寄存器2初始0到第5位（含）重新安置
      - move：src[start,end]-->dst[start,end]
        - 将src指定位图转移dst指定位，必须为NXM字段
          - move：NXM_NX_REG0 [0..5]；-> NXM_NX_REG1 [26..31]将寄存器0中最每为0至5的六个位图转移到2-26至31（含）
          - move：NXM_NX_REG0 [0..15]；-> NXM_OF_VLAN_TCI 将寄存器0的最低有效16位图转移VLAN TCI字段中。
      - set_tunnel:id / set_tunnel64:id
      - learn(argument,argument....)
        - 此操作在OpenFlow表中添加或修改表项，类似于ovs-ofctl -strict mod-flows，参数指定流的匹配字段，操作和其他属性
          - hard_timeout=seconds/idle_timeout=seconds
          - priority=value
          - table=number — 插入新流表项所属的表，未指明默认为1，数值为0-254
          - field[start,end]=src[start,end]/load:value->dst[start,end]/output:field[start,end] — 匹配条件/输出（NXM字段）
- 流表项结构
  - 1 Match Fields(40)
    - Ether
      - 3 Ether Des (48 bits)
      - 4 Ether Source (48 bits)
      - 5 Ether Type (16 bits)
      - 6 VLAN ID(12+ 1 bits)
      - 7 VLAN Priority (3 bits)
    - IP
      - IP 区分服务指南，IP头部的TOS(Type of service)标识符，使用6bit
      - 8 IP DSCP (6 bits)
      - 显示拥塞通知（Explicit Congestion Notification）,2 bits in ToS
      - 9 IP ECN (2bits)
      - 10 IP Proto(8 bits)
    - IPv4
      - 11 IPV4 Source(32 bits)
      - 12 IPV4 Des(32 bits)
    - TCP
      - 13 TCP Source(16 bits)
      - 14 TCP Des(16 bits)
    - UDP
      - 15 UDP Source(16 bits)
      - 16 UDP Des(16 bits)
    - SCTP
      - 17 SCTP Source(16 bits)
      - 18 STCP Des(16 bits)
    - ICMPv4
      - 19 ICMPv4 Type(8 bits)
      - 20 ICMPv4 Code(8bits)
    - ARP
      - ARP 操作字段，对应四种操作类型（ARP请求：1 ARP响应：2 RARP请求：3 RARP响应：4 ）
      - 21 ARP OP(16 bits)
      - 22 ARP Source (IPv4) (32 bits)
      - 23 ARP Target (IPv4)(32 bits)
      - 24 ARP Source hardware(48 bits)
      - 25 ARP Target hardware(48 bits)
    - IPv6
      - 26 IPv6 Source(128 Bits)
      - 27 IPv6 Des(128 Bits)
      - 28 IPv6 Flow Label(20 bits)
    - ICMPv6
      - 29 ICMPv6 Type(8 bits)
      - 30 ICMPv6 Code(8 bits)
    - IPv6 ND
      - 31 IPv6 Target (ND)(128 bits)
      - IPv6 邻居发现协议目标地址
      - 32 IPv6 Source LL (ND)(48 bits)
      - IPv6 邻居发现协议的链路层目标地址
      - 33 IPv6 Target LL(ND)(48 bits)
      - IPv6 邻居发现协议的链路层目标地址
    - MLPS
      - 34 MLPS Label(20 bits)
      - 35 MPLS TC(3 Bits)
      - 36 MPLS BoS (1 bit)
    - PBB
      - 37 PBB ISID (24 bits)
    - TUNNEL
      - 38 TUNNEL ID (64 bits)
    - IPv6 EXTHDR
      - IPv6 Extension Header Pseudo-field
      - 39 IPv6 EXTHDR (9 bits)
    - OXM TLVs
      - 端口数字从1开始，可能是物理端口，也可能是逻辑端口
        - 0 Ingress port (32bits)
      - 数据包在逻辑端口接收时的底层物理端口
        - 1 Ingress physical port (32 bits)
      - 用于在表之间传递信息，起标记作用
        - 2 Metadata (64 bits)
    - Packet headers
    - **上子结构中某体加粗的为交换机必须支持的匹配项，但这些流表项不一定在流表中实现特定匹配**
      - Require Match
  - 2 Counters
    - Priority
    - 说明
      - Counters是基于流表，流表项，端口，队列，组表，动作桶，计量表，计量带
      - per flow table — Active Flow Pkts Matched Pkts Looked-up
      - 当前有效流表数，查表的数据包数，匹配的数据包次数
      - per flow entry — 到达包数，到达字节数，秒级流间隔时间，纳秒级流间隔时间
      - 分类
        - per port — Rx Pkts Tx Pkts Rx Bytes Tx Bytes Rx Drops Tx Drops Rx Errs Tx Errs Rx Frame Errs Rx OverRun Errs Rx CRC Errs Collisions
          - 接收包数，字节数，发送包数，字节数，丢弃的接收数据包数，丢弃的发送数据包数，接收错误，接收帧校验字节校验错误，接收溢出错误，接收帧CRC错误，冲突，接收时长（秒数），接收时间（纳秒级）
        - per Queue — 发送包数，字节数，接收溢出错误，持续时间（秒数），持续时间（纳秒级）
      - per Group
      - per Group bucket
      - per Meter
      - per Meter Band
  - 3 Instructions
    - Instructions（作用流表之间，processing和modify action set，可对应多个action）
      - Required Instruction
        - 指定Action写入 Action Set，是对每个数据包的操作，执行下流表之间
        - Write-Actions action(s)
        - 流表流水线处理的下一条流表 next-table-id
        - Goto-Table next_table_id
        - Meter meter_id
      - Optional Instruction
        - 作用指定Action（Action List中Action），不改变Action Set, 作用于数据包
        - Apply- Actions action(s)
        - 清除Action Set中所有Actions
        - Clean- Actions
        - Write-Metadata metadata/mask
        - Experimenter
    - 区分
    - Action（作用于数据包）
      - 分类1（按一定顺序执行）
        - **Action Set(每个数据包都有动作集，Action Set作用于流表之间）**
        - 1 copy TTL Inwards
        - 2 pop
        - 3 push-MLPS
        - 4 push-PBB
        - 5 push-VLAN
        - 6 copy TTL outwards
        - 7 decrement TTL
        - 8 set
        - 9 qos
        - 10 group
        - 11 output
        - Action List
          - **按列表顺序执行，新动作覆盖旧动作，动作写入Action Set种动作类型，Apply -Action和Packet-out中包含Action List**
      - 分类2（按required和Optional分类）
        - Required Action
          - Output
            - 转发给所有的端口（入端口除外）
            - ALL
            - 数据包转发至控制器
            - CONTROLLER
            - 对Packet-out报文执行Action List中动作
            - TABLE
            - 从入端口发出
            - IN_PORT
            - 本地定义的端口转发
            - ANY
            - 没有明确的output Action
          - Drop
            - 指令集（Instruction Set）为空或执行Clear-Action之后
          - Group
            - 由特定组表执行，从而流表选择group中的action bucket
        - Optional Action
          - Output
            - 本地的网络栈和管理栈
            - LOCAL
            - 传统的二层三层转发，不经过OpenFLow匹配动作处理
            - NORMAL
            - 数据包泛洪发出，不包括入端口
            - FLOOD
          - Set-Queue（QoS）
            - 将数据包转发发到出端口对应的转发队列
          - Push-Tag/Pop-tag
            - Push/Pop VLAN header
            - Push/Pop MPLS header
            - Push/Pop PBB header
          - Set-Field
            - 修改数据包的头内容
          - Change-TTL
            - Set/Decrement MPLS TTL
            - Set/Decrement IP TTL
            - Copy TTL outwards
            - Copy TTL Inwards
  - 4 TimeOuts
    - 分类
      - 空闲超时时间（idle_time）
      - 硬超时时间（hard_timeout）
    - 说明
      - 1. hard_time设置不为0，从流表项创建时间-hard_timeout 流表项有效
      - 2. idle_time设置不为0，从上次流表项匹配成功时间-idle_timeout 流表项有效
  - Cookie