How to extend the OF protocol with OpenvSwitch+Floodlight

Baohua Yang

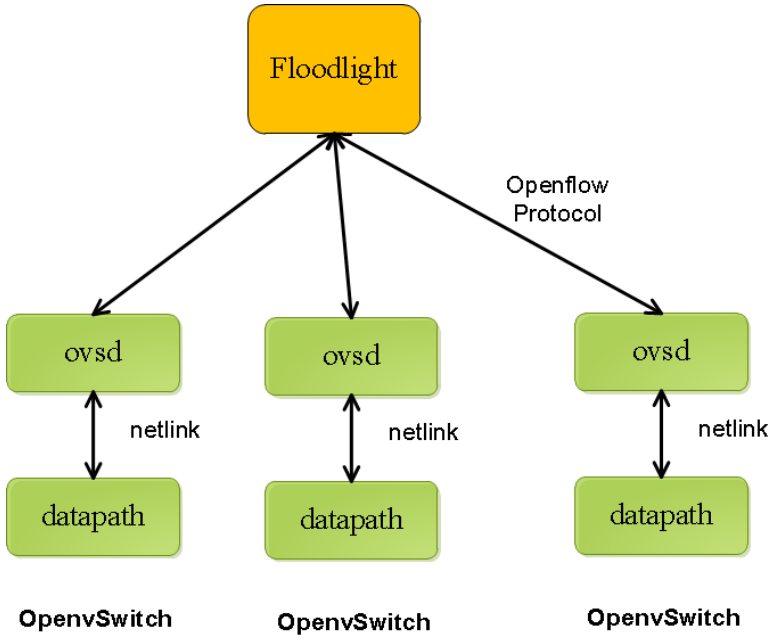
Updated: 2012-12-15

1. 整体思路

要在 floodlight+ovs 的模型下扩展 OF 协议，增加新的功能，主要包括两个基本点，涉及到三部分模块的修改。

两个基本点包括要扩展的消息和可能涉及的行动。

三部分模块包括控制器模块、ovsd 模块（用户态）和 datapath 模块（内核态）。



本文中，以添加新的 PacketRemote 消息为例（涉及添加新的行动 ActionRemote），介绍这一过程。该消息可以让被处理的网包从指定端口发出去，并且在发出去之前，封装上给定的 IP 地址信息。

2. Floodlight

2.1. 行动

在 src/main/java/org/openflow/protocol/action/OFActionType.java 文件中声明新的消息类型。主要是在 public enum OFActionType {}结构中添加，注意新消息的 id 要跟 ovs 中使用的一致。

```
public enum OFActionType {
   OUTPUT           (0, OFActionOutput.class, new Instantiable<OFAction>() {
                    @Override
                    public OFAction instantiate() {
                       return new OFActionOutput();
                    }}),
   SET_VLAN_ID      (1, OFActionVirtualLanIdentifier.class, new Instantiable<OFAction>() {
                    @Override
                    public OFAction instantiate() {
                       return new OFActionVirtualLanIdentifier();
                    }}),
   SET_VLAN_PCP     (2, OFActionVirtualLanPriorityCodePoint.class, new
Instantiable<OFAction>() {
                    @Override
                    public OFAction instantiate() {
                       return new OFActionVirtualLanPriorityCodePoint();
                    }}),
   STRIP_VLAN       (3, OFActionStripVirtualLan.class, new Instantiable<OFAction>() {
                    @Override
                    public OFAction instantiate() {
                       return new OFActionStripVirtualLan();
                    }}),
   SET_DL_SRC       (4, OFActionDataLayerSource.class, new Instantiable<OFAction>() {
                    @Override
                    public OFAction instantiate() {
                       return new OFActionDataLayerSource();
                    }}),
   SET_DL_DST       (5, OFActionDataLayerDestination.class, new
Instantiable<OFAction>() {
                    @Override
                    public OFAction instantiate() {
                       return new OFActionDataLayerDestination();
                    }}),
```

```
   SET_NW_SRC        (6, OFActionNetworkLayerSource.class, new
Instantiable<OFAction>() {
                    @Override
                    public OFAction instantiate() {
                       return new OFActionNetworkLayerSource();
                    }}),
   SET_NW_DST        (7, OFActionNetworkLayerDestination.class, new
Instantiable<OFAction>() {
                    @Override
                    public OFAction instantiate() {
                       return new OFActionNetworkLayerDestination();
                    }}),
   SET_NW_TOS        (8, OFActionNetworkTypeOfService.class, new
Instantiable<OFAction>() {
                    @Override
                    public OFAction instantiate() {
                       return new OFActionNetworkTypeOfService();
                    }}),
   SET_TP_SRC        (9, OFActionTransportLayerSource.class, new
Instantiable<OFAction>() {
                    @Override
                    public OFAction instantiate() {
                       return new OFActionTransportLayerSource();
                    }}),
   SET_TP_DST        (10, OFActionTransportLayerDestination.class, new
Instantiable<OFAction>() {
                    @Override
                    public OFAction instantiate() {
                       return new OFActionTransportLayerDestination();
                    }}),
   OPAQUE_ENQUEUE     (11, OFActionEnqueue.class, new Instantiable<OFAction>() {
                    @Override
                    public OFAction instantiate() {
                       return new OFActionEnqueue();
                    }}),
```

```
REMOTE        (12, OFActionRemote.class, new Instantiable<OFAction>() {
              @Override
              public OFAction instantiate() {
                return new OFActionRemote();
              }}),
VENDOR        (0xffff, OFActionVendor.class, new Instantiable<OFAction>() {
              @Override
              public OFAction instantiate() {
                return new OFActionVendor();
              }});
```

之后，创建行动类，主要在文件 src/main/java/org/openflow/protocol/action/OFActionRemote.java 中定义。注意相关的长度信息要满足 OF 协议的规范定义和实际需求。

2.2. 消息

在 src/main/java/org/openflow/protocol/OFType.java 文件中声明新的消息类型。主要是在 public enum OFType {}结构中添加，注意新消息的 id 要跟 ovs 中的一致。

```
public enum OFType {
   HELLO           (0, OFHello.class, new Instantiable<OFMessage>() {
                   @Override
                   public OFMessage instantiate() {
                     return new OFHello();
                   }}),
   ERROR           (1, OFError.class, new Instantiable<OFMessage>() {
                   @Override
                   public OFMessage instantiate() {
                     return new OFError();
                   }}),
   ECHO_REQUEST      (2, OFEchoRequest.class, new Instantiable<OFMessage>() {
                   @Override
                   public OFMessage instantiate() {
                     return new OFEchoRequest();
                   }}),
   ECHO_REPLY        (3, OFEchoReply.class, new Instantiable<OFMessage>() {
                   @Override
                   public OFMessage instantiate() {
                     return new OFEchoReply();
                   }}),
   VENDOR          (4, OFVendor.class, new Instantiable<OFMessage>() {
                   @Override
                   public OFMessage instantiate() {
                     return new OFVendor();
                   }}),
   FEATURES_REQUEST   (5, OFFeaturesRequest.class, new Instantiable<OFMessage>() {
                   @Override
                   public OFMessage instantiate() {
                     return new OFFeaturesRequest();
                   }}),
   FEATURES_REPLY     (6, OFFeaturesReply.class, new Instantiable<OFMessage>() {
                   @Override
```

```
                    public OFMessage instantiate() {
                        return new OFFeaturesReply();
                    }}),
    GET_CONFIG_REQUEST  (7, OFGetConfigRequest.class, new Instantiable<OFMessage>()
{
                    @Override
                    public OFMessage instantiate() {
                        return new OFGetConfigRequest();
                    }}),
    GET_CONFIG_REPLY    (8, OFGetConfigReply.class, new Instantiable<OFMessage>() {
                    @Override
                    public OFMessage instantiate() {
                        return new OFGetConfigReply();
                    }}),
    SET_CONFIG          (9, OFSetConfig.class, new Instantiable<OFMessage>() {
                    @Override
                    public OFMessage instantiate() {
                        return new OFSetConfig();
                    }}),
    PACKET_IN           (10, OFPacketIn.class, new Instantiable<OFMessage>() {
                    @Override
                    public OFMessage instantiate() {
                        return new OFPacketIn();
                    }}),
    FLOW_REMOVED        (11, OFFlowRemoved.class, new Instantiable<OFMessage>() {
                    @Override
                    public OFMessage instantiate() {
                        return new OFFlowRemoved();
                    }}),
    PORT_STATUS         (12, OFPortStatus.class, new Instantiable<OFMessage>() {
                    @Override
                    public OFMessage instantiate() {
                        return new OFPortStatus();
```

```
                }}),
PACKET_OUT        (13, OFPacketOut.class, new Instantiable<OFMessage>() {
                @Override
                public OFMessage instantiate() {
                  return new OFPacketOut();
                }}),
FLOW_MOD          (14, OFFlowMod.class, new Instantiable<OFMessage>() {
                @Override
                public OFMessage instantiate() {
                  return new OFFlowMod();
                }}),
PORT_MOD          (15, OFPortMod.class, new Instantiable<OFMessage>() {
                @Override
                public OFMessage instantiate() {
                  return new OFPortMod();
                }}),
STATS_REQUEST     (16, OFStatisticsRequest.class, new Instantiable<OFMessage>() {
                @Override
                public OFMessage instantiate() {
                  return new OFStatisticsRequest();
                }}),
STATS_REPLY       (17, OFStatisticsReply.class, new Instantiable<OFMessage>() {
                @Override
                public OFMessage instantiate() {
                  return new OFStatisticsReply();
                }}),
BARRIER_REQUEST   (18, OFBarrierRequest.class, new Instantiable<OFMessage>() {
                @Override
                public OFMessage instantiate() {
                  return new OFBarrierRequest();
                }}),
BARRIER_REPLY     (19, OFBarrierReply.class, new Instantiable<OFMessage>() {
                @Override
```

```
                public OFMessage instantiate() {
                    return new OFBarrierReply();
                }}),
    QUEUE_GET_CONFIG_REQUEST    (20, OFQueueGetConfigRequest.class, new
Instantiable<OFMessage>() {
                    @Override
                    public OFMessage instantiate() {
                        return new OFQueueGetConfigRequest();
                    }}),
    QUEUE_GET_CONFIG_REPLY      (21, OFQueueGetConfigReply.class, new
Instantiable<OFMessage>() {
                    @Override
                    public OFMessage instantiate() {
                        return new OFQueueGetConfigReply();
                    }}),
    PACKET_REMOTE      (22, OFPacketRemote.class, new Instantiable<OFMessage>() {
                                    @Override
                                    public OFMessage instantiate() {
                                        return new OFPacketRemote();
                                    }});
```

之后，添加新的消息类的实现 src/main/java/org/openflow/protocol/OFPacketRemote.java。

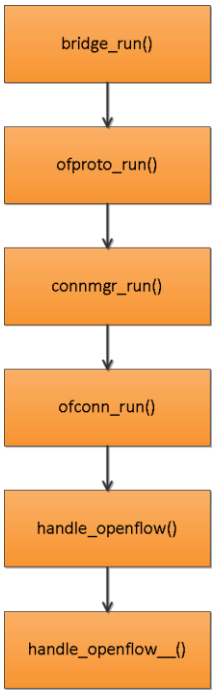
2.3. 调用

之后，可在自定义的 app 中调用该消息。一般通过在 processPacketIn()接口中执行相关的操作。

3. OpenvSwitch 用户态

OpenvSwitch 处理 of 消息，主要通过，bridge_run()→ofproto_run()→connmgr_run()→ofconn_run()来进行处理，ofconn_run()更进一步的调用传入的 handle_openflow()对 of 消息进行解析。

对 of 消息的解析在 ofproto/ofproto.c 文件中完成，主要通过 handle_openflow()→handle_openflow__()进行处理。



其中，handle_openflow__()首先调用 ofptype_decode()（位于 lib/ofp_msg.c）来解析消息的类型（返回的实际上就是 OF 消息头中 type 的值）。然后根据解析的类型进行相应的消息处理。因此，整个过程是在 handle_openflow__()中添加 OFPTYPE_PACKET_REMOTE 类型的处理 handle_packet_remote()，这是最上层的处理封装。为了实现这个函数，要先预先添加相关的数据结构定义和子函数。

3.1. OpenFlow 行动处理

以 openflow-1.0 为例，在 include/openflow/openflow-1.0.h 中添加新的 remote 行动的结构体 ofp_action_remote 和消息头结构体 ofp_packet_remote。

3.1.1. 新的行动结构体

注意已有行动结构体的抽象 union ofp_action 的大小为 8 字节，新的行动结构为 16 字节。在 include/openflow/openflow-1.0.h 中添加

```
struct ofp_action_remote {
   ovs_be16 type;              /* OFPAT10_REMOTE. */
   ovs_be16 len;               /* Length is 8. */
   ovs_be16 port;              /* Output port. */

   uint8_t pad[6];             /* pad. */
   ovs_be32 ip;                /* Remote ip. */
};
OFP_ASSERT(sizeof(struct ofp_action_remote) == 16);
```

3.1.2. 新的消息头结构体

新的消息头结构体 ofp_packet_remote 大小为 8 字节，在 include/openflow/openflow-1.0.h 中添加

```
struct ofp_packet_remote {
   ovs_be32 buffer_id;         /* ID assigned by datapath or UINT32_MAX. */
   ovs_be16 in_port;           /* Packet's input port (OFPP_NONE if none). */
   ovs_be16 actions_len;       /* Size of action array in bytes. */
   /* Followed by:
    *   - Exactly 'actions_len' bytes (possibly 0 bytes, and always a multiple
    *     of 8) containing actions.
    *   - If 'buffer_id' == UINT32_MAX, packet data to fill out the remainder
    *     of the message length.
    */
};
OFP_ASSERT(sizeof(struct ofp_packet_remote) == 8);
```

3.1.3. 声明新的行动类型

还要在 enum ofp10_action_type 中添加新的行动 id，需要注意，此处与控制器端行动类型定义数值要一致。

```
enum ofp10_action_type {
    OFPAT10_OUTPUT,          /* Output to switch port. */
    OFPAT10_SET_VLAN_VID,     /* Set the 802.1q VLAN id. */
    OFPAT10_SET_VLAN_PCP,     /* Set the 802.1q priority. */
    OFPAT10_STRIP_VLAN,       /* Strip the 802.1q header. */
    OFPAT10_SET_DL_SRC,       /* Ethernet source address. */
    OFPAT10_SET_DL_DST,       /* Ethernet destination address. */
    OFPAT10_SET_NW_SRC,       /* IP source address. */
    OFPAT10_SET_NW_DST,       /* IP destination address. */
    OFPAT10_SET_NW_TOS,       /* IP ToS (DSCP field, 6 bits). */
    OFPAT10_SET_TP_SRC,       /* TCP/UDP source port. */
    OFPAT10_SET_TP_DST,       /* TCP/UDP destination port. */
    OFPAT10_ENQUEUE,          /* Output to queue. */
    OFPAT10_REMOTE=12,        /* Remote to switch port. */
    OFPAT10_VENDOR = 0xffff
};
```

3.2. OVS 抽象行动处理

3.2.1. ofp-msgs.h

ofp-msgs.h 中主要定义了两个 enum 类型来表示 OF 的消息类型：enum ofptype{}和 enum ofpraw{}。前者是基本抽象的消息类型，后者则代表真实的原始消息，进一步进行细分（例如不同的 OF 版本等）。

所有 OF 消息的语义类型在 ofp-msgs.h 中的 enum ofptype{}中定义。此处，我们添加 OFPTYPE_PACKET_REMOTE 类型，并注意在注释中添加对应的元消息类型 OFPRAW_OFPT10_PACKET_REMOTE（其中元消息类型的数值必须跟控制器发出的 OF 消息头中类型的数值相一致）。

enum ofptype{}类型定义如下：

```
/* Semantic identifiers for OpenFlow messages.
 *
 * Each OFPTYPE_* enumeration constant represents one or more concrete format
 * of OpenFlow message.  When two variants of a message have essentially the
 * same meaning, they are assigned a single OFPTYPE_* value.
 *
 * The comments here must follow a stylized form because the "extract-ofp-msgs"
 * program parses them at build time to generate data tables.  The format is
 * simply to list each OFPRAW_* enumeration constant for a given OFPTYPE_*,
 * each followed by a period. */
enum ofptype {
    /* Immutable messages. */
    OFPTYPE_HELLO,              /* OFPRAW_OFPT_HELLO. */
    OFPTYPE_ERROR,              /* OFPRAW_OFPT_ERROR. */
    OFPTYPE_ECHO_REQUEST,       /* OFPRAW_OFPT_ECHO_REQUEST. */
    OFPTYPE_ECHO_REPLY,         /* OFPRAW_OFPT_ECHO_REPLY. */

    /* Switch configuration messages. */
    OFPTYPE_FEATURES_REQUEST,   /* OFPRAW_OFPT_FEATURES_REQUEST. */
    OFPTYPE_FEATURES_REPLY,     /* OFPRAW_OFPT10_FEATURES_REPLY.
                                 * OFPRAW_OFPT11_FEATURES_REPLY. */
    OFPTYPE_GET_CONFIG_REQUEST, /* OFPRAW_OFPT_GET_CONFIG_REQUEST. */
    OFPTYPE_GET_CONFIG_REPLY,   /* OFPRAW_OFPT_GET_CONFIG_REPLY. */
    OFPTYPE_SET_CONFIG,         /* OFPRAW_OFPT_SET_CONFIG. */

    /* Asynchronous messages. */
    OFPTYPE_PACKET_IN,          /* OFPRAW_OFPT10_PACKET_IN.
                                 * OFPRAW_OFPT11_PACKET_IN.
                                 * OFPRAW_OFPT12_PACKET_IN.
                                 * OFPRAW_NXT_PACKET_IN. */
    OFPTYPE_FLOW_REMOVED,       /* OFPRAW_OFPT10_FLOW_REMOVED.
                                 * OFPRAW_OFPT11_FLOW_REMOVED.
                                 * OFPRAW_NXT_FLOW_REMOVED. */
```

```
    OFPTYPE_PORT_STATUS,        /* OFPRAW_OFPT10_PORT_STATUS.
                                 * OFPRAW_OFPT11_PORT_STATUS. */

    /* Controller command messages. */
    OFPTYPE_PACKET_OUT,         /* OFPRAW_OFPT10_PACKET_OUT.
                                 * OFPRAW_OFPT11_PACKET_OUT. */
    OFPTYPE_FLOW_MOD,           /* OFPRAW_OFPT10_FLOW_MOD.
                                 * OFPRAW_OFPT11_FLOW_MOD.
                                 * OFPRAW_NXT_FLOW_MOD. */
    OFPTYPE_PORT_MOD,           /* OFPRAW_OFPT10_PORT_MOD.
                                 * OFPRAW_OFPT11_PORT_MOD. */

    /* Barrier messages. */
    OFPTYPE_BARRIER_REQUEST,    /* OFPRAW_OFPT10_BARRIER_REQUEST.
                                 * OFPRAW_OFPT11_BARRIER_REQUEST. */
    OFPTYPE_BARRIER_REPLY,      /* OFPRAW_OFPT10_BARRIER_REPLY.
                                 * OFPRAW_OFPT11_BARRIER_REPLY. */

    /* Statistics. */
    OFPTYPE_DESC_STATS_REQUEST,     /* OFPRAW_OFPST_DESC_REQUEST. */
    OFPTYPE_DESC_STATS_REPLY,       /* OFPRAW_OFPST_DESC_REPLY. */
    OFPTYPE_FLOW_STATS_REQUEST,     /* OFPRAW_OFPST10_FLOW_REQUEST.
                                     * OFPRAW_OFPST11_FLOW_REQUEST.
                                     * OFPRAW_NXST_FLOW_REQUEST. */
    OFPTYPE_FLOW_STATS_REPLY,       /* OFPRAW_OFPST10_FLOW_REPLY.
                                     * OFPRAW_OFPST11_FLOW_REPLY.
                                     * OFPRAW_NXST_FLOW_REPLY. */
    OFPTYPE_AGGREGATE_STATS_REQUEST, /* OFPRAW_OFPST_AGGREGATE_REQUEST.
                                     * OFPRAW_NXST_AGGREGATE_REQUEST. */
    OFPTYPE_AGGREGATE_STATS_REPLY,   /* OFPRAW_OFPST_AGGREGATE_REPLY.
                                     * OFPRAW_NXST_AGGREGATE_REPLY. */
    OFPTYPE_TABLE_STATS_REQUEST,    /* OFPRAW_OFPST_TABLE_REQUEST. */
    OFPTYPE_TABLE_STATS_REPLY,      /* OFPRAW_OFPST10_TABLE_REPLY.
```

```
                        * OFPRAW_OFPST11_TABLE_REPLY.
                        * OFPRAW_OFPST12_TABLE_REPLY. */
    OFPTYPE_PORT_STATS_REQUEST,     /* OFPRAW_OFPST_PORT_REQUEST. */
    OFPTYPE_PORT_STATS_REPLY,       /* OFPRAW_OFPST_PORT_REPLY. */
    OFPTYPE_QUEUE_STATS_REQUEST,    /* OFPRAW_OFPST_QUEUE_REQUEST. */
    OFPTYPE_QUEUE_STATS_REPLY,      /* OFPRAW_OFPST_QUEUE_REPLY. */
    OFPTYPE_PORT_DESC_STATS_REQUEST, /* OFPRAW_OFPST_PORT_DESC_REQUEST. */
    OFPTYPE_PORT_DESC_STATS_REPLY,   /* OFPRAW_OFPST_PORT_DESC_REPLY. */



    /* Nicira extensions. */
    OFPTYPE_ROLE_REQUEST,       /* OFPRAW_NXT_ROLE_REQUEST. */
    OFPTYPE_ROLE_REPLY,         /* OFPRAW_NXT_ROLE_REPLY. */
    OFPTYPE_SET_FLOW_FORMAT,    /* OFPRAW_NXT_SET_FLOW_FORMAT. */
    OFPTYPE_FLOW_MOD_TABLE_ID,  /* OFPRAW_NXT_FLOW_MOD_TABLE_ID. */
    OFPTYPE_SET_PACKET_IN_FORMAT, /* OFPRAW_NXT_SET_PACKET_IN_FORMAT. */
    OFPTYPE_FLOW_AGE,           /* OFPRAW_NXT_FLOW_AGE. */
    OFPTYPE_SET_ASYNC_CONFIG,   /* OFPRAW_NXT_SET_ASYNC_CONFIG. */
    OFPTYPE_SET_CONTROLLER_ID,  /* OFPRAW_NXT_SET_CONTROLLER_ID. */


    /* Flow monitor extension. */
    OFPTYPE_FLOW_MONITOR_STATS_REQUEST, /*
OFPRAW_NXST_FLOW_MONITOR_REQUEST. */
    OFPTYPE_FLOW_MONITOR_STATS_REPLY,   /*
OFPRAW_NXST_FLOW_MONITOR_REPLY. */
    OFPTYPE_FLOW_MONITOR_CANCEL,       /* OFPRAW_NXT_FLOW_MONITOR_CANCEL.
*/
    OFPTYPE_FLOW_MONITOR_PAUSED,       /* OFPRAW_NXT_FLOW_MONITOR_PAUSED.
*/
    OFPTYPE_FLOW_MONITOR_RESUMED,      /*
OFPRAW_NXT_FLOW_MONITOR_RESUMED. */


    OFPTYPE_PACKET_REMOTE,      /* OFPRAW_OFPT10_PACKET_REMOTE. */
};
```

enum ofpraw{}类型定义如下：

```
/* Raw identifiers for OpenFlow messages.
 *
 * Some OpenFlow messages with similar meanings have multiple variants across
 * OpenFlow versions or vendor extensions.  Each variant has a different
 * OFPRAW_* enumeration constant.  More specifically, if two messages have
 * different types, different numbers, or different arguments, then they must
 * have different OFPRAW_* values.
 *
 * The comments here must follow a stylized form because the "extract-ofp-msgs"
 * program parses them at build time to generate data tables.  The syntax of
 * each comment is:
 *
 *    type versions (number): arguments.
 *
 * where the syntax of each part is:
 *
 *    - type: One of OFPT (standard OpenFlow message), OFPST (standard OpenFlow
 *      statistics message), NXT (Nicira extension message), or NXST (Nicira
 *      extension statistics message).
 *
 *      As new vendors implement extensions it will make sense to expand the
 *      dictionary of possible types.
 *
 *    - versions: The OpenFlow version or versions in which this message is
 *      supported, e.g. "1.0" or "1.1" or "1.0+".
 *
 *    - number:
 *        For OFPT, the 'type' in struct ofp_header.
 *        For OFPST, the 'type' in struct ofp_stats_msg or ofp11_stats_msg.
 *        For NXT, the 'subtype' in struct nicira_header.
 *        For NXST, the 'subtype' in struct nicira10_stats_msg or
 *          nicira11_stats_msg.
 *
```

```
 *    - arguments: The types of data that follow the OpenFlow headers (the
 *      message "body").  This can be "void" if the message has no body.
 *      Otherwise, it should be a comma-separated sequence of C types.  The
 *      last type in the sequence can end with [] if the body ends in a
 *      variable-length sequence.
 *
 *      The arguments are used to validate the lengths of messages when a
 *      header is parsed.  Any message whose length isn't valid as a length of
 *      the specified types will be rejected with OFPERR_OFPBRC_BAD_LEN.
 *
 *      A few OpenFlow messages, such as OFPT_PACKET_IN, intentionally end with
 *      only part of a structure, up to some specified member.  The syntax "up
 *      to <member>" indicates this, e.g. "struct ofp11_packet_in up to data".
 */
enum ofpraw {
/* Standard messages. */

    /* OFPT 1.0+ (0): uint8_t[]. */
    OFPRAW_OFPT_HELLO,

    /* OFPT 1.0+ (1): struct ofp_error_msg, uint8_t[]. */
    OFPRAW_OFPT_ERROR,

    /* OFPT 1.0+ (2): uint8_t[]. */
    OFPRAW_OFPT_ECHO_REQUEST,

    /* OFPT 1.0+ (3): uint8_t[]. */
    OFPRAW_OFPT_ECHO_REPLY,

    /* OFPT 1.0+ (5): void. */
    OFPRAW_OFPT_FEATURES_REQUEST,

    /* OFPT 1.0 (6): struct ofp_switch_features, struct ofp10_phy_port[]. */
```

```
    OFPRAW_OFPT10_FEATURES_REPLY,
    /* OFPT 1.1+ (6): struct ofp_switch_features, struct ofp11_port[]. */
    OFPRAW_OFPT11_FEATURES_REPLY,

    /* OFPT 1.0+ (7): void. */
    OFPRAW_OFPT_GET_CONFIG_REQUEST,

    /* OFPT 1.0+ (8): struct ofp_switch_config. */
    OFPRAW_OFPT_GET_CONFIG_REPLY,

    /* OFPT 1.0+ (9): struct ofp_switch_config. */
    OFPRAW_OFPT_SET_CONFIG,

    /* OFPT 1.0 (10): struct ofp_packet_in up to data, uint8_t[]. */
    OFPRAW_OFPT10_PACKET_IN,
    /* OFPT 1.1 (10): struct ofp11_packet_in up to data, uint8_t[]. */
    OFPRAW_OFPT11_PACKET_IN,
    /* OFPT 1.2 (10): struct ofp12_packet_in, uint8_t[]. */
    OFPRAW_OFPT12_PACKET_IN,
    /* NXT 1.0+ (17): struct nx_packet_in, uint8_t[]. */
    OFPRAW_NXT_PACKET_IN,

    /* OFPT 1.0 (11): struct ofp_flow_removed. */
    OFPRAW_OFPT10_FLOW_REMOVED,
    /* OFPT 1.1+ (11): struct ofp11_flow_removed, uint8_t[8][]. */
    OFPRAW_OFPT11_FLOW_REMOVED,
    /* NXT 1.0+ (14): struct nx_flow_removed, uint8_t[8][]. */
    OFPRAW_NXT_FLOW_REMOVED,

    /* OFPT 1.0 (12): struct ofp_port_status, struct ofp10_phy_port. */
    OFPRAW_OFPT10_PORT_STATUS,
    /* OFPT 1.1+ (12): struct ofp_port_status, struct ofp11_port. */
    OFPRAW_OFPT11_PORT_STATUS,
```

```
    /* OFPT 1.0 (13): struct ofp_packet_out, uint8_t[]. */
    OFPRAW_OFPT10_PACKET_OUT,
    /* OFPT 1.1+ (13): struct ofp11_packet_out, uint8_t[]. */
    OFPRAW_OFPT11_PACKET_OUT,

    /* OFPT 1.0 (14): struct ofp10_flow_mod, struct ofp_action_header[]. */
    OFPRAW_OFPT10_FLOW_MOD,
    /* OFPT 1.1+ (14): struct ofp11_flow_mod, struct ofp11_instruction[]. */
    OFPRAW_OFPT11_FLOW_MOD,
    /* NXT 1.0+ (13): struct nx_flow_mod, uint8_t[8][]. */
    OFPRAW_NXT_FLOW_MOD,

    /* OFPT 1.0 (15): struct ofp10_port_mod. */
    OFPRAW_OFPT10_PORT_MOD,
    /* OFPT 1.1+ (16): struct ofp11_port_mod. */
    OFPRAW_OFPT11_PORT_MOD,

    /* OFPT 1.0 (18): void. */
    OFPRAW_OFPT10_BARRIER_REQUEST,
    /* OFPT 1.1 (20): void. */
    OFPRAW_OFPT11_BARRIER_REQUEST,

    /* OFPT 1.0 (19): void. */
    OFPRAW_OFPT10_BARRIER_REPLY,
    /* OFPT 1.1 (21): void. */
    OFPRAW_OFPT11_BARRIER_REPLY,

    /* OFPT 1.0 (22): struct ofp_packet_remote, uint8_t[]. */
    OFPRAW_OFPT10_PACKET_REMOTE,

/* Standard statistics. */
```

```
/* OFPST 1.0+ (0): void. */
OFPRAW_OFPST_DESC_REQUEST,

/* OFPST 1.0+ (0): struct ofp_desc_stats. */
OFPRAW_OFPST_DESC_REPLY,

/* OFPST 1.0 (1): struct ofp10_flow_stats_request. */
OFPRAW_OFPST10_FLOW_REQUEST,
/* OFPST 1.1+ (1): struct ofp11_flow_stats_request, uint8_t[8][]. */
OFPRAW_OFPST11_FLOW_REQUEST,
/* NXST 1.0 (0): struct nx_flow_stats_request, uint8_t[8][]. */
OFPRAW_NXST_FLOW_REQUEST,

/* OFPST 1.0 (1): uint8_t[]. */
OFPRAW_OFPST10_FLOW_REPLY,
/* OFPST 1.1+ (1): uint8_t[]. */
OFPRAW_OFPST11_FLOW_REPLY,
/* NXST 1.0 (0): uint8_t[]. */
OFPRAW_NXST_FLOW_REPLY,

/* OFPST 1.0 (2): struct ofp10_flow_stats_request. */
OFPRAW_OFPST_AGGREGATE_REQUEST,
/* NXST 1.0 (1): struct nx_flow_stats_request, uint8_t[8][]. */
OFPRAW_NXST_AGGREGATE_REQUEST,

/* OFPST 1.0 (2): struct ofp_aggregate_stats_reply. */
OFPRAW_OFPST_AGGREGATE_REPLY,
/* NXST 1.0 (1): struct ofp_aggregate_stats_reply. */
OFPRAW_NXST_AGGREGATE_REPLY,

/* OFPST 1.0-1.2 (3): void. */
OFPRAW_OFPST_TABLE_REQUEST,
```

```
    /* OFPST 1.0 (3): struct ofp10_table_stats[]. */
    OFPRAW_OFPST10_TABLE_REPLY,
    /* OFPST 1.1 (3): struct ofp11_table_stats[]. */
    OFPRAW_OFPST11_TABLE_REPLY,
    /* OFPST 1.2 (3): struct ofp12_table_stats[]. */
    OFPRAW_OFPST12_TABLE_REPLY,

    /* OFPST 1.0 (4): struct ofp10_port_stats_request. */
    OFPRAW_OFPST_PORT_REQUEST,

    /* OFPST 1.0 (4): struct ofp10_port_stats[]. */
    OFPRAW_OFPST_PORT_REPLY,

    /* OFPST 1.0 (5): struct ofp10_queue_stats_request. */
    OFPRAW_OFPST_QUEUE_REQUEST,

    /* OFPST 1.0 (5): struct ofp10_queue_stats[]. */
    OFPRAW_OFPST_QUEUE_REPLY,

    /* OFPST 1.0 (13): void. */
    OFPRAW_OFPST_PORT_DESC_REQUEST,

    /* OFPST 1.0 (13): struct ofp10_phy_port[]. */
    OFPRAW_OFPST_PORT_DESC_REPLY,

/* Nicira extension messages.
 *
 * Nicira extensions that correspond to standard OpenFlow messages are listed
 * alongside the standard versions above. */

    /* NXT 1.0+ (10): struct nx_role_request. */
    OFPRAW_NXT_ROLE_REQUEST,
```

```
    /* NXT 1.0+ (11): struct nx_role_request. */
    OFPRAW_NXT_ROLE_REPLY,

    /* NXT 1.0+ (12): struct nx_set_flow_format. */
    OFPRAW_NXT_SET_FLOW_FORMAT,

    /* NXT 1.0+ (15): struct nx_flow_mod_table_id. */
    OFPRAW_NXT_FLOW_MOD_TABLE_ID,

    /* NXT 1.0+ (16): struct nx_set_packet_in_format. */
    OFPRAW_NXT_SET_PACKET_IN_FORMAT,

    /* NXT 1.0+ (18): void. */
    OFPRAW_NXT_FLOW_AGE,

    /* NXT 1.0+ (19): struct nx_async_config. */
    OFPRAW_NXT_SET_ASYNC_CONFIG,

    /* NXT 1.0+ (20): struct nx_controller_id. */
    OFPRAW_NXT_SET_CONTROLLER_ID,

    /* NXT 1.0+ (21): struct nx_flow_monitor_cancel. */
    OFPRAW_NXT_FLOW_MONITOR_CANCEL,

    /* NXT 1.0+ (22): void. */
    OFPRAW_NXT_FLOW_MONITOR_PAUSED,

    /* NXT 1.0+ (23): void. */
    OFPRAW_NXT_FLOW_MONITOR_RESUMED,

/* Nicira extension statistics.
 *
 * Nicira extension statistics that correspond to standard OpenFlow statistics
```

```
 * are listed alongside the standard versions above. */

   /* NXST 1.0 (2): uint8_t[8][]. */
   OFPRAW_NXST_FLOW_MONITOR_REQUEST,

   /* NXST 1.0 (2): uint8_t[8][]. */
   OFPRAW_NXST_FLOW_MONITOR_REPLY,
};
```

3.2.2. ofp-util.def

在 lib/ofp-util.def 中添加对应映射关系，格式为 行动类型+数据结构+名字。

```
#ifndef OFPAT10_ACTION
#define OFPAT10_ACTION(ENUM, STRUCT, NAME)
#endif
OFPAT10_ACTION(OFPAT10_OUTPUT,      ofp10_action_output, "output")
OFPAT10_ACTION(OFPAT10_SET_VLAN_VID, ofp_action_vlan_vid, "mod_vlan_vid")
OFPAT10_ACTION(OFPAT10_SET_VLAN_PCP, ofp_action_vlan_pcp, "mod_vlan_pcp")
OFPAT10_ACTION(OFPAT10_STRIP_VLAN,   ofp_action_header,   "strip_vlan")
OFPAT10_ACTION(OFPAT10_SET_DL_SRC,   ofp_action_dl_addr,  "mod_dl_src")
OFPAT10_ACTION(OFPAT10_SET_DL_DST,   ofp_action_dl_addr,  "mod_dl_dst")
OFPAT10_ACTION(OFPAT10_SET_NW_SRC,   ofp_action_nw_addr,  "mod_nw_src")
OFPAT10_ACTION(OFPAT10_SET_NW_DST,   ofp_action_nw_addr,  "mod_nw_dst")
OFPAT10_ACTION(OFPAT10_SET_NW_TOS,   ofp_action_nw_tos,   "mod_nw_tos")
OFPAT10_ACTION(OFPAT10_SET_TP_SRC,   ofp_action_tp_port,  "mod_tp_src")
OFPAT10_ACTION(OFPAT10_SET_TP_DST,   ofp_action_tp_port,  "mod_tp_dst")
OFPAT10_ACTION(OFPAT10_ENQUEUE,     ofp_action_enqueue,  "enqueue")
OFPAT10_ACTION(OFPAT10_REMOTE,      ofp_action_remote, "remote")
```

3.2.3. ofp-actions.h

添加新行动的映射关系：类型（OFPACT_REMOTE）和对应的数据结构 struct ofpact_remote。

```
/* List of OVS abstracted actions.
 *
 * This macro is used directly only internally by this header, but the list is
 * still of interest to developers.
 *
 * Each DEFINE_OFPACT invocation has the following parameters:
 *
 * 1. <ENUM>, used below in the enum definition of OFPACT_<ENUM>, and
 *    elsewhere.
 *
 * 2. <STRUCT> corresponding to a structure "struct <STRUCT>", that must be
 *    defined below.  This structure must be an abstract definition of the
 *    action.  Its first member must have type "struct ofpact" and name
 *    "ofpact".  It may be fixed length or end with a flexible array member
 *    (e.g. "int member[];").
 *
 * 3. <MEMBER>, which has one of two possible values:
 *
 *       - If "struct <STRUCT>" is fixed-length, it must be "ofpact".
 *
 *       - If "struct <STRUCT>" is variable-length, it must be the name of the
 *         flexible array member.
 */
#define OFPACTS                                     \
    /* Output. */                                   \
    DEFINE_OFPACT(OUTPUT,        ofpact_output,      ofpact)   \
    DEFINE_OFPACT(CONTROLLER,    ofpact_controller,  ofpact)   \
    DEFINE_OFPACT(ENQUEUE,       ofpact_enqueue,     ofpact)   \
    DEFINE_OFPACT(REMOTE,        ofpact_remote,      ofpact)   \
    DEFINE_OFPACT(OUTPUT_REG,    ofpact_output_reg,  ofpact)   \
    DEFINE_OFPACT(BUNDLE,        ofpact_bundle,      slaves)   \
                                                    \
    /* Header changes. */                           \
```

```
DEFINE_OFPACT(SET_VLAN_VID,   ofpact_vlan_vid,    ofpact)   \
DEFINE_OFPACT(SET_VLAN_PCP,   ofpact_vlan_pcp,    ofpact)   \
DEFINE_OFPACT(STRIP_VLAN,     ofpact_null,        ofpact)   \
DEFINE_OFPACT(SET_ETH_SRC,    ofpact_mac,         ofpact)   \
DEFINE_OFPACT(SET_ETH_DST,    ofpact_mac,         ofpact)   \
DEFINE_OFPACT(SET_IPV4_SRC,   ofpact_ipv4,        ofpact)   \
DEFINE_OFPACT(SET_IPV4_DST,   ofpact_ipv4,        ofpact)   \
DEFINE_OFPACT(SET_IPV4_DSCP,  ofpact_dscp,        ofpact)   \
DEFINE_OFPACT(SET_L4_SRC_PORT, ofpact_l4_port,    ofpact)   \
DEFINE_OFPACT(SET_L4_DST_PORT, ofpact_l4_port,    ofpact)   \
DEFINE_OFPACT(REG_MOVE,       ofpact_reg_move,    ofpact)   \
DEFINE_OFPACT(REG_LOAD,       ofpact_reg_load,    ofpact)   \
DEFINE_OFPACT(DEC_TTL,        ofpact_cnt_ids,     cnt_ids)  \
                                                    \
/* Metadata. */                                     \
DEFINE_OFPACT(SET_TUNNEL,     ofpact_tunnel,      ofpact)   \
DEFINE_OFPACT(SET_QUEUE,      ofpact_queue,       ofpact)   \
DEFINE_OFPACT(POP_QUEUE,      ofpact_null,        ofpact)   \
DEFINE_OFPACT(FIN_TIMEOUT,    ofpact_fin_timeout, ofpact)   \
                                                    \
/* Flow table interaction. */                       \
DEFINE_OFPACT(RESUBMIT,       ofpact_resubmit,    ofpact)   \
DEFINE_OFPACT(LEARN,          ofpact_learn,       specs)    \
                                                    \
/* Arithmetic. */                                   \
DEFINE_OFPACT(MULTIPATH,      ofpact_multipath,   ofpact)   \
DEFINE_OFPACT(AUTOPATH,       ofpact_autopath,    ofpact)   \
                                                    \
/* Other. */                                        \
DEFINE_OFPACT(NOTE,           ofpact_note,        data)     \
DEFINE_OFPACT(EXIT,           ofpact_null,        ofpact)
```

添加新的抽象 OVS 行动结构体为

```
/* OFPACT_REMOTE.
 *
 * Used for OFPAT10_REMOTE. */
struct ofpact_remote {
    struct ofpact ofpact;
    uint16_t port;            /* Output port. */
    uint32_t ip;            /* Remote ip. */
};
```

3.2.4. ofp-actions.c

添加对 openflow 消息的处理 remote_from_openflow10()。

```
static enum ofperr
remote_from_openflow10(const struct ofp_action_remote *oar,
                       struct ofpbuf *out)
{
#ifdef DEBUG
   VLOG_INFO(">>>remote_from_openflow10():oar->len=%u,port=%u,ip=0x%x",ntohs(oar->len),ntohs(oar->port),ntohl(oar->ip));
#endif
   struct ofpact_remote *remote;

   remote = ofpact_put_REMOTE(out);
   /*
   struct ofpact *ofpact;
   ofpact_pad(out);
   ofpact = out->l2 = ofpbuf_put_uninit(out, sizeof(struct ofpact_remote));
   ofpact_init(ofpact, OFPACT_REMOTE, sizeof(struct ofpact_remote));
   remote = ofpact;
   */

   remote->port = ntohs(oar->port);
   remote->ip = ntohl(oar->ip);
#ifdef DEBUG
   VLOG_INFO("<<<remote_from_openflow10(): remote->len=%u,port=%u,ip=0x%x",remote->ofpact.len,remote->port,remote->ip);
#endif

   return ofputil_check_output_port(remote->port, OFPP_MAX);
}
```

注意，如果新的行动结构大小为 8，并且已经添加到 union ofp_action 中，则在 ofpact_from_openflow10()中添加对应的处理。

添加 ofpact_from_openflow10_remote()函数。

```
static enum ofperr
ofpact_from_openflow10_remote(const struct ofp_action_remote *a, struct ofpbuf *out)
{
   enum ofputil_action_code code;
   enum ofperr error;

   error = decode_openflow10_action((const union ofp_action*)a, &code);//get the code
   if (error) {
      return error;
   }

#ifdef DEBUG
   VLOG_INFO(">>>ofpact_from_openflow10_remote():
code=%u,len=%u,port=%u,ip=0x%x", code,ntohs(a->len),ntohs(a->port),ntohl(a->ip));
#endif
   if(code == OFPUTIL_OFPAT10_REMOTE) {
#ifdef DEBUG
      VLOG_INFO("handle OFPUTIL_OFPAT10_REMOTE");
#endif
      return remote_from_openflow10(a, out);
   }
#ifdef DEBUG
   VLOG_INFO("<<<ofpact_from_openflow10_remote()");
#endif

   return error;
}
```

添加一批 openflow 行动到 ovs 行动的转换函数 ofpacts_from_openflow_remote()。

```
static enum ofperr
ofpacts_from_openflow_remote(const struct ofp_action_remote *in, size_t n_in,
                  struct ofpbuf *out,
                  enum ofperr (*ofpact_from_openflow)(
                     const struct ofp_action_remote *a, struct ofpbuf *out))
{
   const struct ofp_action_remote *a;
   size_t left;

#ifdef DEBUG
   VLOG_INFO(">>>ofpacts_from_openflow_remote(): n_in=%u, in->port=%u,in->ip=0x%x",n_in,ntohs(in->port),ntohl(in->ip));
#endif
   ACTION_FOR_EACH_REMOTE (a, left, in, n_in) {
#ifdef DEBUG
      VLOG_INFO("a->len=%u",ntohs(a->len));
#endif
      enum ofperr error = ofpact_from_openflow(a, out);
      if (error) {
         log_bad_action((const union ofp_action *)in, n_in, a - in, error);
         return error;
      }
   }
   if (left) {
#ifdef DEBUG
      VLOG_WARN("still left %u bytes",left);
#endif
      enum ofperr error = OFPERR_OFPBAC_BAD_LEN;
      log_bad_action(in, n_in, n_in - left, error);
      return error;
   }

   ofpact_pad(out);
```

```
#ifdef DEBUG
      VLOG_INFO("<<<ofpacts_from_openflow_remote() done");
#endif
   return 0;
}
```

OVS 抽象行动到 openflow10 的行动转换函数 ofpact_remote_to_openflow10()。

```
static void
ofpact_remote_to_openflow10(const struct ofpact_remote *remote,
                        struct ofpbuf *out)
{
   struct ofp_action_remote *oar;

   oar = ofputil_put_OFPAT10_REMOTE(out);
   oar->port = htons(remote->port);
   oar->ip = htonl(remote->ip);
}
```

在 ofpact_to_openflow10()函数中添加对新行动的处理分支。

```
case OFPACT_REMOTE:
      ofpact_remote_to_openflow10(ofpact_get_REMOTE(a), out);
      break;
```

3.3. OVS 抽象消息处理

在 lib/ofp-util.h 中定义抽象的消息结构，在 lib/ofp-util.c 中定义对该新消息的解析函数。

3.3.1. ofp-util.h

添加消息结构体 struct ofputil_packet_remote。

```
struct ofputil_packet_remote {
    const void *packet;        /* Packet data, if buffer_id == UINT32_MAX. */
    size_t packet_len;         /* Length of packet data in bytes. */
    uint32_t buffer_id;        /* Buffer id or UINT32_MAX if no buffer. */
    uint16_t in_port;          /* Packet's input port. */
    struct ofpact *ofpacts;    /* Actions. */
    size_t ofpacts_len;        /* Size of ofpacts in bytes. */
};
```

并声明对该消息的解析函数

```
enum ofperr
ofputil_decode_packet_remote(struct ofputil_packet_remote *pr,
                             const struct ofp_header *oh, struct ofpbuf *ofpacts);
```

3.3.2. ofp-util.c

定义对新消息的解析函数 ofputil_decode_packet_remote()。

```
enum ofperr
ofputil_decode_packet_remote(struct ofputil_packet_remote *pr,
                    const struct ofp_header *oh,
                    struct ofpbuf *ofpacts)
{
#ifdef DEBUG
      VLOG_INFO(">>>ofputil_decode_packet_remote(), oh_len=%u",ntohs(oh->length));
#endif
   enum ofperr bad_in_port_err;
   enum ofpraw raw;
   struct ofpbuf b;

   ofpbuf_use_const(&b, oh, ntohs(oh->length)); //b->data points to oh
   raw = ofpraw_pull_assert(&b);

   if (raw == OFPRAW_OFPT10_PACKET_REMOTE) {
      enum ofperr error;
      const struct ofp_packet_remote *opr = ofpbuf_pull(&b, sizeof *opr);//pull bytes from
b->data into opr

      pr->buffer_id = ntohl(opr->buffer_id);
      pr->in_port = ntohs(opr->in_port);

#ifdef DEBUG
      VLOG_INFO("act_len=%u",ntohs(opr->actions_len));
#endif
      error = ofpacts_pull_openflow10(&b, ntohs(opr->actions_len), ofpacts);//convert
from b into ofpacts
      if (error) {
         return error;
      }

      bad_in_port_err = OFPERR_NXBRC_BAD_IN_PORT;
   } else {
```

```
        NOT_REACHED();
    }

    if (pr->in_port >= OFPP_MAX && pr->in_port != OFPP_LOCAL
        && pr->in_port != OFPP_NONE && pr->in_port != OFPP_CONTROLLER) {
        VLOG_WARN_RL(&bad_ofmsg_rl, "packet-remote has bad input port %#"PRIx16,
                     pr->in_port);
        return bad_in_port_err;
    }

    pr->ofpacts = ofpacts->data;
    pr->ofpacts_len = ofpacts->size;

    if (pr->buffer_id == UINT32_MAX) {
        pr->packet = b.data;
        pr->packet_len = b.size;
    } else {
        pr->packet = NULL;
        pr->packet_len = 0;
    }
#ifdef DEBUG
    VLOG_INFO("<<<ofputil_decode_packet_remote(): ofpacts->act_type=%u,
ofpacts_len=%u, packet_len=%u",pr->ofpacts->type,pr->ofpacts_len,pr->packet_len);
#endif
    return 0;
}
```

3.4. 完整流程 handle_packet_remote()

对 openflow 消息的处理都是在 ofproto/ofproto.c 中的 handle_openflow__()函数中根据消息头类型进行调用不同的处理函数。

例如

```
case OFPTYPE_PACKET_REMOTE:
#ifdef DEBUG
    VLOG_INFO("handle PKT_REMOTE from controller.");
#endif
    return handle_packet_remote(ofconn, oh);
```

以 REMOTE 行为为例，在 ofproto/ofproto.c 中添加 handle_packet_remote()函数，完成对从控制器收到新消息的完整响应和处理。包括解析消息头中的行动和负载信息，并创建发送到 datapath 的具体指令，发出 remote 信息。

3.4.1. 解析消息：ofputil_decode_packet_remote()

主要调用 ofpacts_pull_openflow10()来解析消息头，该函数位于 lib/ofp-actions.c。依次调用 ofpacts_pull_openflow10()→ofpacts_pull_actions()。

如果新添加行动结构大小跟 union ofp_aciton 的大小一致（8 字节），则直接在 union ofp_aciton 中添加声明即可，并无需更多改动，会调用 ofpacts_from_openflow10()进行处理。

如果大小不一致，则需要指定调用自己定义的函数，此处为 ofpacts_from_openflow10_remote()→ofpacts_from_openflow_remote()→ofpact_from_openflow_remote()。

其中 ofpact_from_openflow_remote()对单独一条 remote 行动进行解析，代码为

```
static enum ofperr
ofpact_from_openflow10_remote(const struct ofp_action_remote *a, struct ofpbuf *out)
{
   enum ofputil_action_code code;
   enum ofperr error;

   error = decode_openflow10_action((const union ofp_action*)a, &code);//get the code
   if (error) {
      return error;
   }

#ifdef DEBUG
   VLOG_INFO(">>>ofpact_from_openflow10_remote():
code=%u,len=%u,port=%u,ip=0x%x", code,ntohs(a->len),ntohs(a->port),ntohl(a->ip));
#endif
   if(code == OFPUTIL_OFPAT10_REMOTE) {
#ifdef DEBUG
      VLOG_INFO("handle OFPUTIL_OFPAT10_REMOTE");
#endif
      return remote_from_openflow10(a, out);
   }
#ifdef DEBUG
   VLOG_INFO("<<<ofpact_from_openflow10_remote()");
#endif

   return error;
}
```

3.4.2. 执行行动：packet_remote()

主要修改 ofproto/ofproto-dpif.c，实现 packet_remote()，以完成将 REMOTE 行动指令发给 datapath 的过程。

主要设计添加的新函数包括如下

构造发给 datapath 的 remote 行动

```
/**
 * compose remote action, stored in ctx->odp_actions.
 */
static void
compose_remote_action__(struct action_xlate_ctx *ctx, uint16_t ofp_port, uint32_t ip,
                        bool check_stp)
{
    const struct ofport_dpif *ofport = get_ofp_port(ctx->ofproto, ofp_port);
    uint16_t odp_port = ofp_port_to_odp_port(ofp_port);
    ovs_be16 flow_vlan_tci = ctx->flow.vlan_tci;
    uint8_t flow_nw_tos = ctx->flow.nw_tos;
    uint16_t out_port;

    if (ofport) {
        struct priority_to_dscp *pdscp;

        if (ofport->up.pp.config & OFPUTIL_PC_NO_FWD
            || (check_stp && !stp_forward_in_state(ofport->stp_state))) {
            return;
        }

        pdscp = get_priority(ofport, ctx->flow.skb_priority);
        if (pdscp) {
            ctx->flow.nw_tos &= ~IP_DSCP_MASK;
            ctx->flow.nw_tos |= pdscp->dscp;
        }
    } else {
        /* We may not have an ofport record for this port, but it doesn't hurt
         * to allow forwarding to it anyhow.  Maybe such a port will appear
         * later and we're pre-populating the flow table.  */
    }

    out_port = vsp_realdev_to_vlandev(ctx->ofproto, odp_port,
```

```
                                ctx->flow.vlan_tci);
#ifdef DEBUG
      VLOG_INFO(">>>compose_remote_action__():
odp_port=%u,out_port=%u,ip=0x%x",odp_port,out_port,ip);
#endif
   if (out_port != odp_port) {
      ctx->flow.vlan_tci = htons(0);
   }
   commit_odp_actions(&ctx->flow, &ctx->base_flow, ctx->odp_actions); //check flow key
   nl_msg_put_u64(ctx->odp_actions, OVS_ACTION_ATTR_REMOTE,
((uint64_t)out_port<<32)+ip); //add output port,ip


   ctx->sflow_odp_port = odp_port;
   ctx->sflow_n_outputs++;
   ctx->nf_output_iface = ofp_port;
   ctx->flow.vlan_tci = flow_vlan_tci;
   ctx->flow.nw_tos = flow_nw_tos;
#ifdef DEBUG
      VLOG_INFO("<<<compose_remote_action__(): size=%u, nla_data=0x%llx",ctx-
>odp_actions->size,nl_attr_get_u64(ctx->odp_actions->data));
#endif
}
static void
compose_remote_action(struct action_xlate_ctx *ctx, uint16_t ofp_port, uint32_t ip)
{
#ifdef DEBUG
      VLOG_INFO(">>>compose_remote_action(): port=%u,ip=0x%x",ofp_port,ip);
#endif
   compose_remote_action__(ctx, ofp_port, ip, false);//no need to check stp actually
#ifdef DEBUG
      VLOG_INFO("<<<compose_remote_action() done");
#endif
}
```

将 remote 行动指令发给 datapath。

```
static void
xlate_remote_action(struct action_xlate_ctx *ctx,
                uint16_t port, uint32_t ip)
{
#ifdef DEBUG
      VLOG_INFO(">>>xlate_remote_action(): port=%u, ip=0x%x.",port,ip);
#endif
   uint16_t prev_nf_output_iface = ctx->nf_output_iface;

   ctx->nf_output_iface = NF_OUT_DROP;
   if (port != ctx->flow.in_port) {
      compose_remote_action(ctx, port,ip);
   }

   if (prev_nf_output_iface == NF_OUT_FLOOD) {
      ctx->nf_output_iface = NF_OUT_FLOOD;
   } else if (ctx->nf_output_iface == NF_OUT_DROP) {
      ctx->nf_output_iface = prev_nf_output_iface;
   } else if (prev_nf_output_iface != NF_OUT_DROP &&
            ctx->nf_output_iface != NF_OUT_FLOOD) {
      ctx->nf_output_iface = NF_OUT_MULTI;
   }
#ifdef DEBUG
      VLOG_INFO("<<<xlate_remote_action() done.");
#endif
}
```

在 do_xlate_actions()函数中添加对新行动的处理分支，调用 xlate_remote_action()。

```
      case OFPACT_REMOTE:
         xlate_remote_action(ctx, ofpact_get_REMOTE(a)->port,
               ofpact_get_REMOTE(a)->ip);
         break;
```

3.5. 新建 netlink 消息属性

修改 include/linux/openvswitch.h 文件中 enum ovs_action_attr{}。

```
enum ovs_action_attr {
  OVS_ACTION_ATTR_UNSPEC,
  OVS_ACTION_ATTR_OUTPUT,       /* u32 port number. */
  OVS_ACTION_ATTR_USERSPACE,    /* Nested OVS_USERSPACE_ATTR_*. */
  OVS_ACTION_ATTR_SET,        /* One nested OVS_KEY_ATTR_*. */
  OVS_ACTION_ATTR_PUSH_VLAN,    /* struct ovs_action_push_vlan. */
  OVS_ACTION_ATTR_POP_VLAN,     /* No argument. */
  OVS_ACTION_ATTR_SAMPLE,      /* Nested OVS_SAMPLE_ATTR_*. */
#ifdef LC_ENABLE
  OVS_ACTION_ATTR_REMOTE,       /* encapulate and send pkt to remote sw. */
#endif
  __OVS_ACTION_ATTR_MAX
};
```

4. OpenvSwitch 内核态

主要是实现对封装了 packet_remote 行动的 netlink 消息的处理。

内核态对包消息处理，在 datapath/datapath.c 中的 ovs_packet_cmd_execute()函数中。

首先，调用 validate_actions()对 action 域的长度进行验证，需要添加自定义的行动长度。

之后，调用 ovs_execute_actions()→do_execute_actions()。该函数位于 datapath/actions.c，添加对 OVS_ACTION_ATTR_REMOTE 属性的处理。

```
case OVS_ACTION_ATTR_REMOTE: //TODO: print to test here.
        payload = nla_get_u64(a);
        remote_ip = payload & 0xffffffff; //remote_ip
        prev_port = (payload >>32) & 0xffffffff; //port_no
#ifdef DEBUG
        pr_info(">>>DP will execute remote cmd, encap first: oport=%u, ip=0x%x\n",prev_port,remote_ip);
#endif
        do_remote_encapulation(dp,skb,remote_ip); //encapulate with new l2 and l3 header
        break;
```