ISBN: 978-89-6211-682-3

오픈플로우 프로토콜(OpenFlow Protocol) 분석

일 자: 2010년 11월 12일

부 서: 슈퍼컴퓨팅 본부/융합자원실

제출자: 권윤주, 홍원택

KiSTi www.kisti.re.kr
한국과학기술정보연구원
Korea Institute of Science and Technology Information
305-806 대전광역시 유성구 어은동 52번지
TEL (042)869-0676 / FAX (042)869-0679
www.kisti.re.kr

오픈플로우 프로토콜 분석

작성자: 권윤주, 홍원택

융합자원실, 한국과학기술정보연구원

{yulli,wthong}@kisti.re.kr

최종수정일: 2010년 11월 12일

Abstract

본 문서는 로컬 네트워크에 OpenFlow 네트워크 테스트베드를 구성하여 NOX(Controller)의 가장 기본적인 컴포넌트인 pyswitch, packetdump를 구동시키고, 그에 따라 OpenFlow 네트워크의 호스트들을 연결하는 데 있어서 Controller와 OF 스위치간 주고받는 OF 프로토콜들을 분석해보고자 한다.

Topics

1. 서론

2. OpenFlow Protocol의 개요

2.1 OpenFlow Protocol Message Format

2.2 OpenFlow Protocol Message Types

3. 각 이벤트에 따른 메시지 교환

3.1 Connection Setup

3.2 Packet In

3.3 Flow Entry Modification

3.4 Flow Entry Removal

4. 결론

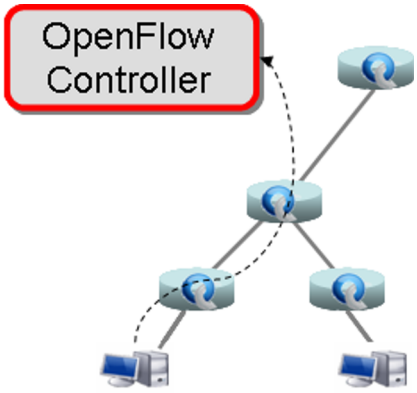
1. 서론

미래인터넷에 대한 연구와 네트워크 가상화 분야에 대한 연구가 지속되고 있는 가운데 선진국에서는 미래인터넷 프로젝트들을 통하여 미래인터넷분야에 대한 기술 및 표준화 선점을 위해 연구에 주력하고 있다. 이러한 배경 속에서 스탠포드 대학에서는 OpenFlow라는 새로운 개념의 네트워크 테스트 환경을 개발했는데 이것은 실험망과 운영망의 분리를 통하여 실제 네트워크에 영향을 주지 않으면서 실제 네트워크 환경에서 연구자의 새로운 네트워크 프로토콜 및 실험이 가능한 기술이다.

특히, OpenFlow 기술은 IP 라우팅 의존도를 낮추고 다양한 메트릭들을 통해 라우팅을 가능하게 함으로써 다양한 방식으로 네트워크를 운영할 수 있다. 이러한 형태로 네트워크를 운영시키기 위해서 OpenFlow 네트워크 내에서는 라우팅 결정 엔진인 NOX와 패킷 포워딩 엔진인 OF 스위치간 긴밀한 통신을 수행하야 하는 데, 이 때 사용되는 것이 오픈플로우 프로토콜(OpenFlow Protocol)이다.

본 문서에서는 로컬 네트워크에 OpenFlow 네트워크 테스트베드를 구성하여 NOX(Controller)의 가장 기본적인 컴포넌트인 pyswitch, packetdump를 구동시키고, 그에 따라 OpenFlow 네트워크의 호스트들을 연결하는 데 있어서 Controller와 OF 스위치간 주고받는 OF 프로토콜들을 분석해보고자 한다.

2. OpenFlow(OF) 프로토콜 개요



[그림 1] OpenFlow 네트워크

OpenFlow(네트워크 테스트베드)는 왼쪽 그림과 같이 다 수개의 OF 스위치와 하나의 Controller로 구성되어 있으며, OF 스위치는 패킷 포워딩하는 역할을, Controller는 해당 네트워크에 진입하는 패킷들에 대한 경로 설정 등의 액션을 정의하는 역할을 한다.

각각의 OF 스위치는 OF 네트워크에 처음 진입하는 패킷을 모두 Controller로 전달해주고, Controller는 아래 그림과 같이 10개의 튜플을 이용하여 OF 스위치에게 해당 플로우에 대한 action을 정의해준다. 이때, Controller는 10개의 튜플을 모두 사용할 수도 있고, 자기가 사용하지 않는 요소에 대해서는 'wildcard'를 이용하여 배제시킬 수도 있다.

Ingress port	Ether src	Ether dst	Ehter type	VLAN id	IP src	IP dst	IP proto	TCP/UDP src port	TCP/UDP dst port

[그림 2] OpenFlow의 플로으(flow entry header)

이처럼 OF 스위치와 Controller는 OF 네트워크의 패킷들에 대한 경로 설정 및 관리를 위해 다양한 메시지를 주고 받게 되는 데 이때 사용하게 되는 것이 OF 프로토콜이다. OF 프로토콜은 Controller와 OF 스위치 간에 Secure channel을 통해 OpenFlow 네트워크를 관리하는 규약이며, 다음과 같이 3가지 메시지 타입을 지원한다.

- Controller-to-Switch 메시지: controller에 의해 시작(initiate)되고, 직접적으로 switch의 상태를 관리한다거나 감시할 때 사용된다.
- Asynchronous 메시지 : OF 스위치에 의해 시작(initiate)되고 controller의 네트워크 이벤트와 스위치 상태 변화를 갱신하는 데 사용된다.
- Symmetric 메시지 : solicitation 없이 controller 또는 OF 스위치에 의해서 시작(initiate)될 수 있다.

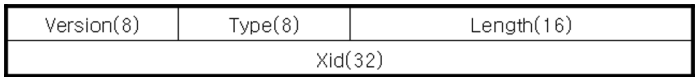
2.1 OpenFlow Protocol Message Format

OpenFlow 프로토콜(OpenFlow Protocol, OFP) 메시지 포맷은 아래 그림에서 보는 것처럼 헤더와 페이로드로 구성되어 있다.

OFP header	OFP 타입에 따른 메시지 포맷의 페이로드

[그림 3] OFP 메시지 포맷

OFP 헤더는 아래 그림에서 보는 것과 같이 OpenFlow 스위치 버전(version), OFP 타입(type), OFP 헤더를 포함한 전체길이(length), 본 패킷을 식별하는 트랜잭션 ID(xid)로 구성되어 있다.



[그림 4] OFP 헤더 구조

OFP 페이로드는 OFP 헤더내의 'type' 필드 값에 따라 해당 타입에 맞는 페이로드로 구성된다. OFP 메시지 타입은 다음과 같다.

```
enum ofp_type {
        /* Immutable messages. */
        OFPT_HELLO,
        OFPT_ERROR,
        OFPT_ECHO_REQUEST,
        OFPT_ECHO_REPLY,
        OFPT_VENDOR,

        /* Switch configuration messages. */
        OFPT_FEATURES_REQUEST,
        OFPT_FEATURES_REPLY,
        OFPT_GET_CONFIG_REQUEST,
        OFPT_GET_CONFIG_REPLY,
        OFPT_SET_CONFIG,

        /* Asynchronous messages. */
        OFPT_PACKET_IN,
        OFPT_FLOW_EXPIRED,
        OFPT_PORT_STATUS,

        /* Controller command messages. */
        OFPT_PACKET_OUT,
        OFPT_FLOW_MOD,
        OFPT_PORT_MOD,

        /* Statistics messages. */
        OFPT_STATS_REQUEST,
        OFPT_STATS_REPLY
};
```

[표1] OFP Message Types

2.2 OpenFlow Protocol Message Types

2.2.1 Controller-to-Switch

본 메시지는 controller에 의해 시작되며 OF 스위치로부터 응답이 요구될 수도 있고 안될 수도 있다.

(1) Features 관련 메시지

▶ OFP_TYPE : OFPT_FEATURES_REQUEST, OFPT_FEATURES_REPLY

▶ SSL session이 연결된 상태에서 controller가 switch에 대한 feature들을 요청하는 메시지로서, OFPT_FEATURES_REQUEST는 body없이 OFP 헤더만으로 구성되어있고, OFPT_FEATURES_REPLY는 다음과 같은 구조로 body를 구성한다.

```
struct ofp_switch_features{
        struct ofp_header header

        uint64_t datapath_id;
        uint32_t n_buffers;         /* Max packets buffered at once. */
        uint8_t n_tables;  /* Number of tables supported by datapath. */
        uint8_t pad[3];             /* Align to 64-bits */

        /* Features. */
        uint32_t capabilities;
        uint32_t actions;

        /* Port info. */
        struct ofp_phy_port ports[0];       /* Port definitions */

};
OFP葦ASSERT(sizeof(struct ofp_switch_features) == 32);
```

[표 2] OFPT_FEATURES_REPLY 내용

datapath에 의해 지원되는 capabilities는 다음과 같다.

```
/* Capabilities supported by the datapath. */
enum ofp_capabilities {
cOFPC_FLOW_STATS   = 1 << 0,
        OFPC_TABLE_STATS   = 1 << 1,
        OFPC_PORT_STATS    = 1 << 2,
        OFPC_STP           = 1 << 3,
        OFPC_MULTI_PHY_TX  = 1 << 4,
        OFPC_IP_REASM      = 1 << 5
};
```

[표 3] Capabilities

(2)설정(Configuration) 관련 메시지

- ▶ OFP_TYPE : OFPT_GET_CONFIG_REQUEST, OFPT_GET_CONFIG_REPLY, OFPT_SET_CONFIG
- ▶ Controller가 OF 스위치 내 파라미터들을 설정하거나 설정된 내용을 요청할 때 사용하는 메시지로서, OFPT_GET_CONFIG_REQUEST는 OFP 헤더만으로 구성되어 있고, OFPT_SET_CONFIG와 OFPT_GET_CONFIG_REPLY는 다음과 같은 바디로 구성되어 있다.

```
/* Switch configuration. */
struct ofp_switch_config {
        struct ofp_header header;
        uint16_t flags; /* OFPC_* flags. */
        uint16_t miss_send_len; /* Max bytes of new flow that datapath should
                        send to the controller. */
};
OFP_ASSERT(sizeof(struct ofp_switch_config) == 12);
```

[표 4] OFPT_GET_CONFIG_REPLY/OFPT_SET_CONFIG 내용

OFPC_* 플래그는 아래와 같다.

```
enum ofp_config_flags {
/* Tells datapath to notify the controller of expired flow entries. */
        OFPC_SEND_FLOW_EXP = 1 << 0,
/* Handling of IP fragments. */
        OFPC_FRAG_NORMAL = 0 << 1, /* No special handling for fragments. */
        OFPC_FRAG_DROP = 1 << 1, /* Drop fragments. */
        OFPC_FRAG_REASM = 2 << 1, /* Reassemble (only if OFPC_IP_REASM
set). */
        OFPC_FRAG_MASK = 3 << 1
};
```

[표 5] OFPC_* 플래그

OFPC_SEND_FLOW_EXP 플래그가 세팅되면, OF 스위치는 플로우 엔트리의 timeout이 끝날 때 Controller로 Flow Expiration 메시지를 보낸다. 디폴트값은 0으로, 이것은 Flow Expiration 메시지 보내지 않는 것을 의미한다.
OFPC_FRAG_* 플래그는 IP 프레그먼트들이 어떻게 처리될 지를 명시한다. 처리방식에는 NORMAL, DROP, REASSEMBLE 등이 존재한다.

(3) OF 스위치 상태 변화(Modify-State) 관련 메시지

▶ **OFP_TYPE : OFPT_FLOW_MOD, OFPT_PORT_MOD**

▶ Controller가 OF 스위치의 상태를 변경시키기 위해 보내는 메시지로서, 스위치 플로우 테이블과 (물리적) 포트에 대한 변경 내용을 담고 있다.

① **Modify Flow Entry Message** : Controller로부터 OF 스위치의 플로우 테이블에 플로우 엔트리를 추가/수정/삭제하는 메시지

```
/* Flow setup and teardown (controller --> datapath). */

struct ofp_flow_mod{
        struct ofp_header header;
        struct ofp_match match;                 enum ofp_flow_mod_command {

        /* Flow actoins. */
        uint16_t command;                               OFPFC_ADD,
        uint16_t idle_timeout;                          OFPFC_MODIFY,
        uint16_t hard_timeout;                          OFPFC_MODIFY_STRICT,
        uint16_t priority;                              OFPFC_DELETE,
        uinit32_t buffer_id;                            OFPFC_DELETE_STRICT
                                                };
        uinit16_t out_port;

        uint8_t pad[2];
        uint32_t reserved;
        struct ofp_action_header actions[0];
};

OFP_ASSERT(sizeof(struct ofp_flow_mod) == 64);
```

[표 6] OFPT_FLOW_MOD 내용

OF 스위치에서 하나의 플로우를 결정하는 10개 튜플은 'match(struct ofp_match)' 구조체가 가지고 있고, /*Flow actions*/ 이하의 코드에서 해당 플로우에 대한 액션을 정의하고 있다.

② Port Modification Message : Controller가 물리적 포트(physical port)의 행위를 변경하고자 할 때 사용하는 메시지

```
/* Modify behavior of the physical port */
struct ofp_port_mod {
        struct ofp_header header;
        uint16_t port_no;
        uint8_t hw_addr[OFP_ETH_ALEN];

        uint32_t config;
        uint32_t mask;

        uint32_t advertise;
        uint8_t pad[4];
};
OFP_ASSERT(sizeof(struct ofp_port_mod) == 32);
```

[표 7] OFPT_PORT_MOD 내용

(4) OF 스위치 상태 수집(Read-State) 관련 메시지

▶ OFP_TYPE : OFPT_STATS_REQUEST, OFPT_STATS_REPLY)

▶ flow-tables, ports 그리고 플로우 엔트리들의 통계정보를 수집하기 위한 메시지로서, Controller의 요청 메시지와 OF 스위치의 응답 메시지는 아래와 같다.

```
struct ofp_stats_request {
        struct ofp_header header;
        uint16_t type;
        uint16_t flags;
        uint8_t body[0];
};
OFP_ASSERT(sizeof(struct ofp_stats_request) == 12);
```

[표 8] OFPT_STATS_REQUEST 내용

```
struct ofp_stats_reply {
        struct ofp_header header;
        uint16_t type;
        uint16_t flags;
        uint8_t body[0];
};
```

[표 9] OFPT_STATS_REPLY 내용

요청메시지와 응답메시지에 있는 'body[0]'필드의 구성은 요청된 통계 타입('type'필드)에 따라 달라진다.

```
enum ofp_stats_types {
        OFPST_DESC,
        OFPST_FLOW,
        OFPST_AGGREGATE,
        OFPST_TABLE,
        OFPST_PORT,
        OFPST_VENDOR = 0xffff
};
```

[표 10] 통계 타입

다음은 각 통계 타입에 따른 'body[0]'필드 구조 및 내용이다.

① Description Statistics : 요청타입이 OFPST_DESC인 경우, 요청메시지의 body는 없고 응답 메시지는 아래 표와 같이 switch manufacturer, hw revision, sw revision 과 serial number 정보를 갖는다.

```
struct ofp_desc_stats{
        char mfr_desc[DESC_STR_LEN];
        char hw_desc[DESC_STR_LEN];
        char sw_desc[DESC_STR_LEN];
        char serial_num[DESC_STR_LEN];
};
OFP_ASSERT(sizeof(struct ofp_desc_stats) == 800);
```

[표 11] 통계타입이 OFPST_DESC인 경우, 응답메시지의 body[0] 내용

② Indivisual Flow Statistics : 요청 타입이 OFPST_FLOW인 경우. Controller는 개별 플로우에 대한 통계정보를 요청하는 메시지이며, OF 스위치는 가지고 있는 개별 플로우에 대한 통계 정보를 응답 메시지로 보낸다.

```
/* Body for ofp_stats_request of type OFPST_FLOW */
struct ofp_flow_stats_request {
        struct ofp_match match;
        uint8_t table_id;
        uint8_t pad;
        uint16_t out_port;
};
OFP_ASSERT(sizeof(struct ofp_flow_stats_request) == 40);
```

[표 12] 통계타입이 OFPST_FLOW인 경우, 요청메시지의 body[0] 내용

```
/* Body of reply to OFPST_FLOW request. */
struct ofp_flow_stats {
        uint16_t length;
        uint8_t table_id;
        uint8_t pad;

        struct ofp_match match;
        uint32_t duration;
        uint16_t priority;
        uint16_t idle_timeout;
        uint16_t hard_timeout;
        uint16_t pad2[3];
        uint64_t packet_count;
        uint64_t byte_count;
        struct ofp_action header actions[0];
};
OFP_ASSERT(sizeof(struct ofp_flow_stats) == 72);
```

[표 13] 통계타입이 OFPST_FLOW인 경우, 응답메시지의 body[0] 내용

③ Aggregate Flow Statistics : 요청타입이 OFPST_AGGREGATE인 경우, Controller는 다수개의 플로우에 대한 통계정보 요청하는 것으로 요청메시지와 응답메시지의 body는 아래 표와 같다.

```
/* Body for ofp_stats_request of type OFPST_AGGREGATE. */
struct ofp_aggregate_stats_request {
        strcut ofp_match match;
        uint8_t table_id;
        uint8_t pad;
        uint16_t out_port;
};
OFP_ASSERT(sizeof(struct ofp_aggregate_stats_request) == 40);
```

[표 14] 통계타입이 OFPST_AGGREGATE인 경우, 요청메시지의 body[0] 내용

```
/* Body for reply to OFPST_AGGREGATE request. */
struct ofp_aggregate_stats_reply {
        uint64_t packet_count;
        uint64_t byte_count;
        uint32_t flow_count;
        uint8_t pad[4];
};
OFP_ASSERT(sizeof(struct ofp_aggregate_stats_reply) == 24);
```

[표 15] 통계타입이 OFPST_AGGREGATE인 경우, 응답메시지의 body[0] 내용

④ Table Statistics : 요청타입이 OFPST_TABLE인 경우, Controller가 플로우 테이블의 플로우 엔트리들의 통계정보를 얻기 위해 사용되는 메시지로서, 요청시 바디는 없고, 응답 메시지는 아래와 같은 바디 구조를 갖는다.

```
/* Body of reply to OFPST_TABLE request. */
struct ofp_table_stats {
        uint8_t table_id;

        uint8_t pad[3];
        char name[OFP_MAX_TABLE_NAME_LEN];
        uint32_t wildcards;
        uint32_t max_entries; /* Max number of entries supported */
        uint32_t active_count; /* Number of active entries */
        uint64_t lookup_count; /* Number of packets looked up in table */
        uint64_t matched_count; /* Number of packets that hit table */
};
OFP_ASSERT(sizeof(struct ofp_table_stats) == 64);
```

[표 16] 통계타입이 OFPST_TABLE인 경우, 응답메시지의 body[0] 내용

⑤ Port Statistics : OFPST_PORT인 경우, Controller가 OF 스위치의 포트단위의 통계정보를 얻기 위해 사용하는 메시지로서, 요청시 바디는 없고 응답 메시지는 아래와 같은 바디 구조를 갖는다.

```
/* Body of reply to OFPST_PORT request. */

struct ofp_port_stats {
        uint16_t port_no;                       uint64_t tx_errors;
        uint8_t pad[6];
        uint64_t rx_packets;                    uint64_t rx_frame_err;
        uint64_t tx_packets;                    uint64_t rx_over_err;
        uint64_t rx_bytes;                      uint64_t rx_crc_err;
        uint64_t tx_bytes;                      uint64_t collisions;
        uint64_t rx_dropped;            };
        uint64_t tx_dropped;            OFP_ASSERT(sizeof(struct    ofp_port_stats)
        uint64_t rx_errors;             == 104);
```

[표 17] 통계타입이 OFPST_PORT인 경우, 응답메시지의 body[0] 내용

(5) 패킷 보내기 관련 메시지(Send-Packet)

▶ OFP_TYPE : OFPT_PACKET_OUT

▶ Controller가 데이터패스 상에 패킷을 내보내기 위해 사용하는 메시지로서, buffer_id가 '-1'인 경우는 Controller가 'data[0]'에 담긴 데이터를 보내고자 하는 경우이며, 그 외의 경우에는 OF 스위치내에 같은 'buffer_id'를 갖고 있는 패킷을 데이터패스로 전송하게 하는 메시지이다.

```
/* Send packet (controller -> datapath). */
struct ofp_packet_out {
        struct ofp_header header;
        uint32_t buffer_id;          /* ID assigned by datapath (-1 if none). */
        uint16_t in_port;
        uint16_t actions_len;
        struct ofp_action_header actions[0];
        /* uint8_t data[0]; */ /* Packet data. The length is inferred
                               from the length field in the header.
                               (Only meaningful if buffer_id == -1.) */
};
OFP_ASSERT(sizeof(struct ofp_packet_out) == 16);
```

[표 18] OFPT_PACKET_OUT 내용

2.2.2 비동기 메시지 (Asynchronous Messages)

비동기 메시지는 OF 스위치에서 Controller로 solicitation 없이 보내진다. 그리고 이 메시지는 OF 스위치나 네트워크 상태 변화를 Controller에게 알리기 위해 사용된다.

(1) Packet In 메시지

▶ OFP_TYPE : OFPT_PACKET_IN

▶ OF 스위치에 유입된 패킷 중에서 플로우 엔트리에 매칭되지 않는 모든 패킷은 packet-in 메시지를 이용하여 Controller에 통보된다. OF 스위치가 Controller로 보낼 패킷을 버퍼링할 수 있는 정도의 메모리를 가지고 있으면 packet-in 메시지는 **패킷 헤더 일부**(default 128bytes)와 OF 스위치가 패킷을 포워드할 때 Controller에 의해 사용될 **buffer ID**만 포함한다. 메모리에 여유없는 OF 스위치의 경우 packet-in 메시지는 full-packet을 포함하여 controller로 보내야만 한다.

```
/* Packet received on port (datapath -> controller). */
struct ofp_packet_in {
        struct ofp_header header;
        uint32_t buffer_id;         /* ID assigned by datapath. */
        uint16_t total_len;/* full length of frame. */
        uint16_t in_port;           /* Port on which frame was received. */

        uint8_t reason;
        uint8_t pad;
        uint8_t data[0];
};
OFP_ASSERT(sizeof(struct ofp_packet_in) == 20);

/* Why is this packet being sent to the controller? */
enum ofp_packet_in_reason {
        OFPR_NO_MATCH,
        OFPR_ACTION
};
```

[표 19] OFP_PACKET_IN 내용

(2) Flow Expiration 관련 메시지

▶ OFP_TYPE : OFPT_FLOW_EXPIRED

▶ 플로우 엔트리가 OF 스위치에 추가될 때, hard timeout 뿐만이 아니라 idle timeout도 포함된다. ‘idle timeout’은 해당 플로우가 어떤 액티비티 없이 존재하고 있을 때 엔트리가 삭제되는 시간을 의미하고, ‘hard timeout’은 액티비티와 관계없이 엔트리가 삭제되어야 하는 시간을 의미한다. Controller는 configuration message를 통해서 switch에 엔트리가 삭제될 때 그 내용을 보고받는 지 여부를 지정할 수 있는데 Flow Expiration 메시지는 configuration message를 통해서 controller에 의해 명시적으로 활성화(enable)될 때에만 전송된다.

```
/* Flow expiration (datapath -> controller). */
struct ofp_flow_expired {
        struct ofp_header header;
        struct ofp_match match;

        uint16_t priority;
        uint8_t reason;
        uint8_t pad[1];

        uint32_t duration; /* Time flow was alive in seconds. */
        uint8_t pad2[4];
        uint64_t packet_count;
        uint64_t byte_count;
};
OFP_ASSERT(sizeof(struct ofp_flow_expired) == 72);

/* Why did this flow expire? */
enum ofp_flow_expired_reason {
        OFPER_IDLE_TIMEOUT,
        OFPER_HARD_TIMEOUT
};
```

[표 20] OFPT_FLOW_EXPIRED 내용

(3) 포트 상태 관련 메시지

▶ OFP_TYPE : OFPT_PORT_STATUS

▶ OF 스위치는 (물리적) 포트 설정 상태에 변화가 생겼을 때 Controller로 Port_status 메시지를 보낸다.

```
/* A physical port has changed in the datapath */
struct ofp_port_status {
        struct ofp_header header;
        uint8_t reason;
        uint8_t pad[7];
        struct ofp_phy_port_desc;
};
OFP_ASSERT(sizeof(struct ofp_port_status) == 64);

/* What changed about the physical port */
enum ofp_port_reason {
        OFPPR_ADD,
        OFPPR_DELETE,
        OFPPR_MODIFY
};
```

[표 21] OFPT_PORT_STATUS 내용

(4) 에러 관련 메시지

▶ OFP_TYPE : **OFPT_ERROR**

▶ OF 스위치는 Controller부터의 요청 메시지들에 대해 에러를 보고해야 할 경우에 OFPT_ERROR 메시지를 보낸다.

```
/* OFPT_ERROR : Error message (datapath -> controller). */
struct ofp_error_msg {
        struct ofp_header header;

        uint16_t type;
        uint16_t code;
        uint8_t data[0];            /* Variable-length data. Interpreted based on the type
and code */
};
OFP_ASSERT(sizeof(struct ofp_error_msg) == 12);

enum ofp_error_type {
        OFPET_HELLO_FAILED,
        OFPET_BAD_REQUEST,
        OFPET_BAD_ACTION,
        OFPET_FLOW_NOD_FAILED
};

/* ofp_error_msg 'code' values for OFPET_HELLO_FAILED. */
enum ofp_hello_failed_code {
        OFPHFC_INCOMPATIBLE
};

/* ofp_error_msg 'code' valued for OFPET_BAD_REQUEST. */
enum ofp_bad_request_code {
        OFPBRC_BAD_VERSION,
        OFPBRC_BAD_TYPE,
        OFPBRC_BAD_STAT,
        OFPBRC_BAD_VENDOR,
        OFPBRC_BAD_SUBTYPE
};
```

[표 22] OFPT_ERROR 내용

2.2.3 Symmetric

Symmetric message는 solicitation 없이 어떤 방향에서든 보낼 수 있다.

1) Hello(OFPT_HELLO) : Hello message는 switch와 controller간 connection startup시에 교환하는 메시지이다. (The OFPT_HELLO message has no body.)

2) Echo(OFPT_ECHO_REQUEST/OFPT_ECHO_REPLY) : Echo 메시지는 OF 스위치와

Controller간 지연을 체크한다거나 밴드위스를 측정하거나 상대 시스템의 생존 여부를 확인하기 위해 사용된다.

3. 메시지 교환

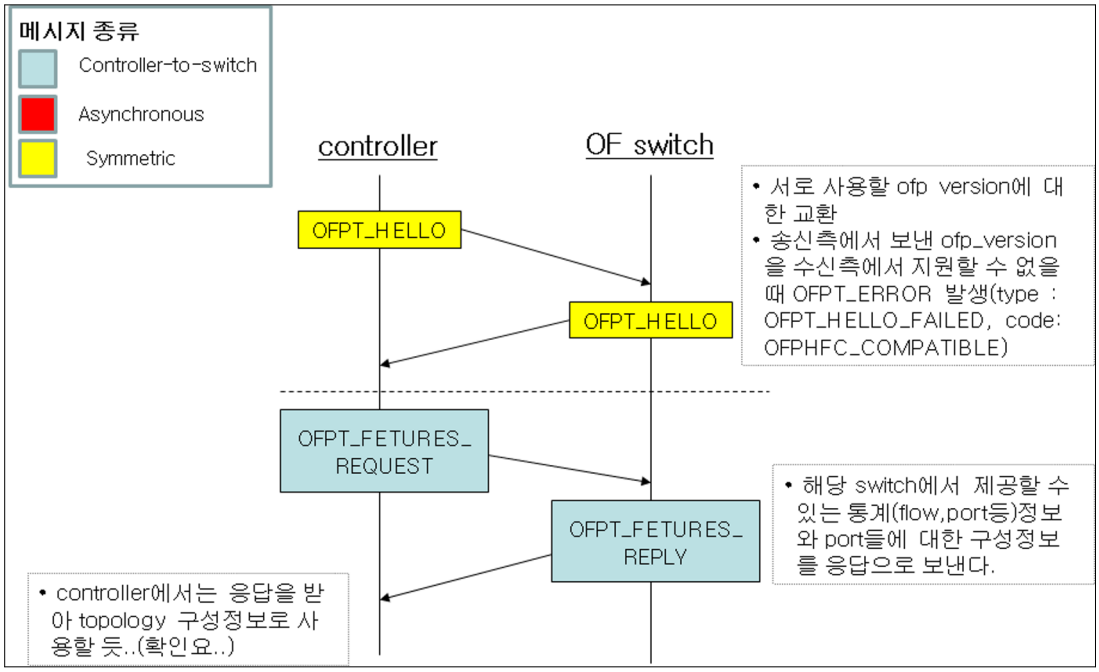
3.1 Connection Setup

OF 스위치에서는 다음의 명령을 통해 Controller와 OF 스위치간 secure channel을 형성한다.

secchan/secchan *nl:0* tcp:*10.0.0.2*:*2525*
'nl:0'은 해당 switch에 network interface들을 등록시켜놓은 network link ID를 의미하고 *'10.0.0.2'는 controller IP를 의미하고* *'2525'는 controller에 띄어놓은 nox_core 데몬의 포트번호를 의미한다.*

[표 23] Secure Channel 형성 명령

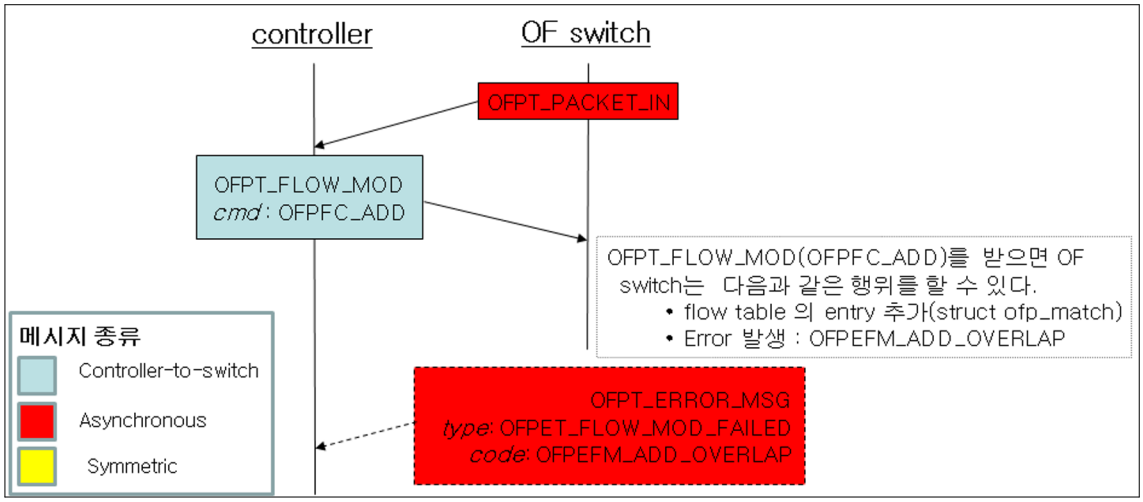
이 명령을 통해 switch와 controller간 다음과 같은 메시지 교환(OFP handshake)이 수행된다.



[그림 5] Controller와 OF스위치 간 Handshake 과정

3.2 Packet-In

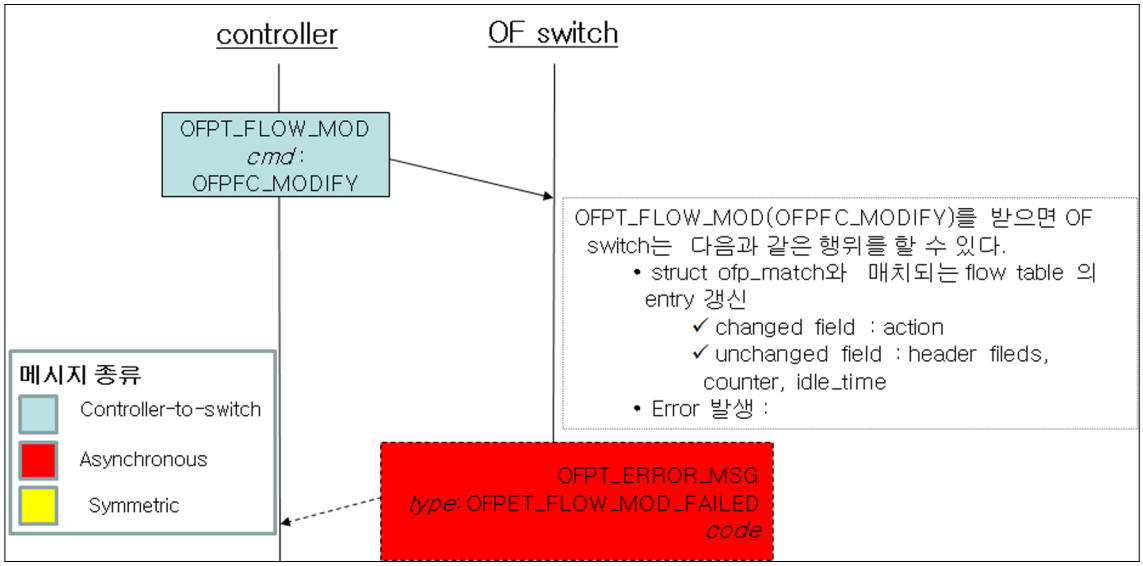
OF 스위치로 새로운 패킷이 유입되면 OF 스위치는 Controller로 OFPT_PACKET_IN 메시지를 보낸다. Controller는 OFPT_PACKET_IN 메시지를 분석하여 해당 플로우에 대한 플로우 액션 정의가 필요하다면 OFPT_FLOW_MOD(cmd : OFPFC_ADD)를 통해 각각의 OF 스위치들에게 새로운 플로우 엔트리를 추가시킨다.



[그림 6] Paket_In 에 대한 메시지 교환

3.3 Flow Entry Modification

OF 스위치에 이미 존재하는 플로우 엔트리를 변경해야 하는 경우, Controller는 OFPT_FLOW_MOD(cmd : OFPFC_MODIFY) 메시지를 통해 플로우 엔트리 변경을 요청한다. 이 때, OF 스위치는 OFPT_FLOW_MOD 메시지내에 존재하는 struct ofp_match를 flow table의 header fields와 매치하여 해당하는 entry에 대해서 “action field"만을 수정할 수 있다.

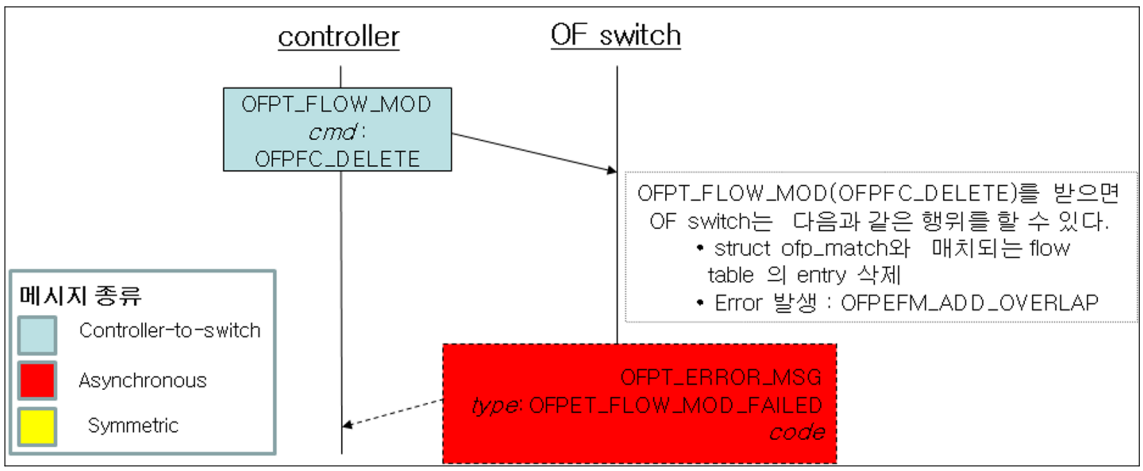


[그림 7] Flow Entry 수정에 대한 메시지 교환

3.4 Flow Entry Removal

OF 스위치에서 플로우 테이블에 존재하는 플로우 엔트리가 삭제되는 방법은 두 가지이다. 플로우 엔트리 생성시 지정된 ‘timeout(idle_time, hard_timeout)’에 의해 삭제되는 방법과 Controller에서 명시적으로 플로우 엔트리 삭제 명령을 주는 방법이다.

두 번째 방법으로 플로우 엔트리를 삭제해야 할 경우, Controller는 OFPT_FLOW_MOD(cmd : OFPFC_DELETE) 메시지를 통해 플로우 엔트리 삭제를 요청한다. 이때, OF 스위치는 OFPT_FLOW_MOD 메시지내 에 존재하는 struct ofp_match를 플로우 테이블의 헤더 필드들과 매치하여 해당하는 플로우 엔트리를 삭제한다.



[그림 8] Flow Entry 삭제에 대한 메시지 교환

3. 결론

최근 미래네트워크 또는 미래인터넷에 대한 요구사항들이 많이 나오고 있다. 미래인터넷의 요구사항은 확장성(Scalability/Ubiquity), 보안성/견고성(Security/Robustness), 이동성(Mobility), 이질성(Heterogeneity), 서비스품질(Quality of Service), 재설정/자동설정(Re-configurability), 상황인지(Context-awareness), 관리성(Manageability), 데이터 중심(Data-centric), 경제성(Economics) 등이 있다. 이러한 요구사항을 중심으로 해서 여러 연구그룹에서 다양한 형태의 미래 인터넷을 시도하고 있다.

국가연구망인 KREONET에서는 OpenFlow 네트워크 구축을 통해 연구자 기반의 동적네트워크 구성 환경을 미래연구망의 모습으로 갖추고자 한다. 이는 공급자 기반의 고정된 네트워크 환경을 KREONET을 이용하는 연구자가 필요로하는 형태의 네트워크를 구성하고, 트래픽을 전송할 수 있는 전체 환경을 의미한다.

본 문서에서는 이러한 OpenFlow 네트워크를 구동하게 하는 OpenFlow 프로토콜에 대하여 살펴봄으로써, OpenFlow를 이용한 네트워크 운영에 대하여 이해를 넓히고자 하였다. 모든 Controller와 OF스위치간의 통신은 Secure Channel을 통한 OpenFlow 프로토콜을 이용하여 이루어지므로 응용 특성에 맞는 Controller를 만들기 위해 OFP 메시지의 내용과 메시지 교환 방식의 습득은 중요하다. 앞으로 KREONET의 미래연구망으로서의 모습에 다양한 유연함을 가지고 있는 OpenFlow 네트워크가 일조할 수 있기를 기대한다.

참고문헌

[1] 이현용, 안서현, 김종원, "VINI(PlanetLab 가상화 확장) 이해 및 활용방안", Future Internet Winter Camp 2009

[2] 신명기, "미래인터넷 기술 및 표준화 동향", 전자통신동향분석, 제22권 제6호, 2009년 12월, pp.116~128

[3] 권순종, "미래인터넷을 위한 네트워크 가상화 기술", 정보과학회지 제26권 제10호, 2008년 10월, pp.76~81

[4] Brandon Heller, "OpenFlow switch Specification version 0.8.9", December 2008