sysmocom
systems for mobile communications GmbH

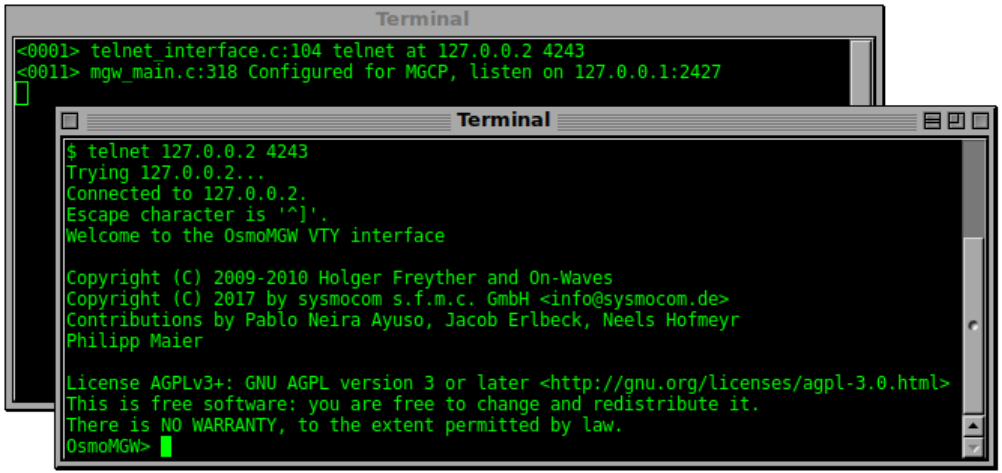
osm-mgw:
The new Osmocom Media Gateway

Philipp Maier <pmaier@sysmocom.de>

Agenda

- Introduction
- Practical example

osmo-mgw – an overview

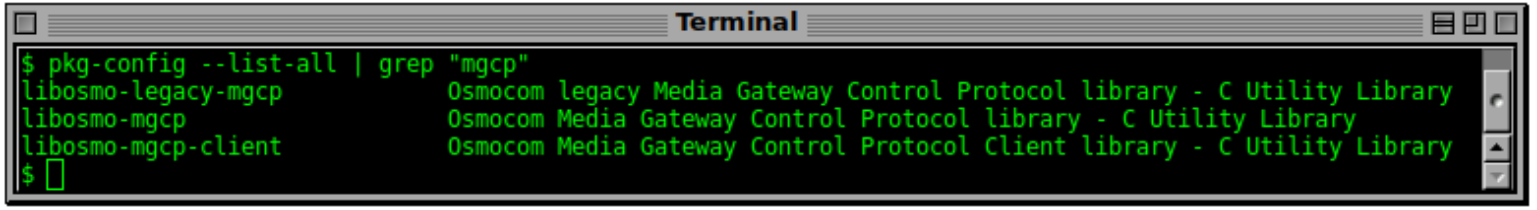


- Relatively new, around since 2017
- Predecessor: osmo-bsc_mgcp
- Assumptions and Hacks known from osmo-bsc_mgcp are removed.
  (e.g. BTS auto dectection)
- Temporarly removes OSMUX and built in transcoding

- New features:
  - Dual ported Endpoints (two connections per endpoint, no BTS/NET associations)
  - Wildcarded CRCX

mgcp_client – controls osmo-mgw

- Applications that use osmo-mgw (MGCP) do not have to implement the protocol stack themselves

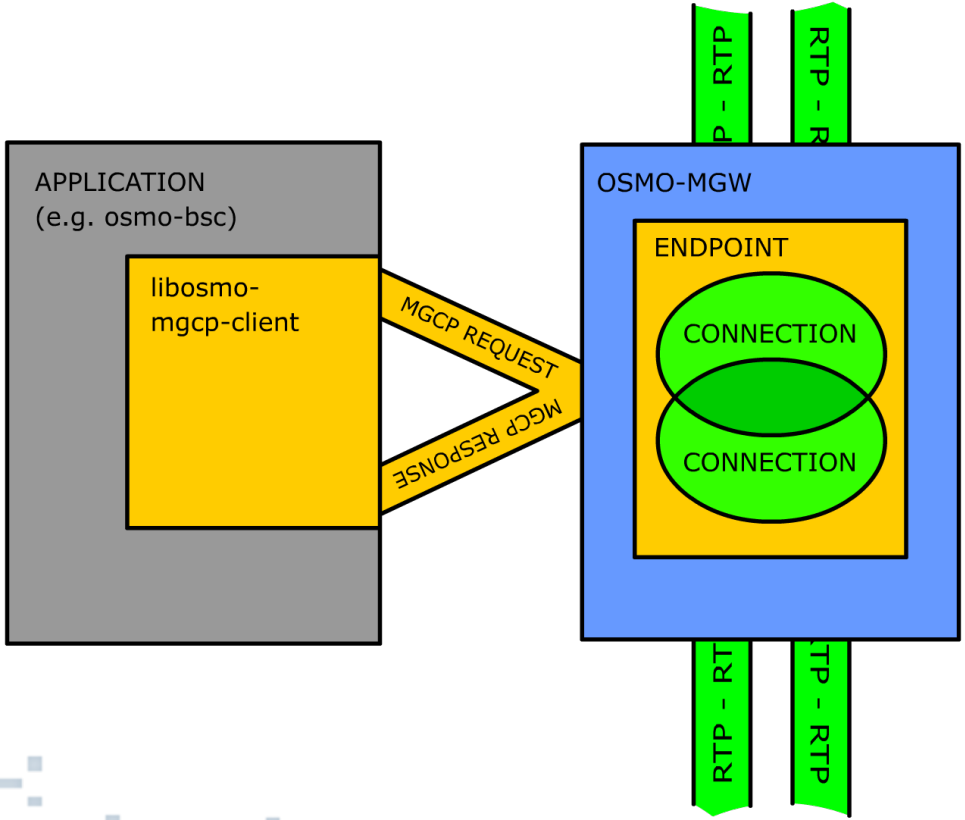


- Client library available, installed along with osmo-mgw as libosmo-mgcp-client
- MGCP-Client comes with its own VTY, crafting of own configuration interfaces is not needed

mgcp_client – API in two flavours

Low level API: mgcp_client (.h/.c)	High level API: mgcp_client_fsm (.h/.c)
· Handcrafted MGCP messages · Responses received via callback functions · Response handling up to the user · Message sequence implemented by user · No built-in timeout handling · A lot of implementation freedom	· MGCP messages are abstracted away · Implemented as child FSM (osmo-fsm), user must provide parent FSM · Abstract commands (create, modify, delete) · Less implementation freedom, but more comfort and safety · Uses mgcp_client internally

Endpoints and connections



- Application uses library provided by the osmo-mgw installtion
- Each RTP side gets its own connection on the same endpoint.
- Both connections are implicitly connected to each other

Agenda

- Introduction
- Practical example

MGCP transaction with mgcp_client

```
/* Send a simple CRCX message to the MGW */
void crcx(void)
{
        struct mgcp_msg mgcp_msg;
        struct msgb *msg;
        int rc;

        /* Generate MGCP message string */
        mgcp_msg = (struct mgcp_msg) {
                .verb = MGCP_VERB_CRCX,
                .presence = (MGCP_MSG_PRESENCE_ENDPOINT | MGCP_MSG_PRESENCE_CALL_ID | MGCP_MSG_PRESENCE_CONN_MODE),
                .call_id = 1234567,
                .conn_mode = MGCP_CONN_LOOPBACK};
        if (osmo_strlcpy
            (mgcp_msg.endpoint, "rtpbridge/*@mgw.osmocom.org", sizeof(mgcp_msg.endpoint)) >= MGCP_ENDPOINT_MAXLEN) {
                LOGP(DTEST, LOGL_DEBUG, "=== ENDPOINT NAME TOO LONG!\n");
                return;
        }

        msg = mgcp_msg_gen(mgcp, &mgcp_msg);
        LOGP(DTEST, LOGL_DEBUG, "=== GENERATED MGCP MESSAGE:\n%s\n", msg->data);

        /* Transmit MGCP message to MGW */
        rc = mgcp_client_tx(mgcp, msg, crcx_cb, NULL);
        if (rc < 0) {
                LOGP(DTEST, LOGL_DEBUG, "=== ERROR SENDING MGCP MSG!\n");
                return;
        }
}
```

MGCP transaction with mgcp_client

```
▼ Media Gateway Control Protocol
    CRCX (CreateConnection)
    Transaction ID: 1
    Endpoint: rtpbridge/*@mgw.osmocom.org
    Version: MGCP 1.0
    [The response to this request is in frame 2]
  ▼ Parameters
      CallId (C): 12d687 = 1234567
    ▼ LocalConnectionOptions (L): L: p:20, a:AMR, nt:IN
        Packetization period (p): 20
        Codecs (a): AMR
        Network Type (nt): IN
      ConnectionMode (M): loopback
```

MGCP transaction with mgcp_client

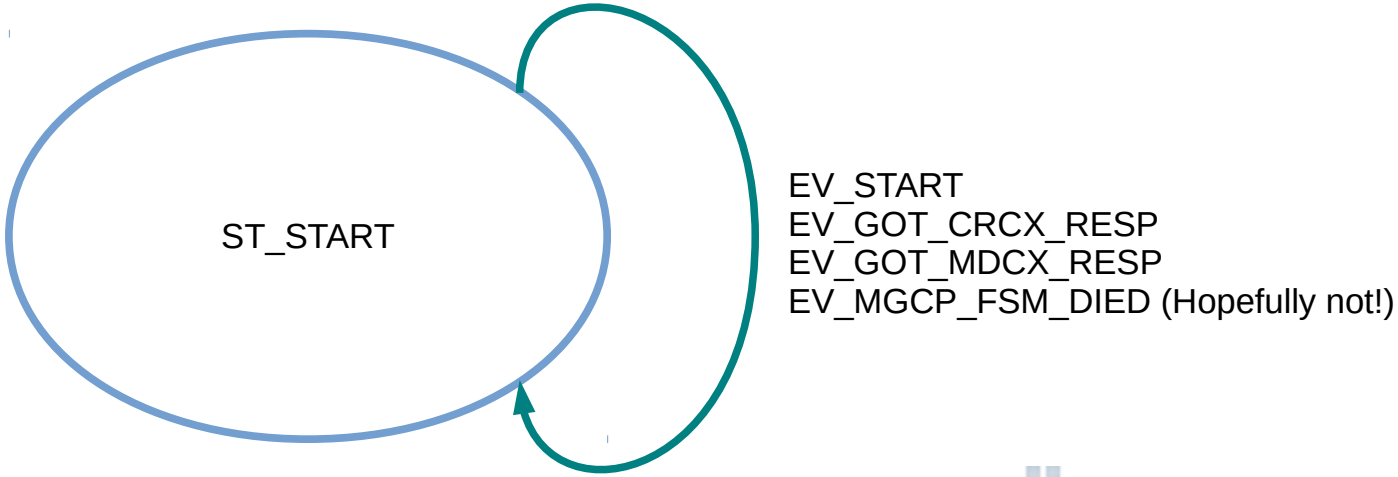
```
/* This callback is executed when the response from the MGW arrives */
static void crcx_cb(struct mgcp_response *r, void *priv)
{
        int rc;

        LOGP(DTEST, LOGL_DEBUG, "=== RECEIVED MGCP MESSAGE:\n%s\n", r->body);

        /* First we check if the MGW was ok with our request */
        if (r->head.response_code != 200) {
                LOGP(DTEST, LOGL_DEBUG,
                     "=== MGW DID NOT FULLFILL THE REQUEST!");
                return;
        }

        /* Parse response */
        rc = mgcp_response_parse_params(r);
        if (rc) {
                LOGP(DTEST, LOGL_DEBUG, "=== RESPONSE NOT PARSEABLE!\n");
                return;
        }

        LOGP(DTEST, LOGL_DEBUG, "==== PORT: %u\n", r->audio_port);
        LOGP(DTEST, LOGL_DEBUG, "==== ADDR: %s\n", r->audio_ip);
        LOGP(DTEST, LOGL_DEBUG, "==== EP: %s\n", r->head.endpoint);
        LOGP(DTEST, LOGL_DEBUG, "==== CONN_ID: %s\n", r->head.conn_id);
}
```

MGCP transaction with mgcp_client

```
▼ Media Gateway Control Protocol
    Response Code: The requested transaction was executed normally. (200)
    Transaction ID: 1
    Response String: OK
    [This is a response to a request in frame 1]
    [Time from request: 0.001045397 seconds]
  ▼ Parameters
      SpecificEndpointID (Z): rtpbridge/0@mgw.osmocom.org
      ConnectionIdentifier (I): 6C180A95E983C472231E008D7450F904
▼ Session Description Protocol
    Session Description Protocol Version (v): 0
  ▼ Owner/Creator, Session Id (o): - 6C180A95E983C472231E008D7450F904 23 IN IP4 127.0.0.1
      Owner Username: -
      Session ID: 6C180A95E983C472231E008D7450F904
      Session Version: 23
      Owner Network Type: IN
      Owner Address Type: IP4
      Owner Address: 127.0.0.1
    Session Name (s): -
  ▼ Connection Information (c): IN IP4 127.0.0.1
      Connection Network Type: IN
      Connection Address Type: IP4
      Connection Address: 127.0.0.1
  ▼ Time Description, active time (t): 0 0
      Session Start Time: 0
      Session Stop Time: 0
  ▼ Media Description, name and address (m): audio 4002 RTP/AVP 96
      Media Type: audio
      Media Port: 4002
      Media Protocol: RTP/AVP
      Media Format: DynamicRTP-Type-96
  ▼ Media Attribute (a): rtpmap:96 AMR
      Media Attribute Fieldname: rtpmap
      Media Format: 96
  ▼ Media Attribute (a): ptime:20
      Media Attribute Fieldname: ptime
      Media Attribute Value: 20
```

MGCP handling with mgcp_client_fsm

- A simple FSM serves as test body, in a real world application, this FSM would handle the surrounding logic (e.g. the handling of a subscriber status)



- See osmo-bsc.git/src/libbsc/bsc_subscr_conn_fsm.c for a real world example

MGCP handling with mgcp_client_fsm

- Similar to mgcp_clint we begin with setting up the connection details
- mgcp_conn_create() is called, connection on the mgw is created.
- EV_GOT_CRCX_RESP is sent to us when done.
- EV_MGCP_FSM_DIED is sent to us on error.

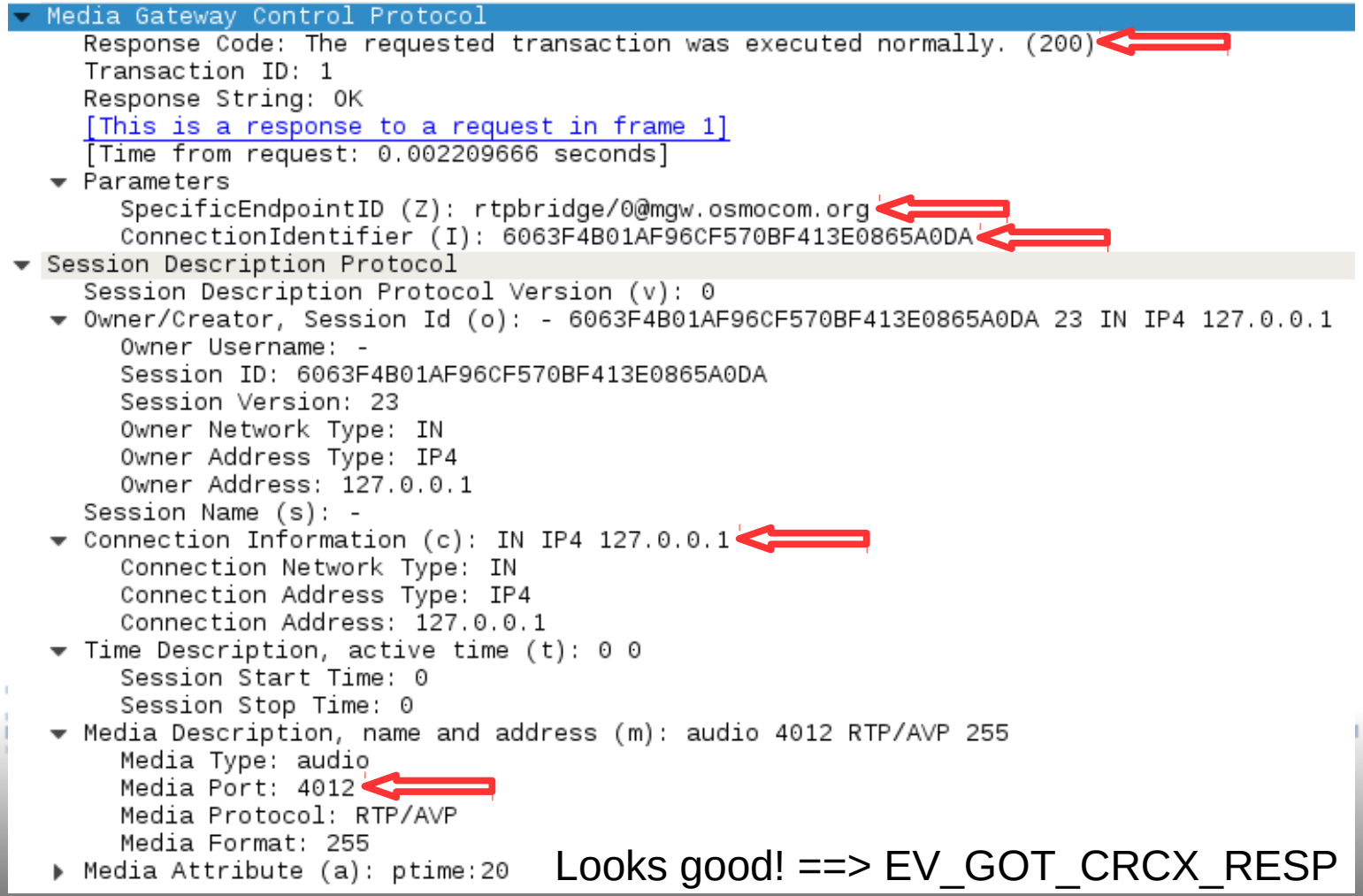
```
struct mgcp_conn_peer {
        /*!< RTP connection IP-Address (optional, st
        char addr[INET_ADDRSTRLEN];

        /*!< RTP connection IP-Port (optional)  */
        uint16_t port;

        /*!< RTP endpoint */
        char endpoint[MGCP_ENDPOINT_MAXLEN];

        /*!< CALL ID (unique per connection) */
        unsigned int call_id;
};
```

```
case EV_START:
        LOGPFSML(fi, LOGL_ERROR, "==== EV_START ====\n");

        memset(&conn_info, 0, sizeof(conn_info));
        osmo_strlcpy(conn_info.endpoint, "rtpbridge/*@mgw.osmocom.org", sizeof(conn_info.endpoint));
        conn_info.call_id = 1234;
        osmo_strlcpy(conn_info.addr, "127.0.0.1", sizeof(conn_info.addr));
        conn_info.port = 4711;

        mgcp_fsm_inst = mgcp_conn_create(mgcp, fi, EV_MGCP_FSM_DIED, EV_GOT_CRCX_RESP, &conn_info);
        break;
```

MGCP handling with mgcp_client_fsm

```
▼ Media Gateway Control Protocol
    CRCX (CreateConnection)
    Transaction ID: 1
    Endpoint: rtpbridge/*@mgw.osmocom.org
    Version: MGCP 1.0
    [The response to this request is in frame 2]
  ▼ Parameters
      CallId (C): 4d2
    ▼ LocalConnectionOptions (L): L: p:20, a:AMR, nt:IN
        Packetization period (p): 20
        Codecs (a): AMR
        Network Type (nt): IN
      ConnectionMode (M): sendrecv
▼ Session Description Protocol
    Session Description Protocol Version (v): 0
  ▼ Owner/Creator, Session Id (o): - 4d2 23 IN IP4 127.0.0.1
      Owner Username: -
      Session ID: 4d2
      Session Version: 23
      Owner Network Type: IN
      Owner Address Type: IP4
      Owner Address: 127.0.0.1
    Session Name (s): -
  ▼ Connection Information (c): IN IP4 127.0.0.1
      Connection Network Type: IN
      Connection Address Type: IP4
      Connection Address: 127.0.0.1
  ▼ Time Description, active time (t): 0 0
      Session Start Time: 0
      Session Stop Time: 0
  ▼ Media Description, name and address (m): audio 4711 RTP/AVP 255
      Media Type: audio
      Media Port: 4711
      Media Protocol: RTP/AVP
      Media Format: 255
```

MGCP handling with mgcp_client_fsm



Looks good! ==> EV_GOT_CRCX_RESP

MGCP handling with mgcp_client_fsm

- We can now pick up and process the connection info the MGW has passed to us.
- Modification only requires to fill up conn_info again and a call to mgcp_conn_modify()

```
case EV_GOT_CRCX_RESP:
        LOGPFSML(fi, LOGL_ERROR, "==== EV_GOT_CRCX_RESP ====\n");

        conn_info_remote = data;
        LOGPFSML(fi, LOGL_ERROR, "==== PORT: %u\n", conn_info_remote->port);
        LOGPFSML(fi, LOGL_ERROR, "==== ADDR: %s\n", conn_info_remote->addr);
        LOGPFSML(fi, LOGL_ERROR, "==== EP: %s\n", conn_info_remote->endpoint);
        LOGPFSML(fi, LOGL_ERROR, "==== CALLID: %u\n", conn_info_remote->call_id);

        memset(&conn_info, 0, sizeof(conn_info));
        osmo_strlcpy(conn_info.endpoint, conn_info_remote->endpoint, sizeof(conn_info.endpoint));
        conn_info.call_id = 1234;
        osmo_strlcpy(conn_info.addr, "127.0.0.2", sizeof(conn_info.addr));
        conn_info.port = 4712;

        rc = mgcp_conn_modify(mgcp_fsm_inst, EV_GOT_MDCX_RESP, &conn_info);
        if (rc != 0)
                LOGPFSML(fi, LOGL_ERROR, "==== CAN NOT MODIFY!\n");
        break;
```

MGCP handling with mgcp_client_fsm

- We can now again pick up and process the connection info the MGW has passed to us.
- Removing the connection requires almost no effort, a call to mgcp_conn_delete() is enough.

```
case EV_GOT_MDCX_RESP:
        LOGPFSML(fi, LOGL_ERROR, "==== EV_GOT_MDCX_RESP ====\n");
        conn_info_remote = data;
        LOGPFSML(fi, LOGL_ERROR, "==== PORT: %u\n", conn_info_remote->port);
        LOGPFSML(fi, LOGL_ERROR, "==== ADDR: %s\n", conn_info_remote->addr);
        LOGPFSML(fi, LOGL_ERROR, "==== EP: %s\n", conn_info_remote->endpoint);
        LOGPFSML(fi, LOGL_ERROR, "==== CALLID: %u\n", conn_info_remote->call_id);

        mgcp_conn_delete(mgcp_fsm_inst);
        osmo_fsm_inst_term(fi, OSMO_FSM_TERM_REGULAR, NULL);
        return;
```

osm-mgw:
The new Osmocom Media Gateway

Philipp Maier <pmaier@sysmocom.de>