GAPK (GSM Audio Packet Knife) status

Vadim Yanitskiy, OsmoDevCon 2018, 2018-04-22, Berlin

Introduction

What is GAPK?

- GSM Audio Pocket Knife
- Initially written by Sylvain Munaut (@tnt)
- Audio format encoding / decoding (e.g. *.amr, *.gsm)
- GSM audio codecs (e.g. FR, EFR, AMR)
- ALSA capture / playback support
- Basic RTP support

How can one use it?

- Transcoding from one audio format to another
- Transcoding from one audio codec to another
- Real-time audio encoding / decoding
- Real-time decoding of RTP-streams

Introduction

The previous state of project:

- Just a simple command line tool
- Weak testing coverage
- No shared libraries, single binary :/

```
gapk -i IN_FILE -f IN_FORMAT -g OUT_FORMAT -o OUT_FILE
```

What if we turn it into a shared library?

- Audio coding support for OsmocomBB/mobile
- GR-GSM blocks for format / audio transcoding
- OsmoMGW audio transcoding
- Other projects?

Introduction

What was done?

- Introduced a shared `libosmogapk` library
- Install GAPK headers to `${includedir}/osmocom/gapk/`
- Add an `osmo_gapk` prefix to the exposed symbols
- Add a pkg-config manifest for `libosmogapk`
- Add the symbol export map for `libosmogapk`
- Use talloc for memory management
- Better testing coverage

What was changed?

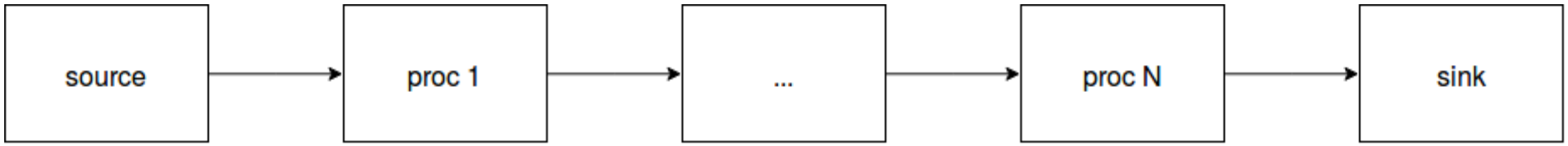
- The `gapk` binary was renamed to `osmo-gapk`
  - and linked against `libosmogapk`
- Use Osmocom logging framework instead of `printf()`

Internal architecture

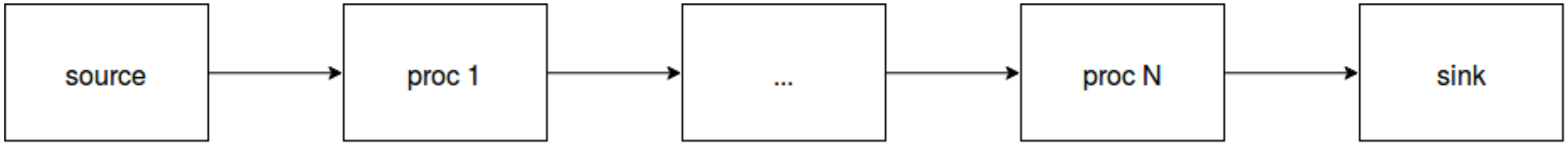
Some basics:

- Block-based architecture (like in GNU Radio):
  - linear processing chains (queues),
  - source, sink and processing blocks,
  - no scheduler, just iterate over all blocks in chain.

Processing chain X:



Processing chain Y:



Internal architecture

A block structure:

```
struct osmo_gapk_pq_item {
  /* I/O data length */
  unsigned int len_in;
  unsigned int len_out;

  /* Internal state */
  void *state;
  /* Output buffer */
  uint8_t *buf;

  /* Processing / exit call-backs */
  int (*proc)(...); // Like work() in GNU Radio
  int (*exit)(...); // Destructor

  /* Meta info ... */
}
```

Internal architecture

A source block example:

```
/* Block definition */
struct osmo_gapk_pq_item *item;

item = osmo_gapk_pq_add_item(pq);
if (!item)
  return -ENOMEM;

item->type     = OSMO_GAPK_ITEM_TYPE_SOURCE;
item->cat_name = OSMO_GAPK_CAT_NAME_SOURCE;
item->sub_name = "file";

item->len_in  = 0; /* Because it's a source */
item->len_out = DATA_LEN;
item->proc    = pq_cb_file_input;
item->exit    = pq_cb_file_exit;
```

Internal architecture

A source block example:

```
/* Proc function definition */
static int
pq_cb_file_input(void *_state, uint8_t *out,
  const uint8_t *in, unsigned int in_len)
{
  struct pq_state_file *state = _state;
  int rv;
  rv = fread(out, state->blk_len, 1, state->fh);
  if (rv <= 0)
    return -1;
  return rv * state->blk_len;
}
```

Internal architecture

API/header files

- Processing queue and block definition:
  - `osmocom/gapk/procqueue.h`
- Formats:
  - `osmocom/gapk/formats.h`
- Formats:
  - `osmocom/gapk/codecs.h`

API/symbol names

- All are starting from `osmo_gapk_` prefix

EOF

Question processing queue: