]pexip[

dcSCTP in GStreamer

GStreamer Conference 2025

Tulio Beloqui <tulio@pexip.com>

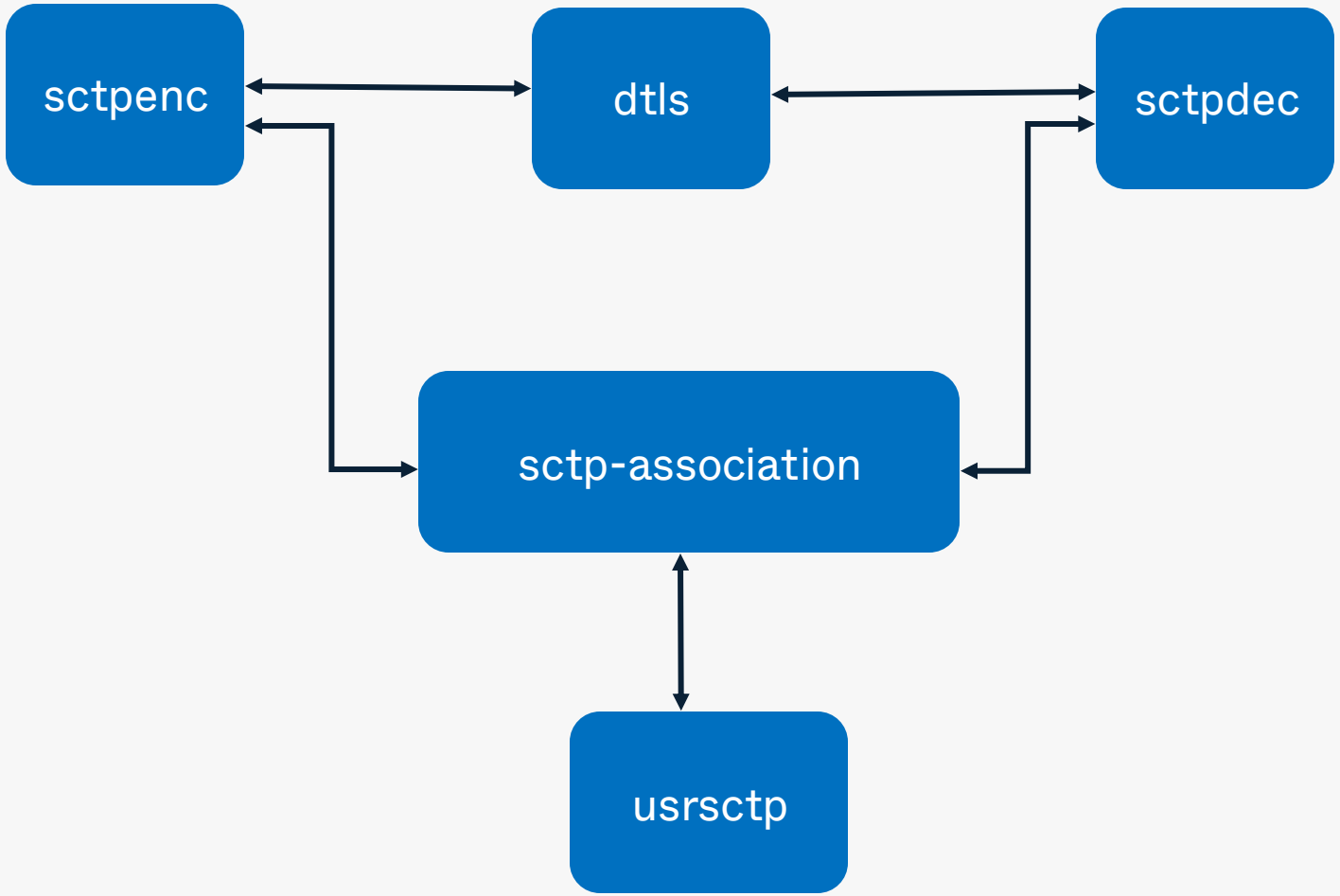
SCTP – the "p" stands for PAIN!

- We have been doing interop with webrtc for a long time and since then, we have been dealing with data-channels.
- We also have interop with the google meeting service and they use data-channels as well... so we have used the SCTP stack in gstreamer for all of these.
- **usrsctp** maintained by Michael Tüxen (https://github.com/sctplab/usrsctp)
- We have also been struggling with the technical debt the usrsctp project:
  - use-after-free bugs
  - memory leaks
  - race conditions
  - if-defs jungle
- Not to shame usrsctp! as it has been a great stack to begin with, but it doesn't have the traction we need.
- It has done it well so far, but we need to move forward, to a more modern stack.

dcSCTP to the rescue

- So we started looking at what SCTP implementations were available and open-source... and the webrtc's intiative came up!
- Written in C++.
- Licensed under Apache-2.0.
- It is very **well written**, is up-to-date and **very well** TESTED.
- Lives in https://chromium.googlesource.com/external/webrtc/+/master/net/dcsctp
- Big thanks to the dcSCTP owner Victor Boivie <boivie@webrtc.org> and the webrtc team involved!
- We looked into the public API and it looked pretty easy to use...so we gave it a chance!

Current plugin structure



Let's start porting!

- We started porting the library into GStreamer...

- The goal was simple: remove usrsctp, put in place the new library and rework the sctp-association to use it.

- The library lives in the webrtc git repo, so we have to do some scripting to get only the bits we needed from the code.
- Cloned the repo, copied some files over.. pulled in some of its dependencies.
- Wrote a meson project to start building it...

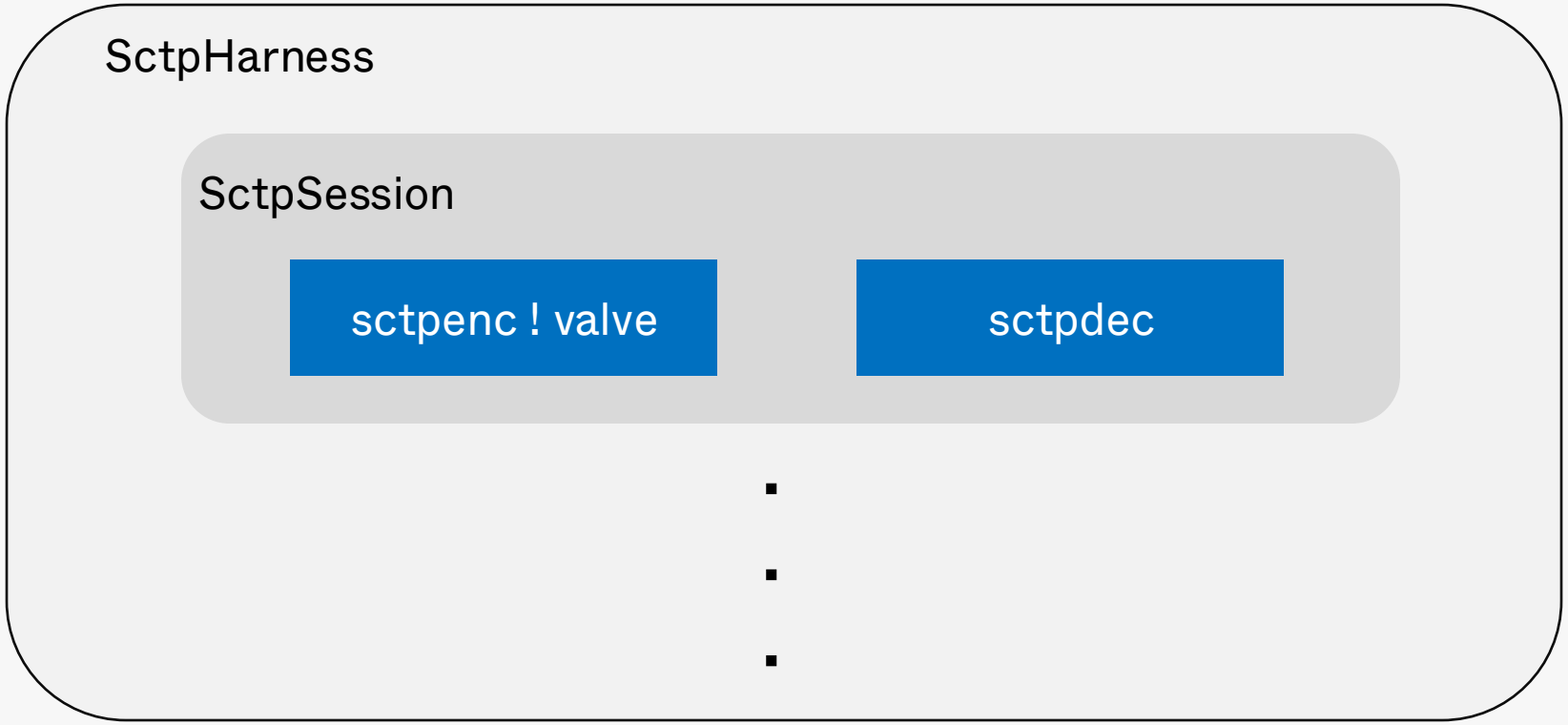
- We wrote a thin layer on top of a sctp-socket that the library comes with, which felt the natural place to make the C++ <-> C interop, and we replaced the usrsctp socket with this new one.

- We introduced a sctp-association-factory responsible for handling the lifetime of the sctp-associations, as they could be shared between the encoder and decoders.
- The factory also contains its own **GMainLoop** to handle the lifetime of the objects, sctp timers, so all SCTP sockets operations are single-threaded.

- **Everything fall into place! and when we ran our test suite... all test passed!**

SCTP-harness! (1 of 3)

A harness for writing classic abe to bob style tests!



SCTP-harness! (2 of 3)

```
guint sctp_harness_session_new (SctpHarness * h, guint id,
    gboolean aggressive_heartbeat);

GstHarness * sctp_harness_create_send_stream (SctpHarness * h,
    guint session_id, guint stream_id);

GstHarness * sctp_harness_wait_for_stream_created (SctpHarness * h,
    guint session_id, guint stream_id);
```

```c
GST_START_TEST (sctp_stream_create_and_destroy)
{
  const guint a = 1, b = 2;
  const guint sctp_ppid = 51;
  SctpHarness *h = sctp_harness_new (FALSE);
  GstHarness *a_send, *b_recv;

  sctp_harness_session_new (h, a, FALSE);
  sctp_harness_session_new (h, b, FALSE);

  sctp_harness_connect_sessions (h, a, b);

  fail_unless (sctp_harness_wait_for_association_established (h, a));
  fail_unless (sctp_harness_wait_for_association_established (h, b));

  /* Create send stream */
  a_send = sctp_harness_create_send_stream (h, a, a);
  gst_harness_set_src_caps_str (a_send, "application/x-data");

  /* Create and push buffer */
  guint8 *data = ...
  GstBuffer *buf = ...
  gst_sctp_buffer_add_send_meta (buf, sctp_ppid, TRUE, 0, 0);
  gst_harness_push (a_send, buf);

  /* Expect recv stream to be created */
  b_recv = sctp_harness_wait_for_stream_created (h, b, a);
  fail_unless (b_recv != NULL);

  /* Retrieve buffer from recv stream */
  GstBuffer *recvbuf = gst_harness_pull (b_recv);

  /* Ensure it looks sane */
  fail_unless_equals_int ((gint) gst_buffer_get_size (recvbuf), sizeof (dat
  fail_unless_equals_int ((gint) gst_buffer_memcmp (recvbuf, 0, data,
      sizeof (data)), 0);
  GstSctpReceiveMeta *recv_meta = gst_sctp_buffer_get_receive_meta (recvbuf
  fail_unless_equals_int (recv_meta->ppid, sctp_ppid);

  gst_buffer_unref (recvbuf);

  /* Destroy the send stream */
  sctp_harness_destroy_send_stream (h, a, a);
  /* Expect the corresponding recv stream to be destroyed */
  fail_unless (sctp_harness_wait_for_stream_destroyed (h, b, a));

  sctp_harness_teardown (h);
}
```

```c
GST_START_TEST (sctp_disconnect_unclean_stream_sock)
{
  const guint a = 1, b = 2;
  SctpHarness *h = sctp_harness_new (TRUE);

  sctp_harness_session_new (h, a, TRUE);
  sctp_harness_session_new (h, b, FALSE);

  sctp_harness_connect_sessions (h, a, b);

  fail_unless (sctp_harness_wait_for_association_established (h, a));
  fail_unless (sctp_harness_wait_for_association_established (h, b));

  sctp_harness_break_network (h, a);

  fail_unless (sctp_harness_wait_for_association_disestablished (h, a))

  sctp_harness_unbreak_network (h, a);
  sctp_harness_reconnect_session (h, a);

  fail_unless (sctp_harness_wait_for_association_established (h, a));
  fail_unless (sctp_harness_wait_for_association_restarted (h, b));

  sctp_harness_teardown (h);
}

GST_END_TEST;
```

Final conclusions

- Even if it sounds scary, sometimes is okay to replace building blocks
- dcSCTP proved to be a great upgrade for us
  - We have not gotten a **single crash report** since we upgraded to dcSCTP (jinx)
  - Performance was also impacted massively, our test suite runtime cut in half!
  - The "dc" might stand for does-not-crash!
- Thanks to the dcSCTP owner Victor Boivie <boivie@webrtc.org> andthe webrtc team involved!

What's next?

- dcSCTP rust successor: https://github.com/webrtc/dcsctp
- We will upstream this work during the hackfest ! all of it!
- Help needed testing it within webrtcbin, any takers?!

Thanks!

- Contact
  - tulio@pexip.com
  - tulio @ GStreamer Community in Matrix

Questions?