ALSA and DAHDI Audio Architecture Analysis

ALSA Audio Driver Logic

The ALSA framework controls the timing of audio data exchange using two core mechanisms: `pointer()` and `snd_pcm_period_elapsed()`.

Mono channel (1 channel), 16-bit samples (S16_LE)
✅ 1 frame = 2 bytes

Step 1: Period Elapsed Notification
The driver periodically calls `snd_pcm_period_elapsed()`. This can be triggered by a timer or hardware interrupt. This tells ALSA that a period (e.g., 20ms of audio) has been processed and requests ALSA to continue.

Step 2: ALSA Calls pointer()
In response, ALSA invokes `substream->ops->pointer(substream)`, which returns the current frame position (i.e., where the DMA has reached). This is not a period count, but rather a pointer in the ring buffer (unit: frame).

```
Example: pos = (delta_ms * runtime->rate) / 1000
```

Step 3: Determine if a new period has elapsed
ALSA compares the current and previous pointer values:

```
(pos - last_pos + buffer_size) % buffer_size >= period_size
```

If the condition is true, a new period has elapsed.

✅ ALSA then calls `substream->ops->copy_user()`
This pulls audio data from userspace and invokes:
play

```
copy_from_user(kbuf, user_buf, bytes);
write_pcm_20ms(kbuf, bytes);
```

Record

```
read_pcm_20ms(kbuf, bytes);
copy_to_user(user_buf, kbuf, bytes);
```

ALSA and Hardware Timing and Buffering

In a typical system, timing mechanisms differ across ALSA, hardware, and userspace applications:

Module	Timing Source
ALSA kernel layer	`snd_pcm_timer_function()`, jiffies, hrtimer
Hardware driver	DMA + Interrupt + Hardware Clock
Userspace program	Relies on `poll()` or `select()` (no timer)

Typical use of ring buffers:

Name	Writer	Reader
RX ringbuf	Hardware	ALSA
TX ringbuf	ALSA	Hardware

Concepts:
Overrun: Writing too fast, overwriting unread data.
Underrun: Reading too fast, no data available.

RX Ring Buffer:
Overrun → Data loss, frame drop
Underrun → Short reads, I/O errors

TX Ring Buffer:
Underrun → Clicks, hardware faults
Overrun → Frame drop (acceptable in moderation)

⚠ Important Notes:
1. On `SNDRV_PCM_TRIGGER_START/STOP`, ring buffers and pointer state must be reset.
2. For echo cancellation (EC/AEC), ring buffer length must be < EC window:
EC window: 128ms → buffer ≤ 2048B (2KB)
AEC window: 200–1000ms → buffer ≤ 4096B (4KB)

DAHDI Audio Driver Logic

DAHDI architecture is similar to ALSA in terms of driver design but uses different APIs.

ALSA exchange interface:
snd_pcm_copy_user(substream, channel, pos, user_buf, count);

DAHDI exchange interface:

dahdi_transmit(&d_span); → pushes writechunk
bcm_pull_dahdi_chunk() → writechunk → TX ringbuf
bcm_push_dahdi_chunk() → RX ringbuf → readchunk
dahdi_receive(&d_span); → pull readchunk to DAHDI core

RX direction (capture):
Hardware → RX ringbuf → bcm_push_dahdi_chunk() → readchunk → DAHDI

TX direction (playback):
DAHDI → writechunk → bcm_pull_dahdi_chunk() → TX ringbuf → Hardware

Comparison between ALSA and DAHDI:

Aspect	ALSA	DAHDI
User interface	snd_pcm_copy_user()	dahdi_transmit()/receive()
Ring buffer	RX/TX ringbuf	Same
Data flow	user buf ↔ ringbuf ↔ hardware	writechunk/readchunk ↔ ringbuf ↔ hardware

Note: Both use ringbufs for hardware-driver interaction. Only the interface layer differs.