What are you listening to now?

Implementing "Now Playing" feature in modern XMPP

Özcan Oğuz
ozcan@oyd.org.tr

1st February 2026, Brussels
FOSDEM 2026

It's the year 2005...

MSN Messenger was the king at the time. Some friends were using official client, some Pidgin, some Adium...

Below your name, everyone can see:

> *♪ Iron Maiden - Wasted Years*

It's awesome! You can brag about your *music taste* :)

The Golden Era of "Now Playing"

- **MSN Messenger** (via WMP)
- **Pidgin** (with plugins)
- **aMSN**
- **Adium**
- ...

All had some form of music status sharing

Then it disappeared

- MSN Messenger and similar apps are gone
- Mobile messaging took over
- Centralised platforms never brought it back
- Rising streaming services did not implement such feature
- Within time, we forgot it existed

I use XMPP+OMEMO **daily and regularly**.

A few months ago, with a friend of mine, we figured out that we were listening to the same song.

It was a song from an old Turkish symphonic metal band **Almôra**.

It was the time when I remembered the old times.

And there comes the question:

"Can I bring this "Now Playing" feature to my daily XMPP client?"

I started to think about building such protocol...

But wait...

The XMPP specification already exists!

XEP-0118: User Tune

Published: 2004, Last updated: 2008

The journey

1. **Discovery**: Finding XEP-0118
2. **Research**: Understanding PEP and MPRIS
3. **Implementation**: Writing the code for Dino
4. **Contribution**: Open a PR and publish it

Phase 1: Discovery

The protocol already exists!

XEP-0118: User Tune

Published: March 2004

Last Updated: 2008

Status: Draft Standard

22 years old but *barely* implemented.

What XEP-0118 defines

A simple way to tell your contacts:

- **What** you're listening to (artist, track name)
- **From where** (album)
- **How long** the track is
- **User rating** of the track, stars 1-10
- **And much more**: a link to it, extra details like genre, composer, performer etc.

Built on a widely used standard

Personal Eventing Protocol (XEP-0163)

XMPP clients already use this XEP for status, mood etc.

The protocol part seems OK

The XMPP spec was straightforward, seems easy to implement and the base protocol is already widely known/ used.

But implementation is an issue

I am always using XMPP with **OMEMO**, and personally my desktop client is **Dino**, while my mobile client is **Conversations**.

Surprisingly, *neither* of them had support on User Tune; the supported clients usually do not have good OMEMO implementation...

...so I decided to implement it in modern clients. But here comes the question (again)...

Phase 2: Research

How do I know what is playing?

The challenge

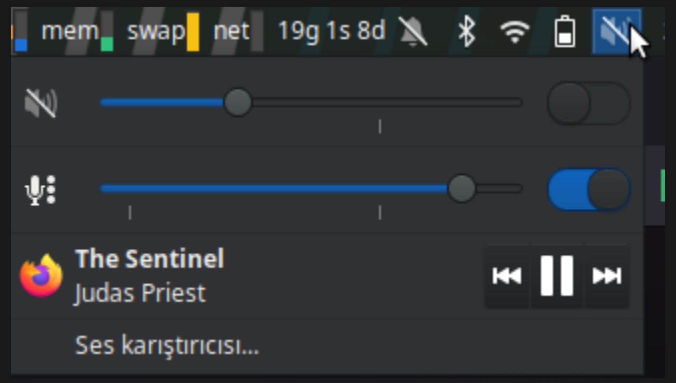
XMPP clients doesn't play music.

Lollypop does. VLC does. Firefox does.

How do I ask them what's playing?

Then I realised...

When I play a song regardless of the app I used, its structured details and controls appear in the system tray's PulseAudio menu.



So, there must be a system wide API!

After a little research

Platform	API
GNU/BSD	**MPRIS (DBus)**
Android	**MediaSessionAPI**
Windows	SystemMediaTransportControls
Mac OS	(private API)

GNU/Linux and Android are my main targets, for Dino I will use DBus.

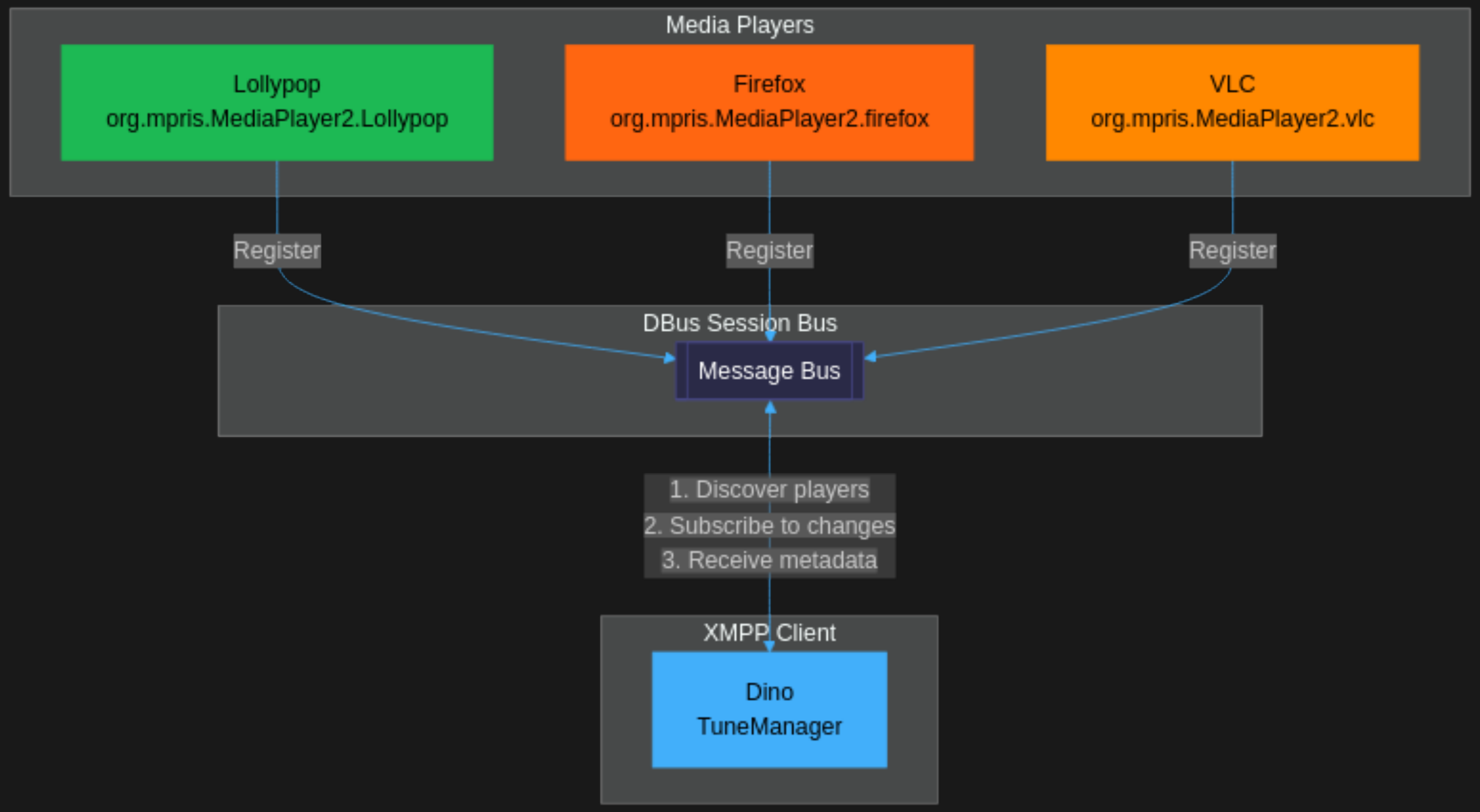
MPRIS

Media Player Remote Interfacing Specification

Almost every media player speaks it:

- VLC
- Lollypop
- Firefox
- MPV
- DRM bloatware (Spotify etc.)
- ...

D-Bus: The Message Bus



Which data I get from MPRIS?

- **PlaybackStatus**: playing, paused, stopped
- **Metadata**: artist, title, album, length, URL
- **Signals**: real time change notifications!

So, I do not need anything else for XEP-0118!

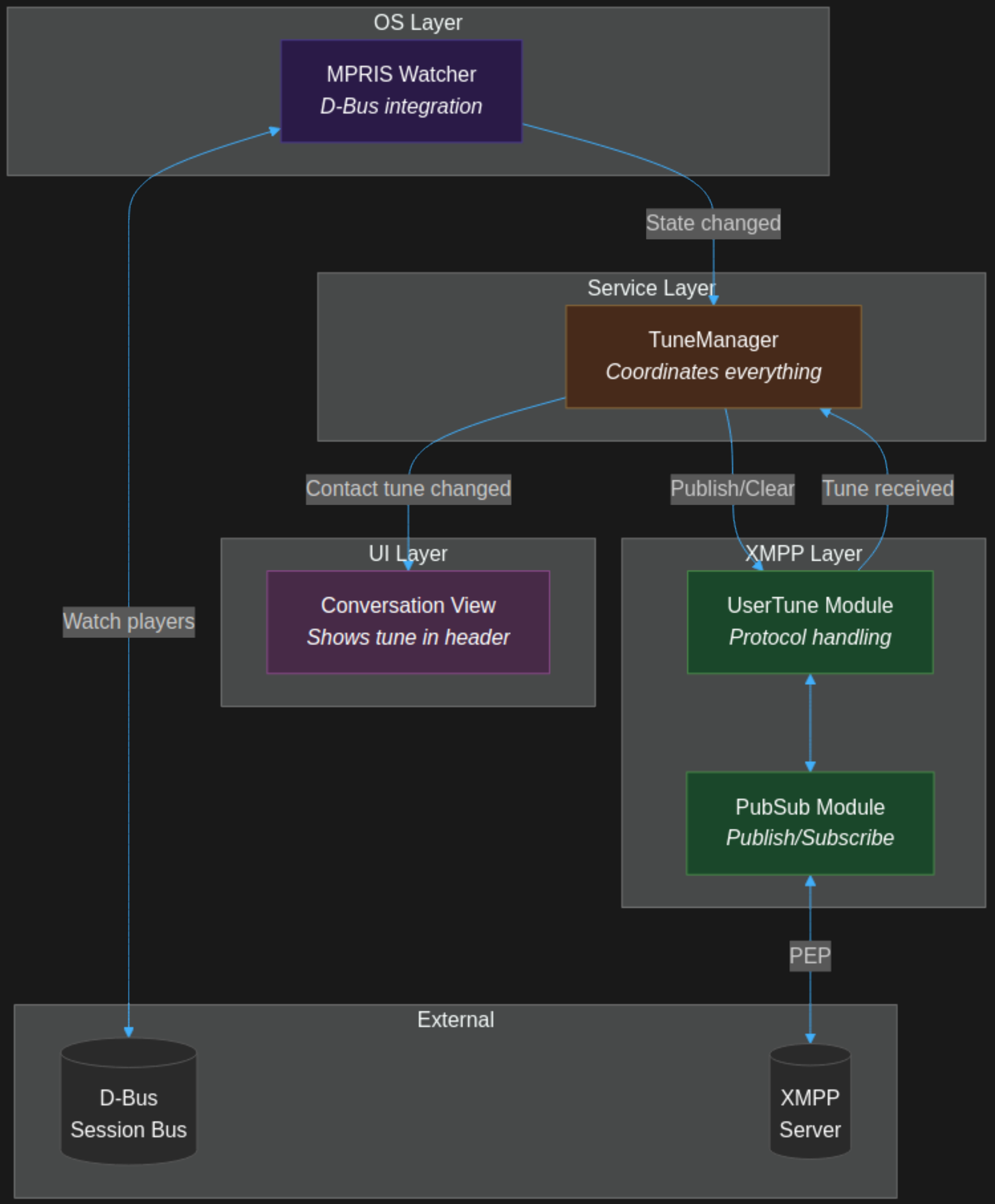
- MPRIS tells me what's playing
- D-Bus delivers the notifications
- The mapping to XEP-0118 is direct

Time to write code.

Phase 3: Implementation

Connecting everything together

The architecture



Component 1: XMPP Module

```
// xmpp-vala/src/module/xep/0118_user_tune.vala

public class Tune : Object {
    public string? artist { get; set; }
    public string? title { get; set; }
    public string? source { get; set; }  // album
    public int length { get; set; default = -1; }
    public string? track { get; set; }
    public string? uri { get; set; }
}
```

And create the XML stanza.

Publishing via PEP

```
public async bool publish_tune(XmppStream stream, Tune? tune) {
    StanzaNode tune_node = build_tune_node(tune);

    Pubsub.PublishOptions options = new Pubsub.PublishOptions()
        .set_persist_items(true)
        .set_max_items("1")
        .set_send_last_published_item("on_sub_and_presence")
        .set_access_model(Pubsub.ACCESS_MODEL_PRESENCE);

    return yield stream.get_module(Pubsub.Module.IDENTITY)
        .publish(stream, null, NS_URI, "current", tune_node, options);
}
```

Receiving Notifications

```
public class Module : XmppStreamModule {
    public signal void tune_received(XmppStream stream,
                                     Jid jid, Tune? tune);

    public override void attach(XmppStream stream) {
        stream.get_module(Pubsub.Module.IDENTITY)
            .add_filtered_notification(
                stream, NS_URI,
                on_pubsub_item,
                on_pubsub_retract,
                on_pubsub_delete
            );
    }
}
```

Component 2: D-Bus Interfaces

```
// libdino/src/dbus/mpris.vala

[DBus (name = "org.freedesktop.DBus")]
public interface FreedesktopDBus : Object {
    public abstract string[] list_names() throws Error;
    public signal void name_owner_changed(string name,
                                          string old_owner,
                                          string new_owner);
}

[DBus (name = "org.mpris.MediaPlayer2.Player")]
public interface MprisPlayer : Object {
    public abstract string playback_status { owned get; }
    public abstract HashTable<string, Variant> metadata { owned get; }
}
```

Component 3: TuneManager

```
public class TuneManager : StreamInteractionModule, Object {
    private const int DEBOUNCE_MS = 1000;

    private HashMap<string, MprisPlayerWatcher> player_watchers;
    private Tune? last_published_tune = null;
    private uint debounce_source = 0;

    // Signals
    public signal void contact_tune_changed(Account account,
                                            Jid jid, Tune? tune);
}
```

The tricky parts

1. **Multiple players**: What if VLC ****and**** Firefox are open?
2. **Rapid changes**: Skip 5 tracks in 3 seconds?
3. **Lifecycle**: Players start, stop, crash
4. **My skills**: I'd not consider myself experienced C and Vala :)

The debouncing problem

Without it:

```
Play → Publish
Skip → Publish
Skip → Publish
Skip → Publish
Pause → Clear
Resume → Publish
```

6 server requests within 5 seconds.

The solution: Wait and see

When something changes, wait 1 second.

If nothing else changes, publish.

If something changes, reset the timer.

Result: One clean publish per user action.

Where to show the tune?

Options:

- Contact list (too crowded)
- Separate panel (too hidden)
- **Conversation header** (great)

Component 4: UI Integration

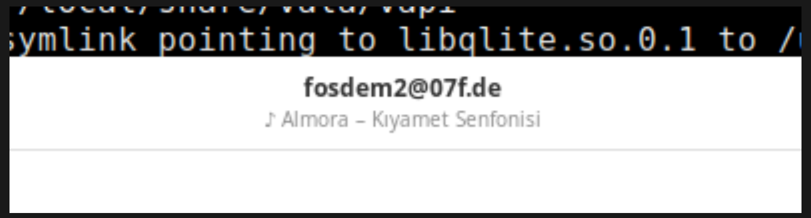
```
// In conversation_view_controller.vala

private void update_conversation_topic() {
    Tune? tune = tune_manager.get_contact_tune(
        conversation.account, conversation.counterpart);

    if (tune != null) {
        conversation_topic = "♪ " + tune.artist + " — " + tune.title;
    }
}
```

The result

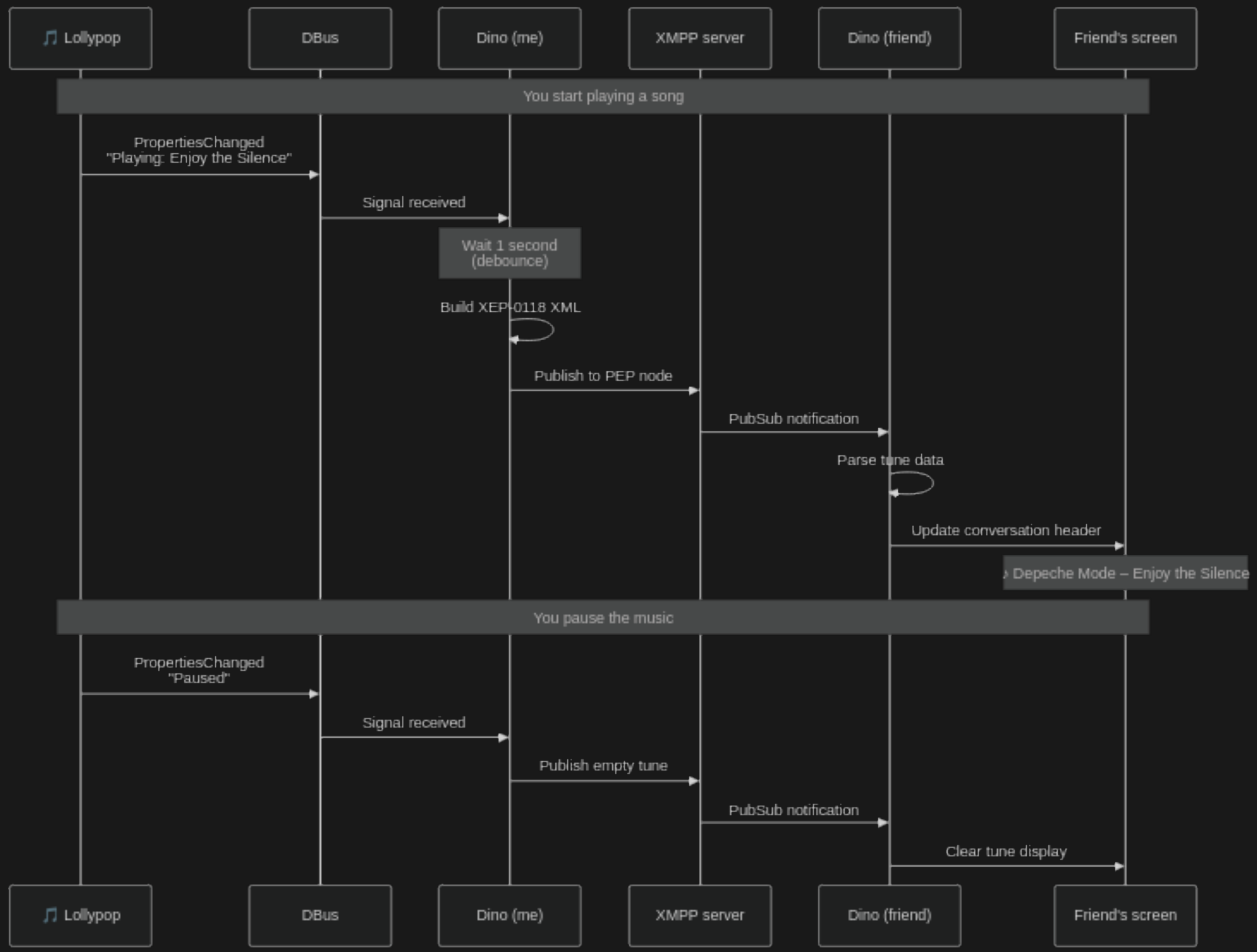
When chatting with someone playing music:



```
/local/share/vala/vapi
symlink pointing to libqlite.so.0.1 to /
```

fosdem2@07f.de

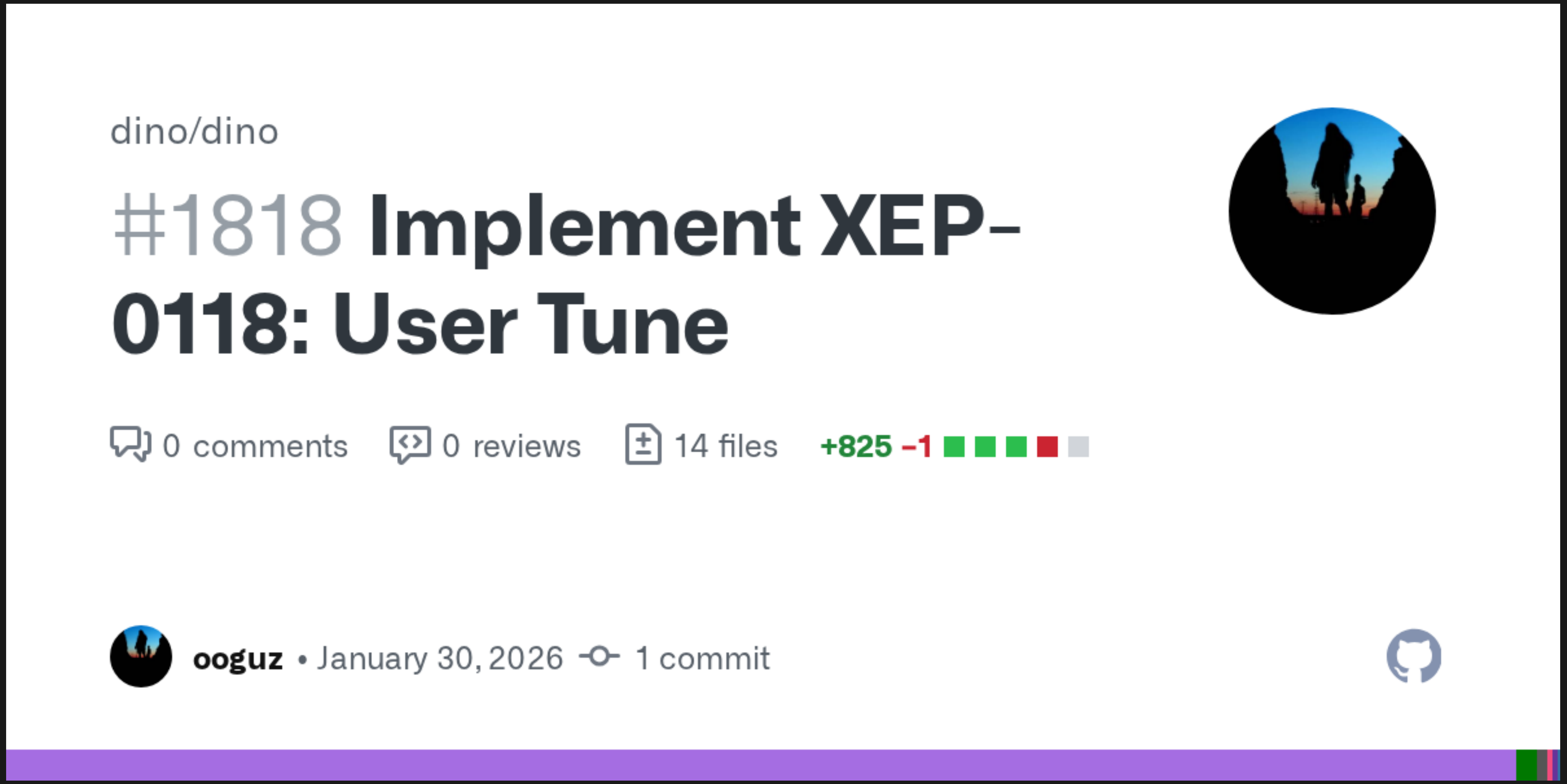
♪ Almora – Kıyamet Senfonisi



From player to your friend's screen.

Phase 4: Contribution

Making it usable for everyone



dino/dino

#1818 Implement XEP-0118: User Tune

0 comments 0 reviews 14 files +825 −1 🟩🟩🟩🟥⬜

ooguz • January 30, 2026 1 commit

The Present

Where are we now?

Compatibility

Client	User tune	OMEMO
Dino	✓ (my patch)	✓ full
Psi+	✓ (built-in)	~ (partial)
Gajim	~ (plugin)	✓ full
Conversations	X	✓ full
Monocles	✓ (built-in)	✓ full
Monal	X	✓ full

Now Dino supports both OMEMO and User tune together natively.

Why it ~~is~~ was not supported in new clients?

- Mobile: APIs are restrictive
- Prioritization: "Nice to have" vs core features
- Awareness: Some people don't know the XEP exists
- Time has changed: We are not looking for it

Server support

Every modern server supports this.

- ejabberd
- Prosody
- Metronome

No extra configuration needed since it uses PEP. But still, public servers may have different configurations, I've tried on 07f.de

Next steps

- Get the PR merged into Dino
- Make improvements on Dino (app selection etc.)
- Conversations will be my next target
- Encourage other clients to implement
- Encourage people to use and contribute to XMPP!

Try it yourself!

The patch is available right now.

1. Clone the repo (my fork @ GH) -> ooguz/dino
2. Build: `meson compile -C build`
3. Enable sharing in preferences
4. Play music!

Let's bring it back

The "Now Playing" feature brought joy in the 2000s.

It can bring joy again.

Thank You!

♪ *Now playing: The Audience - Questions*

ozcan@oyd.org.tr

https://ooguz.dev

`0x2D33E2BD3D975818`

oo@5222.de (XMPP+OMEMO)