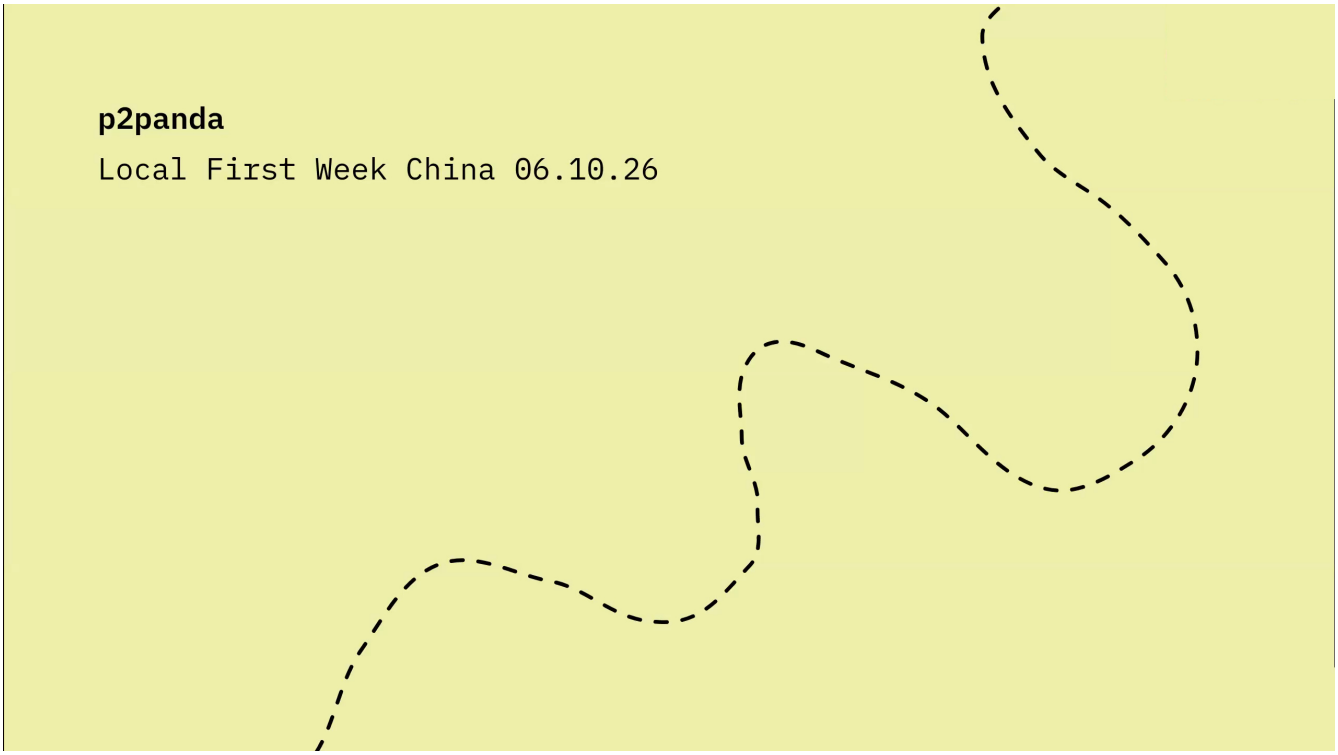
p2panda: Building Blocks for Local-First P2P Systems

Sam Andreae · 2026-10-06 · English

English transcript (with translations of the Chinese questions). Edited transcript of the presentation and discussion on 6 October 2026. Repetitions and clear recognition errors have been corrected while preserving the order, examples, qualifications and questions. Product status and plans reflect the talk date. Timestamps refer to the edited video.

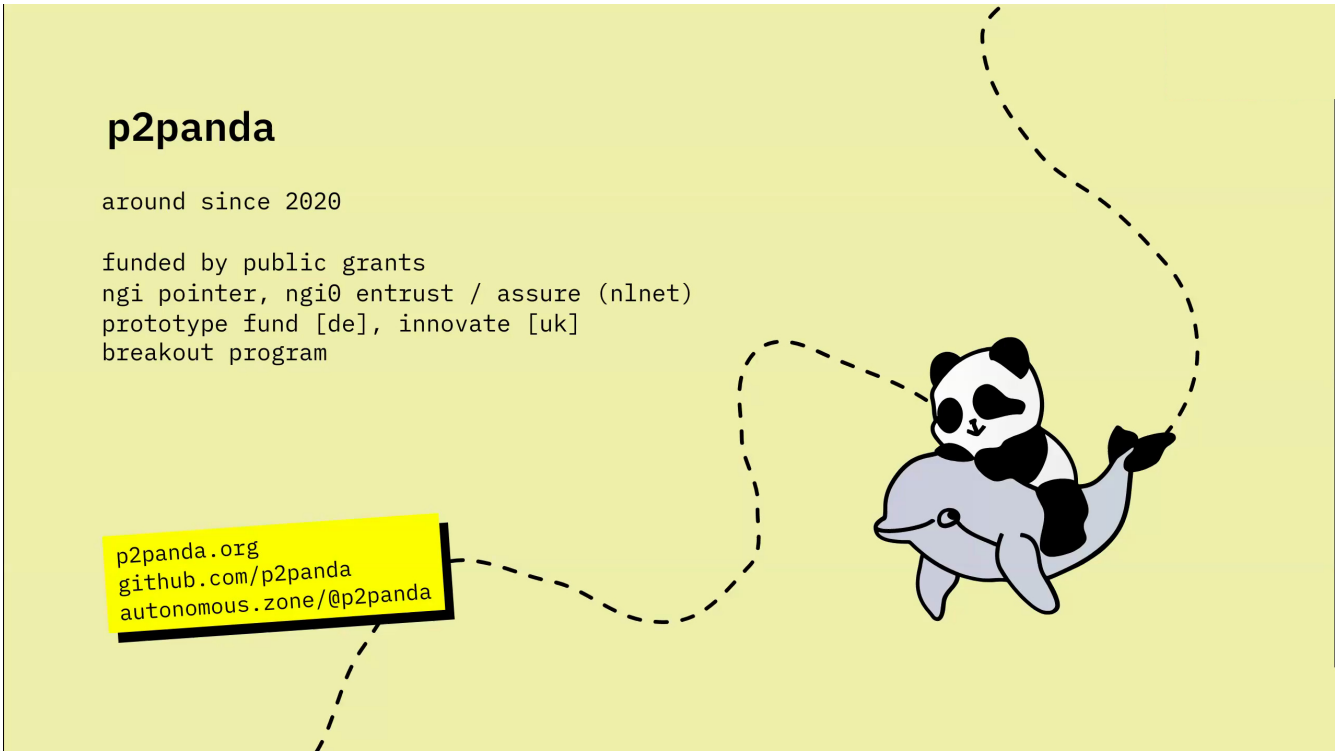
Introduction to p2panda



p2panda · Local First Week China · Frame 00:00:16

00:00:00 · Sam Andreae

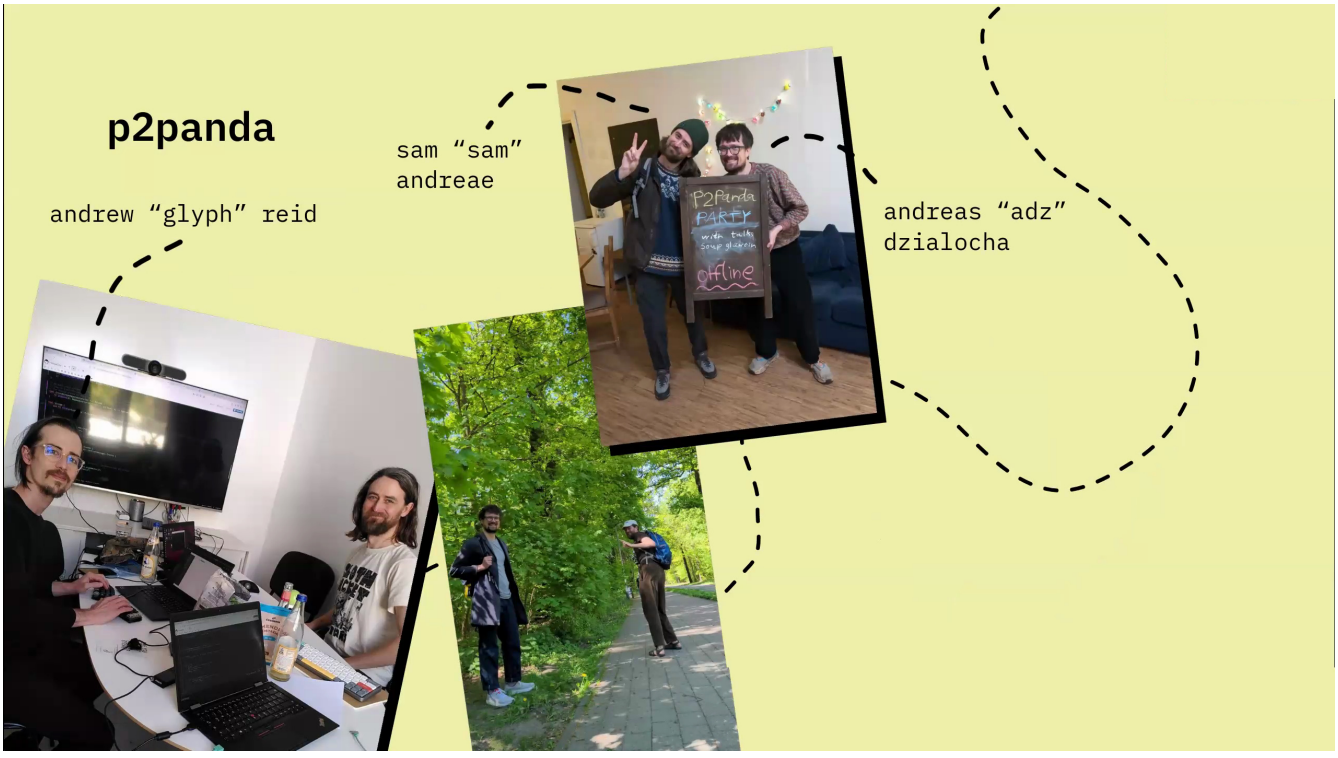
Hello, everybody. Welcome. I'm Sam from p2panda, and I'm here to give a presentation about the project. Thanks very much to the organizer for inviting us. It looks like you've already had many really great presentations from people we're familiar with, and I think there are still a few more. I'm very happy to be here. Next slide, please.



Project introduction, public funding, and links · Frame 00:00:46

00:00:31 · Sam Andreae

We are p2panda. We've been around as a software project since 2020. We're funded mainly by public grants from the EU and the UK—those are the sources so far. This includes NGI, NLnet, Innovate UK, and the eQualitie Breakout Program. We've got a website, our code is on GitHub at the moment, and we're on the Fediverse here.



The p2panda team · Frame 00:01:36

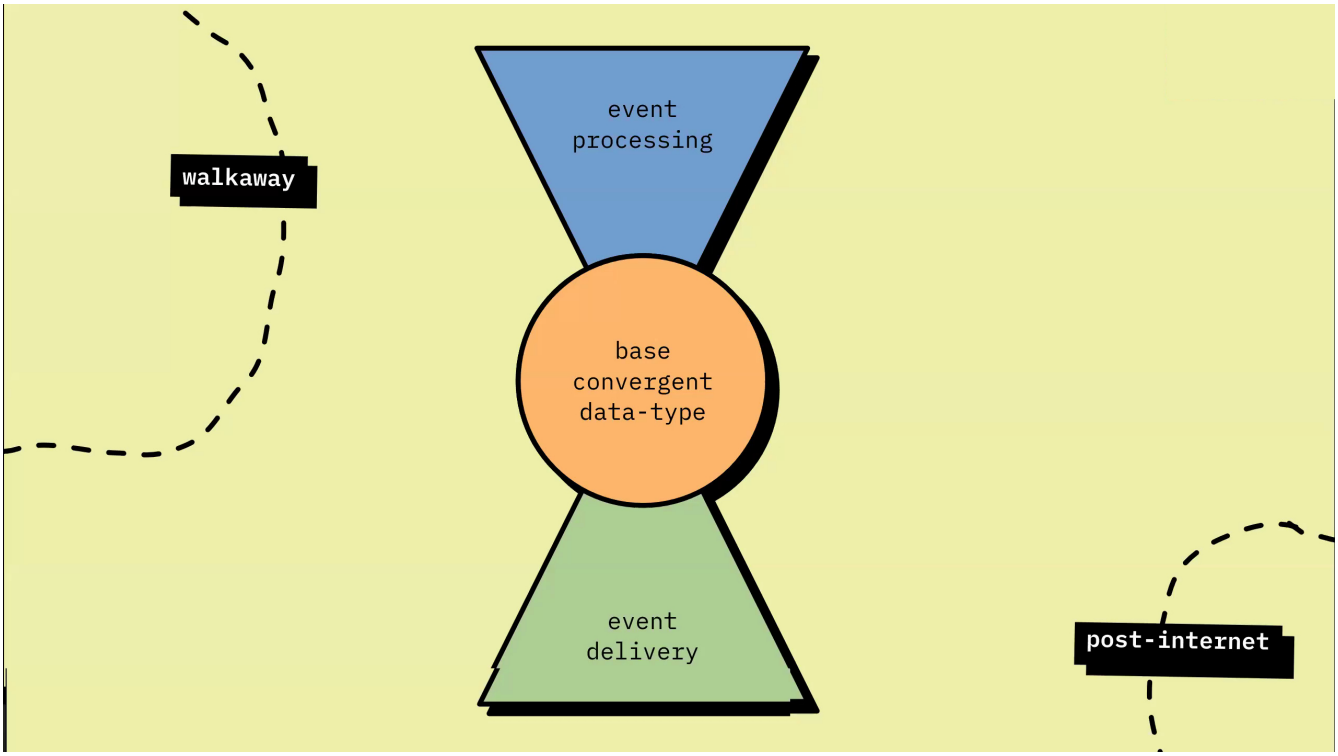
00:01:16 · Sam Andreae

When I say "we, p2panda," the core team is myself, Sam—I live in London—and Glyph and Andreas, who both live in Berlin. We've all been involved in p2panda for many years, and outside of that, for even longer, in related areas around the P2P space, let's say. Oh, sorry, just go back a slide.

00:01:56 · Sam Andreae

We're the core team, but, as you'll see, we do a lot of collaboration. There are many projects in our community. Really, I'd say there are more than 20 or 30 people actively contributing to p2panda: some coding, some building apps, and some working in design and UX. We have quite a broad community.

The Walkaway Stack



Walkaway and post-internet: event processing, convergent data types, and event delivery · Frame 00:03:36

00:02:32 · Sam Andreae

To start with, I want to go into our design philosophy a little: how we approach architecture in the P2P space. There's a term, the "Walkaway Stack," which has been coined not only by us but also by other projects around us. It's based around these three layers. At the top, the blue triangle is called event processing. At the bottom, the green triangle is event delivery. In the middle, we have the base convergent data type layer.

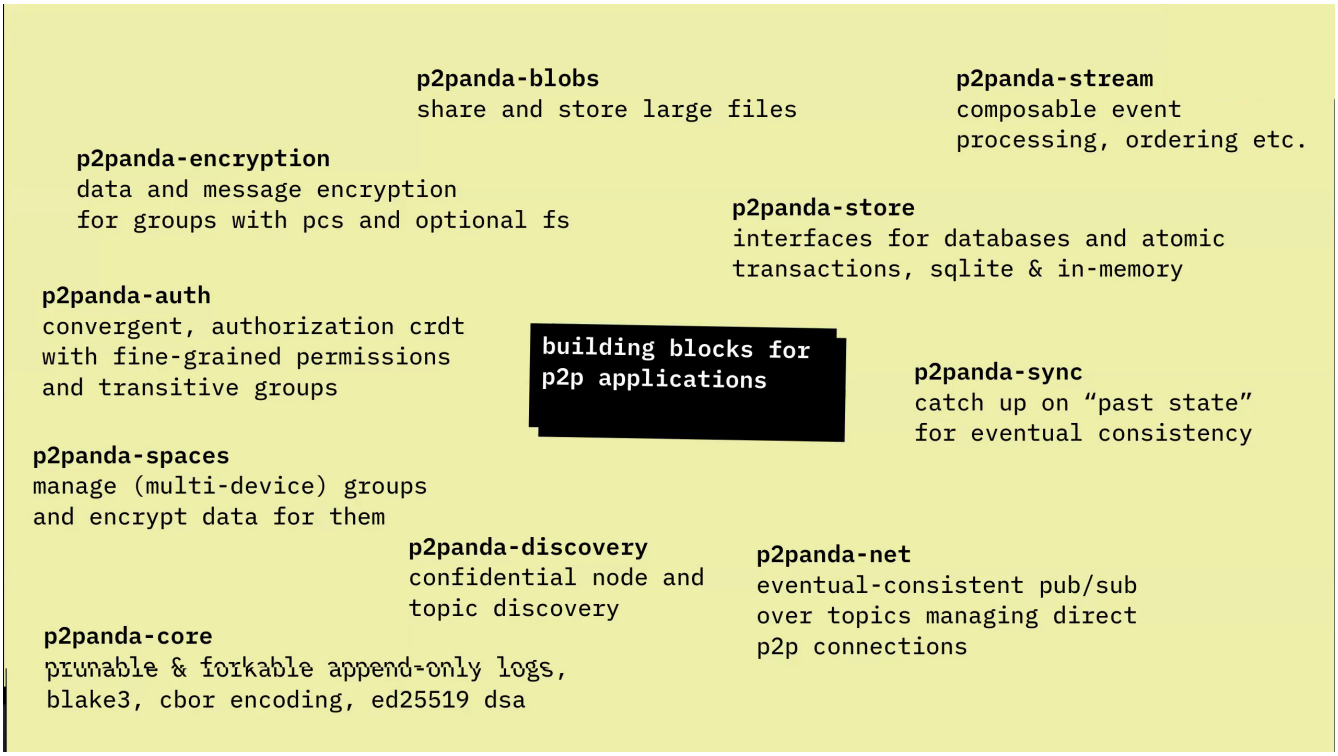
00:03:20 · Sam Andreae

What makes this a "walkaway" stack is that some layers can be taken away and replaced, while the other layers should still work. This hinges on the base convergent data type and event processing being, essentially, user data and application logic. That's the application domain; this is your local data. The event delivery layer is separate from this and can be replaced.

00:04:02 · Sam Andreae

This points toward the kind of post-internet world we work in. Event delivery often happens over the internet, perhaps via servers, but it could actually be replaced with many other transports or routing protocols. They might even retain some memory of what they're sending, to try to reach peers that are hard to reach. But that isn't the concern of the higher layer. As long as you receive what's needed for your base convergent data type, you're okay. This frames a lot of our thinking. Next slide.

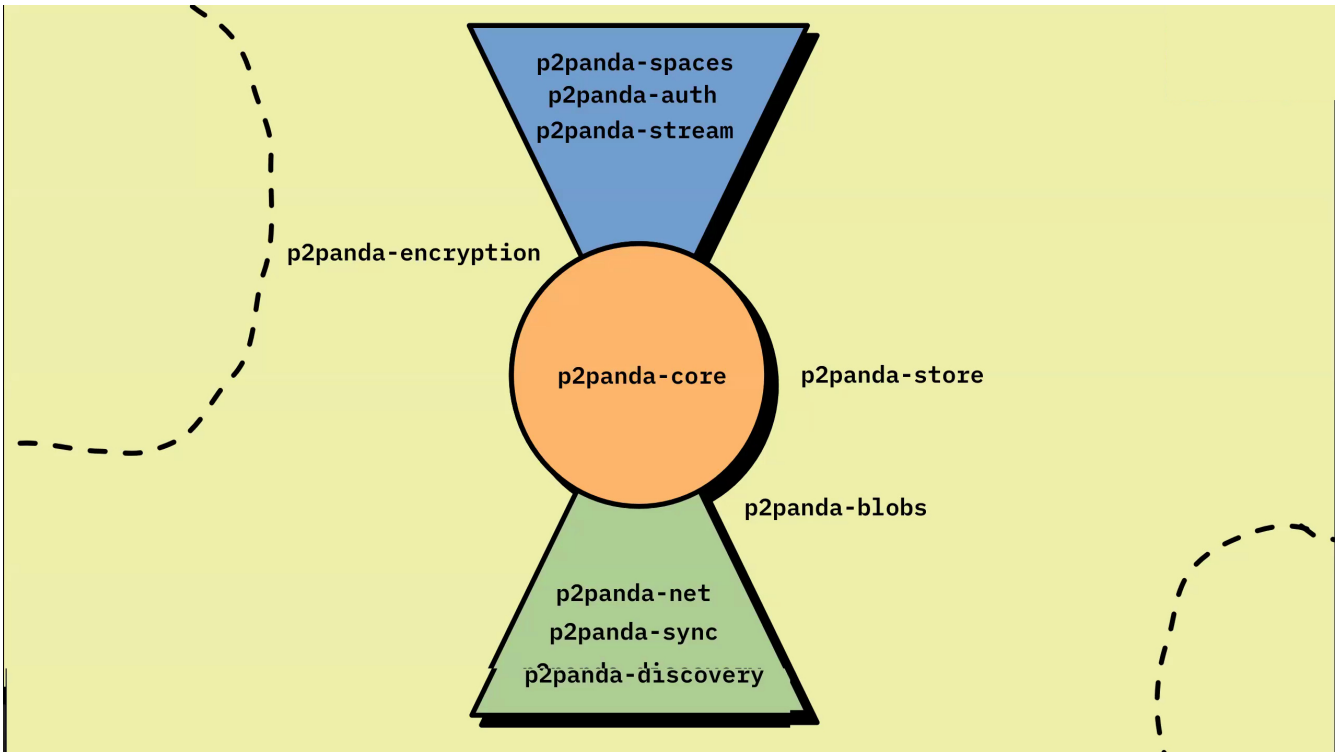
Modular Rust crates and architectural layers



p2panda modules and responsibilities · Frame 00:05:16

00:04:53 · Sam Andreae

What does that actually mean? What do we actually write in p2panda? We write a lot of Rust crates. Here is a cloud of them—an overwhelming cloud of Rust crates. We have a very modular system where we try to separate concerns nicely. It's a software development practice that we enjoy and find beneficial. Many of these crates offer generic interfaces that don't necessarily depend on other parts of p2panda. That helps with modularity and points back to the idea of parts being detachable. I'll go to the next slide, but this is our modular collection of things.



Mapping the modules onto the hourglass architecture · Frame 00:07:56

00:05:53 · Sam Andreae

Here, I tried to place them within the Walkaway Stack architecture. This is a new slide; I only made it today, and some of the placements weren't obvious. But here in the lower delivery layer, we definitely have p2panda-net, p2panda-sync, and p2panda-discovery. Discovery is for privately discovering peers interested in the same things as you. Sync is for actually exchanging the data. Net handles various things around transport and orchestrates those other two layers. These are things we provide as p2panda; these are crates that we write.

00:06:49 · Sam Andreae

But they're in this lower layer, so we're not actually dependent on them. We can use different transports and different approaches to sync. Recently, we've done some work more toward mesh networking. For a broadcast setting, we needed slightly different ideas around sync. This is that post-internet, pure-broadcast world—perhaps just radio. We provide these things, but, again, the upper parts don't depend on them.

00:07:22 · Sam Andreae

p2panda-core contains our core data types. We use append-only logs, and rely heavily on them for sync. We provide these data types here, but, again, you could take them all, plug in a different transport, and they would still work fine. Then, up in the blue event-sourcing layer, you can think of most of these as processes in an event-sourcing pipeline. We define what message types are expected, and as long as we receive those somehow, we can process them.

00:08:13 · Sam Andreae

Auth is about group management in a decentralized setting, which is a very difficult problem area. Spaces is about encrypting data for a group. It uses p2panda-encryption, which is placed a little to the side because it may touch multiple layers. This is a fresh slide, and I thought there was still some ambiguity here: p2panda-encryption is in the event-sourcing pipeline, but it's also encrypting things to be sent, so it's very important to the event delivery layer as well.

00:08:52 · Sam Andreae

We also have p2panda-store and p2panda-blobs at the side, and it's not completely clear where they belong either. It's not all black and white. But this is where our crates sit in our philosophy, you could say. Next slide.

A unified API for application developers



The out-of-the-box p2panda high-level API · Frame 00:10:16

00:09:13 · Sam Andreae

Not everybody wants to work through all these modules and understand how to plug them together. We take responsibility for that as well. Here, we provide p2panda itself: the out-of-the-box p2panda API for application developers. Everything you saw on the previous slide is brought together in one crate, with a unified API that we hope is easy for application developers to use. It still follows event-sourcing API patterns, so people building applications with it will likely want to follow that same pattern. But that's a fairly established approach, not something unique to the P2P space.

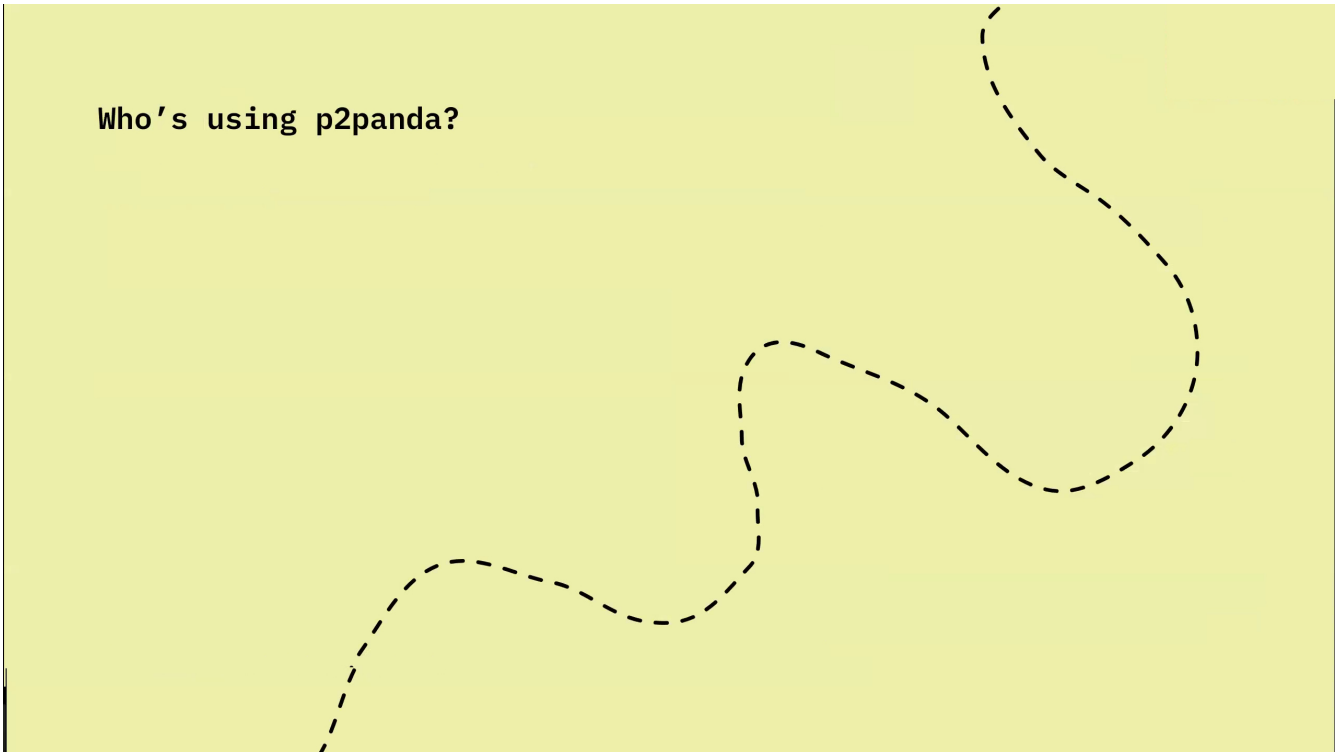
00:10:11 · Sam Andreae

Everything you get with it is listed here. In the all-in-one solution, you get confidential discovery of secret topics, efficient sync, causal ordering, event streaming, ephemeral messaging, direct connections, group encryption, fine-grained access control, multi-device support, crash resilience, and garbage collection.

00:10:38 · Sam Andreae

Here's a small example—oh, wait, sorry, back one slide. You can see a little example of the API here: creating a random private topic and instantiating a node. When you subscribe to this topic, you get a sender and a receiver for a stream. These are eventually consistent streams. If you send into one and somebody else subscribes later, the expectation—the guarantee—is that you'll both arrive at the same state. If they missed messages, they'll receive them eventually. Next slide.

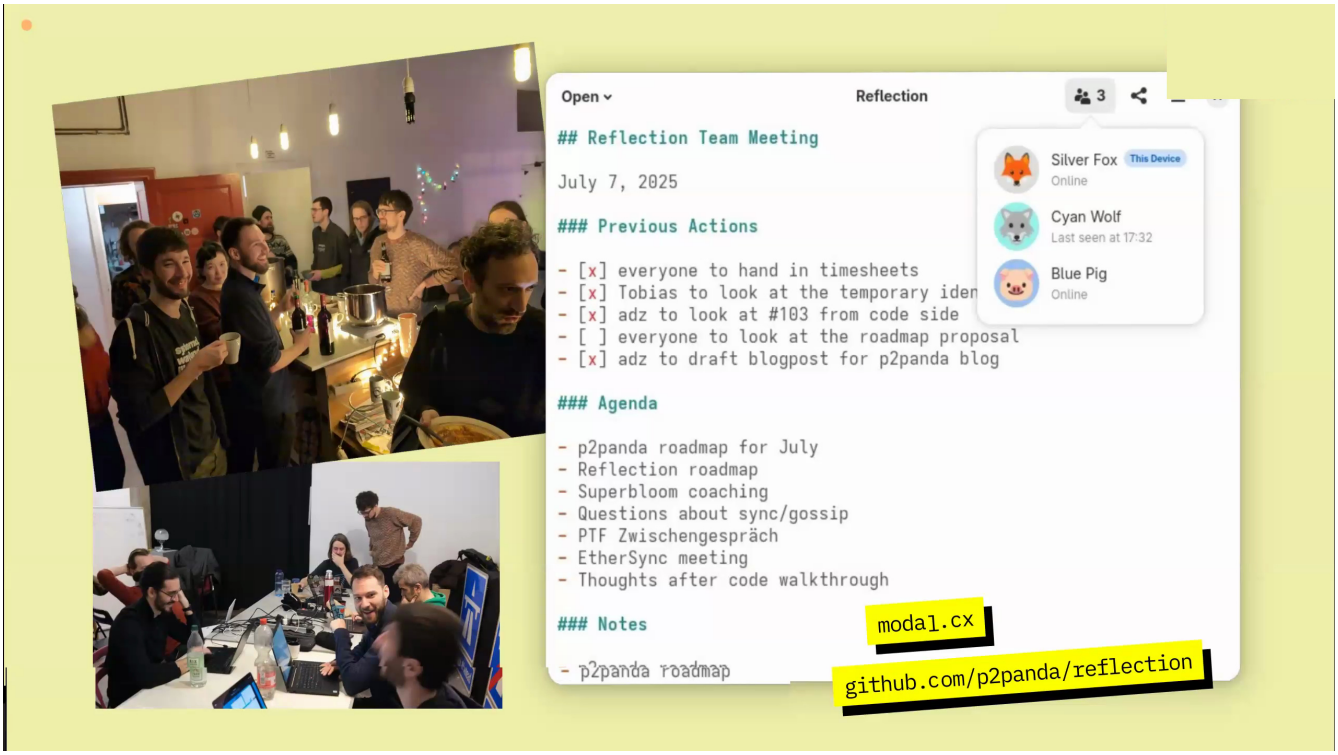
Applications and collaborating communities



Who’s using p2panda? · Frame 00:11:36

00:11:28 · Sam Andreae

As I said earlier, we have a good community around us. Many of these people are using p2panda. We're involved in some of their projects and not in others. I thought it would be nice to share a few of them at this point. Who's using p2panda? I'll go through a few examples.



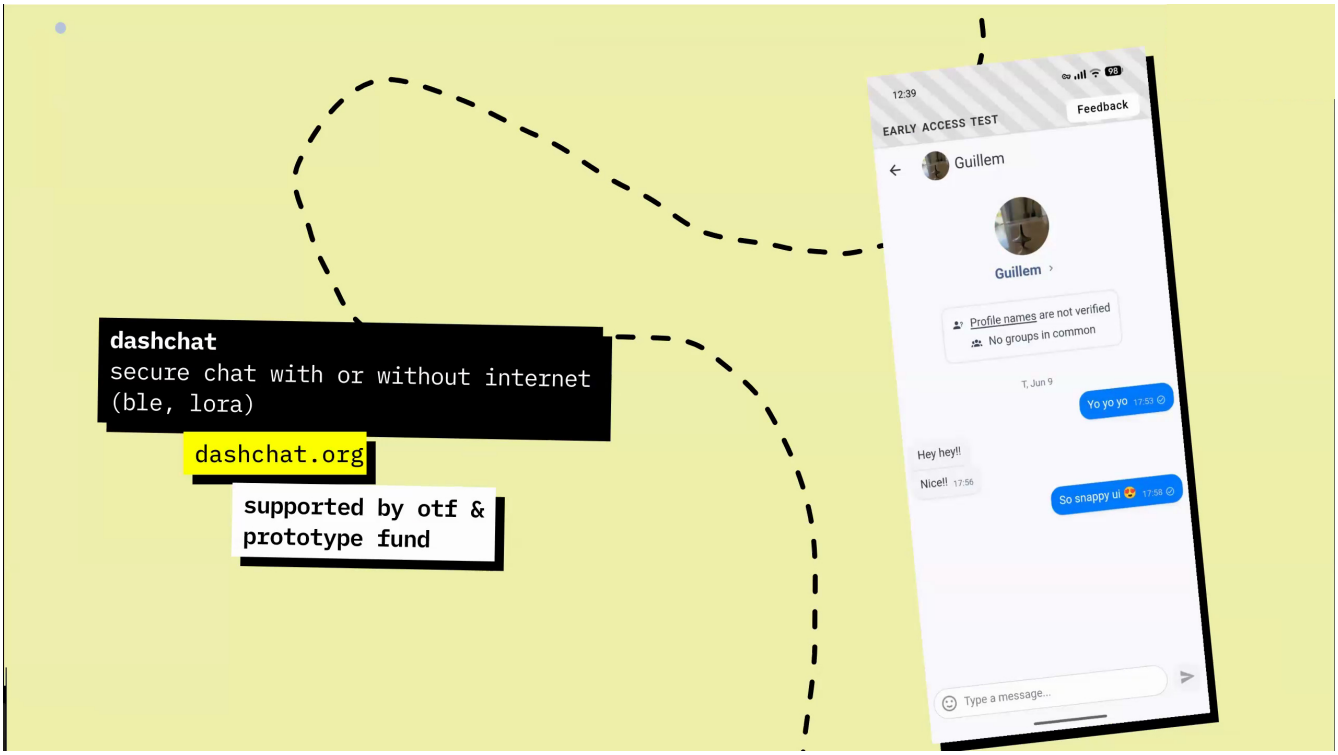
Reflection: collaborative notes · Frame 00:12:16

00:11:51 · Sam Andreae

This is the Reflection application. It's a GTK app built by members of the GNOME community, available on Flathub Beta. It uses p2panda, and it originated in a close collaboration between GNOME and p2panda. Over many years, their feedback has really helped shape the APIs we provide today.

00:12:31 · Sam Andreae

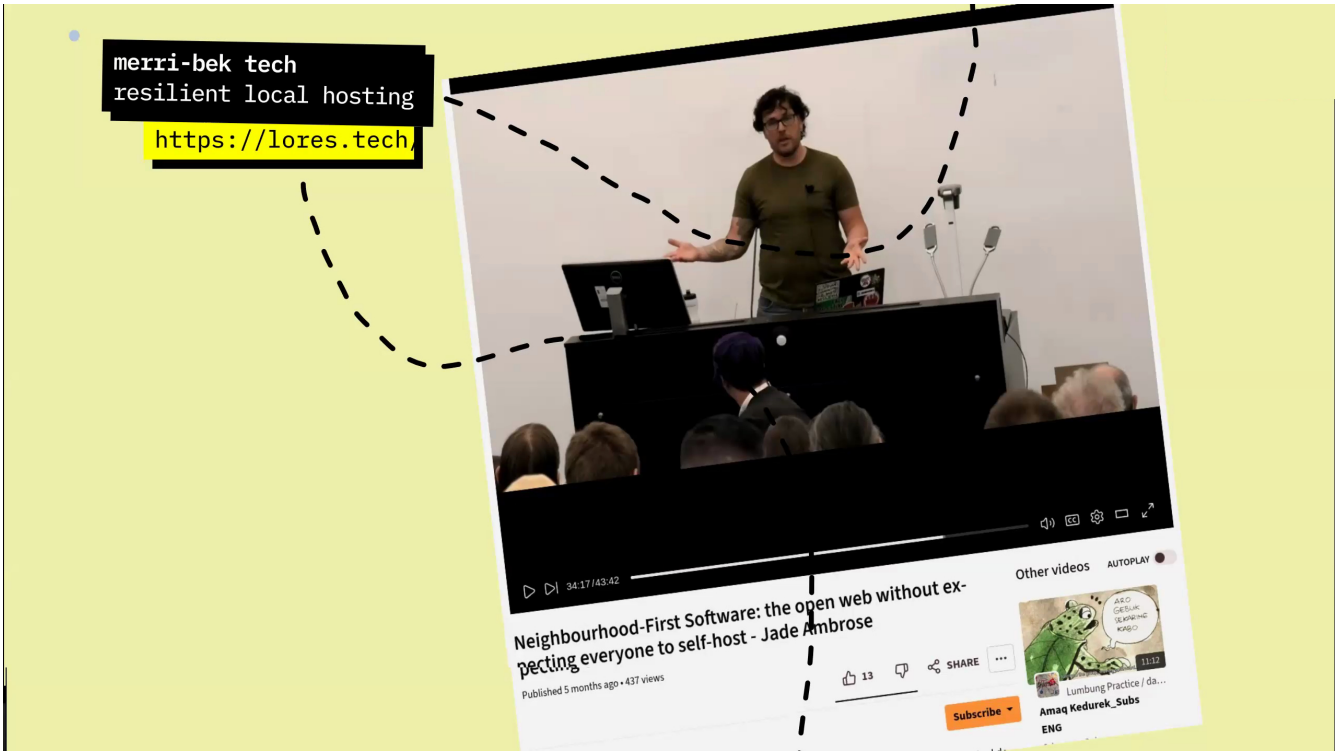
It's a collaborative text editor. We use it every day for our work, and we know other projects do as well. In its current form, it's stable and very reliable. There are still features we want to bring in, but I recommend trying it out. Next slide.



Dash Chat: offline chat over BLE and LoRa · Frame 00:13:16

00:13:04 · Sam Andreae

This is the Dash Chat project. This team is building a secure chat application that works with or without the internet, using p2panda. They have their own grants and other funding. We work quite closely with them in a consulting role: we help them, and they also give us a lot of feedback on our APIs. They're moving very fast. We've been testing the app, and it's really nice.



Merri-bek Tech: resilient local hosting · Frame 00:14:26

00:13:54 · Sam Andreae

This is Merri-bek Tech, and you can see Jade here on the slide. Their focus is more on resilient federated systems for local communities, particularly in the face of climate disasters. I think that's their main focus: communities can remain connected within their locality, and, when possible, bridge to other communities. They're doing really great community work, training people to deploy their own systems and maintain them as well. It's a really great project. Next slide.



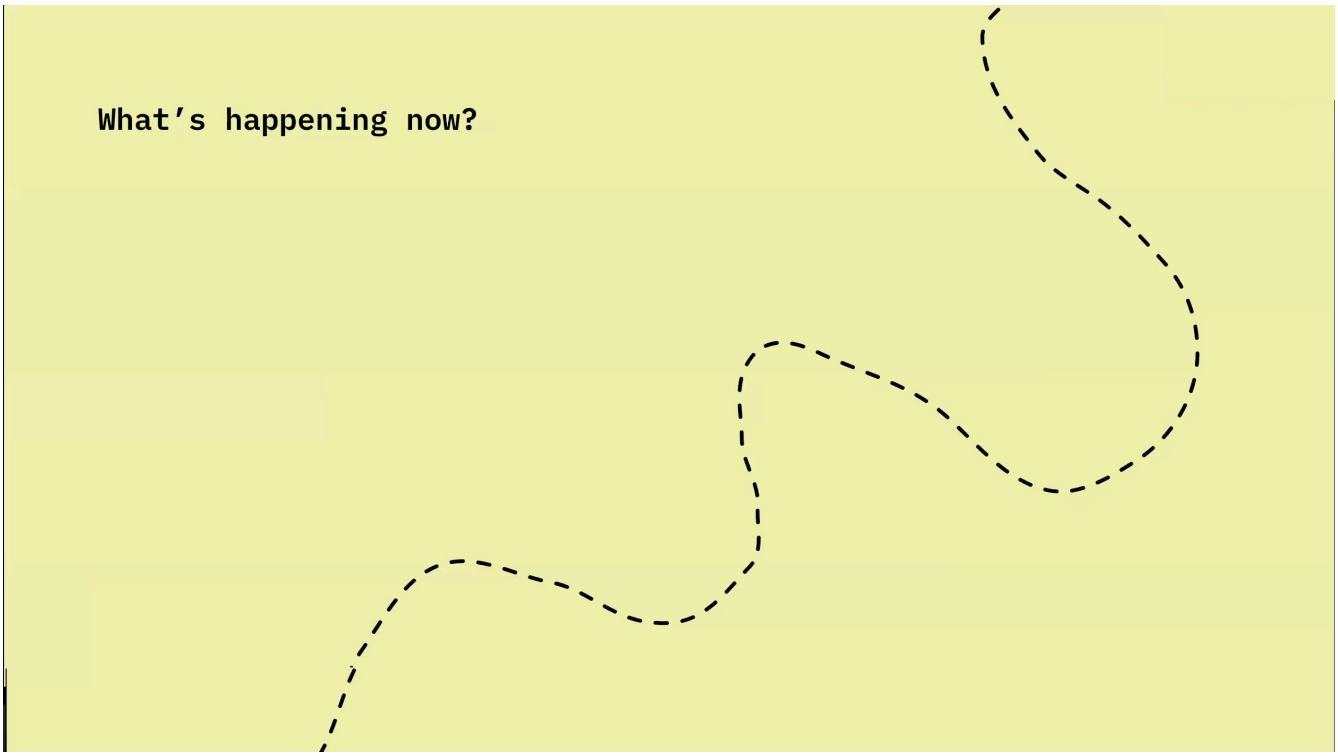
Toolkitty: coordination of people, spaces, and resources · Frame 00:15:56

00:15:02 · Sam Andreae

This is Toolkitty. We're closely involved in the design and in building a prototype of the app, which is funded by Innovate UK. It comes from a time before p2panda even existed, when Andreas and I were involved in organizing festivals and publishing in a music context in Berlin. We were already interested in decentralized systems, and we wanted to organize in a decentralized way. The precursor to Toolkitty is called Hoffnung3000, a platform for running a self-curated festival.

00:16:03 · Sam Andreae

Toolkitty is similar: it's for sharing resources and organizing events in a decentralized manner, on a decentralized network. It now runs on top of p2panda. We got it to the prototype stage, did a lot of good design work, and validated it. But it's on hiatus at the moment. We need a little more funding to really get it going again, and we're working on that.

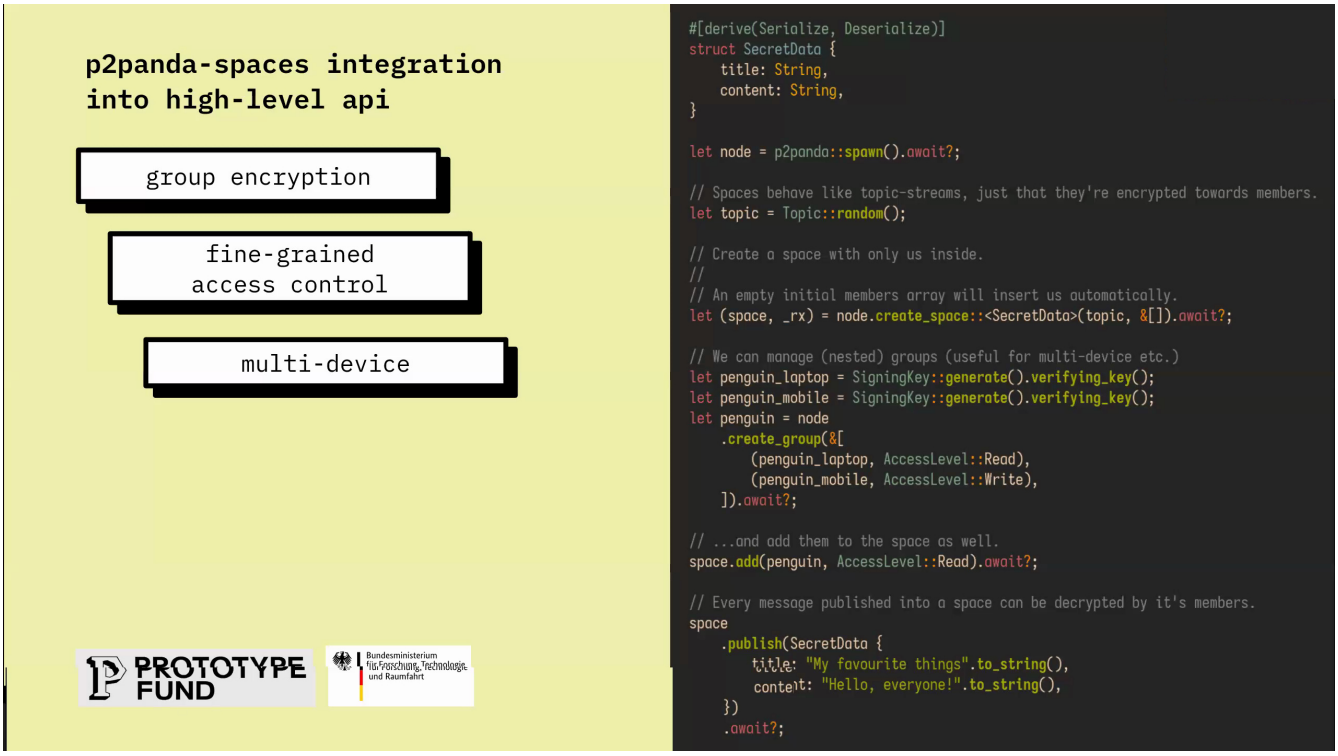


What’s happening now? · Frame 00:16:51

00:16:44 · Sam Andreae

Those are a few of our collaborators and related projects. The final section of the talk is about what's happening in p2panda now. We have quite a lot on the horizon, and I'll go through it.

Spaces: group encryption and permissions



Integrating p2panda-spaces into the high-level API · Frame 00:17:46

00:17:03 · Sam Andreae

The major work we're undertaking at the moment is integrating Spaces—group-encrypted subscriptions—into the high-level API. This provides event-sourcing API patterns such as replays and crash resilience, on top of everything you get from Spaces: group encryption and fine-grained access control. These give you multi-device support, because group management in Spaces allows nested groups. That's how you can model multiple devices.

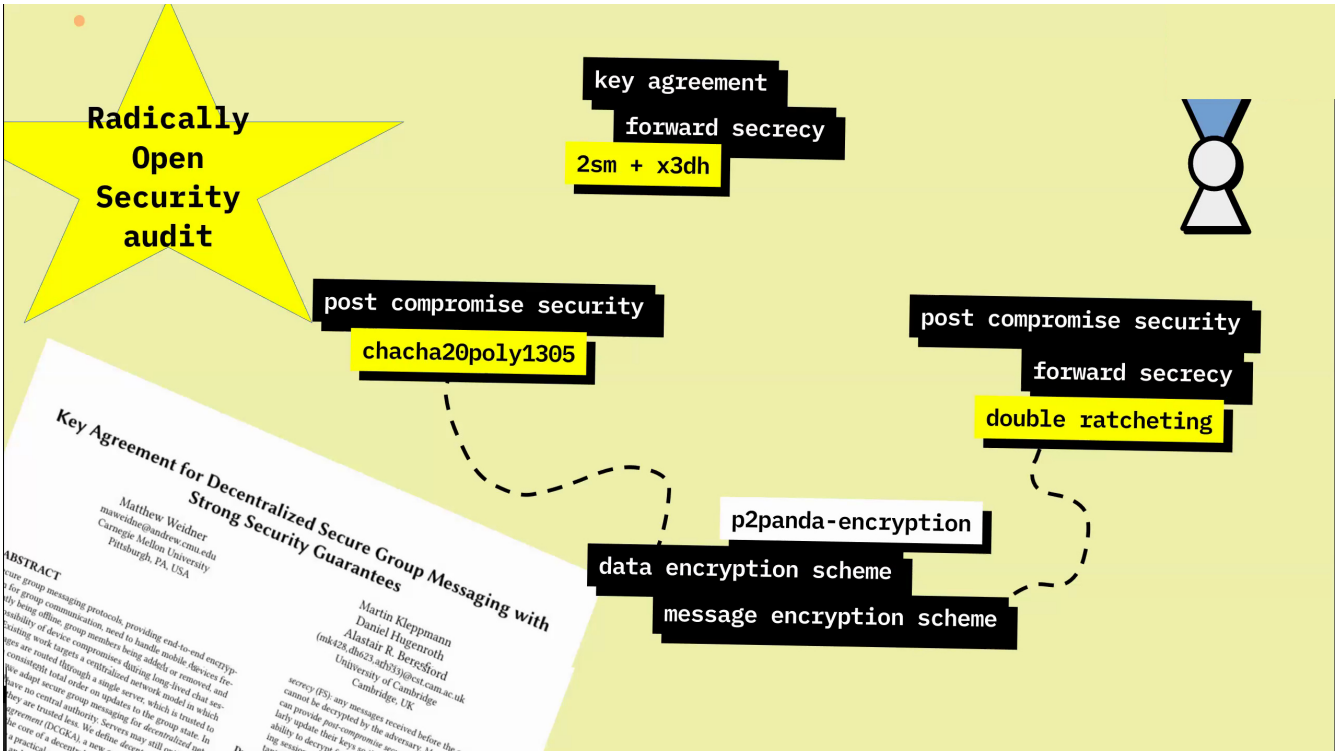
00:18:05 · Sam Andreae

This brings in many features that we've long felt are critical to having a really useful P2P framework. It represents a big milestone for us, and we're really excited. It's all going well. We've done a lot of it and we're in the final stages. There's still a lot to do, but it's going very well. This is part of a project funded by Prototype Fund, which is also a collaboration with the Dash Chat team.

00:18:50 · Sam Andreae

There's an example of the API on the right. It's similar to the example I showed a little earlier. Subscribing to a topic is the same as subscribing to a stream. What you get on top of that is the group management API. Okay, next slide.

Key agreement and security audit



Open security audit and encryption design · Frame 00:19:46

00:19:18 · Sam Andreae

We've talked about all our crates, but part of what spaces actually does is integrate p2panda-encryption with p2panda-auth, which provides group management. And p2panda-encryption is where the key agreement protocol lives. This is based on “Key Agreement for Decentralized Secure Group Messaging with Strong Security Guarantees,” a paper by Martin Kleppmann and others.

00:19:51 · Sam Andreae

We're very happy that our implementation has just had a security audit by Radically Open Security, and it went well. We have some feedback and some changes to make, but essentially it was a very positive process. We'll be happy to share that and share the findings quite soon. This is something we've been really hoping for for a few years now, so it's another milestone we're really happy to get to.

Support nodes: shared infrastructure and privacy



Support nodes: key-value storage, quotas, and flattened data · Frame 00:21:56

00:20:31 · Sam Andreae

There are two projects related to this subject, so I'll introduce the subject first: support nodes. This is another really important feature for local-first or P2P application development. It's all around the problem of how you get data when everybody's offline, or how you have some shared infrastructure. We call these support nodes. The concern here is: how do you share infrastructure while still keeping your sensitive data private?

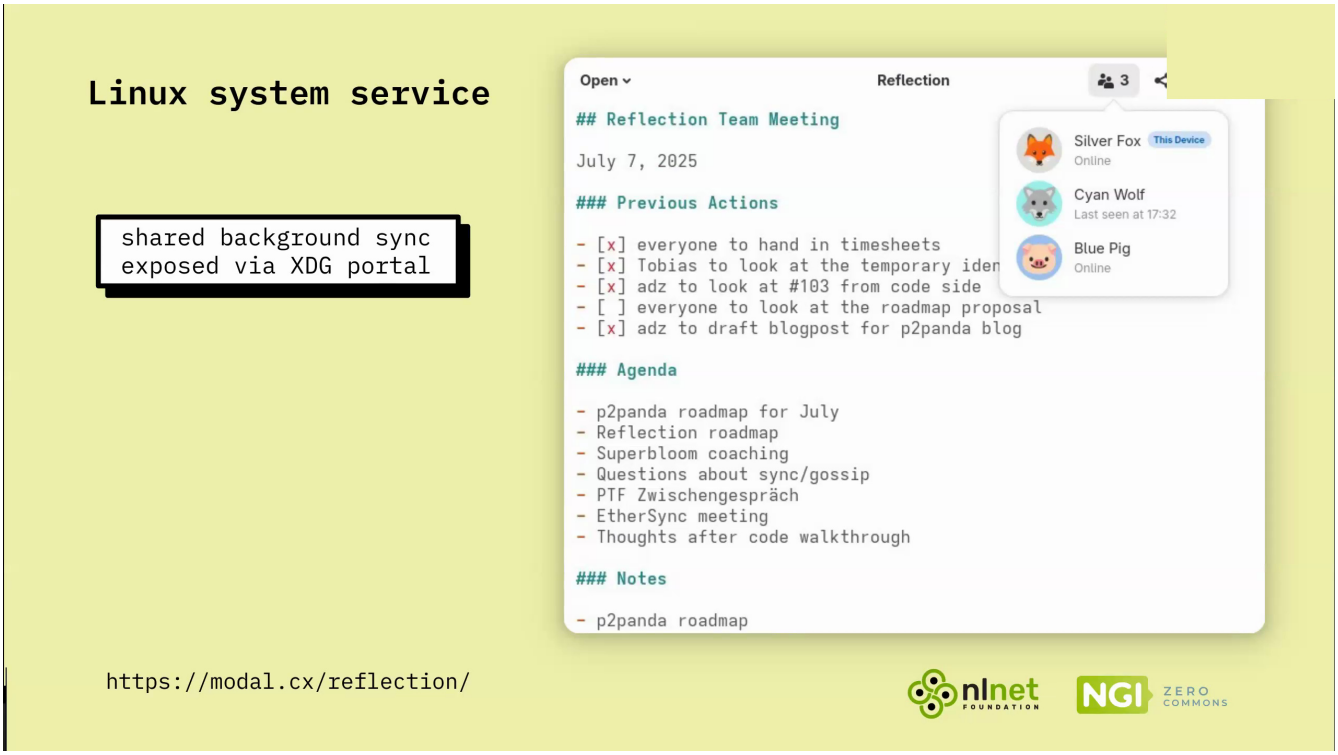
00:21:16 · Sam Andreae

With the work I was just talking about on group encryption, we can now encrypt all the data. So how do we store and share it while still exposing as little metadata as possible? With the work on spaces, we end up being able to have almost-zero-metadata encryption of operations. Then, with a support node, we can have quite a simple design: a key-value store. You can just put fully encrypted data there. The support node doesn't even know this is p2panda data. It just knows, “I'm going to store this content-hashed data for you, and you can retrieve it later.”

00:22:16 · Sam Andreae

With the support node unaware of the particularities of our data types and protocol, it can have its own logic around things like garbage collection or how many resources a user can have, using things like time-to-live or space-to-live. Access control can be managed by whatever mechanism is chosen for your support node implementation—for example, access tokens. In the slide, you can see how you can represent p2panda operations in this store. That's an introduction to two concrete projects we have underway.

A system-level P2P service for GNOME

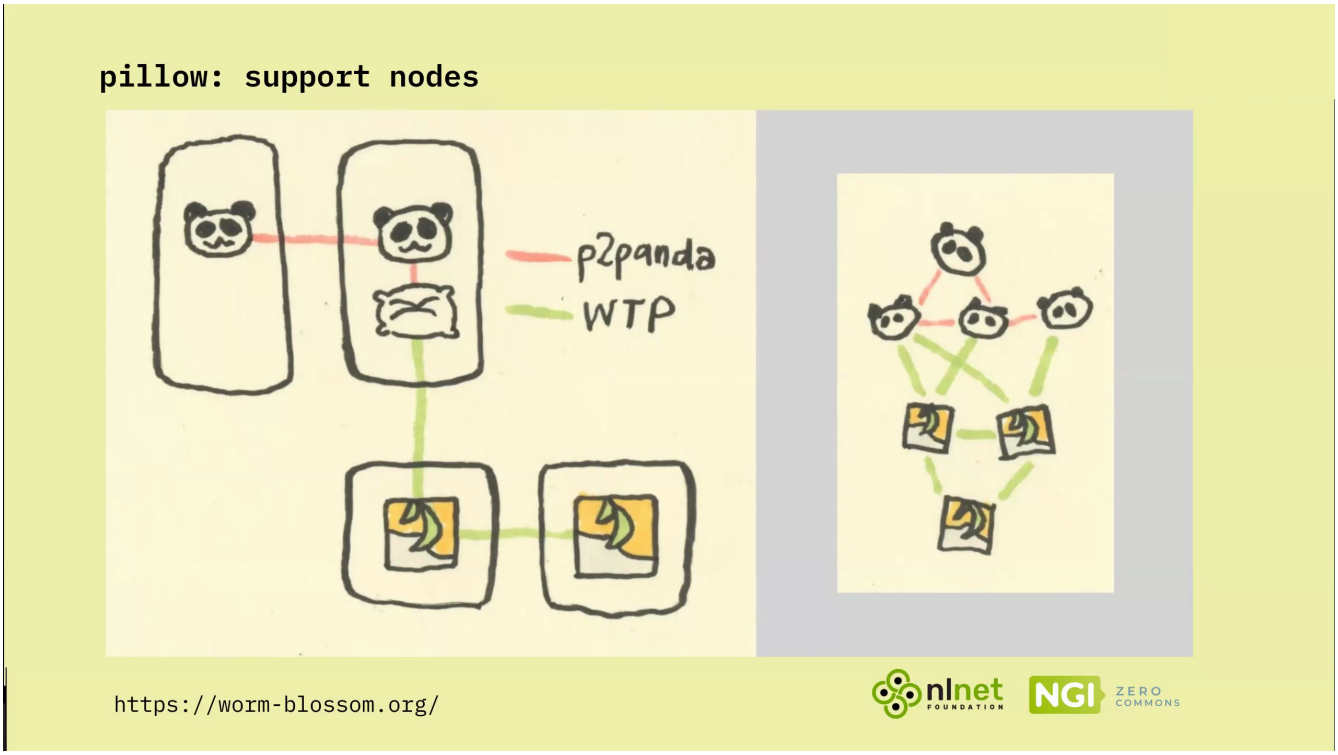


A Linux system service and XDG portal · Frame 00:23:46

00:23:17 · Sam Andreae

This is a Linux system service. It's an NLnet-funded project, again in collaboration with the GNOME community. Here we're exploring building a system-level P2P service that applications can request access to via portals. The background service is essentially a support node: it's doing work for multiple actors in this system. As part of this project, we'll implement a support node that runs as a system service. The example again shows Reflection, because that's one example of an application that might want to use this.

Collaboration with Willow and worm-blossom

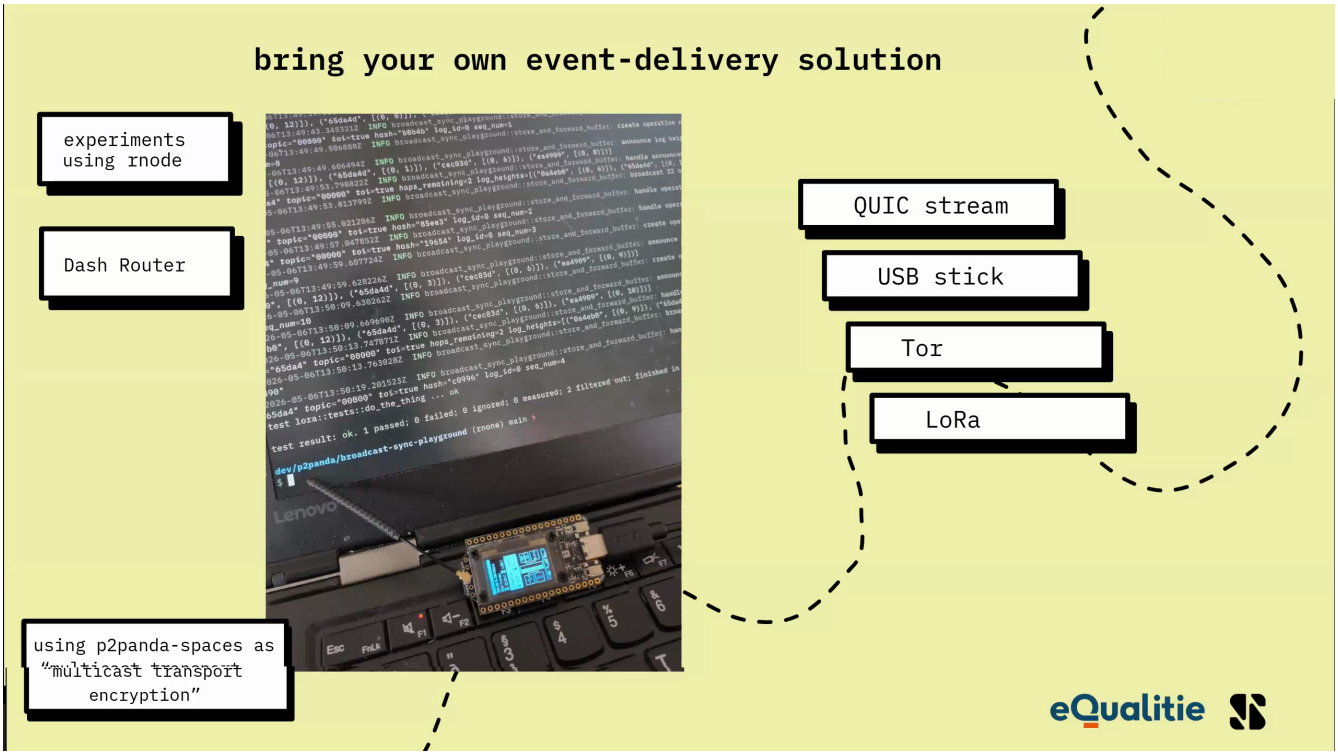


pillow: support nodes · Frame 00:24:56

00:24:23 · Sam Andreae

Here's another project, also around support nodes. This is a collaboration with the Willow protocol, or the worm-blossom organization, who are old friends of ours and old comrades in this space. They will also implement a support node, but using their own data types for synchronization between support nodes, synchronization with end peers, and access control. They have their own access tokens, which are delegatable capabilities. You can then have patterns where a support node gives access to, say, a system service, which can further delegate access to applications on a system. Here are some fantastic diagrams of the system.

Dash Router and multiple transports



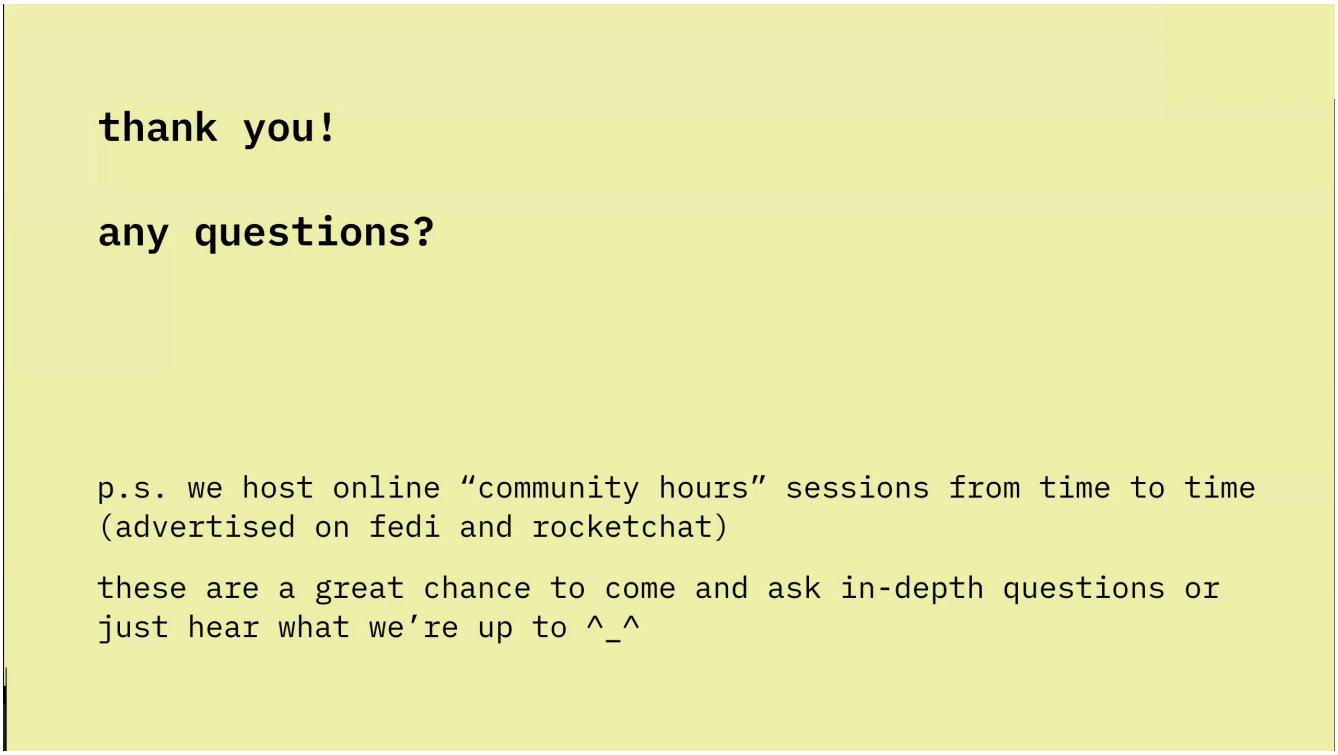
Bring your own event-delivery solution: QUIC, USB, Tor, and LoRa · Frame 00:26:36

00:25:45 · Sam Andreae

I'll point back to the walkaway stack architecture. Here we're very much in the event delivery area, the lower detachable layer. We have a couple of projects at the moment that touch on this. One is the collaboration with Dash Chat. We talked about the spaces integration that they'll be using, but another aspect of that project is their mesh routing protocol. They are implementing Dash Router, an approach to a mesh routing protocol with store-and-forward properties, where peers relay—or hold and relay—data for others.

00:26:43 · Sam Andreae

The other project touching on this is that we'll be working on multicast—or multi-transport—APIs, and a concrete implementation of a new transport for p2panda, funded by eQualitie. Both of these can be understood as new event delivery implementations that the p2panda stack is fully ready to integrate.



Questions and community channels · Frame 00:27:56

00:27:26 · Sam Andreae

Okay, great. Thank you. That's a fairly fast introduction to p2panda and the p2panda projects. We're going to open up for questions now. I know we have a few others here—definitely Andreas, and... Hi. Oh.

00:28:08 · Host

Is that okay?

00:28:11 · Sam Andreae

Hi, sorry. I thought I got kicked out, but I seem to still be here. One minute.

00:28:18 · Host

No worries.

00:28:21 · Sam Andreae

Can you still hear me? That's strange. Okay, great. Thank you very much for listening. We're going into the questions round now. I'm happy to answer any questions, and I believe Andreas and Glyph are here as well.

00:28:38 · Host

If anyone has questions, you can start asking now. There's already one question in the chat box.

00:28:53 · Sam Andreae

Right.

Q&A: root authority and everyday administrators

00:28:54 · Host

It says, “How would you recommend modeling a separate root authority and day-to-day administrators with p2panda-auth?”

00:29:08 · Sam Andreae

That's a good question. This is about how you would model a superuser admin, I think. For this, you would have nested groups. There's a concept of nested groups in p2panda-auth: you can have a root group, and other groups can be added as members of that group. Their members become transitive members of the root group.

00:29:47 · Sam Andreae

Here you can see how the root authority would be the root group. You can add members who have a lower access level, so they will never have more access than the root group. That's a very brief explanation. Obviously, there are lots of details depending on the use case, but that's one way you could do it.

Q&A: bottlenecks in very low-bandwidth networks

00:30:36 · Host

I have one question. When the internet is unstable or bandwidth is extremely limited—for example, over LoRa or packet radio—what becomes the main bottleneck for the...?

00:30:56 · Sam Andreae

It's a good question. I'm not sure we have a conclusive answer to this. The problems described there—payload size, metadata overhead, cryptographic overhead, or reconciliation—are indeed all challenges. I also want to hear what Andreas or Glyph might say, because they've both worked very closely with mesh networks and encryption in different contexts. Maybe I'll just say that... Well, no, I'd actually like to hear from them. Glyph and Andreas, do you have any thoughts on this?

00:31:46 · Andreas Dzialocha

I can give some insights. I think you already said it yourself: it will require more simulations and deployments on real, larger networks to really know where the bottlenecks are. This is all ongoing research. We just lack real-world deployments of these systems to really know where the pain points are.

00:32:11 · Andreas Dzialocha

One thing we can already observe, and are definitely trying to reduce, is the payload. In LoRa, we're limited to 255 bytes per packet. Obviously, we can introduce framing, but we also try to reduce the actual operation size to that level. That's very hard, especially for key agreement schemes, where we have to send a lot of cryptographic material before we can start communicating. There will definitely be some interesting trade-offs we need to explore.

00:32:51 · Andreas Dzialocha

The two factors we're currently trying to improve are the number of messages sent and their size. On message count, we're looking into a reconciliation or synchronization protocol inspired by research from a friend at the University of Basel. We explored using Bloom filters to exchange the state vector between peers. We could reduce that Bloom filter to below 255 bytes, allowing us to represent a cache of 64 messages—sorry, 128 messages, I think, or up to 128. It was really nice to see, in this first exploration, that we can deliver the state vector within one LoRa message.

00:33:52 · Andreas Dzialocha

The same goes for append-only logs. If your application is organized with very few logs, you only need one message to announce your current state. It's literally just the sequence number or the hash of the latest operation you have. Those are the kinds of considerations we have. If you're interested in this topic in general, and in people who have experimented a little further with append-only logs, I recommend checking out the tinySSB project. I can put a link in the chat.

00:34:27 · Andreas Dzialocha

They're also experimenting with removing metadata in the header that may be redundant. For example, we're currently resending the public key in every message, which you obviously don't need to do all the time. Maybe it's enough to include it only in the first message. With the append-only log structure, you can go back to the beginning of the log to find the public key. Things like that.

00:34:55 · Andreas Dzialocha

We've also been exploring ideas around not including the signature in every message, but only at checkpoints, every hundred messages or something. You'd have a small risk window where you haven't verified the signature or the authenticity of the message, but you wait for a checkpoint every ten or twenty messages, something like that.

00:35:19 · Andreas Dzialocha

This would save you another 64 bytes in our cryptographic scheme. It's also interesting for post-quantum cryptography, where signatures might be two kilobytes. These considerations are interesting in many respects, but they're just some ideas. I think the tinySSB folks have pushed pretty far in seeing how low you can go, including even sending packet radio over the Pacific Ocean via—what's it called?—radio waves.

00:35:59 · Host

Cool, thank you. There's a new question in the chat box.

Q&A: can browsers act as independent replication peers?

00:36:19 · Sam Andreae

I can read it and then maybe answer it. It says, “Have you experimented with running p2panda in the browser? Based on that experience, could a browser act as an independent replication peer using IndexedDB and core/auth/sync? What are the main limitations around persistence and sync recovery?”

00:36:46 · Sam Andreae

Yes. An earlier iteration of p2panda was actually quite heavily browser-focused. We had the architecture of having clients and nodes, with the clients in the browser communicating with a node. Often, a design pattern could be wrapping these together in, say, a Tauri app. There was a more opinionated, object-based query API for this, which was really nice for some settings—perhaps for web developers.

00:37:25 · Sam Andreae

But part of the feedback from the GNOME community was that they wanted a very different architecture. That started a multi-year journey toward this event-sourcing pattern. In the current form of p2panda, we don't have browser support implemented. We haven't experimented much with packaging the current framework for the browser. With the previous one, we did a lot of work with—I forget the name now.

00:38:04 · Andreas Dzialocha

WebAssembly?

00:38:05 · Sam Andreae

WebAssembly, yes, as well. We had Rust bindings and so on, and this worked fine. Related to that, we currently have FFI bindings into many other languages, though that's a slightly different subject. But in terms of whether it's possible: yes, definitely. As for the main limitations around persistence and sync recovery, I don't know if I fully understand that question, actually.

00:38:47 · Andreas Dzialocha

Maybe this is pointing toward the limits of IndexedDB and how much you can store in the browser.

00:38:53 · Sam Andreae

Yes.

00:38:54 · Andreas Dzialocha

Of course, this depends a little on the browser vendor. Every browser has different limitations, so it's a little hard to give an exact answer. I think in some browsers you can store a couple of gigabytes per scope.

00:39:13 · Andreas Dzialocha

Our approach so far, including with previous versions of p2panda, and our recommendation, has been not to push too much work or too much data into the browser, simply because the experience can get slow or laggy. There are ways to design web apps smartly, with service workers and moving everything off the main loop, and so on. But suddenly it's much more complicated than classic web development. Skilled developers can definitely do great things, but we've always tried to keep building an app as simple as possible.

00:39:57 · Andreas Dzialocha

Our usual recommendation was to rely on proxies. The proxy is the instance that keeps the data: a server that is online and reachable by the website. We now call these brokers—event-streaming brokers. The client can connect to a broker and consume data from it. If you think about the separation between event delivery and event processing, the browser would contain only a WebAssembly compilation of the event-processing layer. It wouldn't do synchronization, or even store the data. It would just process it and perhaps keep the materialized results, rather than the packets or operations themselves.

00:40:52 · Andreas Dzialocha

That could reduce the work the website or browser needs to do. I think it's one approach for very heavy applications. The browser client would also keep a state vector of what it has already seen, similar to a synchronization protocol. It can tell the broker its state, and the broker can stream the missing operations to the website. You can then have multiple websites in different states, each fetching the latest data from that broker. This is the pattern we're currently thinking about.

00:41:31 · Andreas Dzialocha

But I can also imagine someone one day building a WebSocket or WebRTC event-delivery solution inside the browser, actually allowing peer-to-peer sync and event delivery within it. I think it would be even trickier to make that a good experience for everyone visiting the website. Our friends at Hyper Hyper Space got quite far with this, but I think they also stopped doing it at some point because fighting the browser sandbox and making it a great experience on mobile devices was really hard. Technology has changed, though, and our smartphones are much faster. Who knows? Maybe it's worth doing it now.

Q&A: sharing a local P2P service across applications

00:42:32 · Sam Andreae

I see another question I can take. It says: “Have you thought about multiple local-first applications on the same device sharing a common local peer or networking substrate, rather than each app running its own discovery, gossip, storage and identity stack? Do you think this could become shared infrastructure for local-first applications, or would it introduce too much coupling between otherwise independent apps?”

00:43:00 · Sam Andreae

This idea is actually what I described in one of the slides: the system service we're working on with the GNOME community. The things you've listed are the layers that could be included in this substrate: discovery, gossip, storage and identity. We've discussed them a lot. In this project, we followed the GNOME community's preferences—or almost what they were ready to do—because there are many questions about what can be shared.

00:43:48 · Sam Andreae

Our initial approach is to share the very minimum, which in this case is synchronization using support nodes. There will be a shared service that all apps communicate with: the locally running support node. It can synchronize with other support nodes on your behalf, and each application can pull the data it is interested in from it. So there we will already have multiple apps sharing the same substrate, the same sync layer.

00:44:31 · Sam Andreae

A shared identity service is another thing we've discussed a lot. It's something we want and are very interested in, as is the GNOME community. It brings more concerns around privacy and how the user experience would work in the GTK environment. I think it's very interesting and very possible. Again, it depends very much on the context you're working in, so this is a nuanced area. But I think sharing these systems is very desirable.

00:45:28 · Host

Cool, thank you. Any other questions? Has anyone thought of another question?

Q&A: community applications and user experience

00:46:17 · Host

May I ask one more? What are some of the most interesting cases you've encountered when using p2panda in real-life community projects—for example, the use cases around Toolkitty?

00:46:58 · Sam Andreae

So you mean, some interesting…

00:47:01 · Host

Yes.

00:47:02 · Sam Andreae

Use cases?

00:47:04 · Host

Yes.

00:47:05 · Sam Andreae

Right. Toolkitty is a really interesting use case. It brings up user-experience challenges around the concurrent nature of P2P systems, because you're handling requests for a resource. The conflict is no longer in a text document—“I put an A here, and you deleted this character; resolve it.” It's something different, something more human: “I want to borrow your car. How do I know this has been accepted and nobody else got there first?” Very tangible problems come up, and communicating them to potential users is a challenge.

00:48:03 · Sam Andreae

People are so used to instant responses: I press “reserve,” get an immediate response, and it's confirmed. But actually, you need to wait for somebody to receive your message and for a response to come back. We learned a lot in that project about UX and UI design, and about how to implement it on top of p2panda. There were many good lessons. What about Andreas and Glyph—do you have any favorite challenging use cases?

00:48:57 · Andreas Dzialocha

For me, it's perhaps a more general issue, going back to the previous question about a daemon or a common substrate or service. There's an app-bootstrapping problem: every app needs to introduce a new idea of how to manage identities, how to manage your own identity, and whom you trust to exchange data with. You end up reinventing all of these things with every app, and every app does it slightly differently. If it changes every time, it's hard for users to adopt or get used to it. I think we do need to change it a lot right now, to learn different ways of doing it, because the best way hasn't been found yet—and perhaps the best way will never exist.

00:49:57 · Andreas Dzialocha

In any case, the biggest challenge for me is figuring out which common patterns we could unify and let apps rely on, so they don't have to do the same things over and over again. This is perhaps more of a general, underlying issue.

00:50:33 · Host

Thank you very much.

00:50:37 · Sam Andreae

Thank you for having us. Thanks, everyone, for coming and asking such good questions.

00:50:46 · Host

There is another question.

00:50:50 · Sam Andreae

Oh, yes.

00:50:50 · Host

Another question.

Q&A: AI tools and UI/UX

00:50:52 · Sam Andreae

“Have you ever tried using Claude and Codex together for UI/UX improvement?” No, we haven't, I can say.

00:51:04 · Andreas Dzialocha

It's also worth mentioning that we ourselves aren't UI or UX developers. We're very interested in it and talk to designers and UI/UX developers. We've also experimented with concepts ourselves. Keeping that connection is very important for us; otherwise, we'd be developing blindly. Usually, when we talk about UI or UX development, it's done by other teams, who report back what they've learned and discuss their findings with us. I know, for example, that Dash Chat uses Claude for its front-end work, but I actually don't know how that's going for them.

00:52:02 · Audience

Thank you.

00:52:13 · Host

Any other questions? There is a new message in the chat box. Oh, that seems to be a reply to another participant.

UX: familiar interactions and network health

00:52:46 · Andreas Dzialocha

In terms of UI, this may be very general advice, but I still think it's good: stick to things people already know whenever you can. A delete button should still just be a delete button, even though things can get strange in P2P systems. If you can't follow what people already know, make that difference extremely prominent. If something is special about your system, make it prominent in the UI. With Toolkitty, we tried to keep things intuitive: you request a resource, and while you're waiting, it has a little pending state—things you would feel familiar with.

00:53:41 · Andreas Dzialocha

But we also thought about network health: how much have I synchronized with peers, and how much data have I exchanged? We felt that was a novel concept, something a normal app doesn't have. A normal app might just show whether you're offline. This felt new, so we came up with the idea of a Tamagotchi—a little creature you need to keep happy. It stays happy when you sync often, aren't offline too often, exchange data, and are available to carry data to other people. That's an example of something we featured very prominently.

00:54:30 · Andreas Dzialocha

I'm sure there's more to explore around managing your identity and private key. There are many ideas there. Roughly, those are the two categories: stay as close as possible to what people already know, and then, where you can't…