Hello, and thank you for the introduction. :-)

Before I begin, a quick shout out to all the amazing FSF staff who have been working very hard to get our now-hybrid wonderful conference off the ground this weekend in the face of technical difficulties, still using only free software. Kudos to you all, and thank you very much!

With that, let's jump right in and talk about What's new in Jami.

But before that, first, what is **Jami**?

GNU package for universal communication

Key features of **Jami**

- 1-on-1 and group chats
- calling and conferencing
- audio/video messaging
- screen sharing
- file sharing

- plugins (+ SDK)
  - green screen
  - watermark
  - audio filters
  - ...and more!
- SIP phone

p2p and **distributed**

Jami offers these all the while being peer-to-peer and distributed.

Low-level architecture of **Jami** and how it works are beyond the scope of this talk, but if you'd like to learn more, you can watch my talk `Jami and how it empowers users` from LibrePlanet 2021. Feel free to reach out to me after the conference as well, with any questions.
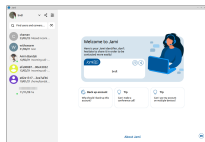
Previously, **Jami** had several clients (user interfaces) for desktop operating systems, namely the GTK-based `jami-gnome` on GNU/Linux, and separate native clients for macOS and Windows as well.

jami — main window

However, in 2020 and 2021, the team started transitioning to the Qt-based `jami-qt` client, initially developed for Windows, across desktop operating systems. We gradually deprecated the other desktop clients, namely the GNOME and macOS clients, as the team shifted focus to the Qt-based client. More recently, with these other clients having been deprecated, we renamed `jami-qt` to simply `jami`.
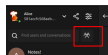
Swarm

1-on-1 + group chats

(currently groups of up to 8 members)

Swarm is arguably the most important change in **Jami** in the recent years that directly affects users and hopefully improves their experience using **Jami**. Swarm is **Jami**'s solution to the difficult problem of building a reliable, distributed chatting system.

In short, a swarm is a git-backed conversation between two or more people that is automatically synchronized across their devices. Before swarm, messaging in **Jami** was rather primitive, with no synchronization across devices: none for the sender, and for the receiver, a message would only be delivered to those of their devices that were online when it was sent. If no device was online, **Jami** would retry for a week, then stop. If by that time the message was not delivered to the receiver, the receiver would simply never know. Swarm addresses these limitations, and does so in a much more robust way compared to **Jami**'s previous generation of conversations, and thanks to git can handle some sophisticated use-cases of partial availability due to connection less, and seamless merging of histories.
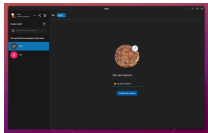
jami — creating a swarm

Let's have a look at a few swarm screenshots.

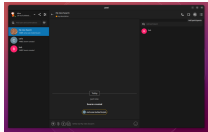Here, we can create a new swarm conversation by clicking the button showing a group of people.

jami — initializing a swarm

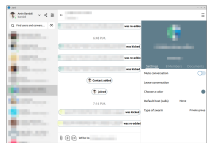We can add an initial set of members for our swarm, and also set an avatar and display name.

jami — adding and removing members for a swarm

We can also add or remove swarm members at any later point.

jami — swarm panel showing conversation settings

We have a new swarm panel showing various aspects of swarm conversations, namely settings, such as conversation colour and notification preferences, . . .

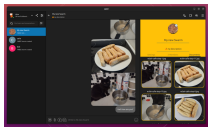jami — swarm panel showing conversation members

. . . conversation members, . . .

jami — swarm panel showing shared documents
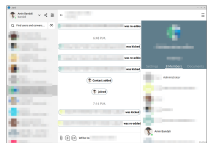
... and documents shared in the conversation. Several of these aspects are synchronized across devices.
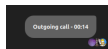
jami — swarm message read status

Swarm conversations also have improved displaying of message read status, and allow replying to specific messages, and reacting to messages using emojis.

Swarm also brings a new and improved file transfer subsystem, as well as an improved calling experience with the improved swarm-based calls subsystem.
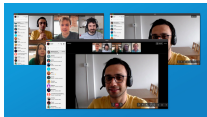
jami — more flexible conference video layouts

There have also been many other additions and improvements in **Jami** that aren't strictly swarm-related.

For instance, we had the 'multistream' and 'videosplit' projects for improving video calls and making the call layouts more flexible, . . .

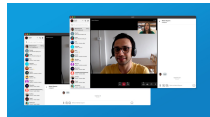jami — conference layout settings, e.g. hiding audio-only participants

. . . as well as adding new options and settings for controlling video layout in calls, for instance allowing the user to hide audio-only participants, per this screenshot.
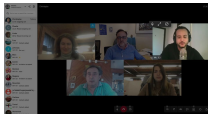
jami — horizontal and vertical in-call chatview split

We also added a setting to allow the user to choose whether they'd like the chatview to appear in a vertical or horizontal split in calls.

jami — conference moderation tools

Also, we added various moderation tools in conference calls, including muting someone (as a moderator), maximizing, kicking, and promoting others to moderator.
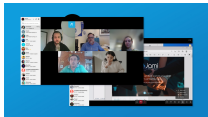
jami — 'raise hand' feature

As well as a raise hand feature that is useful both in an educational classroom setting, and other settings, including regular meetings and conferences.

jami — chatview typing indicator
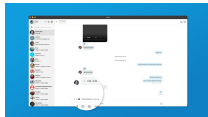
Another small quality of life improvement we see here is the typing indicator in the chatview.

jami — changing ringtone device
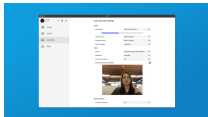
Another would be the ability for the user to choose a different ringtone device than their system default output, useful in cases where you might have headphones plugged into your computer, but want to have **Jami** ring on your speakers, so that you could hear calls when you're not at your desk.

jami — debug logs within the client (user interface)

Another convenience feature is being able to see and copy debug logs right from within the client user interface itself, as opposed to having to launch **Jami** from a terminal.
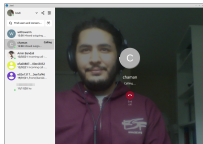
jami — call preview screen

Also, **Jami** now has a call preview screen where you can see your webcam feed before accepting an incoming call or as you wait for the recipient of an outgoing call to answer it.

UnifiedPush
push notifications, in freedom
(for incoming calls and messages on mobile)

jami.net/unifiedpush

Reliably delivering notifications to long-running background applications on mobile has been a long-standing challenge, and problematic especially for applications with a distributed architecture, like **Jami**. On Android, either the user has to disable battery optimizations for each application that wants to remain running in the background, or the application needs to display a permanent notification to prevent it from being killed by the battery manager, or the application needs to use a push notification mechanism.

Issue: no free and privacy-friendly push notification mechanism available, only Google's proprietary (dis)service.

Thankfully, a free replacement has been emerging: UnifiedPush, which is a set of specifications and tools that lets the user choose how push notifications are delivered.

UnifiedPush is now supported by the newest version of **Jami** on F-Droid, version 20230227-01 (361).

In addition to the existing main **Jami** plugins green screen, watermark, audio filter, and auto answer, . . .

Whisper **plugin**
for live captions
(transcription + translation)

... the team will very soon be announcing the addition of a new plugin for **Jami**, the Whisper plugin, for live captions.

Whisper plugin for Jami

Whisper is a general-purpose speech recognition model that can be used to provide automated live transcription or translation. We hope this new plugin will help make calls in **Jami** so much more accessible for folks who may be deaf or hard of hearing, and also be helpful in multi-lingual settings where not all attendees share the same language.

It's worth noting that this feature when offered by other conferencing tools typically either uses or requires nonfree software, or is SaaSS (Service as a Software Substitute), meaning that they send user data to remote servers and do the processing there. **Jami**' Whisper plugin, much like its Green Screen plugin, on the other hand is free software and does all the processing locally, on your own machine.

For future work, the team is looking into and researching the implementation of a new payment plugin to enable users to send and receive money right from **Jami**. Of course, handling payments is a complex topic, and much research and care is needed in how it's implemented, especially in a free software setting. I don't have a lot to share at this point, just that it's something on the team's radar.
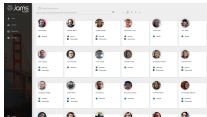
JAMS is a server application for enrolling and managing **Jami** clients in a group/enterprise context. It's a great tool that enables communities organizations to manage their own community of **Jami** users, while benefiting from **Jami**'s distributed network architecture.

Let's have a look at some core features of JAMS.

JAMS — an organization's list of users

JAMS can either run standalone and use its own embedded local database, or integrate with an existing LDAP or AD installation.

Here's how the list of users for an organization might look like in JAMS.

JAMS — a user's profile

For each user, we can see their name, username, profile picture, and account status (active or revoked), along with other useful information such as phone number, organization, and email address. These details can be automatically pulled from LDAP or AD, or entered locally in the case of using a local database embedded in JAMS.
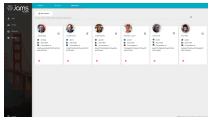
JAMS — a user's devices

JAMS also enables the administrator to manage users' devices, and revoke an individual device if needed (for instance, in the case of a lost or stolen device).

JAMS — a user's contacts

JAMS helps the administrator manage users' contacts, and for instance add a default set of users as someone's contacts. This is useful in an organizational setting for making sure users can find each other very easily by searching their names, without needing to know to search for each other's usernames or **Jami** account IDs.
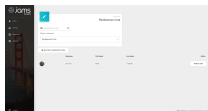
JAMS — creating a group
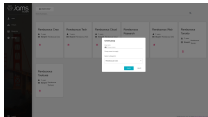
But managing a potentially large number of users individually can be
very tedious, and in very large deployments practically impossible.

So, JAMS allows creating groups of users, to enable administrators
to manage user accounts at scale.

JAMS — editing a group

We can edit groups, . . .

JAMS — adding users to a group

... and add users to them.

JAMS — creating a blueprint
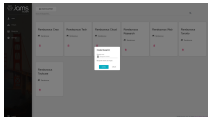
To make groups useful and flexible, JAMS introduces the concept of Blueprints for groups.

A blueprint is essentially a default configuration file, but for a group. When a user logs in with a new device, their device will fetch and use the configuration defined by the blueprint for the group they're a member of.

JAMS — editing a blueprint's permissions

The administrator can edit the permissions associated with a blueprint, for instance whether video calls or calls from unknown contacts are allowed.

JAMS — editing a blueprint's configuration

The administrator can also configure other aspects of a blueprint similarly to how one would with a regular **Jami** account, for instance whether to use UPnP, TURN, and a DHT proxy.

Okay, so much awesome stuff! How can I download **Jami** and start using these awesome new features? You can download the newest version of **Jami** from `jami.net/download`.

You can also install **Jami** from your GNU/Linux distribution's repositories, though with the caveat that it may be outdated.

Debian was one such distribution. However, I'm very happy to be able to say that I was able to take over the maintenance of the package a few months back, and with hard work and help from one of the Debian Developers (shout out Petter Reinholdtsen!) was able to bring it up-to-date just in time for **Debian**'s upcoming Bookworm release.

Additionally, users of the many derivatives of **Debian**, such as Devuan, **Ubuntu**, and the wonderful **Trisquel** will also be able to enjoy up-to-date versions of **Jami** going forward.

What's new in Jami
└─What's new in Jami?
  └─Housekeeping

Housekeeping

A couple of housekeeping changes. . .

What's new in Jami
└─ What's new in Jami?
    └─ Housekeeping

Release tarball naming scheme

```
jami_YYYYMMDD.hhmm.COMMITID.tar.gz → jami-YYYYMMDD.X.tar.gz
jami_20220206.1142.1234abcd.tar.gz → jami-20220206.0.tar.gz
```

dl.jami.net/release/tarballs/

We recently changed the naming scheme we use for the **Jami** release tarballs, from the previous `jami_YYYYMMDD.hhmm.COMMIT.tar.gz` to `jami-YYYYMMDD.X.tar.gz`, where `YYYYMMDD` is the date of the last commit on the current branch, and `X` the number of releases tagged that day, starting from zero. The new format is shorter, omits the unnecessary and confusing commit id, and uses dash instead of underscore. Also, it contains everything in a top-level directory with the same name as the tarball (without the .tar.gz extensions), so that `jami-YYYYMMDD.X.tar.gz` nicely extracts to `jami-YYYYMMDD.X`. With these changes, **Jami** release tarballs will follow more closely the related instructions from the GNU maintainers manual.

What's new in Jami
└─ What's new in Jami?
   └─ Housekeeping

`jami-client-qt.git`
now the primary source repository;
`jami-project.git` deprecated

We also deprecated the use of the `jami-project.git` repository for continuous integration and packaging, in favour of consolidating all of these aspects in `jami-client-qt.git` itself, organized mostly under the `extras` directory. The `daemon` submodule at the repository top-level points to a recent version of the daemon known to build and work with the Qt client. Also, the `build.py` script at the repository top-level can be used to conveniently install the dependencies and build the Jami daemon and client from source.

What's new in Jami
└─What's new in Jami?
　└─Housekeeping

Lastly, we also renamed the main public C++ namespace of the **Jami** daemon/library from DRing to libjami.

Alright, so we've heard seen a couple of new exciting things in **Jami**, and already started using it. But how does one get involved and participate in the community, to ask for help and/or help out others?

We would love to have you along with us in the community! You can join and participate in the community in a number of ways, and here are a few.

On the **Jami** forum you can discuss **Jami**, get help, and help others in the community.

You can also do that via **Jami**'s public mailing list.

Jami IRC channel
#jami on Libera.Chat
(also accessible via Matrix bridge)

We have an IRC channel, #jami, on the Libera.Chat network, that is bridged to Matrix and also accessible there. It's great for quick questions or chatting with fellow users and developers.

Last but most certainly not least, the **Jami** blog, where the team shares blog posts and articles with the latest news and what's new in **Jami**.

We saw how to get involved in the community. But what if you've been enjoying **Jami** and the wonderful free software community around it so much, that you'd like to directly help move **Jami** forward?

Jami has an ambitious goal of becoming the defacto communication application on GNU/Linux, and working towards this goal, the team and the community could use and would appreciate all the help they can get :-)

So, let's look at some of the ways for contributing to **Jami**.

You can contribute to **Jami** by sending patches via our Gerrit code review instance at `review.jami.net`.

Bug reports
git.jami.net
docs.jami.net/user/bug-report-guide.html

You can help by reporting bugs and issues via the issue trackers on git.jami.net. We always greatly appreciate these, and they are one of the most important ways the team receives feedback from the community about new features and changes.

If you do decide to report bugs, please see our bug report guide for details and step-by-step instructions on how to write clear and concise bug reports, and where and how to submit them.

You can also help by extending and improving the documentation for **Jami**.

Another very important and valuable way of helping is to help by translating **Jami** to your language(s).

Plugins
jami.net/plugins
jami.net/plugins-sdk

You can write your own plugins and share them with the community under a free/libre license.

If you're interested in getting started hacking on plugins and making your own, we have a tutorial that walks you through setting things up and writing a simple plugin that draws a circle at the center of the frame in video calls, and go from there.

Packaging
docs.jami.net/build/

If you can, please help package Jami for your GNU/Linux distribution of choice. If you can't, your distro's developers and maintainers might do it if you ask nicely. :-)

The team gratefully accepts and appreciate any and all donations from the community. If you'd simply like to make a monetary donation towards the development **Jami**, you can do so by clicking the **Donate** button on the `jami.net` website.

Fund a feature
contact@jami.net
docs.jami.net/developer/feature-requests.html

Another great way to support the development of **Jami** is to fund the development of specific features you'd like to see in **Jami**.

You can see a list of potential features for funding at this address, as well as proposing your own. In either case, you can get in touch with the team at `contact@jami.net` to discuss the details further.

Another way for supporting the development of **Jami**, particularly if you're in a company or organization setting, is by purchasing one of the JAMS support packages offered by the team. There are several tiers at different price levels, depending on the type of support and services you're looking to get.

Last, but most certainly not least, you can help **Jami** by spreading the word about it, and introducing it to your friends and family and coworkers.

What's new in Jami
└─Thanks!

bandali@gnu.org
kelar.org/~bandali

Thank you!