Jami and how it empowers users

LibrePlanet 2021

Amin Bandali

Jami and how it empowers users

LibrePlanet 2021

Amin Bandali

hello, and thank you for the introduction!
so, let's talk about Jami and how it empowers users

What is Jami?

Jami is free software for universal communication
that respects the freedom and privacy of its users

Jami is free software for universal communication
that respects the freedom and privacy of its users

GNU package for universal communication

Jami is developed by **Savoir-faire Linux®** based in Montréal, Canada. Savoir-faire Linux is a team of free software consultants providing training, consulting, development, and support services for free software technologies.

Savoir-faire
LINUX

Jami is developed by **Savoir-faire Linux®** based in Montréal, Canada. Savoir-faire Linux is a team of free software consultants providing training, consulting, development, and support services for free software technologies.

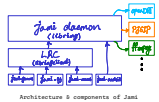
• one-on-one conversations
• file sharing
• audio/video calls & conferences
• screen sharing in video calls & conferences
• recording & sending audio/video messages
• SIP phone software functionality
• cross-platform communication framework

- one-on-one conversations
- file sharing
- audio/video calls & conferences
- screen sharing in video calls & conferences
- recording & sending audio/video messages
- SIP phone software functionality
- cross-platform communication framework

some of the high-level features of Jami include …

Architecture overview

Architecture overview



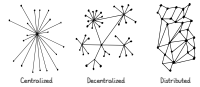
Architecture & components of Jami

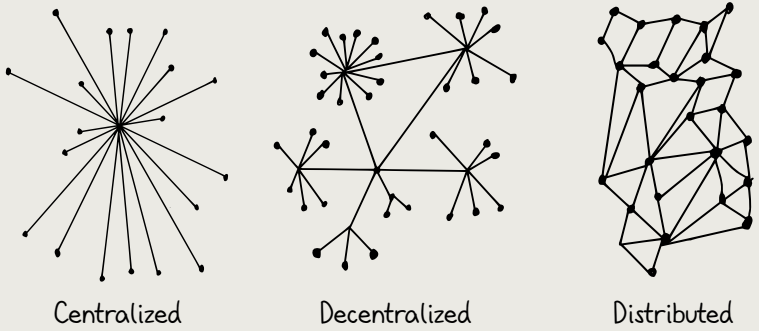
Jami's architecture can be divided into three layers:

1. daemon: core of Jami. is not user-facing, and contains all the connectivity, communication, crypto, media, etc logic. interacts with libs like opendht, pjsip, ffmpeg, and so on. has several APIs, namely: dbus, libwrap, JNI, REST
2. lrc: depends on QtCore and interfaces with the daemon, providing shared code for clients across several platforms (except for android)
3. clients: or frontends, which are user-facing applications that Jami users interact with

Jami is
an **end-to-end encrypted** secure
and **distributed** voice, video,
and chat **communication platform**
that requires **no central server**
and leaves the power of **privacy**
and **freedom** in the hands of *users*

Jami is
an **end-to-end encrypted** secure
and **distributed** voice, video,
and chat **communication platform**
that requires **no central server**
and leaves the power of **privacy**
and **freedom** in the hands of *users*





typology of networks, dating back to the early 1960s. from a research memorandum by paul baran, one of the two independent inventors of packet switching

OpenDHT

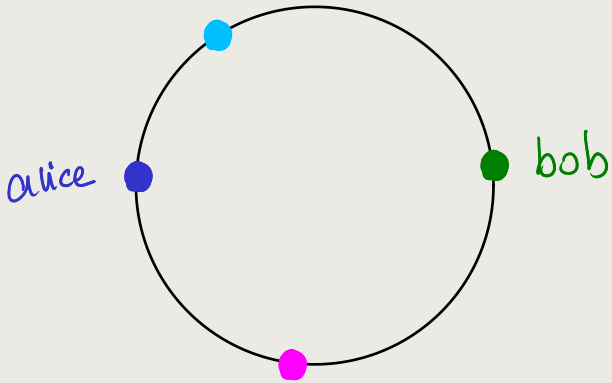
- lightweight and scalable, designed for large networks and small devices
- high resilience to network disruption
- IPv4 and IPv6 support
- clean and powerful C++14 *map* API
- C, Rust, and Python 3 bindings
- public key crypto layer providing optional data signing and encryption (using GnuTLS)
- REST API with optional HTTP client+server and push notification support

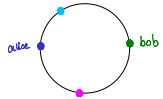
establishing p2p connections today in the face of NATs and firewalls can be quite tricky. so how does Jami achieve distributed communication? OpenDHT is at the heart of Jami's p2p communication

C++14 distributed hash table implementation

API similar to mainline BitTorrent DHT

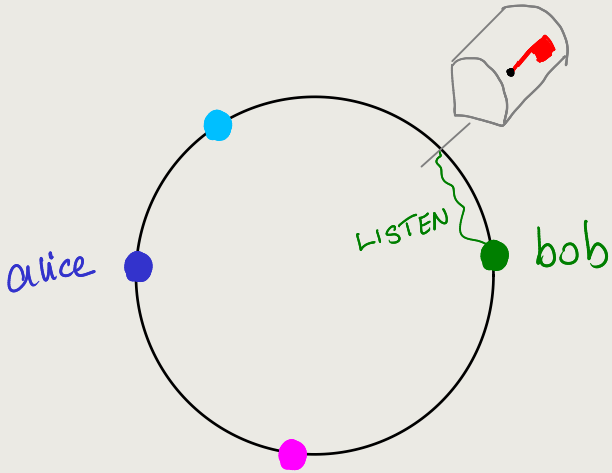
provides easy-to-use distributed in-memory key-value data store. every node in the network can read and write values to the store. values are distributed over the network, with redundancy.

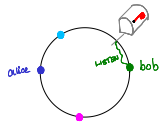




say we have a network of users, including alice and bob who would like to establish a connection

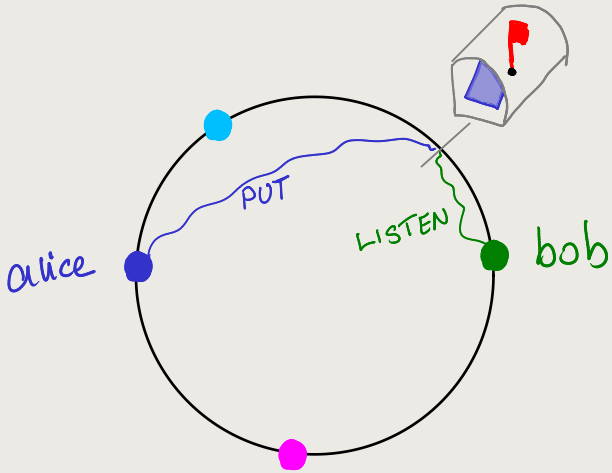
the circle is our network and the points are nodes

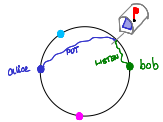




bob performs a `listen` operation on the network (typically served by closest node on the network), with the 160-bit key being the hash of his public key

the mailbox represents the 'key' in the dictionary

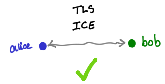


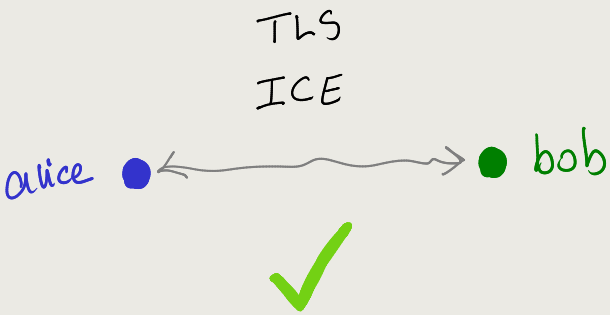


alice then performs a put operation at the same key, which bob will be immediately notified of and receive the value that was put on the dht

the value put on the dht (the blue envelope in the mailbox in our diagram) is an encrypted list of ICE candidates (IP addresses used to try and establish p2p communication)

this is used to establish p2p communication using ICE (interactive connectivity establishment)

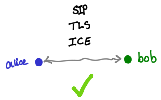


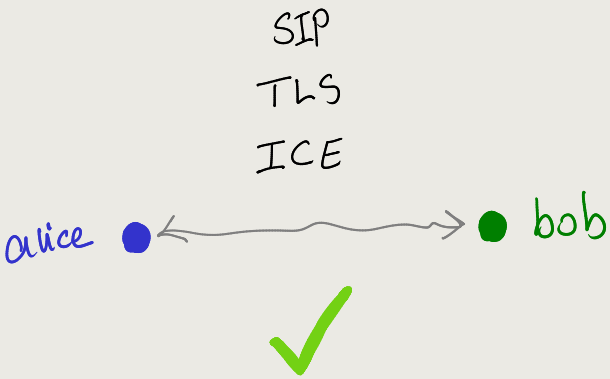


over the ICE connection we establish a TLS connection

the connection is authenticated both ways:
in contrast to web browsers where only clients authenticate the web servers and the servers typically don't authenticate clients using certs (but usually use other web technologies like SSO, cookie-based auth, etc), in Jami both parties exchange cert chains and check each other's certs and keys

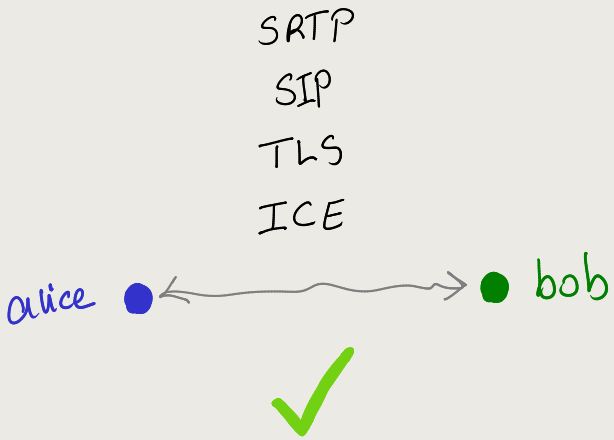
we have a p2p authenticated e2e-encrypted connection, including if through a relay (via TURN) …





over which we do SIP (using pjsip)

we use SIP to negotiate the call and its parameters
(like which audio and/or video codecs to use, etc)



which we use to perform SRTP communication (Secure Real-time Transport Protocol) which carries the media streams

notice the key aspect of Jami establishing this p2p ICE communication without use of a central server and instead over the opendht distributed network

Development news

some important recent or upcoming changes/features

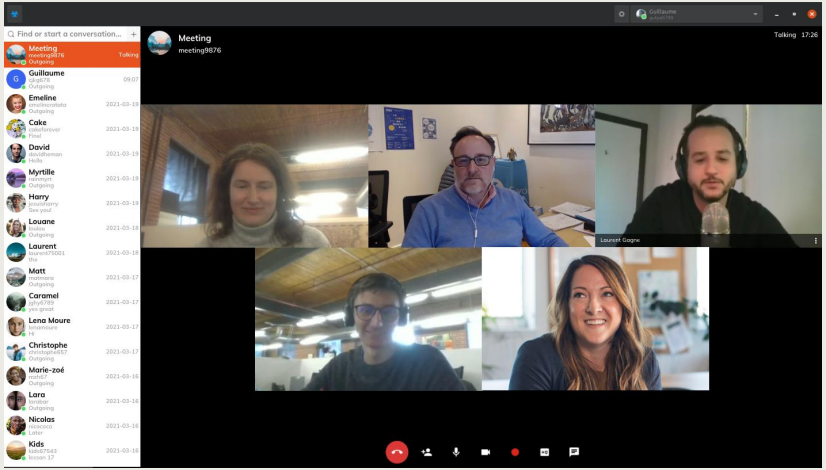
- many connectivity improvements
- automatic video bitrate adjustment
- detached host rendezvous points
- notification improvements
- bidi text in the chat view
- video renderer improvements
- conference call moderation
- improved error handling
- object copy reductions
- plugin API overhaul
- … and more!

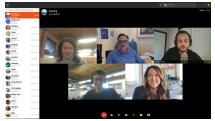
many fixes & improvements throughout Jami's codebases

major UPnP refactoring adding handler for automatic port mapping provisioning to keep a pool of mappings ready for use for incoming or outgoing calls

Rendezvous points

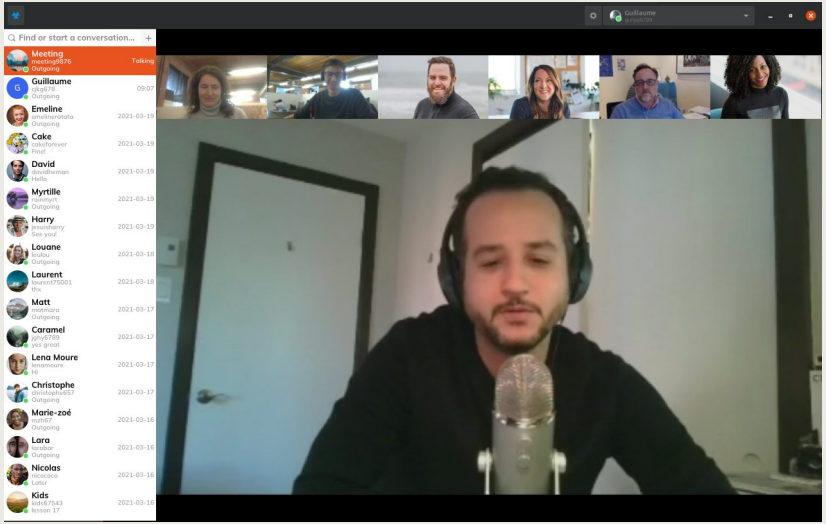
Rendezvous points

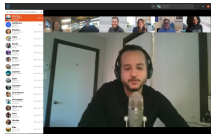




Jami has long supported group calls with multiple participants. rendezvous point accounts introduced last year take that to the next level, essentially allowing you to turn your machine running Jami into a conference server.

rendezvous points automatically answer incoming calls, and by default are in a 'detached' or 'headless' state and don't transmit the host's audio/video





you can create new rendezvous point accounts or easily turn any existing account into a rendezvous point from the account settings page

rendezvous points help empower people who don't have the means or interest in setting up and running self-hosted instances of other free/libre conferencing tools like Jitsi and BigBlueButton

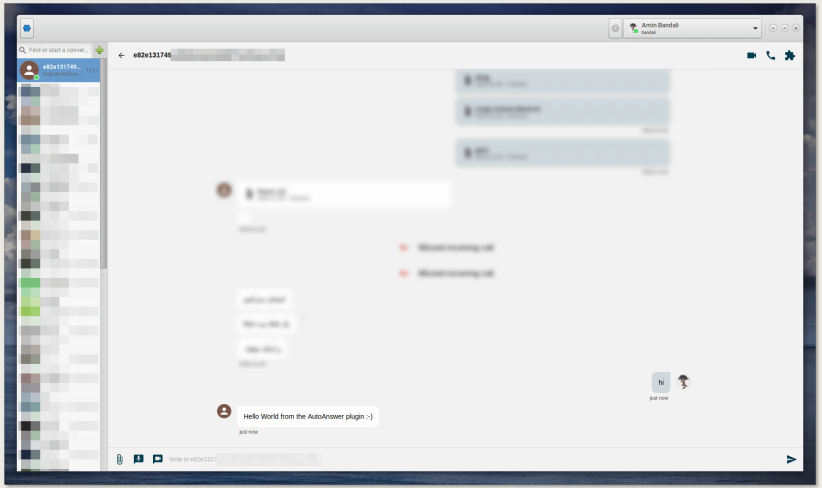
Plugins

are a new addition to the Jami universe

written in C++ using the ChatHandler and MediaHandler APIs of Jami daemon

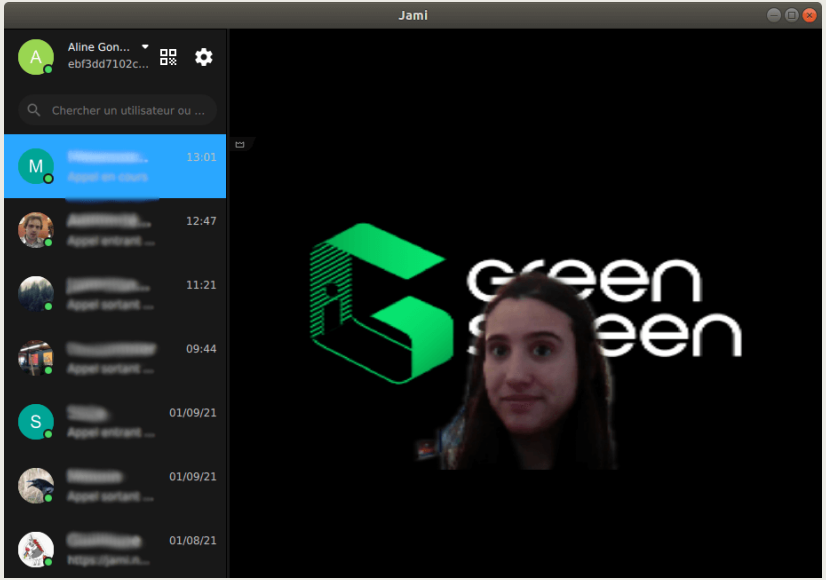
Python SDK for easily creating new plugins from skeleton projects

AudioFilter: audio-related filters for audio/video calls. for instance, a reverb effect

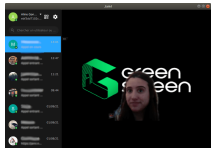


AutoAnswer

AutoAnswer: chat bot-like plugin for automated replies

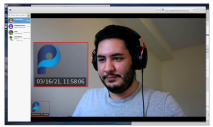


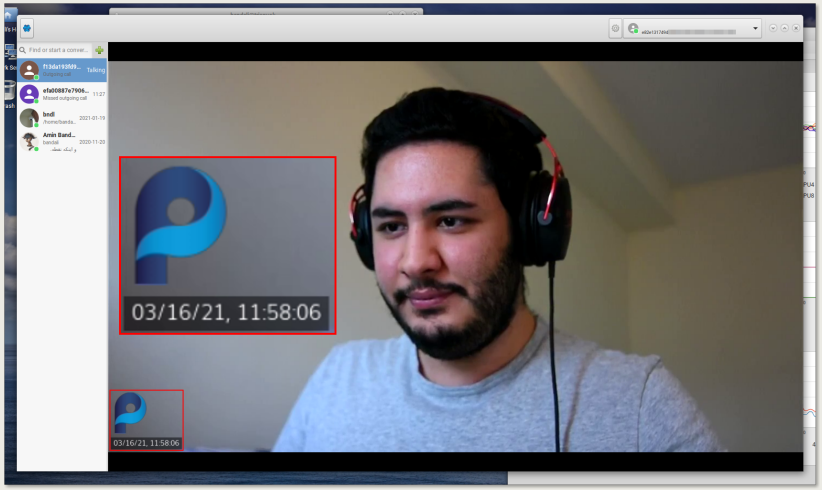
GreenScreen



GreenScreen: a neural network-based plugin for adding a green screen effect of putting you in the foreground of an image

note that the green screen feature of many other conferencing tools are either nonfree, or worse are SaaSS (Service as a Software Substitute), meaning that they send the data to remote servers and do the processing there. the Jami GreenScreen plugin on the other hand is free software and does all the processing locally, on your own machine





WaterMark

WaterMark: enables you to add a watermark in video calls. can be an image and can additionally contain location, date, and time as well

discover more Jami plugins
jami.net/plugins

learn more about the plugins SDK
jami.net/plugins-sdk

discover more Jami plugins
jami.net/plugins
learn more about the plugins SDK
jami.net/plugins-sdk

discover more Jami plugins
jami.net/plugins

learn more about the plugins SDK
jami.net/plugins-sdk

discover more Jami plugins
jami.net/plugins

learn more about the plugins SDK
jami.net/plugins-sdk

if you're interested in getting started hacking on plugins and creating 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

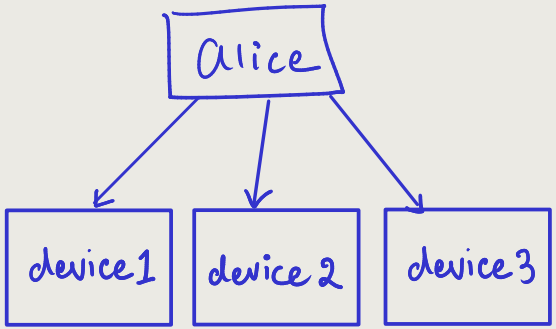
the plugins API can be a really useful method for extending Jami's behaviour with regards to text and audio/video call processing and distributing your changes to others, without necessarily concerning yourself with other parts of the jami-daemon codebase

JAMS

Jami Account Management Server enables organizations to manage their own community of Jami users while taking advantage of Jami's distributed network architecture

JAMS can either run standalone and use its own local database, or integrate with an existing LDAP or AD installation, and allows managing contact lists and pushing specific configurations to groups of users

SFL offers paid commercial support via the Jami Store



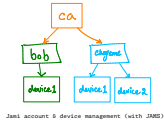
Jami account & device management

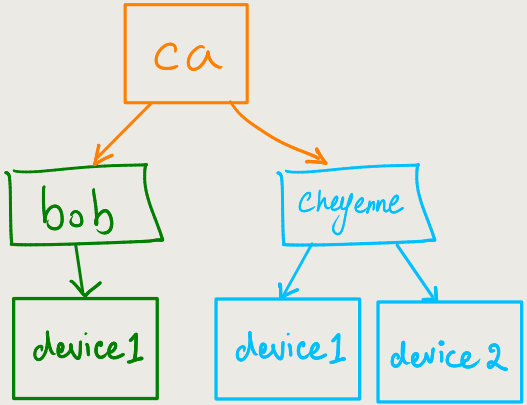
Jami account & device management

how does Jami authenticate users and manage accounts in a non-centralized way and without any central authority? using X.509 certs & cert chains

it's a well-known and widely used standard (e.g. by web browsers when visiting web sites over TLS, and various large organizations use it internally for user authentication and account management)

Jami extends the use of cert chains to device management, covering the multiple-devices-per-account use-case, enabling easy addition or revocation of devices. contacts are managed similarly to openssh





Jami account & device management (with JAMS)

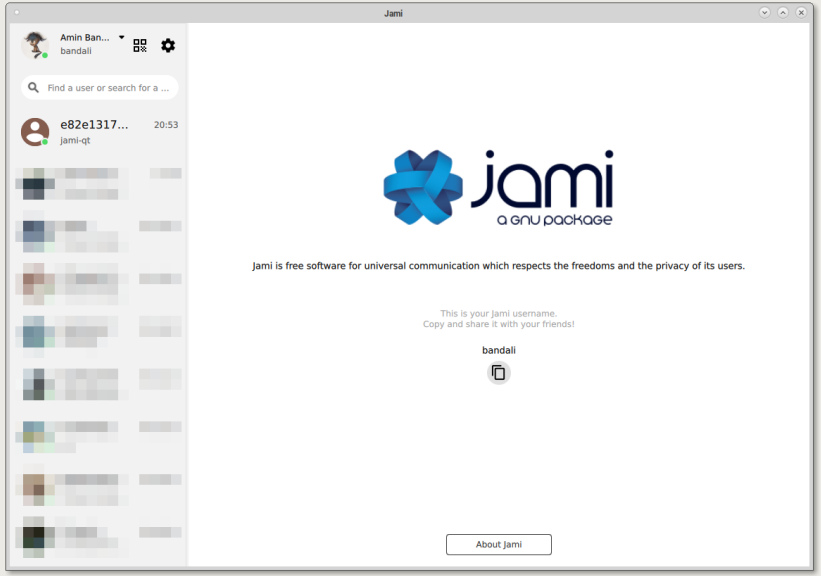
JAMS brings in a CA into the picture, enabling organizations to manage and revoke users and so on. the JAMS server will fulfill CSRs for users via the org CA (when using Jami outside this kind of setting, account certs are self-signed)

using X.509 certs in this way enables mixed distributed & centralized authentication: in an org setting, allows authenticating users and members of the same org, or even another org you're working with, if configured

thus JAMS helps empower Jami users to form and manage their own smaller communities in a hybird way

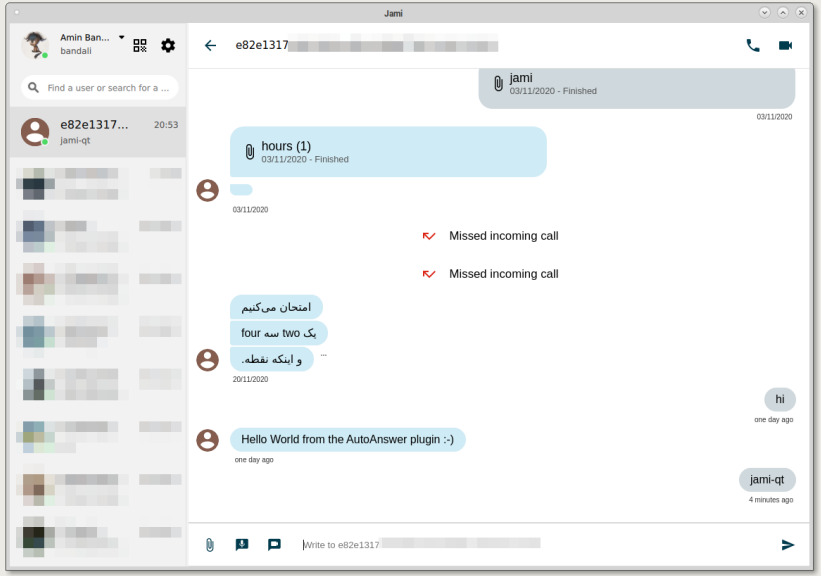
jami-qt **client**

Jami and how it empowers users
└Development news

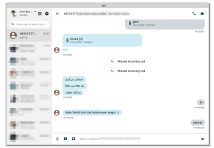


jami-qt – main window

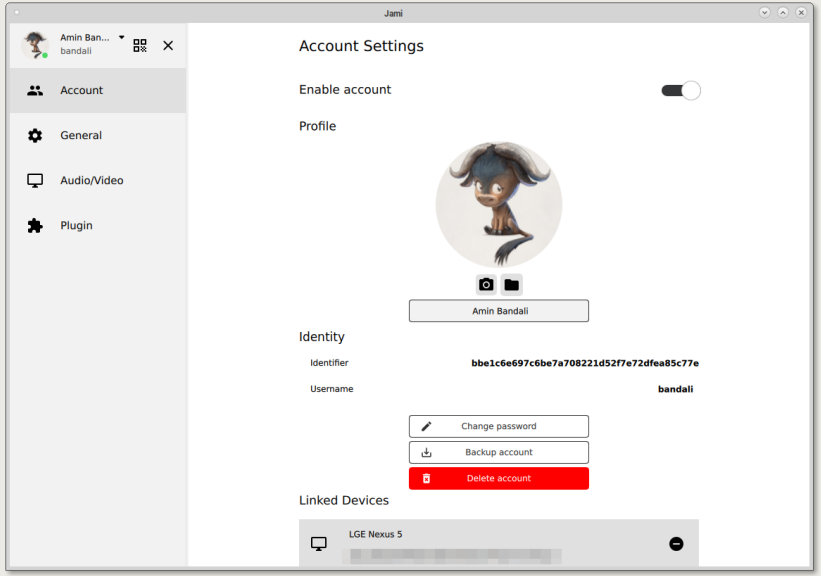
jami-qt – main window



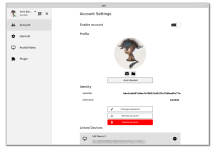
jami-qt – chat view



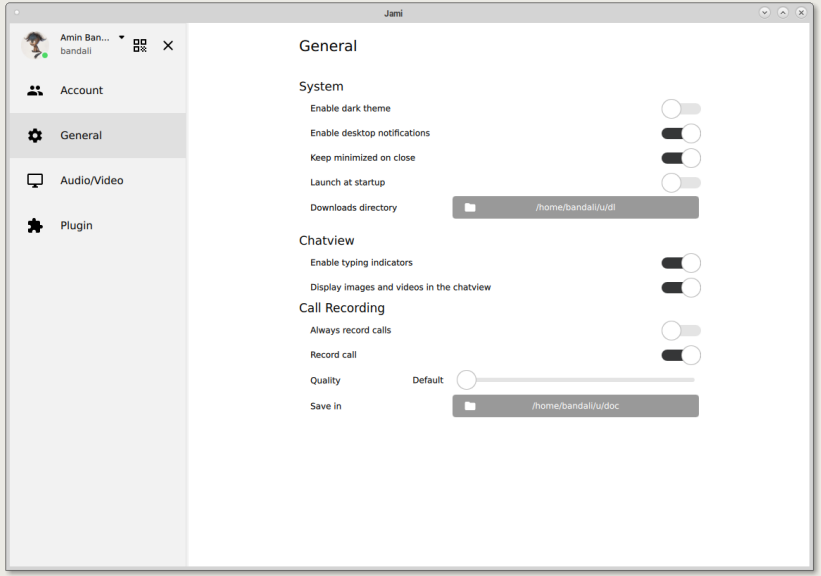
jami-qt – chat view



jami-qt – settings view (account)



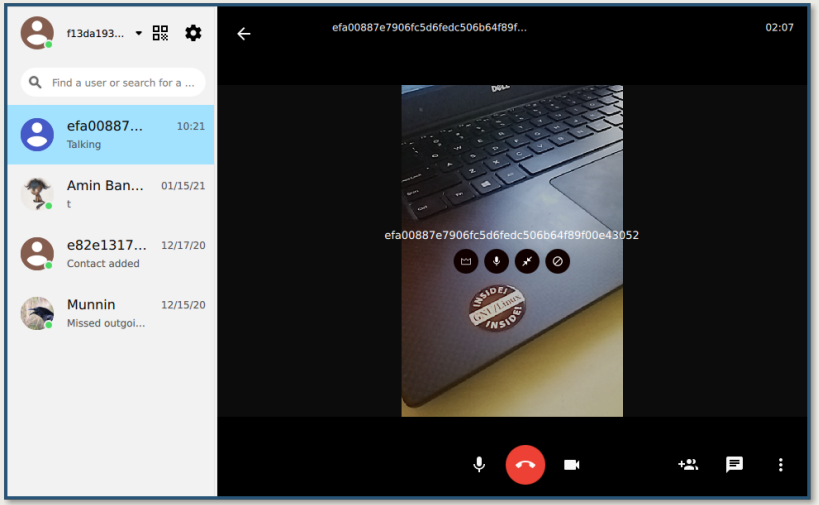
jami-qt – settings view (account)



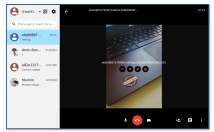
jami-qt — settings view (general)

jami-qt — settings view (general)

Moderation



jami-qt – moderation



jami-qt – moderation

Swarm chat

currently Jami supports linking multiple devices to a single account; however, as most Jami users already know and would point it out, message histories are not synchronized. this has been an important missing feature and probably the most frequently requested one so far, along with group chats.

there's been a reason; implementing these features in a fully distributed setting like Jami is a real challenge: it needs to be done without any central server or authority.

Jami team proud to introduce Swarm chats

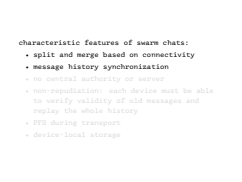
characteristic features of swarm chats:

- split and merge based on connectivity
- message history synchronization
- no central authority or server
- non-repudiation: each device must be able to verify validity of old messages and replay the whole history
- PFS during transport
- device-local storage

swarms are fully distributed p2p text conversations, with a potentially unlimited number of participants, and are characterized by the following features …

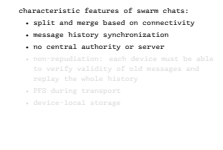
characteristic features of swarm chats:

- split and merge based on connectivity
- message history synchronization
- no central authority or server
- non-repudiation: each device must be able to verify validity of old messages and replay the whole history
- PFS during transport
- device-local storage



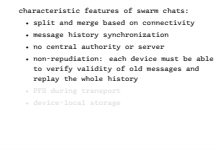
characteristic features of swarm chats:

- split and merge based on connectivity
- message history synchronization
- no central authority or server
- non-repudiation: each device must be able to verify validity of old messages and replay the whole history
- PFS during transport
- device-local storage



characteristic features of swarm chats:

- split and merge based on connectivity
- message history synchronization
- no central authority or server
- non-repudiation: each device must be able to verify validity of old messages and replay the whole history
- PFS during transport
- device-local storage



characteristic features of swarm chats:

- split and merge based on connectivity
- message history synchronization
- no central authority or server
- non-repudiation: each device must be able to verify validity of old messages and replay the whole history
- PFS during transport
- device-local storage

PFS (perfect forward secrecy): assurance that session keys will not be compromised, even if longer-term secrets used in the key exchange for session initiation are compromised. in other words, that potential future compromise of these keys does not affect secrecy of past sessions

characteristic features of swarm chats:

- split and merge based on connectivity
- message history synchronization
- no central authority or server
- non-repudiation: each device must be able to verify validity of old messages and replay the whole history
- PFS during transport
- device-local storage

characteristic features of swarm chats:
- split and merge based on connectivity
- message history synchronization
- no central authority or server
- non-repudiation: each device must be able to verify validity of old messages and replay the whole history
- PFS during transport
- device-local storage

four modes for swarm chats:

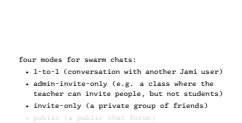
- 1-to-1 (conversation with another Jami user)
- admin-invite-only (e.g. a class where the teacher can invite people, but not students)
- invite-only (a private group of friends)
- public (a public chat forum)

four modes for swarm chats:

- 1-to-1 (conversation with another Jami user)
- admin-invite-only (e.g. a class where the teacher can invite people, but not students)
- invite-only (a private group of friends)
- public (a public chat forum)

four modes for swarm chats:

- 1-to-1 (conversation with another Jami user)
- admin-invite-only (e.g. a class where the teacher can invite people, but not students)
- invite-only (a private group of friends)
- public (a public chat forum)



four modes for swarm chats:

- 1-to-1 (conversation with another Jami user)
- admin-invite-only (e.g. a class where the teacher can invite people, but not students)
- invite-only (a private group of friends)
- public (a public chat forum)

four modes for swarm chats:
- 1-to-1 (conversation with another Jami user)
- admin-invite-only (e.g. a class where the teacher can invite people, but not students)
- invite-only (a private group of friends)
- public (a public chat forum)

maintain a synchronized Merkle tree of messages

each conversation will be a git repository

maintain a synchronized Merkle tree of messages

each conversation will be a git repository

a merkle tree or a hash tree, invented by ralph merkle in 1979, is a tree where each leaf node is labelled with the crypto hash of a piece of data, and each parent node is labelled with the crypto hash of the labels of its children.

have the nice property that verifying a given leaf node is part of the tree has logarithmic complexity (hashing computations proportional to logarithm of number of leaf nodes; so, very fast)

have applications in wide areas of computing such as file systems (like btrfs and zfs), databases, and distributed p2p systems like git and mercurial

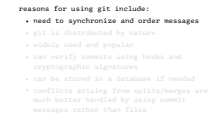
maintain a synchronized Merkle tree of messages

each conversation will be a git repository

maintain a synchronized Merkle tree of messages

each conversation will be a git repository

we decided to go with git, and have each conversation be a git repository



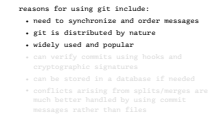
reasons for using git include:

- need to synchronize and order messages
- git is distributed by nature
- widely used and popular
- can verify commits using hooks and cryptographic signatures
- can be stored in a database if needed
- conflicts arising from splits/merges are much better handled by using commit messages rather than files

the Merkle Tree data structure helps us do just that, and can be linearized by merging branches. are widely used by git, and thus easy to synchronize between multiple devices

reasons for using git include:

- need to synchronize and order messages
- git is distributed by nature
- widely used and popular
- can verify commits using hooks and cryptographic signatures
- can be stored in a database if needed
- conflicts arising from splits/merges are much better handled by using commit messages rather than files



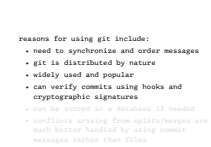
reasons for using git include:

- need to synchronize and order messages
- git is distributed by nature
- widely used and popular
- can verify commits using hooks and cryptographic signatures
- can be stored in a database if needed
- conflicts arising from splits/merges are much better handled by using commit messages rather than files

much potential for further development and extension in the future

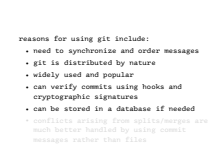
reasons for using git include:

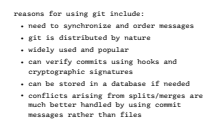
- need to synchronize and order messages
- git is distributed by nature
- widely used and popular
- can verify commits using hooks and cryptographic signatures
- can be stored in a database if needed
- conflicts arising from splits/merges are much better handled by using commit messages rather than files



reasons for using git include:

- need to synchronize and order messages
- git is distributed by nature
- widely used and popular
- can verify commits using hooks and cryptographic signatures
- can be stored in a database if needed
- conflicts arising from splits/merges are much better handled by using commit messages rather than files


reasons for using git include:
• need to synchronize and order messages
• git is distributed by nature
• widely used and popular
• can verify commits using hooks and cryptographic signatures
• can be stored in a database if needed
• conflicts arising from splits/merges are much better handled by using commit messages rather than files



reasons for using git include:

- need to synchronize and order messages
- git is distributed by nature
- widely used and popular
- can verify commits using hooks and cryptographic signatures
- can be stored in a database if needed
- conflicts arising from splits/merges are much better handled by using commit messages rather than files

all git operations, control messages, files, etc will be done over p2p TLS-1.3 connections, using only ciphers guarantying PFS being used. so keys are renegotiated for each new session

demo time!

demo time!

we plan on rolling swarms out in three phases:

1. for new one-on-one conversations, mainly enabling history synchronization across devices
2. group conversations with a limited maximum number of users (our current goal is 8 users in a conversation)
3. increase or lift that limit, and implement public groups

download jami-qt

jami.net/download

download jami-qt
jami.net/download

Community

very happy to say that i think we have a healthy and growing community around Jami, and we'd love to have *you* with us as well!

folks ask "how can i get involved in the community?" here are some ways

- **jami.net**
- forum.jami.net
- jami@gnu.org
- #jami on freenode
- git.jami.net
- review.jami.net
- docs.jami.net
- write and share your own plugins
- help translate Jami to your language(s)
- package Jami in your GNU/Linux distribution
- *use Jami and help spread the word <3*

Jami and how it empowers users
└─Development news

the main Jami website, with blog posts and articles from the Jami team with latest news about Jami

- jami.net
- forum.jami.net
- jami@gnu.org
- #jami on freenode
- git.jami.net
- review.jami.net
- docs.jami.net
- write and share your own plugins
- help translate Jami to your language(s)
- package Jami in your GNU/Linux distribution
- *use Jami and help spread the word <3*

Jami forum for discussing Jami, getting help, and helping others
can also use the mailing list for that
irc for quick questions or chatting with users/devs

- jami.net
- forum.jami.net
- jami@gnu.org
- #jami on freenode
- git.jami.net
- review.jami.net
- docs.jami.net
- write and share your own plugins
- help translate Jami to your language(s)
- package Jami in your GNU/Linux distribution
- *use Jami and help spread the word <3*

report bugs on git.jami.net

- jami.net
- forum.jami.net
- jami@gnu.org
- #jami on freenode
- git.jami.net
- review.jami.net
- docs.jami.net
- write and share your own plugins
- help translate Jami to your language(s)
- package Jami in your GNU/Linux distribution
- *use Jami and help spread the word <3*

send patches on review.jami.net; i'd love to receive patches to jami-gnome and help from the community to maintain it

- jami.net
- forum.jami.net
- jami@gnu.org
- #jami on freenode
- git.jami.net
- review.jami.net
- docs.jami.net
- write and share your own plugins
- help translate Jami to your language(s)
- package Jami in your GNU/Linux distribution
- *use Jami and help spread the word <3*

help write and improve documentation for jami

- jami.net
- forum.jami.net
- jami@gnu.org
- #jami on freenode
- git.jami.net
- review.jami.net
- docs.jami.net
- write and share your own plugins
- help translate Jami to your language(s)
- package Jami in your GNU/Linux distribution
- *use Jami and help spread the word <3*

- jami.net
- forum.jami.net
- jami@gnu.org
- #jami on freenode
- git.jami.net
- review.jami.net
- docs.jami.net
- write and share your own plugins
- help translate Jami to your language(s)
- package Jami in your GNU/Linux distribution
- *use Jami and help spread the word <3*

- jami.net
- forum.jami.net
- jami@gnu.org
- #jami on freenode
- git.jami.net
- review.jami.net
- docs.jami.net
- write and share your own plugins
- help translate Jami to your language(s)
- package Jami in your GNU/Linux distribution
- *use Jami and help spread the word <3*

- jami.net
- forum.jami.net
- jami@gnu.org
- #jami on freenode
- git.jami.net
- review.jami.net
- docs.jami.net
- write and share your own plugins
- help translate Jami to your language(s)
- package Jami in your GNU/Linux distribution
- *use Jami and help spread the word <3*

Thanks!

Amin Bandali <bandali@gnu.org>

kelar.org/~bandali/talks/jami-empowers-users.html

Thanks!

Amin Bandali <bandali@gnu.org>

kelar.org/~bandali/talks/jami-empowers-users.html

(GPLv3+)