



Hello

- Reticulum is an *(inter)-network protocol*
- Attempts to solve the problem of reliably getting data from A -> B
- Uses cryptographic principles for addressing and data transport
- Currently has a fully functional, well-tested and stable reference implementation:

Hello

- Reticulum is an *(inter)-network protocol*
- Attempts to solve the problem of reliably getting data from A -> B
- Uses cryptographic principles for addressing and data transport
- Currently has a fully functional, well-tested and stable reference implementation:

Reticulum Network Stack (*MIT License*)

https://reticulum.network



Nobody actually talks to each other :(

- Even basic compute devices have:
  - Lots of processing power
  - Large amounts of storage and memory
  - Advanced radio and communications hardware



Nobody actually talks to each other :(

- Even basic compute devices have:
  - Lots of processing power
  - Large amounts of storage and memory
  - Advanced radio and communications hardware



Nobody actually talks to each other :(

- Even basic compute devices have:
  - Lots of processing power
  - Large amounts of storage and memory
  - Advanced radio and communications hardware
- Yet, if I want to send a message to a friend in the same room, this happens:



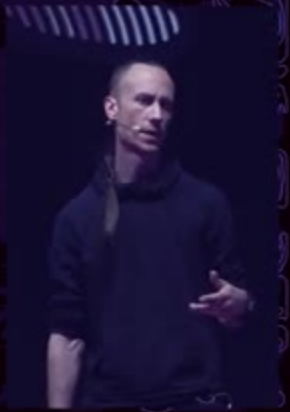
- This situation is so bizarre, that it's strange we've grown completely used to it.

Some applications *try* to alleviate this, but really:

- This should **not be the job of the application developer to solve**
- The **networking protocol** should establish the most efficient data path at runtime.

If you need someone else's server to use your applications...

- They **own your action potential**
- And they can take it away at will
- This is a strange power imbalance
- In terms of compute power and storage, it is also entirely unnecessary



If you need someone else's server to use your applications...

- They **own your action potential**
- And they can take it away at will
- This is a strange power imbalance
- In terms of compute power and storage, it is also entirely unnecessary



If you need someone else's server to use your applications...

- They **own your action potential**
- And they can take it away at will
- This is a strange power imbalance
- In terms of compute power and storage, it is also entirely unnecessary



If you need someone else's server to use your applications...

- They **own your action potential**
- And they can take it away at will
- This is a strange power imbalance
- In terms of compute power and storage, it is also entirely unnecessary

Most current attempts to re-establish end-to-end connectivity between compute devices are *exclusively* overlays to the existing internet protocol stack

- This can work, but:
  - It inherits all of IPs problems and overhead, tries to **work around** them
    - And in the process creates more overhead.



- They own your action potential
- And they can take it away at will
- This is a strange power imbalance
- In terms of compute power and storage, [illegible] entirely unnecessary

Most current attempts to re-establish end-to-end connectivity between compute devices are exclusively overlays to the existing internet [illegible] stack

- This can work, but:
  - It inherits all of IPs problems and overhead, tries to work around them
    - And in the process creates more overhead.
- That overhead effectively excludes:
  - Low-bandwidth radio protocols
  - Disaster relief
  - Resource-constrained environments
  - And many other adverse scenarios

Writing decentralised and self-sovereign applications and systems on top of IP is hard, since 90% of the work goes into making IP do things it was accidentally designed to prevent.

If you need someone else's server to use your applications...

- They **own your action potential**
- And they can take it away at will
- This is a strange power imbalance
- In terms of compute power and storage, it is also entirely unnecessary

Most current attempts to re-establish end-to-end connectivity between compute devices are *exclusively* overlays to the existing internet protocol stack

- This *can* work, but:
  - It inherits all of IPs problems and overhead, tries to **work around** them
    - And in the process creates more overhead.
- That overhead effectively excludes:
  - Low-bandwidth radio protocols
  - Disaster relief
  - Resource-constrained environments
  - And many other adverse scenarios

Writing decentralised and self-sovereign applications and systems on top of IP is **hard,** since 90% of the work goes into making IP do things it was accidentally designed to prevent.

Base-layer infrastructure is inherently centralised

And if not *fixed*, then about as dynamic as slow-flowing pitch



- Base-layer infrastructure is inherently centralised

And if not *fixed*, then about as dynamic as slow-flowing pitch



- Ideally, we want a newly created endpoint address to be reachable in *minutes*

... not days or weeks.

- You never own your endpoint addresses

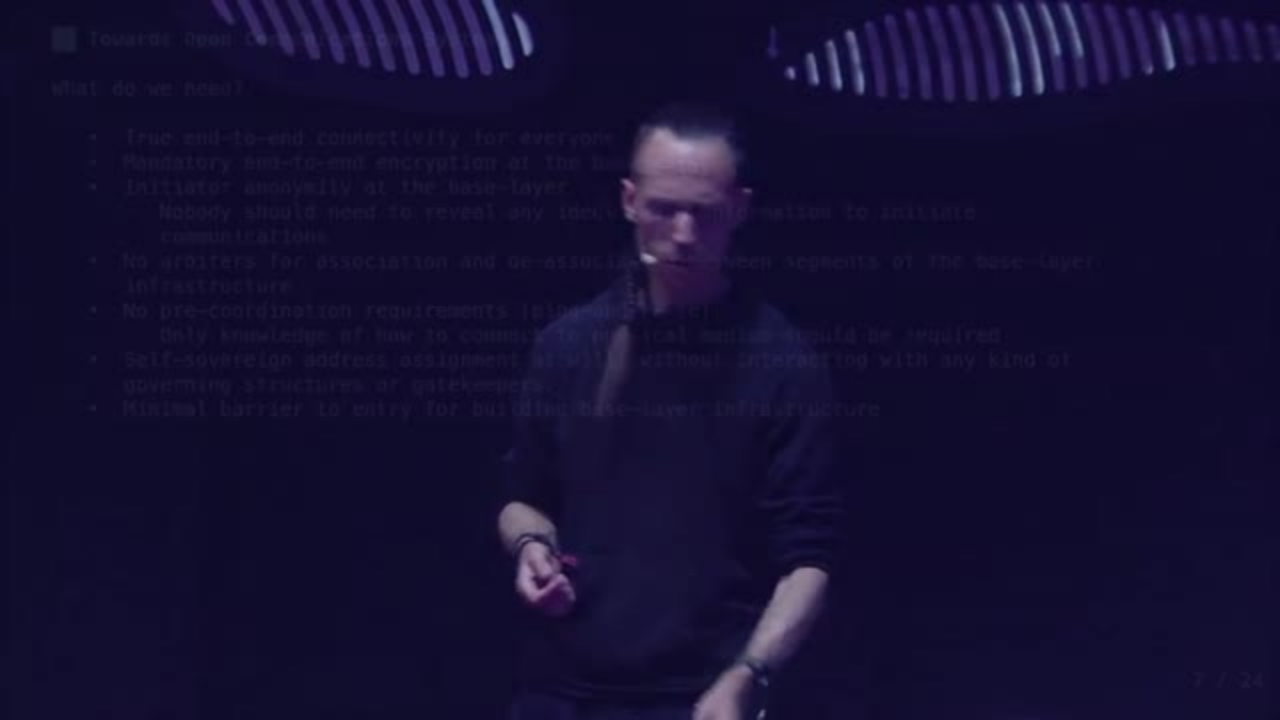


You never own your endpoint addresses

And for them to be discoverable and securely connectable by *anyone*, you **must** use and trust:

- Base-layer infrastructure providers
- Public IP address provider
- DNS providers
- SSL certificate authorities
- And potentially also server infrastructure hosting providers




What do we need?
• True end-to-end connectivity for everyone
• Mandatory end-to-end encryption at the base-layer
• Initiator anonymity at the base-layer
Nobody should need to reveal any identifying information to initiate communications
• No arbiters for association and de-association between segments of the base-layer infrastructure
• No pre-coordination requirements
Only knowledge of how to communicate on the physical medium should be required
• Self-sovereign address assignment at will, without interfering with any kind of governing structures or gatekeepers
• Minimal barrier to entry for building base-layer infrastructure

Towards Open Communications Systems

What do we need?

- True end-to-end connectivity for everyone
- Mandatory end-to-end encryption at the base-layer
- Initiator anonymity at the base-layer
  - Nobody should need to reveal any identifying information to initiate communications
- No arbiters for association and de-association between segments of the base-layer infrastructure
- No pre-coordination requirements (plug-and-route)
  - Only knowledge of how to connect to physical medium should be required
- Self-sovereign address assignment at will, without interacting with any kind of governing structures or gatekeepers.
- Minimal barrier to entry for building base-layer infrastructure

Towards Open Communication Systems

What do we need?

- True end-to-end connectivity for everyone
- Mandatory end-to-end encryption at the base-layer
- Initiator anonymity at the base-layer
  - Nobody should need to reveal any identifying information to initiate communications
- No arbiters for association and de-association between segments of the base-layer infrastructure
- No pre-coordination requirements (plug-and-route)
  - Only knowledge of how to connect to physical medium should be required
- Self-sovereign address assignment at will, without interacting with any kind of governing structures or gatekeepers.
- Minimal barrier to entry for building base-layer infrastructure

It's not possible to just coordinate a large-scale "switch" to a new networking technology

- Isolated small-scale networks must be fully functional and provide value to their users without the rest of the world necessarily caring about it
- Such isolated networks must be able to dynamically converge into larger networks if paths between them become available
  - This **must** occur automatically, without requiring any oversight or user intervention
- Must be able to co-exist with, and bridge over already established infrastructure and protocols

without requiring any kind of massive switchn evolve organically and be immediately useful,

Towards Open Communications Systems

What do we need?

- True end-to-end connectivity for everyone
- Mandatory end-to-end encryption at the base-layer
- Initiator anonymity at the base-layer
  - Nobody should need to reveal any identifying information to initiate communications
- No arbiters for association and de-association between segments of the base-layer infrastructure
- No pre-coordination requirements (plug-and-route)
  - Only knowledge of how to connect to physical medium should be required
- Self-sovereign address assignment at will, without interacting with any kind of governing structures or gatekeepers.
- Minimal barrier to entry for building base-layer infrastructure

It's not possible to just coordinate a large-scale "switch" to a new networking technology

- Isolated small-scale networks must be fully functional and provide value to their users without the rest of the world necessarily caring about it
- Such isolated networks must be able to dynamically converge into larger networks if paths between them become available
  - This **must** occur automatically, without requiring any oversight or user intervention
- Must be able to co-exist with, and bridge over already established infrastructure and protocols

without requiring any kind of massive switchn evolve organically and be immediately useful,



Addressing in Reticulum

- Completely throw out the idea of having random "authorities" assign specific, hierarchical number sequences as endpoint addresses



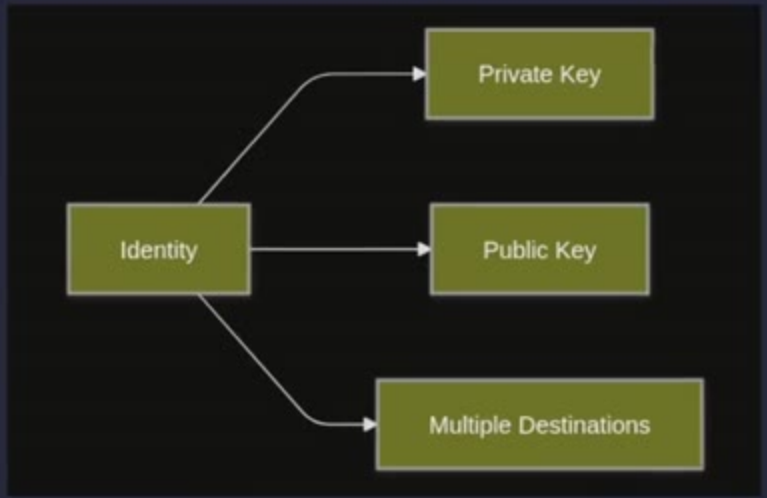
Addressing in Reticulum

- Completely throw out the idea of having random "authorities" assign specific, hierarchical number sequences as endpoint addresses

- Instead, use cryptographic identites as the foundation for addressing
  - Create an Ed25519 / X25519 keypair
  - Derive public keys -> mix in destination type
  - Get a destination hash (address)
  - Sign generated destination hash together with verification data and public key
  - Broadcast this *destination announcement* on the network
  - Everyone can now verify the public key required for initial establishment of communication with this particular destination
  - Routing devices (*transport nodes*) can verify these announces to ascertain the minimum required topological knowledge to route packets towards the destination
    - Nobody knows the full network topology, only the currently best next hop

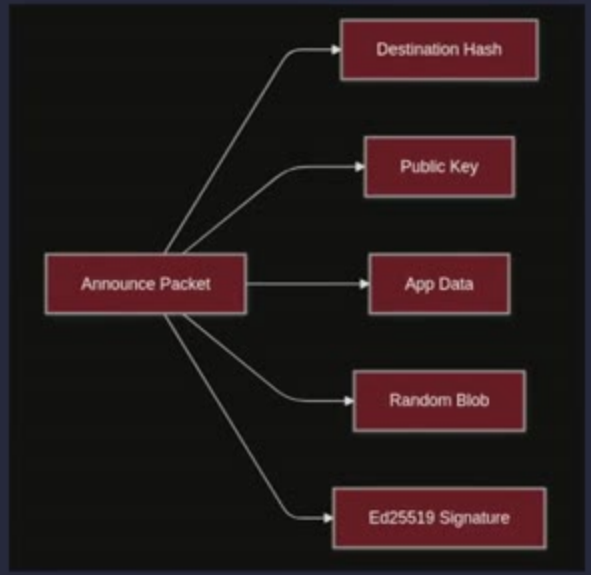


Creating Destinations

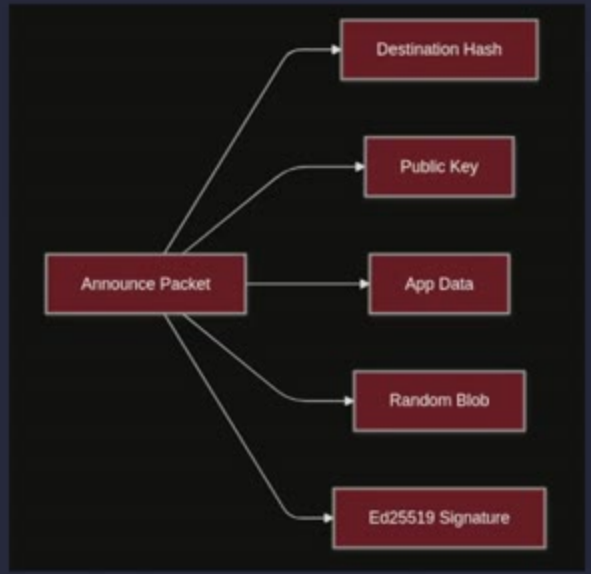


- Multiple destination types for the same identity:
  - *lxmf.delivery*, *lxmf.propagation*, *sensor.temperature*, etc.

Announces


Destination Hash
Public Key
Announce Packet
App Data
Random Blob
Ed25519 Signature

Announces


Destination Hash
Public Key
Announce Packet
App Data
Random Blob
Ed25519 Signature

Network interfaces

- Modular and user-expandable interface system
  - Automatically use any number of heterogeneous physical or virtual interface types to form a self-configuring mesh
  - Works over anything that supports a physical layer MTU of 500 bytes and more than 5 bits per seconds of throughput
  - Radio modems, LoRa radios, Ethernet, Wifi, Serial lines, TCP, UDP, I2P, *and more or less anything else*



Network interfaces

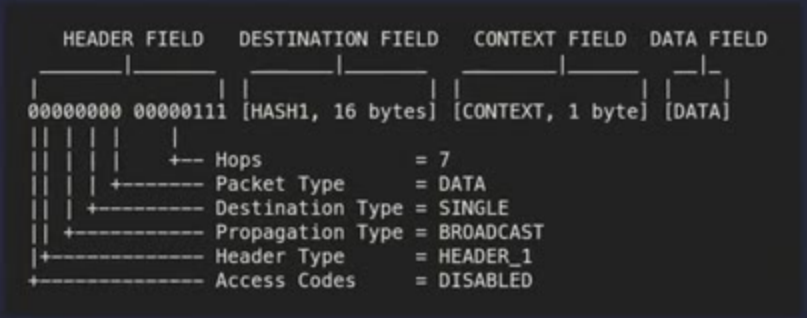
- Modular and user-expandable interface system
  - Automatically use any number of heterogeneous physical or virtual interface types to form a self-configuring mesh
  - Works over anything that supports a physical layer MTU of 500 bytes and more than 5 bits per seconds of throughput
  - Radio modems, LoRa radios, Ethernet, Wifi, Serial lines, TCP, UDP, I2P, *and more or less anything else*

Easily handles *a lot* of interfaces





Packet structure

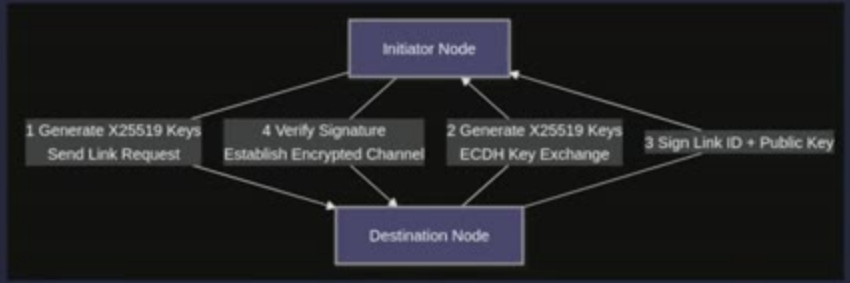


- Simple packet format, low overhead
- Destination field, potential next-hop destination field
- No source address
  - No packets include sender information
  - Can still be ACKed by receiver by signing the packet hash, and directing this "proof" along the reverse path of where it came from

- Works directly even over unmanaged mediums (no MAC layer necessary)
  - Can be tunneled over all existing protocols

Link Establishment

Establishing end-to-end encrypted links in Reticulum is lightweight and efficient.

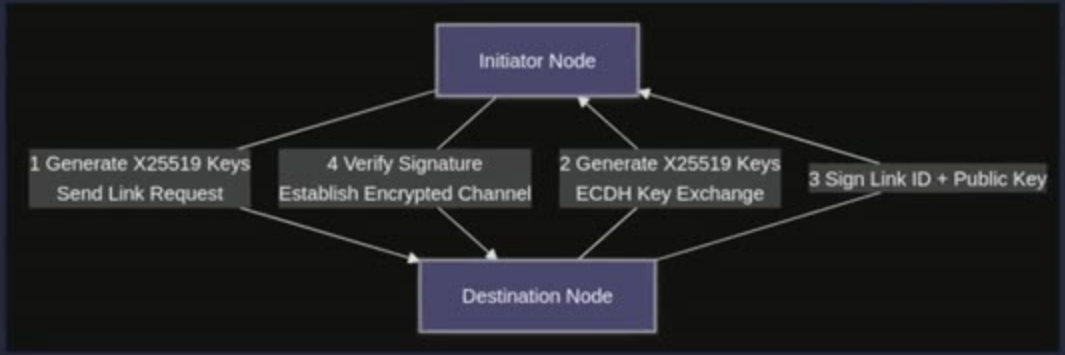


- Total link setup sequence is only 3 packets, totalling 297 bytes
  - Provides an end-to-end encrypted link with forward secrecy
  - Very low upkeep costs at only 0.44 bits per second per link
- Link initiators remain anonymous
  - But *can* identify to remote destination over encrypted link once established



Link Establishment

Establishing end-to-end encrypted links in Reticulum is lightweight and efficient.



- Total link setup sequence is only 3 packets, totalling 297 bytes
  - Provides an end-to-end encrypted link with forward secrecy
  - Very low upkeep costs at only 0.44 bits per second per link
- Link initiators remain anonymous
  - But *can* identify to remote destination over encrypted link once established

Putting it all together

In its current state, Reticulum provides

- Coordination-less globally unique addressing and identification
- Full end-to-end connectivity for all destinations on the network
- Mandatory end-to-end encryption with forward secrecy
  - Routing unencrypted traffic is impossible
- Fully self-configuring multi-hop routing over heterogeneous carriers
  - Ability to carry data over any mixture of physical mediums and topologies
  - Low-bandwidth networks can co-exist with large, high-bandwidth networks
- Initiator anonymity
- Forward Secrecy for all communication types, both for single packets and over links
- Signed packet delivery confirmations
- Flexible and extensible interface system
- Authentication and virtual network segmentation on all supported interface types
- Both high- and low-level APIs
  - Raw packet APIs
  - Reliable and efficient transfer of arbitrary amounts of data
  - Lightweight, flexible and expandable Request/Response mechanism
  - Reliable sequential delivery with Channel and Buffer APIs
- Very broad range of supported hardware
  - Ethernet, WiFi, LoRa transceivers, radio modems, serial and more
  - Anything else with custom interface modules



What works over Reticulum right now?

Useful tools and utilities

- System-wide Reticulum daemon: rnsd
- Fully interactive remote shell: rnsh
- Various management utilities: rnstatus, rnpath, rnprobe
- Identity management and offline file encryption: rnid



What works over Reticulum right now?

Useful tools and utilities

- System-wide Reticulum daemon: rnsd
- Fully interactive remote shell: rnsh
- Various management utilities: rnstatus, rnpath, rnprobe
- Identity management and offline file encryption: rnid

Information sharing

- File transfers, file servers and file drops: rncp
- Lightweight page hosting and browsing: nomadnet
  - Low bandwidth, works over LoRa and packet radio
  - Can run fully interactive and dynamic applications
    - Forums, news aggregators, gemini/gopher proxies



What works over Reticulum right now?

Useful tools and utilities

- System-wide Reticulum daemon: rnsd
- Fully interactive remote shell: rnsh
- Various management utilities: rnstatus, rnpath, rnprobe
- Identity management and offline file encryption: rnid

Information sharing

- File transfers, file servers and file drops: rncp
- Lightweight page hosting and browsing: nomadnet
  - Low bandwidth, works over LoRa and packet radio
  - Can run fully interactive and dynamic applications
    - Forums, news aggregators, gemini/gopher proxies

The LXMF protocol

A universal messaging protocol that provides secure, self-sovereign communications with several different clients implemented.

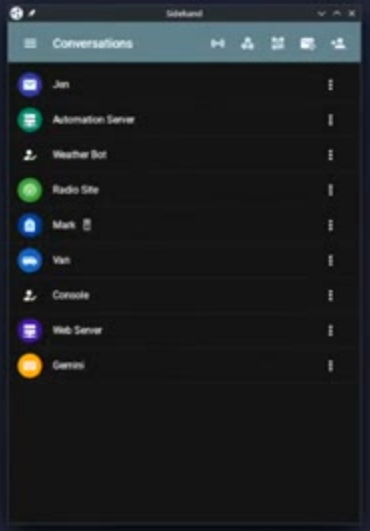
- End-to-end connectivity and encryption
- Short and long-form text messaging
- Voice messaging
- Image and file attachments
- Command / response API
- Seamlessly roams between networks and connection types



LXMF Clients

Sideband

- Works on Linux, Android, macOS, Windows
- Text and voice messaging (Codec2 and Opus)
- Send images and file attachments
- Fully extensible via user plugins
- Peer-to-peer telemetry and location sharing
- Extensible command, request and response system



LXMF Clients

Sideband

- Works on Linux, Android, macOS, Windows
- Text and voice messaging (Codec2 and Opus)
- Send images and file attachments
- Fully extensible via user plugins
- Peer-to-peer telemetry and location sharing
- Extensible command, request and response system



LXMF Clients

Sideband

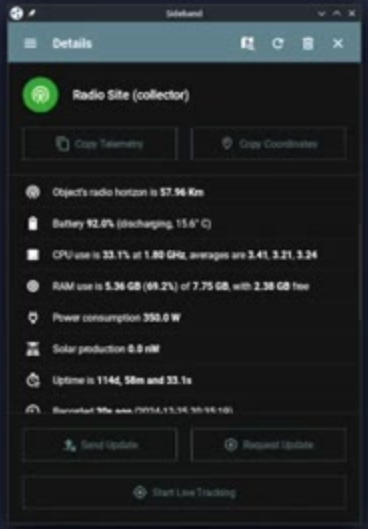
- Works on Linux, Android, macOS, Windows
- Text and voice messaging (Codec2 and Opus)
- Send images and file attachments
- Fully extensible via user plugins
- Peer-to-peer telemetry and location sharing
- Extensible command, request and response system



LXMF Clients

Sideband

- Works on Linux, Android, macOS, Windows
- Text and voice messaging (Codec2 and Opus)
- Send images and file attachments
- Fully extensible via user plugins
- Peer-to-peer telemetry and location sharing
- Extensible command, request and response system



LXMF Clients

MeshChat

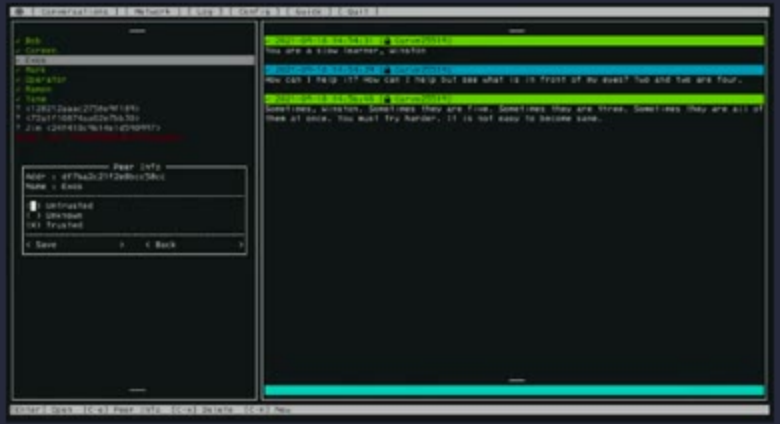
- User-friendly client for desktop OSes
- Text and voice messages
- Image and file attachments
- Real-time audio calls
- Network visualiser
- Nomad Network browser



LXMF Clients

Nomad Network

- Basic LXMF text messaging
- Can host LXMF propagation nodes
- Page hosting
- Page browser





Kthxbye!

- Questions?

Kthxbye!

- Questions?







