Secure Share

Daniel Reusche and Gabor Toth

August 15, 2012

Secure Share

A framework for secure and privacy-protecting social interaction based on peer-to-peer technology

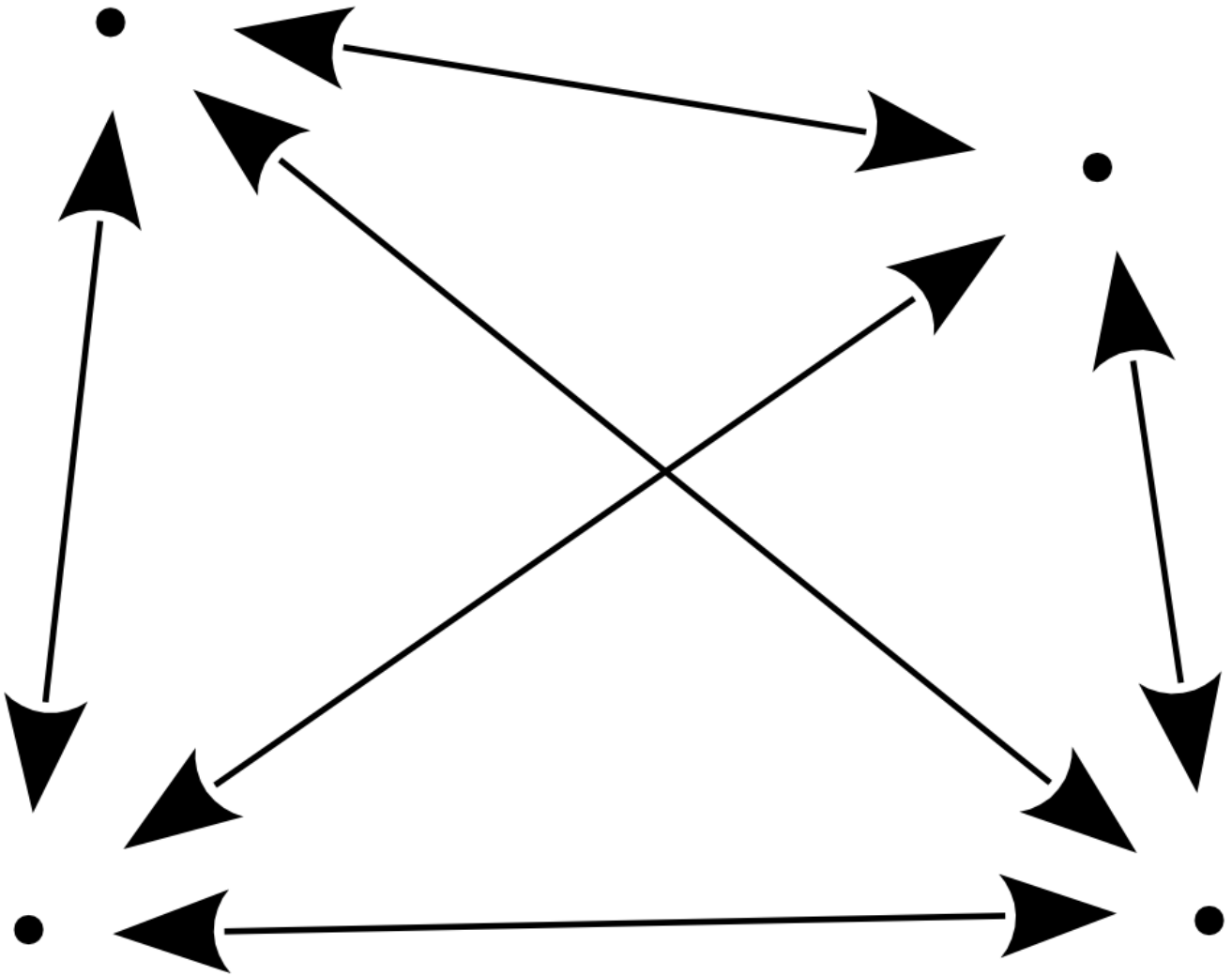
Goals

- arbitrary messanging
  - from one-to-one to many-to-many
  - status updates and messages
- file transfer
  - sharing of pictures, music etc.
  - collaborative document editing

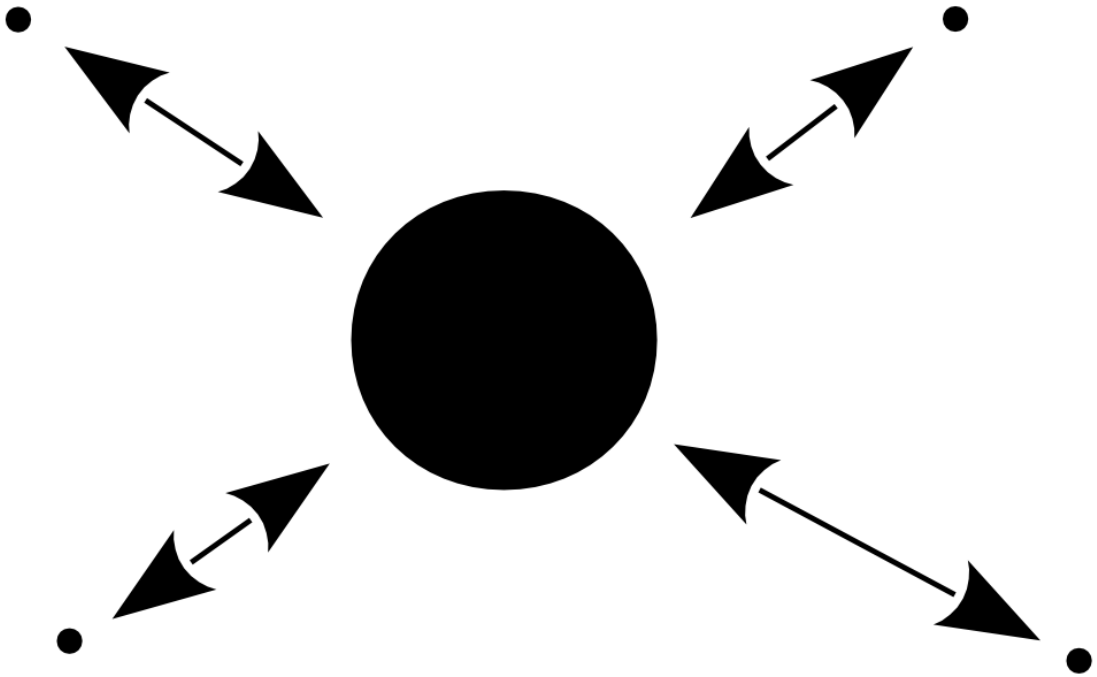
Privacy

Privacy

Ideal Case



Centralized services



Privacy requirements

- end-to-end encryption
- forward secrecy
- padding of packets
- delayed forwarding
- private contact list
- free and open source software

Approach: federated systems

And why not to use them

- personal data on servers
- personal data shared with even more server operators
- only link-level encryption
- PGP, OTR not enough

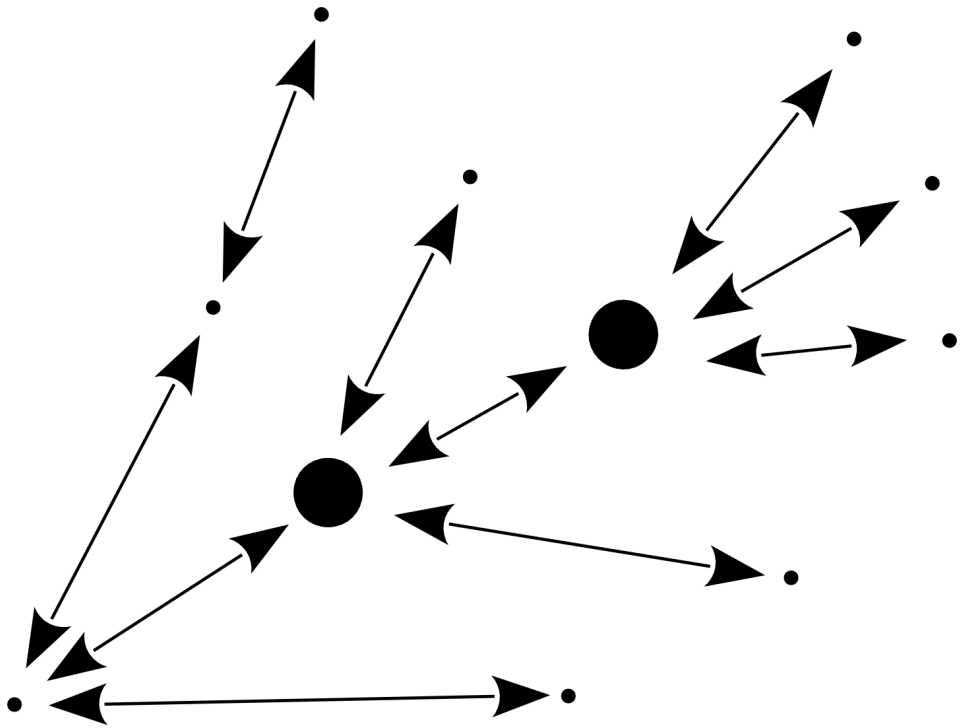
Scalability

Scalability

Social interaction

- one-to-many status updates
- many-to-many group communication

Multicast



Architecture

Architecture

Friend-to-friend architecture

- connect to trusted nodes
- prevents active attacks

Personal devices

- Software runs on personal devices
- Data is stored on personal devices

Personal devices

- laptop, PC
- plug computers, home routers, servers
- smartphones

Peer-to-peer framework requirements

- free/libre/open-source software
- multi-platform, lightweight, written in a compiled language
- provides API for essential P2P features
  - bootstrapping, addressing, routing, encryption, NAT traversal

GNUnet

- written in C
- multi-platform
- modular framework
- advanced NAT traversal

GNUnet

- multiple transport methods
  - TCP, UDP
  - HTTP, HTTPS
  - SMTP
  - ad-hoc WiFi

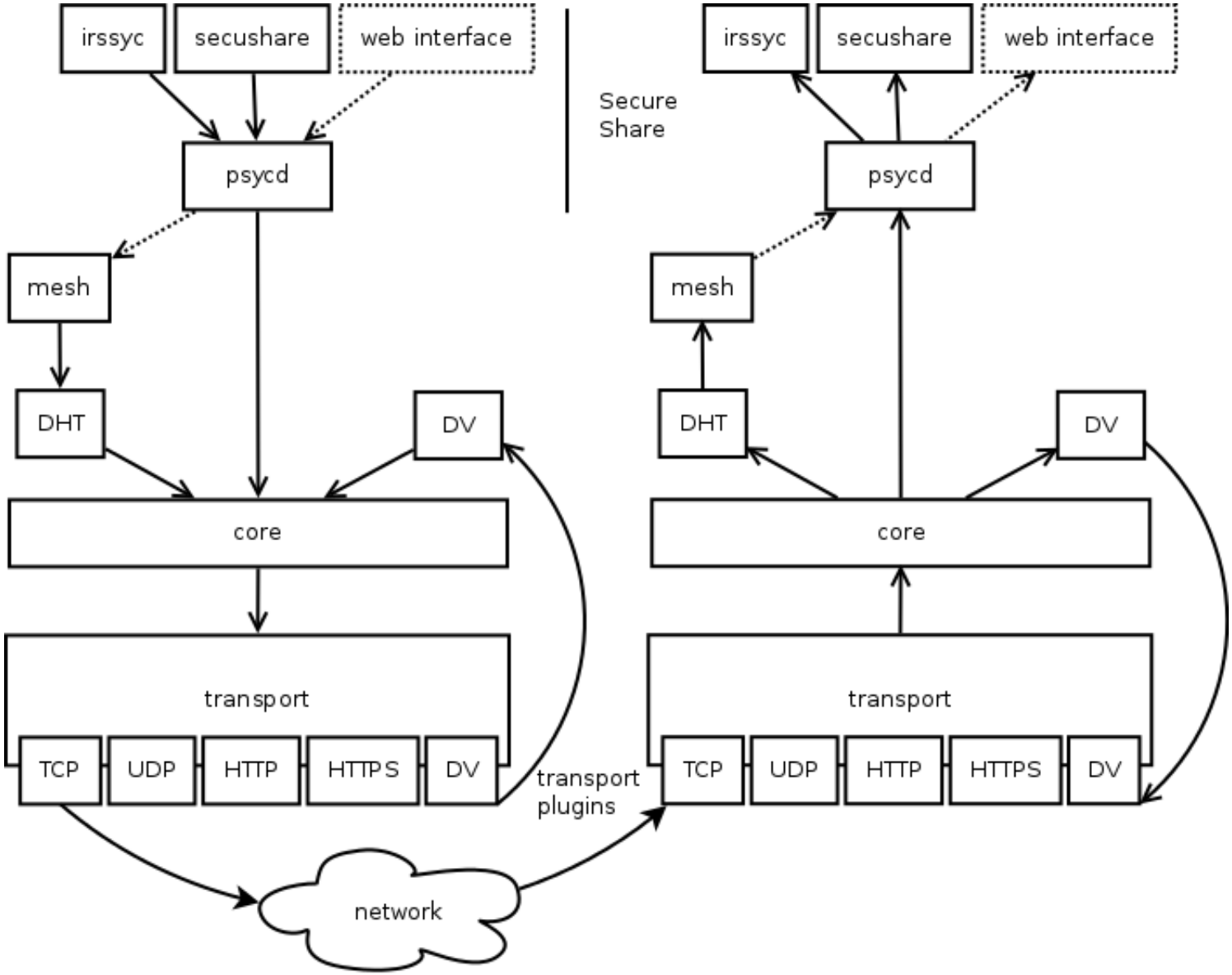
GNUnet

- distributed hash table (DHT)
- file sharing
  - based on DHT and GAP
- various routing schemes
  - fish-eye bounded distance vector protocol
    - gossiping in a limited neighborhood
    - improves connectivity
    - onion routing
  - mesh service
    - supports multicast
    - uses DHT for routing

psycd

- messaging protocol
- manages connections, friendship between users
- client interface

GNUnet - components and message flow



Implementation

Implementation

Components

- libpsyc
- psycd
- GNUnet libraries

Identifiers

- Federated PSYC
  - based on DNS

    `psyc://example.net/~alice#friends`
- P2P PSYC
  - based on public key

    `psyc://I0GC...L29G:g/#friends`

Contacting peers

- initial contact: hello message
  - public key
  - current addresses
- next time contact to same address
- or find new address through other peers

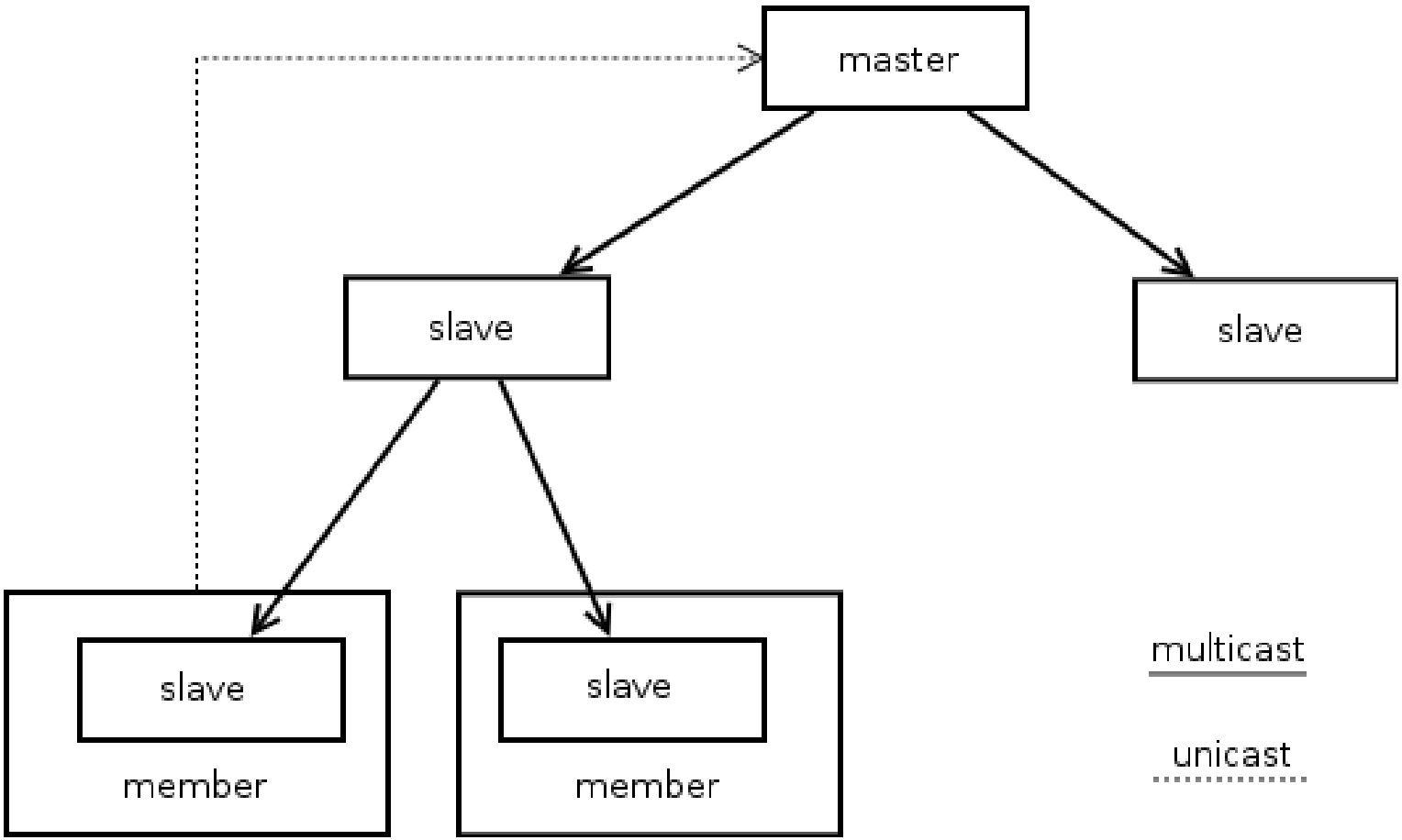
Circuits

- GNUnet
- TCP
- UNIX domain sockets
- TLS

Entities

- person
  - clients link to person entity
  - can subscribe other entities
- place
  - group communication
  - news feeds

Multicast contexts



Distributed state

- profile data, context membership
- push changes once
- synchronize after subscription
- recover lost packets
- syntax changes to support more complex data structures

Storage

- incoming and outgoing packets
- state variables
- SQLite database
  - multiplatform
  - lightweight
  - small memory footprint

Clients

Clients

Desktop clients

- secushare
  - based on Qt/QML
  - multiplatform
  - touch UI
- irssyc
  - based on irssi
  - intended for debugging and for advanced users

Web interface

- JavaScript
- WebSocket

Mobile clients

- port GNUnet to mobile devices
- or client only approach

Extensibility

- channel API
  - using a sandboxed QML or HTML view
  - JavaScript API
  - enables easy app development
  - access only channel data
- client API
  - using libpsycclient
  - allows for developing full-fledged clients

Future work

Future work

Future work

- routing layer - multicast
- separate user and node identities
- user interface
  - improvements on desktop
  - implement web UI
  - implement mobile UI
- file transfers