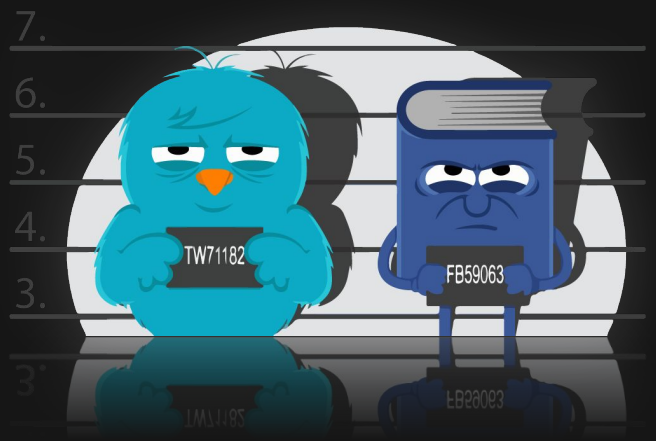
Project Mercury

István Zólyomi
Discord: @Bartmoss

State of the Internet

- Decentralization and distributed networks are not new
- IP + TCP are distributed and reliable as should be
- Too successful => IPv4 addresses ran out
- Network Address Translation
  - ISPs and routers block incoming connections
  - Safe but comes with a price
  - No universal workaround (hole punching/NAT traversal)
- Peer to Peer connection is hard at home

United States of Internet



- Web (HTTP) and cloud wins
- Business prefers central apps
  - IPv6 "sabotaged"
- Google/Facebook/etc lock you up and they
  - give your identity with amazing services
  - “own” your data and spying on you for profit
- No interoperability, you lose “citizenship” changing provider
  - No laws like for phone number migration

Mercury identity model

- Independent from service provider
- Cryptographic public key for identity
- It’s your data: encrypt it by default
- Multiple unconnected profiles (personas, cryptographic keys)
- Restore all your profiles with a single “HD wallet” seed
- Communicate by addressing `profile:app`, forget `IP:port`
- Cryptographically signed profile relations
    - Social graph usable by any application

Mercury network model

- Overlay network on any transport (Tcp, Onion, I2P, mesh, etc)
- Open, fault-tolerant "mobile network"
  - Like BitTorrent or ancient Skype but generic purpose
  - Like SIP or Hamachi but distributed and open
- Clients pick “cells” by trust and free to move if unsatisfied
  - Not automated by signal strength or load balancing
  - Otherwise trustless system
- Encrypted communication, P2P if hole punching possible
- Push notifications
- Plugable hash-based distributed storage
  - Content loaded from “links” is verifiable (Merkle-proof)

Mercury Home protocol

- Nextgen Profile Server and IoP Connect with fixed design
  - E.g. connection multiplexing, better security and simpler client
- Home server: your replaceable entry point to Mercury
  - runs in data center or your Titania box at home
  - provide storage for private client data backups
  - send push notifications on events and calls
  - help initiating P2P connections or relay calls as fallback
- Clients (Pc, phone) contact servers to
  - dedicate home server(s) for profiles and keep connected
  - share public profile data, backup encrypted private data
  - handle profile relations, incoming events and calls

Tech stack

- Safe and strict Rust compiler guarantees
  - no pointer problems, e.g. segfaults or buffer overflows
  - no concurrency problems
  - resource leaks on par with GCs but deterministic
- Rust enables platform compatibility through
  - convenient cross-compiler
  - exposing C API for easier language binding generation (e.g. Swig)
  - WebAssembly output
- TokIO library enables asynchronous (i.e. effective) services
- Cap'n'Proto provides easy asynchronous networking with RPC

Project status

- System foundations in testing
  - Server and client developed together
  - Legacy stack functionality (and some more) provided already
  - Better, safer, simpler, easier to deploy
  - Use our Rust client or your own Cap’n’Proto binding
  - Get your hands dirty with dApp experiments, feedback is welcome
- It’s big, parts still in the works
  - Rough edges: docs, lang bindings, profile search, stronger error typing
  - Tech awesomeness: Diffie-Hellman key exchange, integrating DHTs, hole punching, undelivered message persistency, HD wallets
- Opening source code after conference

dApp SDK

- Design goals
  - Platform-agnostic, convenient SDK for distributed applications
  - Usage complexity comparable to mobile phone plus message recorder
    - Expose social graph, events and calls
    - Hide all possible tech details (homes, networking, cryptography, etc)
  - Use hardware wallets as profile identity without exposing keys
  - Pluggable home selection and social graph management
    - Served e.g. with platform-specific GUI
- Clear initial concept, implementation underway
- What would you like to build with it?