tinySSB
fact sheets

v2b
2024-08-07

tinySSB
in a (nut) shell

Table of Contents

tinySSB = “tiny Secure Scuttlebutt (SSB)”

- tinySSB inherits SSB’s belief system:
  - offline first
  - secure
  - **no dependency on intermediaries**
  - as decentral as it can be

- Race to the bottom: go where nobody else can go

  ... while keeping all desirable properties

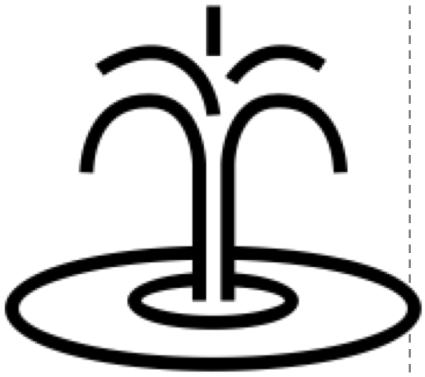
- Unique Selling Points:
  - tinySSB runs on **embedded devices, smartphones**
  - packet size is **120 Bytes**: fits Bluetooth LE, LoRA
  - in the future also short wave radio, satellites ..
  - playground for **teaching** decent(ralized) concepts

tinySSB mindset and tenets

Table of Contents

- evolution of Secure Scuttlebutt
  - binary packet format (instead of JSON)
  - “shadow packet headers”
    (don’t send values that are implicit)
  - no “blobs” outside append-only logs -> side chains

- **data fountain model**
  - data source matters
  - rest is “replication to everywhere”



- replication protocol:
  - **connectionless** (just let data flow everywhere)

- Works without intermediaries:
  - no Internet? no DNS? no IP address? no problem!
  - use “ionosphere bouncing” instead of Starlink

small is beautiful

tech at human scale, not “megalo rocket science”

tinySSB
the Android app

Table of Contents

"PC experience" is vanishing: smart-phones are ubiquitous, even dominant

show QR code (share public key)

view replication status

context-specific menu

the public chat group

for your *encrypted* notes, sketches, passwords

encrypted chats (static groups)

chat is the default "tool"

manage contacts

productivity tools

game collection

tinySSB and BYOD

(bring your own device)

Table of Contents

- currently Android-only

- **How-to:**

  1) if necessary allow *"apps from unknown sources"*, as follows:

  -> settings -> apps -> special access -> install -> Chrome

  2) download APK file (**dWeb release**), install it

  3) grant "Localization permission"

- Go live:

  - enable Bluetooth

  - enable Localization

- Customize:

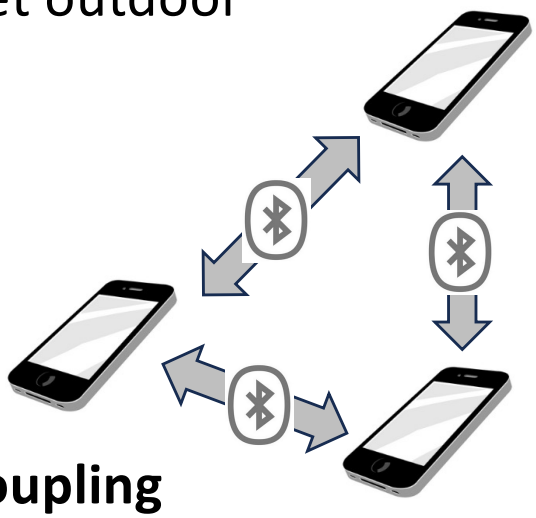
  - go to "contacts" and change "me" to your pseudonym

see peers

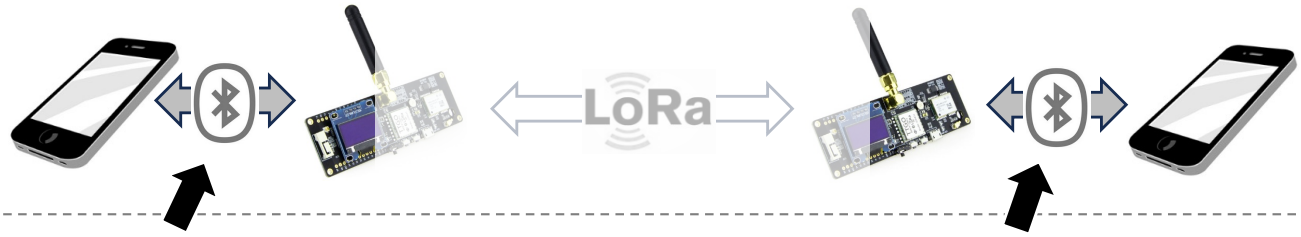
https://github.com/ssbc/tinySSB/releases

tinySSB
Bluetooth Low Energy (BLE)

- Why BLE?
  "hidden" connectivity substrate, used by Apple and others to "identify the context" (AirTags, AppleWatch, etc)
  - unlike Bluetooth **no pairing needed**!
    great for seamless → onboarding
  - range: 30 feet indoors / 300 feet outdoor
    (10 meters / 100 meters)



- BLE used in tinySSB for:
  - **Smartphone to Smartphone coupling**
  - Smartphone to LoRA relais coupling (long range radio)



tinySSB
Long-range RAdio (LoRA)

Table of Contents

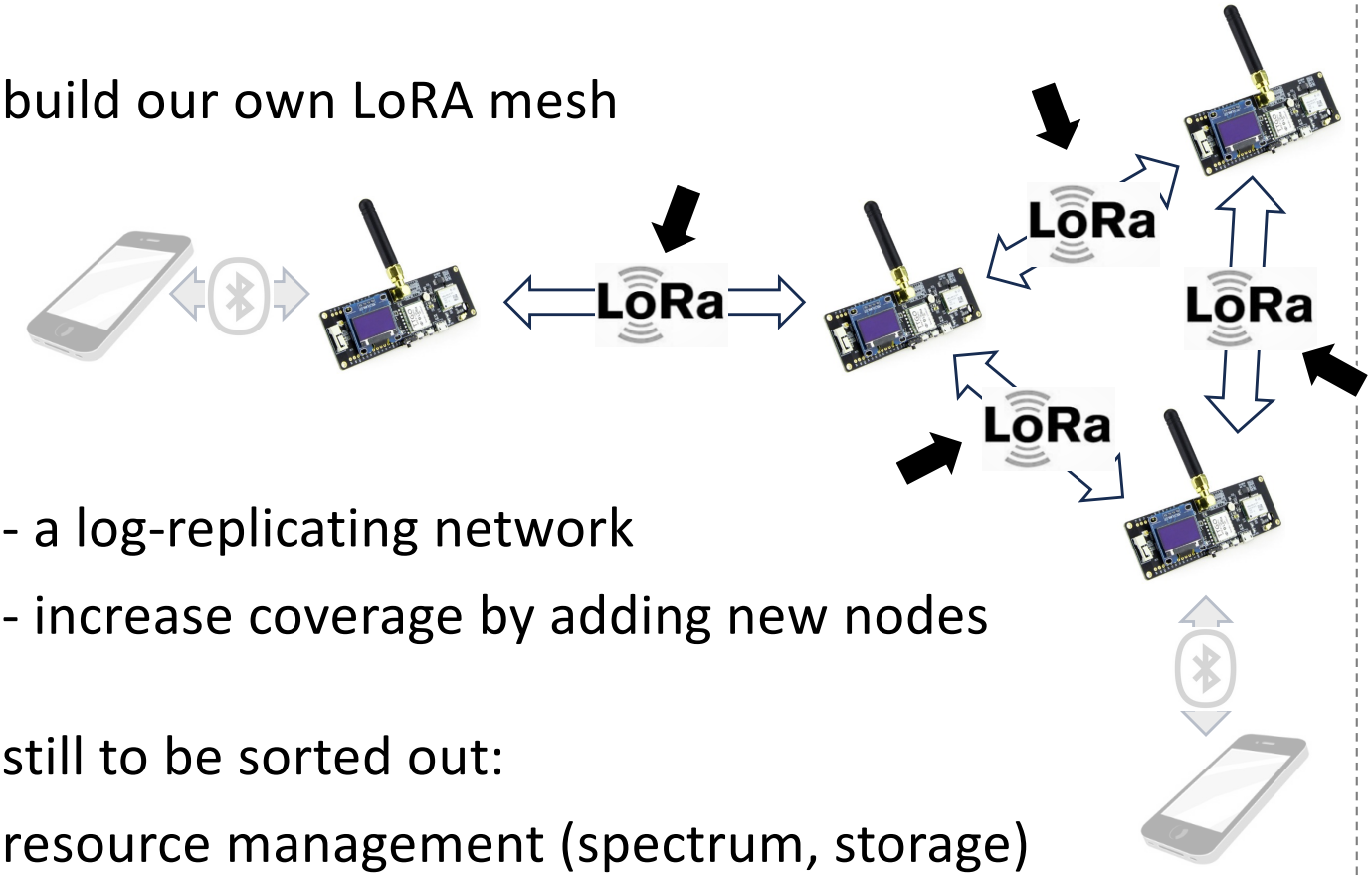
Why LoRA?

Long-range comms requires “carriers” and other intermediaries (Internet, sat).

LoRA goes beyond Bluetooth:

- cheap devices ($30) reaching miles
- no license needed to use this part of the spectrum

• build our own LoRA mesh



- a log-replicating network
- increase coverage by adding new nodes

• still to be sorted out:
resource management (spectrum, storage)

tinySSB data hubs (ws and git)

Table of Contents

Default data sync is between end devices.

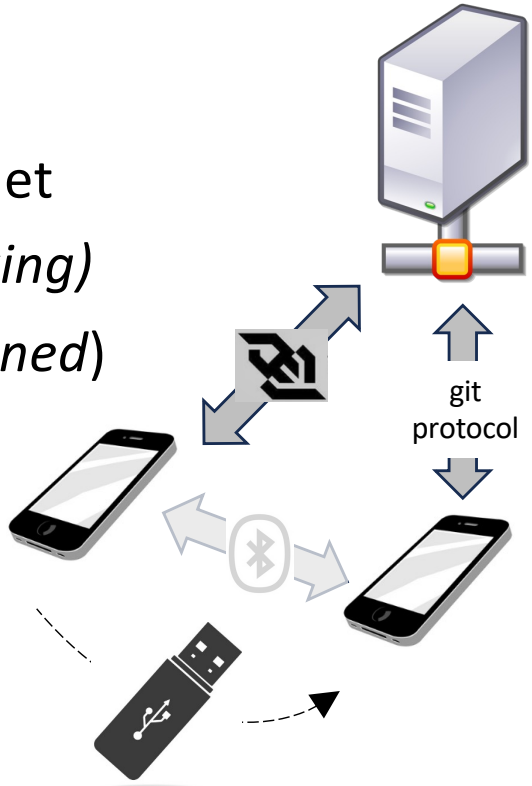
Arbitrary "assists/hubs" can be added:

- on a **voluntary** basis
- can always go back to local contact
- or switch to and **mix in other assists**.

- Useful assists:

  put tinySSB servers on the Internet

  - **websocket** access *(is working)*
  - any **git server**, incl gitHub (*planned*)
  - **USB stick** (*planned*)



- Assists have a problem:

  how to know these hubs?

  (so called rendez-vous problem)

tinySSB
decent teaching, at BSc level

Table of Contents

Communications textbooks are about:

- the Internet
- distributed systems and group comm.

But **almost no teaching resources on "decentral system design"**

- Several reasons for lack of decent resources:
  - client-server much more relevant in practice, today
  - CRDT results rather recent (since 2011)
  - literature still academic, ongoing research
  - few patterns and libraries for a "complete SW stack"

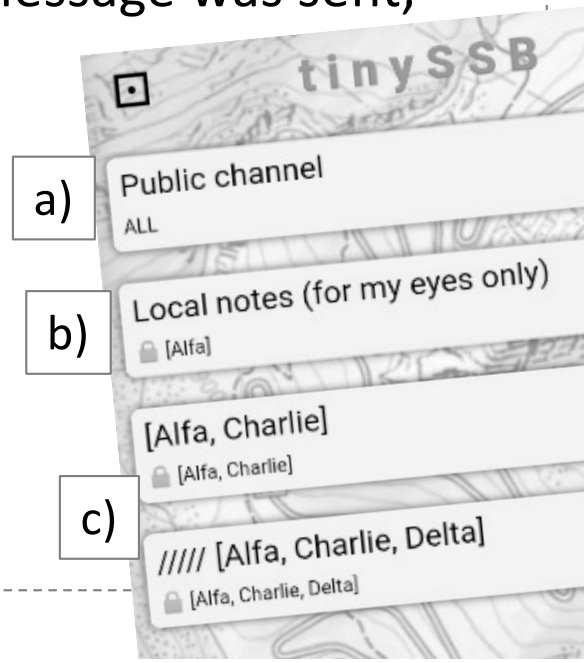
- Good reasons for not waiting:
  - **future engineers are formed now**
  - **must compete for these talents**
  - must push decentral alternatives into BSc studies

  use tinySSB to let students step outside "client/server"

tinySSB
encrypted chat
+ public channel

Table of Contents

- tinySSB dWeb demo version has:
  a) public chat: authenticated but unencrypted
  b) encrypted chat "to self" (used for private notes, memos, writing down passwords)
  c) **encrypted chat groups** (as in Secure Scuttlebutt)
     - up to 7 peers, static
     - **metadata protection**: encrypted blobs do not reveal to which chat this message was sent, nor that this is a chat message



- Plans for enhancements:
  - dynamic group membership (→ **"security bubbles"**)
  - "perfect forward secrecy" (double-ratchet protocol)

tinySSB **security** and resource **bubbles**

Table of Contents

- observation **A**: global reach is evil, leads to influencer wasteland and massive troll farms

- observation **B**: willingness to sponsor technical resources **works IF for the good of your community** (i.e. you should never feed trolls, or X)

- A+B lead to the same goal: protect your infrastructure, your chat room, your scarce wireless bandwidth
  → **be in control of your security perimeter!**

- novel tinySSB concept: **user-defined security bubbles**
  - linking crypto protection of chat rooms with "replication horizon", tasking your set of relais
  - **first prototype is working** (BSc thesis), not yet integrated into the dWeb tinySSB smartphone app

tinySSB onboarding

Table of Contents

The onboarding problem:

*Without servers that store other's data and directories, I have to know from whom I want to read "their data feed". How can **you** tell **me** to consider your feed, without me already listening to you?*

- Two ways of onboarding, both "out-of-band":
  - "direct" (scan QR code with your ID)
  - "by recommendation" (receive your ID via friends)

  Secure Scuttlebutt had some twisted ways ..

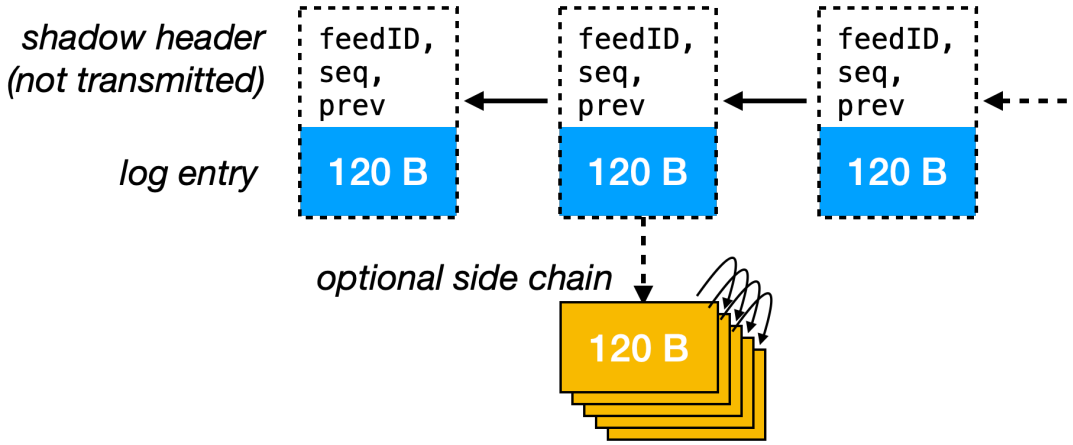
- tinySSB's take:
  - use **local** ad hoc situation (e.g. Bluetooth reach) to **eagerly import** "short-term acquaintances"
  - **adopt their ID** into long-term → "security bubbles" (family, 15 buddies groups, dWeb folk, HAMs..)

tinySSB immutable logs and CRDT

"One append-only log per peer"

we stick to this Secure Scuttlebutt tenet

- **Sender** can only append. Then,
  - everybody can validate and replicate
  - **compressed log format**, all parceled in 120B chunks



- **applications** ship their messages inside log entries
  - use Conflict-Free Replicated Data Types (CRDT) to have "**data convergences**" without any servers
  - the "set of append-only logs" is by itself a CRDT!

  → decentral, peer-to-peer, offline-first apps

tinySSB replication protocol(s)

Trad. replication protocols are heavy or use TCP connections

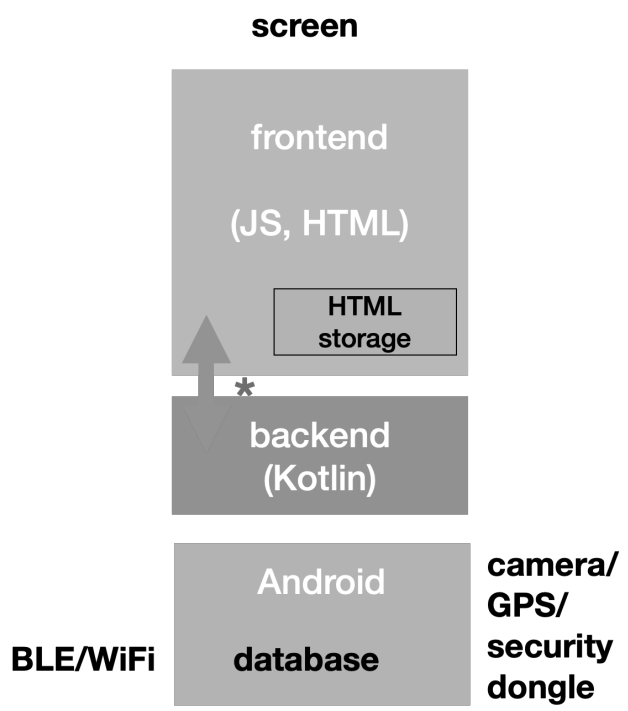
→ we needed something lightweight

- strictly datagrams, limited to 120 Bytes
  - no connections
  - only three message types:
    + WANT (announces own "replication frontier" of logs)
    + CHNK (announces missing side-chain pkts)
    + DATA (actual log entries and side chain pkts)

```
mk_want offs=0, vector=[4.4 0.3 1.86 2.1 3.1
  have entry 3.102 with dmx: a1ddc663070089
  have entry 3.103 with dmx: 058226264f64d6
rcvd WANT vector=[ 2.1 3.102 4.4 0.3 1.86 ]
mk_want offs=1, vector=[0.3 1.86 2.1 3.104
```

- highly compressed WANT, CHNK vectors:
  - "local names" (1B) for crypto identities (32B)
  - established through a local grow-only-set CRDT
  - also uses 120B packets and one CLAIM msg type

tinySSB frontend vs backend



A design challenge: how to keep the "cooked state" of the frontend in sync with the "raw state" in the backend

- Android supports a "browser-in-your-app"
  -> frontend (in JS+HTML) for most of the app logic
- backend (in Kotlin) for the crypto, HW interfaces, raw append-only logs, replication protocol
- iOS port of the backend is ongoing
  (considering using the Socket Supply runtime)

tinySSB
frugal data ...
and compression

Table of Contents

Offline-first means: we have time

-> invitation to give up expectation of

"let's do a facetime"

- starts with images, not advisable:
  - 100kB already too much (=900 tinySSB packets)

- hence limit to **frugal content types**:
  - text, geo-tags, brief acks

- sketch: uses **vector graphics**
  - svg-like encoding, typical sketch has 1.5kBytes

- **voice**: can be done! "codec2" **compression** library
  - at 1300bps-level → 15 sec voice = 2.7 kBytes
  - open source, free-as-in-speech codec

    `https://www.rowetel.com/?page_id=452`

tinySSB Roadmap

Table of Contents

tSSB a prototype, not for daily use:

- no database, hence unable to "select" your data workset – all is in memory
- no dynamic app loading: code of *all* apps is in memory

- Technological **roadmap**:
  - LoRa, shortwaves "end-to-end" *almost there*
  - iOS support *partly works (Socket Supply library)*
  - app store *works.. but not integrated yet*
  - private key on Ubikey dongle + NFC coupling *works..*
  - security bubbles and dyn. groups *works..*
  - replication via websocket *works..*
  - replication via git(hub) and USB sticks *new*
  - log pruning/meta feeds *need more experience*
  - support for Lokens *just concept*

... and your desirable feature is?

leave a comment

https://github.com/ssbc/tinySSB

dWeb camp, Aug 2024