

Secure Scuttlebutt: An Identity-Centric Protocol for Subjective and Decentralized Applications

Dominic Tarr (SSB, New Zealand)

Erick Lavoie (McGill University, Montreal)

Aljoscha Meyer (TU Berlin, Germany)

Christian Tschudin (U of Basel, Switzerland)

September 25, 2019

ACM ICN
2019 Macau



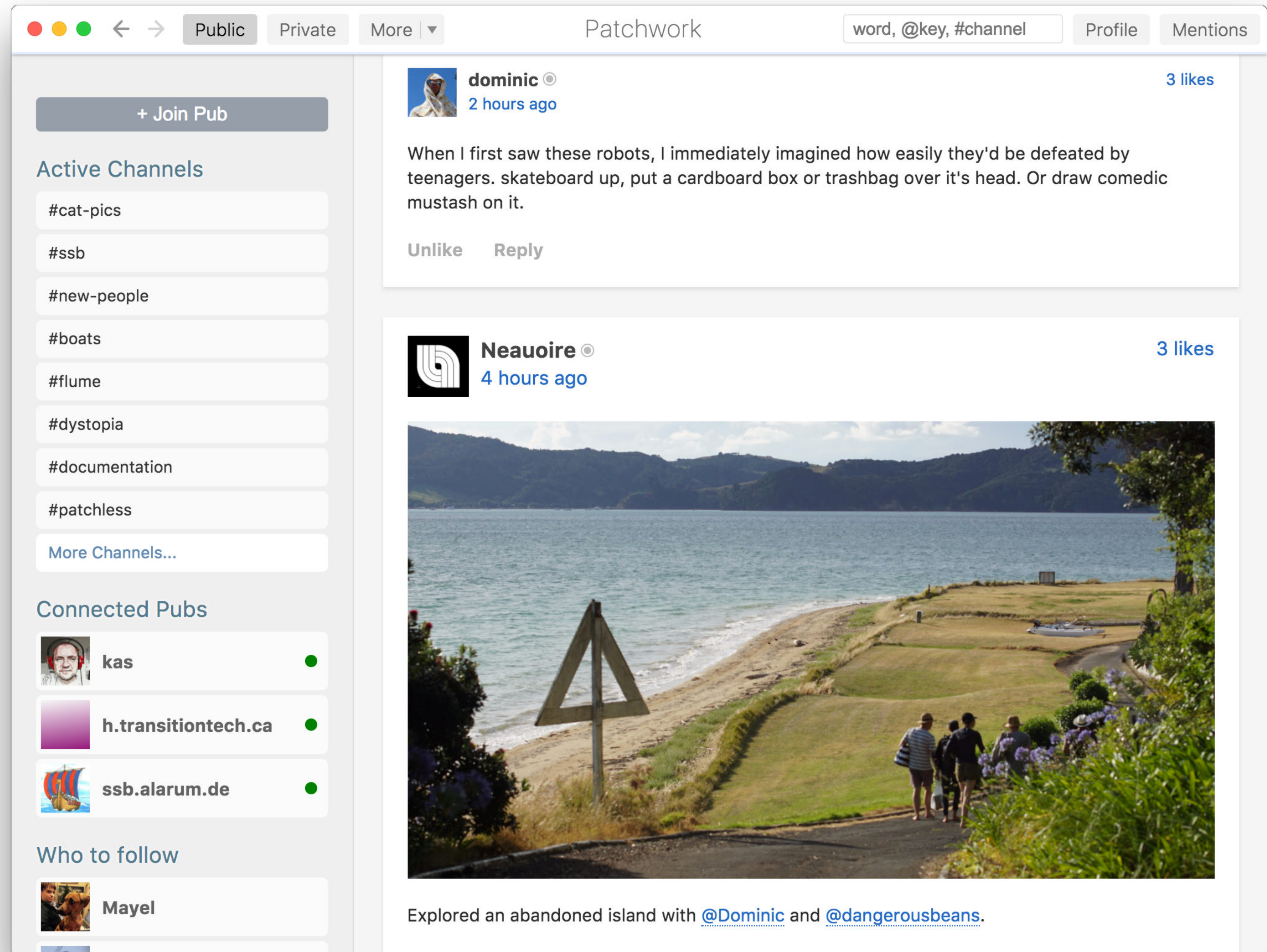
Secure Scuttlebutt: probably the most radical decentralized protocol

- no central authority, data structure, consensus or trust infrastructure
- one source of truth: per-ID cryptographically secured append-only logs
- a log consists of signed events
- Hey, it's named data! The name of a single event is "**id:seqno**", events also have an intrinsic name (hash)
- event content can be encrypted, no leaking of recipient ID
- "distributed app thinking": events for Linda-like "buffered broadcast"
- created in 2014, currently 10'000 users, ca 20-30 core developers
- clients for Win, Mac, Linux, Android, iOS soon



SSB has rich social media clients

- Javascript/Electron
- Several other clients and libraries
 - Rust, C, Python, Kotlin (but not full-fledged yet)
- NaCl crypto library, JS and heavy DB ops



Overview

1. SSB as a technology:
motivation, log replication, subjective reader concept
2. Some example applications
3. The human dimension:
log replication along the social graph, trust and the onboarding problem
4. A comparison of SSB and NDN
5. Challenges, future work

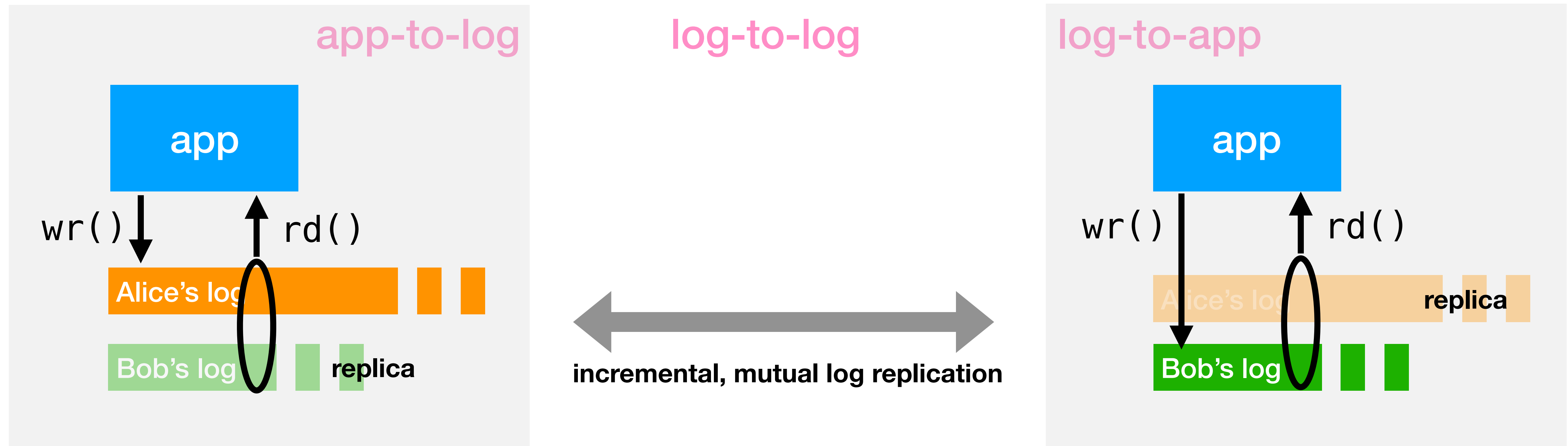
1a) SSB tech: motivation

SSB started with a wish list, empowering the human user:

- no ads, no irrelevant chatter (dial down buddies), yet have social media
- *must* work offline (offline-first)
- *secured* with user-controlled trust (no Cert Authority, please)
- *permissionless* naming (no Naming Authority, please)
- not dependent on consensus (beyond common packet format)
- SSB name: water cooler is where news travels
on ship: water cooler is called scuttlebutt, which also is lingo for gossip



1 b) SSB tech: log replication

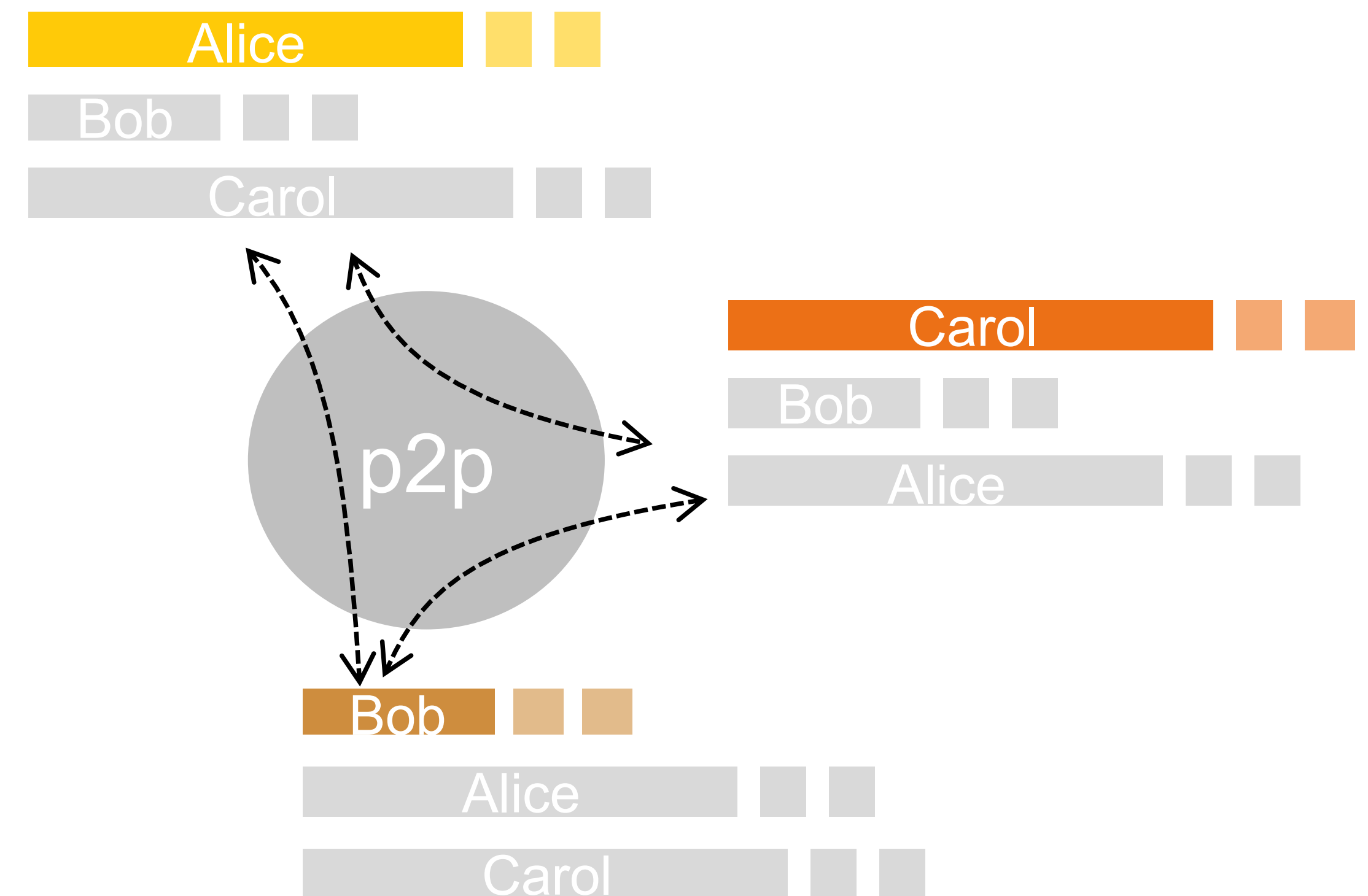


- Application (=frontend) only works on local log replicas
- Log updates are automatically replicated (backend), somehow and at some time

1 c) SSB tech: subjective reader

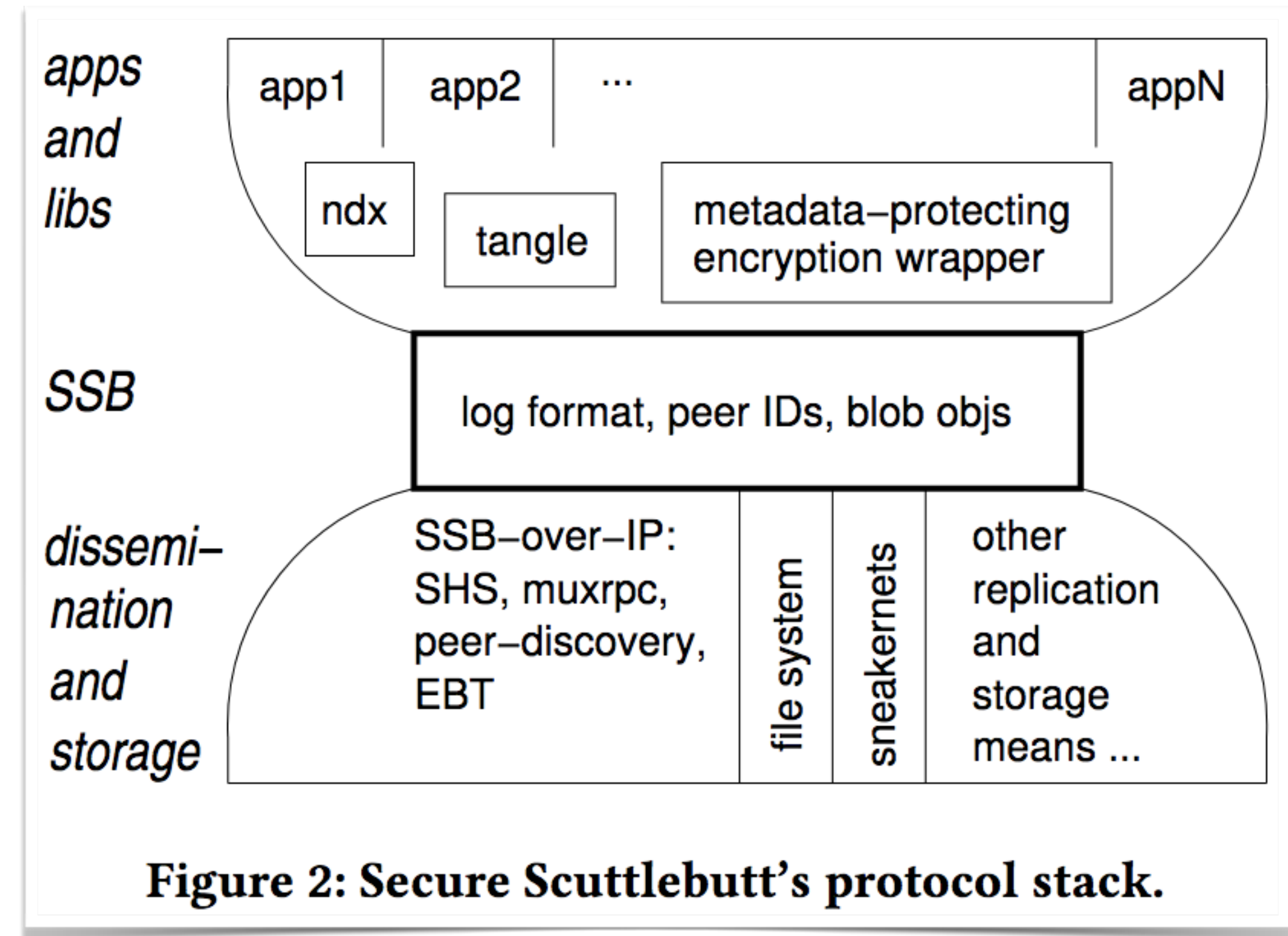
Ground truth are the individual
append-only logs

- Log = hash-chained signed event list
—> causality-preserving
- Applications locally compute their state from events in the available logs:
 - reconstruct e.g. chat dialogue from events in multiple logs (a ‘map-reduce task’)
- “**subjective reader**” (the ‘map’ step):
 - app/user to choose which logs to consider
 - app/user could also lack the replica of some logs that others see



1 d) SSB tech: the protocol

- The thin waist of SSB:
 - ED25519 public keys as IDs
 - log event: up to 4KB, including ID
 - events MUST be signed, chained
= reliable in-order delivery
- Current SSB: an IP overlay
 - epidemic broadcast tree (EBT) routing
 - peering uses a ‘secure handshake protocol’ between trusted nodes
- Plus a separate “blob” replication protocol (a hack, more NDNish client/server)



2a) SSB app: user directory

Q: *How to have a SSB-wide user directory ..
without consensus or central data aggregation?*

A: Be subjective like humans are

- SSB app = common content format
SSB app state = some reduction of all such events
- The “about” app:
 - people assign names to IDs, publish in their log
 - for given about target and log, remember the last
 - from these per-log assignments, pick as you like
(self-assigned first, if absent then name given by
a friend, else name given by a random person etc)

In other words: a directory of display names where IDs are the underlying unique names

```
content = { // by Alice
  type = "about",
  author = "@Ah88Hb.ed25519",
  about = "@B2gg34.ed25519",
  name = "Bob"
}
```

```
content = { // by Alice
  type = "about",
  author = "@Ah88Hb.ed25519",
  about = "@B2gg34.ed25519",
  name = "Bob the guy"
}
```

```
content = { // by Bob
  type = "about",
  author = "@B2gg34.ed25519",
  about = "@B2gg34.ed25519",
  name = "cool Bob"
}
```

```
content = { // by Carl
  type = "about",
  author = "@Ca92c2.ed25519",
  about = "@B2gg34.ed25519",
  name = "Bob the enemy"
}
```

2b) SSB app: Chat, Git, Chess,...

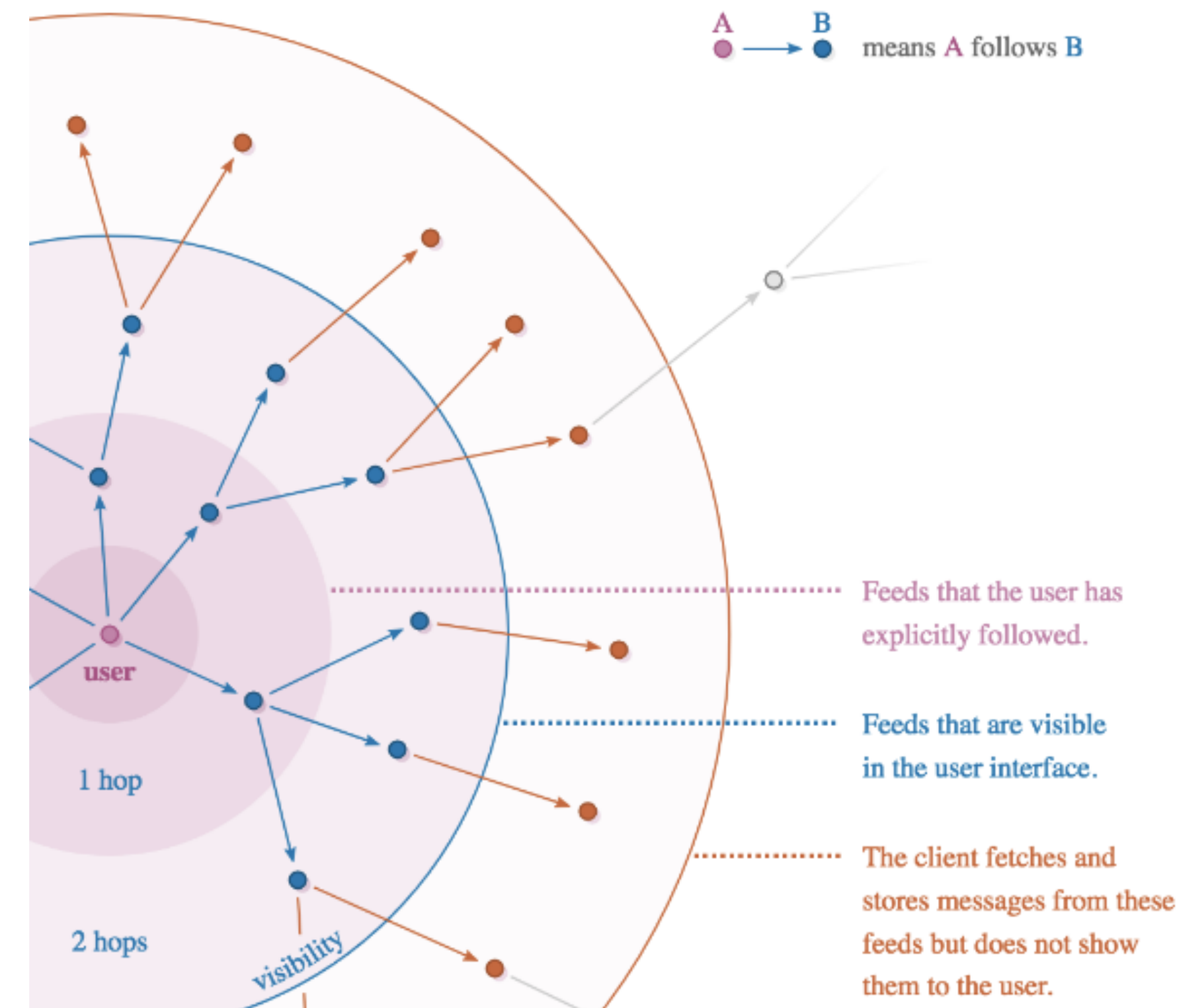
General comment: think CRDTs (conflict-free replicated data types)

- **Chat** uses event records with three fields: `starting-event`, `in-reply-to`, `text` (plus the usual per-event author and sequence number fields)
 - > chat thread = directed acyclic graph with same `starting-event`
 - > state reduction = your subjective linearisation of that DAG
- **Git** naturally is a DAG: just map commits to SSB events
 - > except the singleton decision on “head of the master branch”, where in SSB conflicts are solved by humans instead of central GitHub
- **Chess**: initial mutual “let’s start a game” events, then publish your moves, simply ignore moves by others or moves from other game instances

3a) Human factor: social graph

SSB = “Internet of people” i.e., their log

- self-declared follow events
(mutual follow == “friend”)
- replicate content along the social graph
- When peering, a client subscribes to:
 - it's own log
 - the logs of followed IDs
 - the logs of IDs followed by people whom they follow (friends-of-friends)
- There are also unfollow and block events.



3b) Human factor: trust and onboarding

SSB onboarding as an “existential” experience

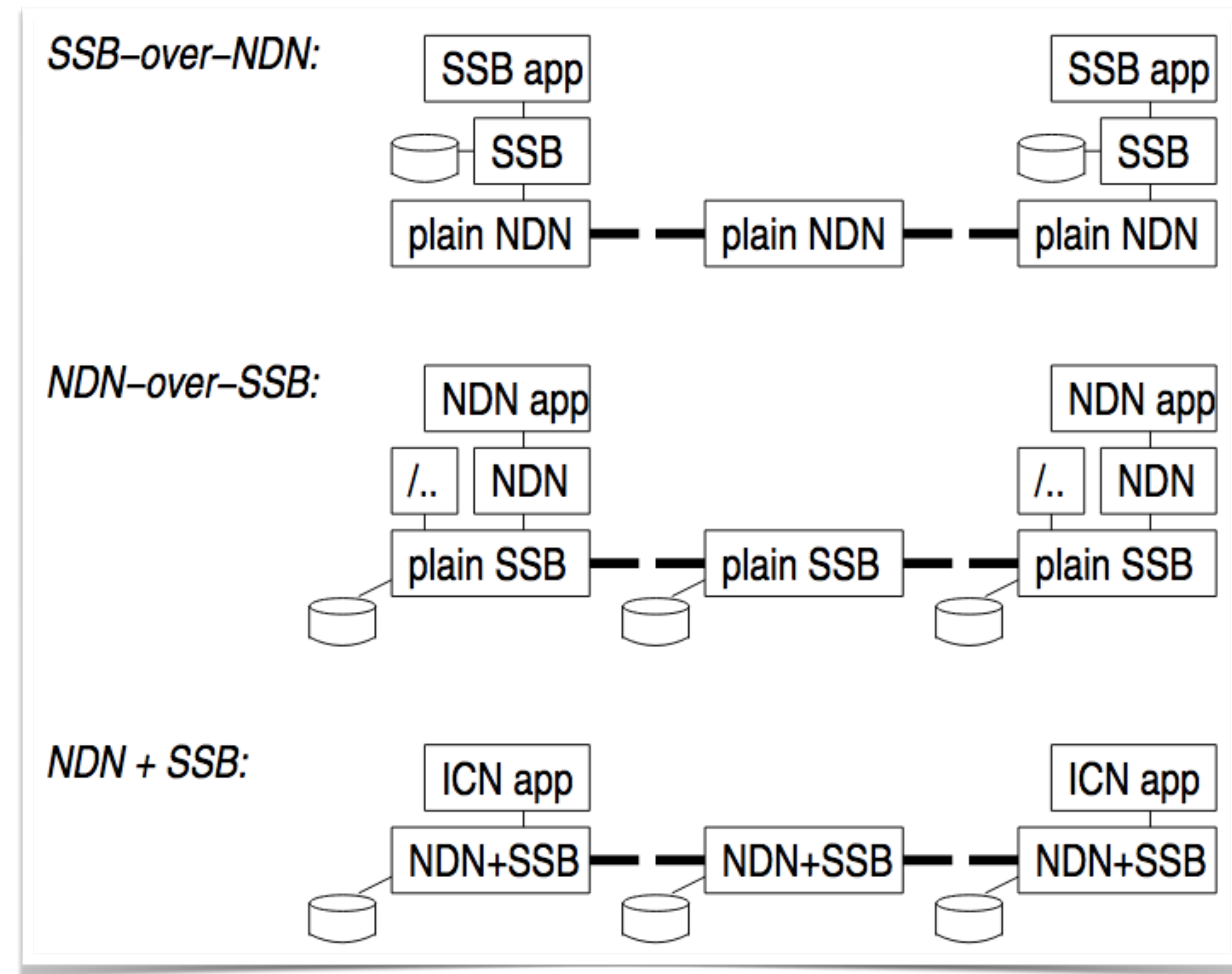
HELP - nobody is following me, hence nobody will see what I post !

- Newcomers MUST be introduced
Some hacks exist: automated “invite codes” by SSB relay nodes
- Receiver-driven mindset, leads to very strict filtering what events you see
- “Web of trust” instead Cert Authority
“follow” declarations replace PGP’s signing parties

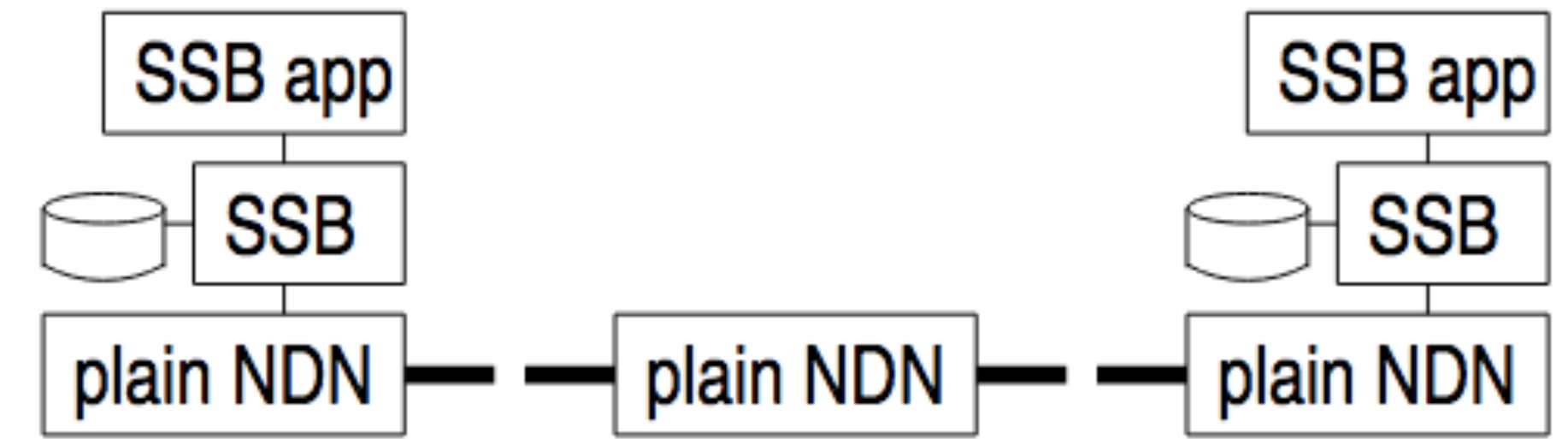
4) Comparing SSB with NDN

Different planets?

- SSB:
 - no consumers, only producers
 - no pull: new content is pushed
 - no hierarchical name space
 - no mutable name binding
 - strict data structure rules (log)
- In the paper:
three attempts to “emulate” or layer



4a) SSB over NDN

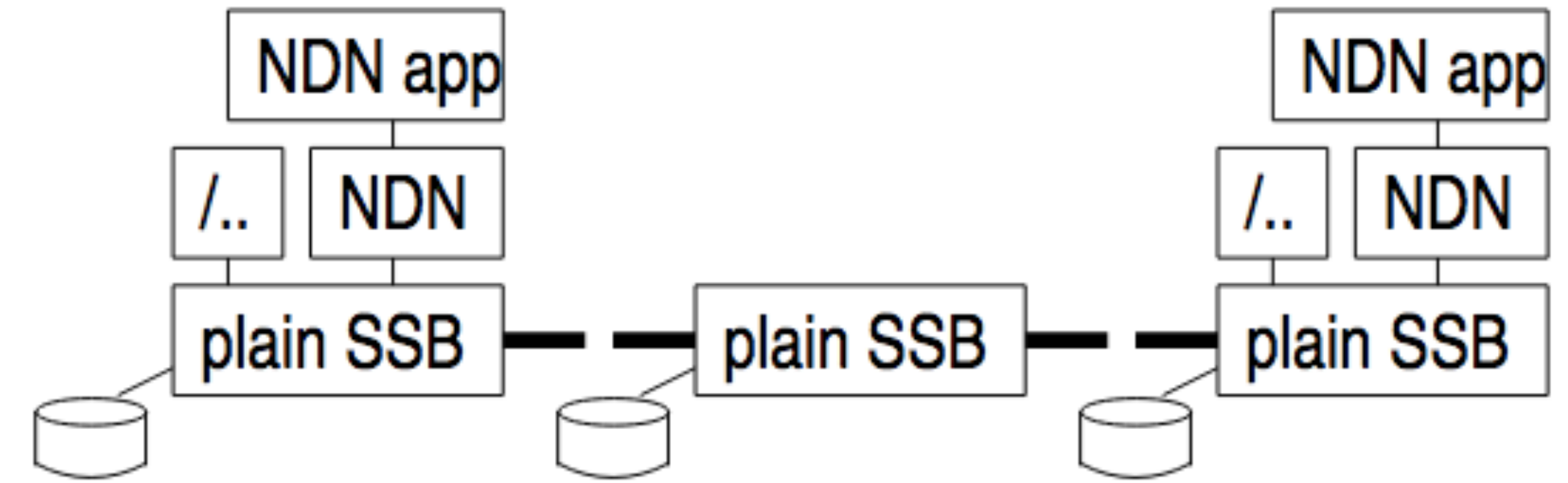


Select insights (about the pain)

- Emulate SSB's push with either polling, or long-lived-interest (reverse pull, parallel pre-registration for efficiency)
- (not in the paper): NDN offers “read” but has unclear “write” operation, linked to “SSB only has producers” property:
 - > perhaps use NDN's mobile producer support?

Still does not match SSB's replication property:
in SSB, your followers are your log's backup

4b) NDN over SSB

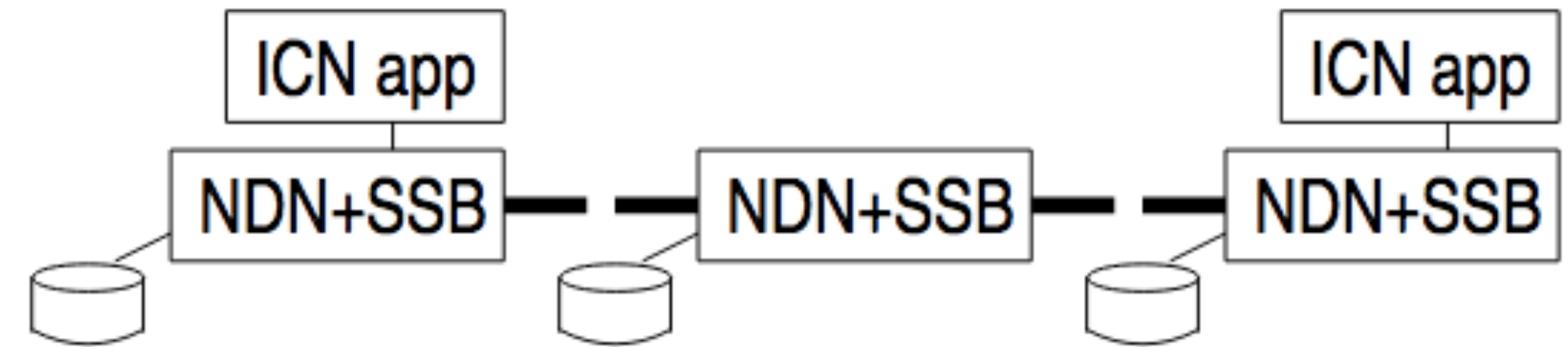


“Pull”: read in your local replicas
or introduce per-data structure IDs, then “subscribe at runtime to a file”

But have to emulate two other NDN aspects: global namespace, forw+trust

- Namespace: SSB must implement a “NDN name authority” ID, plus app-level protocol how to register a NDN name (but how does the authority “follow” all potential requestors?)
- NDN follows IP mindset: “free forwarding”, where free=ISP ecosystem SSB would need to introduce “contractual friends”, happy to replicate your “real friends”’s content

4c) NDN + SSB



Combining the two approaches?

- NDN as “data access” pull protocol, best effort:
 - get random content out of a hierarchical name space
 - or pull single events from logs (out of a flat ID space)
 - optional caching
- SSB integrated as a push service, event streams:
 - a natively supported pub/sub service
 - beyond long-lived interests: “controlled push”
 - reliable, only for signed event chains, some kind of TCP receive window
 - caching mandatory for logs

5) SSB challenges, future work

- Is SSB a risk for “people at risk”?
 - pseudonymous, exposes IDs
 - events are immutable and not refutableOnly for the privileged “who have nothing to hide”? (Google’s Eric Schmidt)
- Scaling concerns from a caring community: What will be the (economic) force that turns SSB into a disruption, could abandon its goals?
- Still global singletons in the system design: e.g. “type” field of events
- Evolution of SSB:
 - binary encoding schema / off-chain content (can delete payload, but keep the event trace) / event-level tangle support / scalable encrypted chat groups / log compaction / log life cycle management / alignment with DAT..

6) Conclusions

SSB's tech choices seem contrarian, but identified a very convincing spot in ICN' solution space

- value system - disintermediation, plurality (SSB a “neutral” infrastructure)
- push - vs pull
- event-source integrity - way better than signing anything
- trustful - instead of trustless crypto currencies: info bubbles are ok
- consensus-avoidance - yet comes with powerful low-level sync

When push comes to shove - does *not* need a network - sneakernet ok