APM 2026 SPECIAL EDITION - TURIN, ITALY

Nearbytes

An Asynchronous Local-First Architecture for Post-Cloud Computing

Vincenzo Ciancia

Joint work with Alexander Kurz • Emilio Tuosto • Gabe Giancarlo Roberto Guanciale • Daniele Strollo

Formal Methods and Tools Lab

Consiglio Nazionale delle Ricerche

Formal Methods and Tools (FMT)

— SYSTEM MODEL

Nearbytes is an asynchronous replayable log

State is not stored directly. It is reconstructed by replaying signed events over immutable encrypted data.

Asynchronous

Peers can write now and synchronize later.

Replayable

Files, chat, and profiles are derived by replay.

Data-carried

Identity, integrity, and history are carried by the data.

Storage keeps bytes; replay gives them meaning.

— DISTRIBUTED CONSEQUENCES

Shared facts give shared state

Distributed

No coordinator is needed to validate or replay.

Eventual consistency

Same validated facts imply the same replayed state.

Conflict-visible

Facts are appended, not overwritten; conflicts remain in the log.

Synchronization is just exchange of missing facts, followed by replay.

THE ORDINARY COMPUTER QUESTION

Where is that file?

This machine. External disk. Other computer. Phone.
Modern storage still makes presence feel uncertain.

NO CORRECT ANSWER

All choices fail.

POSSIBLE ANSWERS

- this Mac
- USB drive
- other PC
- phone

LOCATION IS NOT IDENTITY

STORAGE BECAME GUESSWORK

USERS STILL HUNT

THE FOLLOW-UP

Which copy is current?

Desktop copy. Shared copy. Downloaded copy.
Backup copy.
The same data keeps reappearing as competing presents.

NO CORRECT ANSWER

All choices fail.

COMPETING TRUTHS

- edited here
- uploaded there
- downloaded again
- renamed final

VERSIONING BY PANIC

SYNC AS RITUAL

TRUTH SHOULD NOT FORK

THE SHARING CIRCUS

How do I share this folder now?

Nearby or across the globe, the menu is still absurdly familiar.
Email is too small. Chat is too slow. Bluetooth fails. Cable how, exactly?

NO CORRECT ANSWER

All choices fail.

USUAL WORKAROUNDS

- email
- chat upload
- Bluetooth
- cable transfer

SHARING BECAME CHANNEL ROULETTE

SIZE BREAKS WORKFLOWS

PROXIMITY SHOULD HELP

THE CLOUD DETOUR

Which cloud did I use?

Pick a service, discover they use another one,
pay for more space, lose deduplication,
then remember you sent the file on WhatsApp
because the platforms did not line up.

NO CORRECT ANSWER

All choices fail.

LAST KNOWN LOCATIONS

- OneDrive
- Dropbox
- Mega
- WhatsApp

SERVICES PARTITION PEOPLE

STORAGE GETS BILLED TWICE

THE FILE STILL DISAPPEARS

THE HIDDEN TAX

Why is backup a separate job?

Remember it. Schedule it. Plug something in.

NO CORRECT ANSWER

All choices fail.

WHAT USERS ARE TOLD

- be disciplined
- buy storage
- trust the vendor
- hope it restores

PRESERVATION SHOULD BE AMBIENT

REDUNDANCY SHOULD BE SOCIAL

BACKUP SHOULD DISAPPEAR

THE MOST PERVERSE OUTCOME

Why is local collaboration harder than the cloud?

The people are in the same room,

NO CORRECT ANSWER

All choices fail.

REQUIRED BEFORE WORKING

- same service
- same login
- same terms
- same bill

COORDINATION IS OUTSOURCED

PRESENCE IS IGNORED

TRUST IS DISPLACED

THE NEARBYTES CLAIM

We make these questions obsolete.

We do it through four core primitives and a communication stack built above them.

Four primitive behaviours.

PRIMITIVE I

Immutable encrypted blocks

PRIMITIVE II

Secret-derived authority

PRIMITIVE III

Signed, replayable event logs

PRIMITIVE IV

Transmission-agnostic synchronization

— THE STATUS QUO

We traded control for convenience

In cloud architectures, naming, storage, and communication are concentrated in the same service.

What the service gives you

+ Current state from any device
+ Names resolve instantly
+ Sharing is a URL away

What the service takes

x It goes down -> you wait
x It changes terms -> you adapt
x It shuts down -> your data is orphaned

Convenience and authority sit in the same place.

— THE ALTERNATIVE

Peer-to-peer: process-dependent

The obvious answer is to remove the central service.
But traditional P2P ties state to *running processes*.

Node Online -> **State Exists** - **Node Offline** -> **State Gone**

Think of a BitTorrent swarm with no seeds, or an IPFS pin nobody keeps.
The protocol works, but the data is only as alive as the last peer hosting it.

Decentralized, but fragile.

What if data outlasts processes?

Cloud gives you state but takes away autonomy.
P2P gives you autonomy but state depends on who is online.

If the essential properties - identity, integrity, history -
are carried by the *data encoding itself*,
the system can persist wherever that data persists.

— STRUCTURE - SEPARATION

Nearbytes separates the concerns

STORAGE LAYER

Immutable encrypted blocks

Dumb storage keeps opaque bytes, addressed by content hash.

HISTORY LAYER

Signed, replayable event logs

Authenticated records reference encrypted payloads and can be replayed into current readings.

NAMING LAYER

Secret-derived authority

A remembered anchor deterministically yields the same authority again.

TRANSPORT LAYER

Transmission-agnostic synchronization

The same material can move through any medium without changing its meaning.

EFFECT

Replay gives mutability

Nothing is rewritten in place. New facts shift the current reading.

— PROTOCOL WALKTHROUGH

A chat message enters the protocol

SENDER JOINS HUB

Alice

SHARED SECRET

Knows the hub secret.

1

LOCAL CREATION

Compose and encrypt

"Meeting at 6?"

2

IDENTITY

Derive keypair from secret

secret -> K_{priv}, K_{pub}

3

CONTENT ADDRESS

Hash the ciphertext

H(C) = a7f3...e91b

4

APPEND-ONLY HISTORY

Sign and append to K_{pub}.log

K_{pub}.log <- chat - msg/184 - a7f3...e91b - sig_{Kpriv}(...)

RECIPIENT JOINS SAME HUB

Bob

SAME SHARED SECRET

8

REMOTE RECONSTRUCTION

Read the message

7

MESSAGE RECOVERY

Decrypt and replay chat state

6

CONTENT FETCH

Fetch block by hash

get(a7f3...e91b)

5

REMOTE VERIFICATION

Verify log name and signature

A hub is one collaboration unit, many application channels

HUB CONTAINER

Hub H

- seed -> deterministic authority
- one append-only event history
- many named channels

Open the same seed again and the same collaboration unit comes back.

PROJECTS INTO

CHANNEL

Files

Names, paths, and references to encrypted payloads.

CHANNEL

Chat

Messages are events in the same hub, not another silo.

CHANNEL

Profile

Identity is another reading over the same shared history.

CHANNEL

Future apps

Calendars, boards, or workflows can live in the same hub too.

A hub is both a **namespace** and a **collaboration unit**: several applications, one history, one place to meet.

— PRIMITIVE 1 - CONTENT-BASED ADDRESSING

Immutable Encrypted Blocks

Identity is location-free. The same object remains the same object wherever it is copied. Integrity is intrinsic.

— PRIMITIVE 2 - NAMING BY KEYPAIR DERIVATION

Secret -> Hub Identity

ORIGIN

Secret

passphrase

KDF

ECDSA P-256 keypair

Private key

SIGN

Public key

VERIFY

PUBLIC ADDRESS

STABLE NAME

Hub Identity

stable public address

secret -> keypair -> hub ID

Recovery starts with the secret, not with an account server.

Hash-Named Events, Public-Key Logs

1. Event = signed fact

A payload is signed and becomes an immutable event.

event hash = H(event)

2. Events are hash-addressed

The event itself is named after its own hash, so the identifier is self-verifying.

4fd2...9ac1.json

3. Logs are public-key-addressed

Each event is appended to a log named after the writer's public key.

log/02ab...77ef/4fd2...9ac1.json

Anyone can recompute the event hash and verify the signature against the public-key-named log: content, author, and placement are all checkable.

This is the logging principle: hashes name events, public keys name logs, and replay uses only verifiable records.

— PRIMITIVE 4 - SYNCHRONIZATION

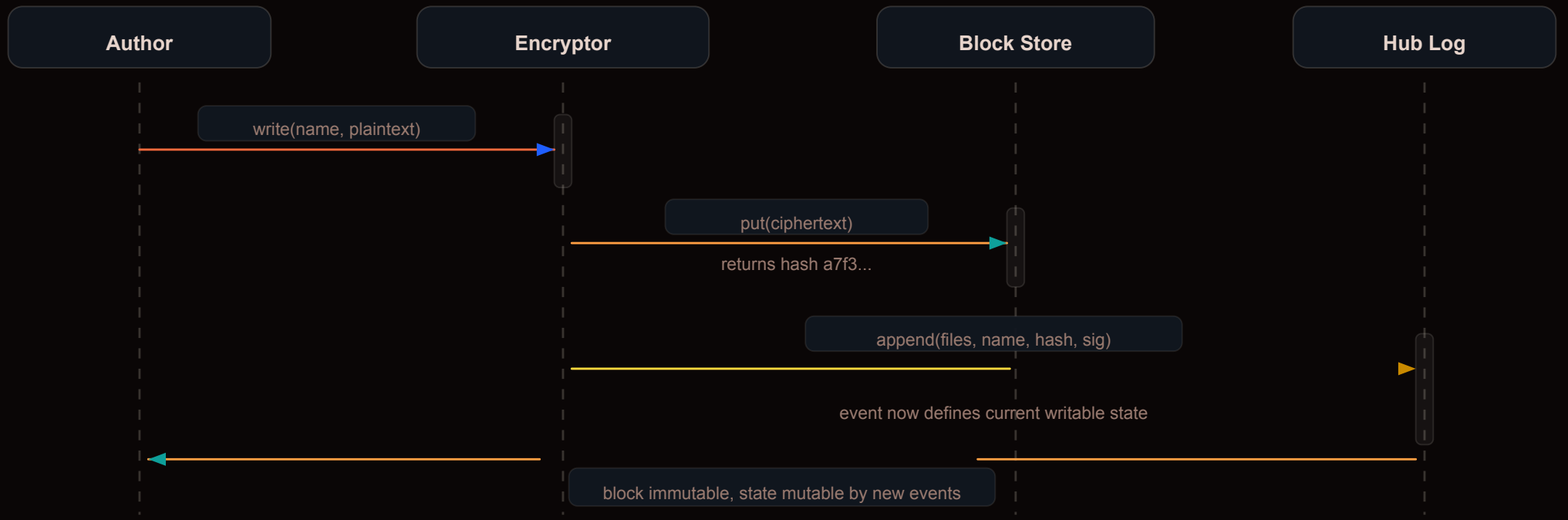
CONVERGENCE

Same facts, same state

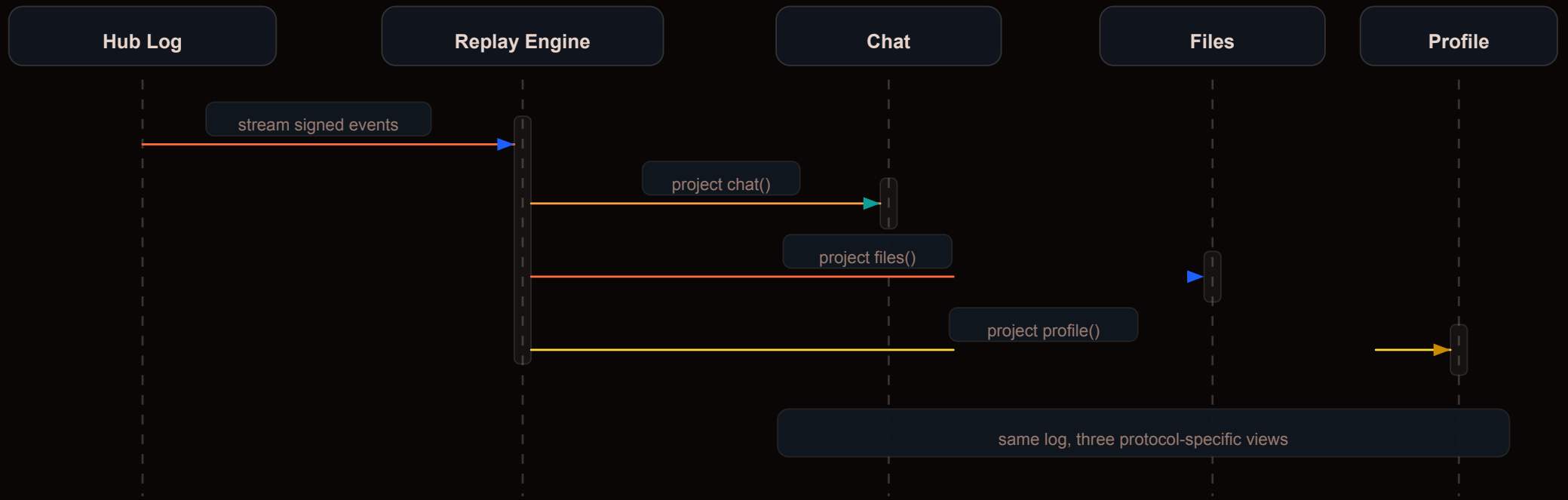
17 - 18 - 19 -> one replay

Reconciliation is not negotiation. It is the recovery of the same fact set.

Immutable content first, mutable state by appended event



Chat & Profiles as replay



The hub carries conversations and profiles, not just files. Same log,

Transport agnosticism

Correctness is a property of *encoded data and signatures*,
not of any particular transport.

WAN

relay or cloud transport

Remote services carry the same bytes, but they do not define the semantics.

USB

removable media

Manual copy over disks and drives preserves exactly the same signed material.

LAN

local network

Nearby peers can exchange the same facts directly, without a privileged channel.

SAME ENCODED TRUTH hashes identify content, signatures validate authority, and the carrier stays incidental

Transport changes latency and convenience, not the meaning of the data.

— CONTEXT - LINEAGE

Related systems

2014	2018	2021	2023	2025
IPFS	Hypercore	ERIS	Willow	Nearbytes
content addressing	append-only logs	encrypted substrate	collaborative spaces	secret-derived hubs

Nearbytes is a synthesis: addressable blocks, append-only history, encrypted persistence, and socially chosen replication.

WHAT MUST SURVIVE

Nearbytes is designed to survive itself.

If today's programs change or disappear, the hub still carries identity, integrity, and history in a form from which new software can start again.

identity survives

history survives

meaning survives

Resilience follows from redundancy, not uptime. Continuity follows from data, not from any one application.

This is not only a technical property, but also a democratic one.

WORK IN PROGRESS

Global choreographies inside Nearbytes

We want protocol logic inside the app to be described globally, not hidden in ad hoc local code.

Global view

Describe who may say what, to whom, and in which order.

Local projection

Derive participant behaviour from the choreography rather than hand-coding each side separately.

Nearbytes use

Sharing, approval flows, collaborative editing, and agent interaction can become explicit protocol objects.

WORK IN PROGRESS

Local AI agents confined to Nearbytes volumes

Locality

The agent runs close to the files, keys, and policies of the user.

Scoped visibility

The agent sees only the volumes and records the user mounts into the task.

Auditable action

Outputs become proposals or appended records in the same replayable log.

— LOOKING AHEAD - OPENINGS

Open questions

01

Theory

What is the correct security model here, and which concrete threats should Nearbytes be designed to resist?

02

Human-Computer Interaction

How do we make a jungle of data, channels, secrets, and identities feel simple, legible, and beautiful?

03

Implementation

The architecture is clear enough to build now, but the system still needs to be fully implemented and tested in practice.

Formalisation, analysis, and implementation

Nearbytes needs formal models, adversarial analysis, and open implementations that can be evaluated in practice.

THEORY

Semantics and types

Asynchrony, replay, convergence, authority, and meaning.

ANALYSIS

Security and cryptography

Threat models, proofs, attacks, protocol scrutiny, and hardening.

SYSTEMS

Implementations and tools

Reference implementations, benchmarks, interoperability, and UX.

The aim is an open system with both theoretical clarity and practical value.

Thank you

Nearbytes: An Asynchronous Local-First Architecture for Post-Cloud Computing

github.com/nearbytes/nearbytes-app

github.com/nearbytes/whitepaper

Vincenzo Ciancia

Joint work with A. Kurz • E. Tuosto • G. Giancarlo • R. Guanciale • D. Strollo