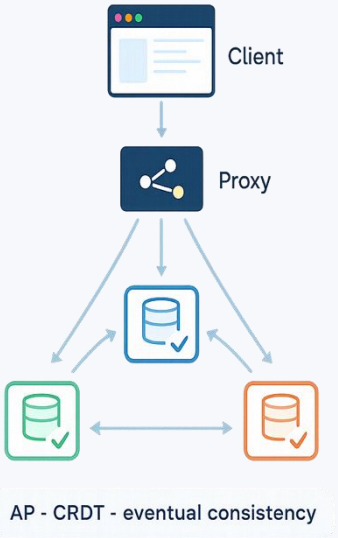
Distributed Voting and Polling System

Modellazione del Sistema e scelte progettuali.

Presentazione progetto · DS 2025/2026

Roberto Pisu



AP / BASE **CRDT** **Anti-entropy**

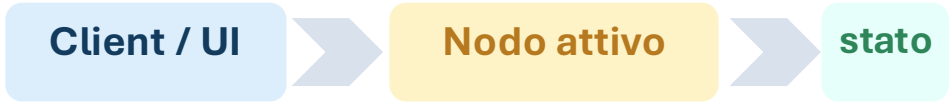
Problema affrontato e obiettivi di sistema

Un servizio di polling distribuito che resta operativo anche con guasti e disconnessioni

Perché distribuire?

- accettare voti e letture da qualunque nodo raggiungibile
- tollerare crash, partizioni temporanee e perdite di messaggi
- replicare lo stato dei sondaggi senza coordinamento centrale
- recuperare rapidamente dopo un riavvio

Modello funzionale



- creazione implicita del poll al primo voto valido
- vote submission con conferma locale immediata
- poll listing e result retrieval da replica locale
- propagazione successiva delle modifiche agli altri nodi

servizio long-lived | active-active | REST + JSON

Modellazione del sistema

Componenti, connettori, dati e stile architetturale

Componenti

- reverse proxy
- repliche del servizio
- modulo CRDT
- WAL / checkpoint
- failure detector

Connectors

- HTTP request/response
- JSON serialization
- heartbeat
- anti-entropy periodica
- accesso al file system locale

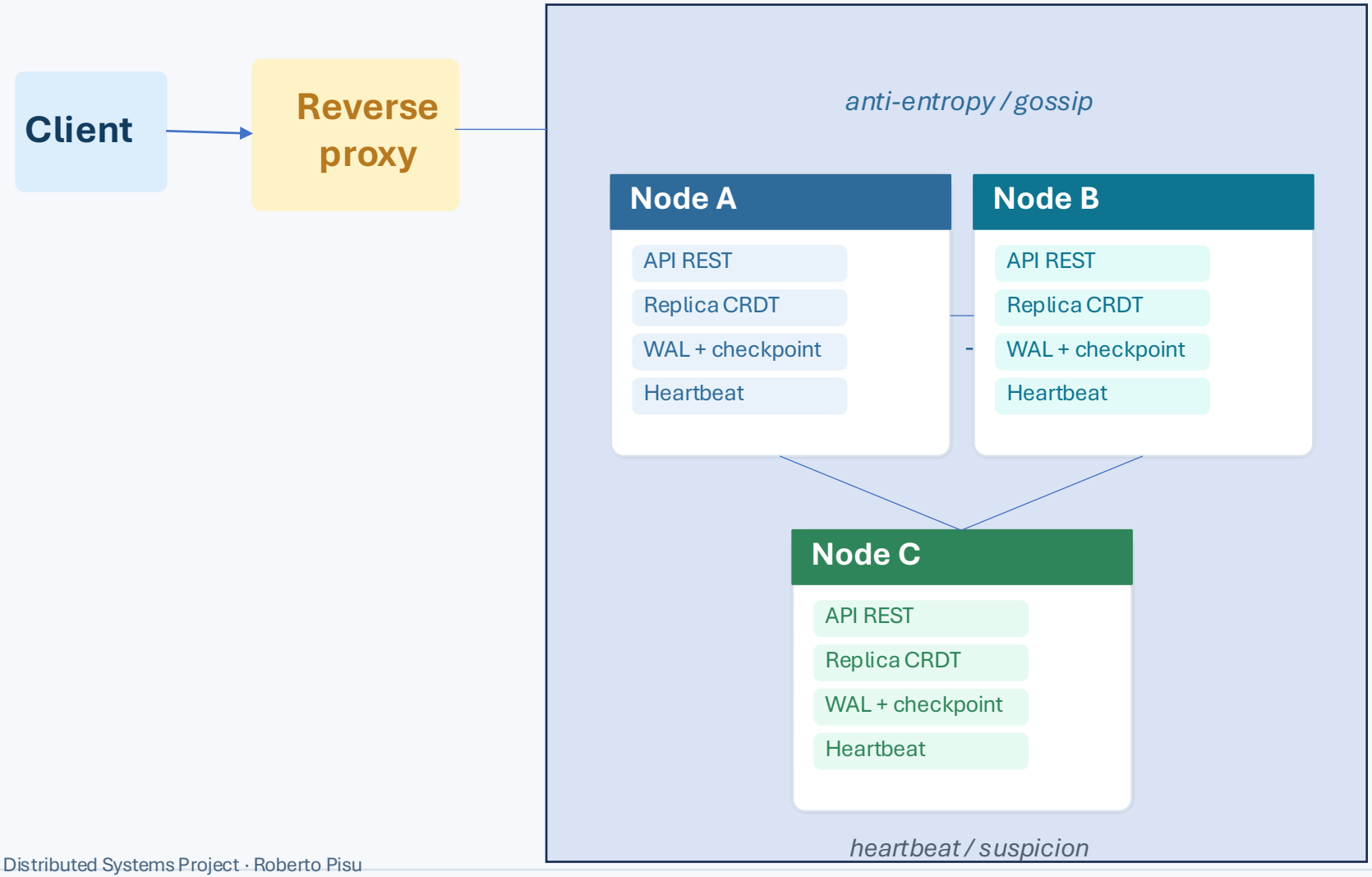
Data

- polls e vote counters
- update CRDT
- log append-only
- snapshot di stato
- membership/config

Stile architetturale principale: data-centered (stato replicato dei poll) con caratteristiche **event-based** (propagazione asincrona degli update). A livello di deployment il sistema è **peer-to-peer active-active** dietro reverse proxy.

Architettura run-time

Repliche attive, sincronizzazione periodica e accesso trasparente



Write path: disponibilità prima della consistenza forte.

La scelta del progetto è AP / BASE con eventual consistency.

Write path

1. richiesta a un nodo raggiungibile
2. append nel WAL locale
3. apply alla replica CRDT
4. risposta immediata al client
5. propagazione asincrona ai peer

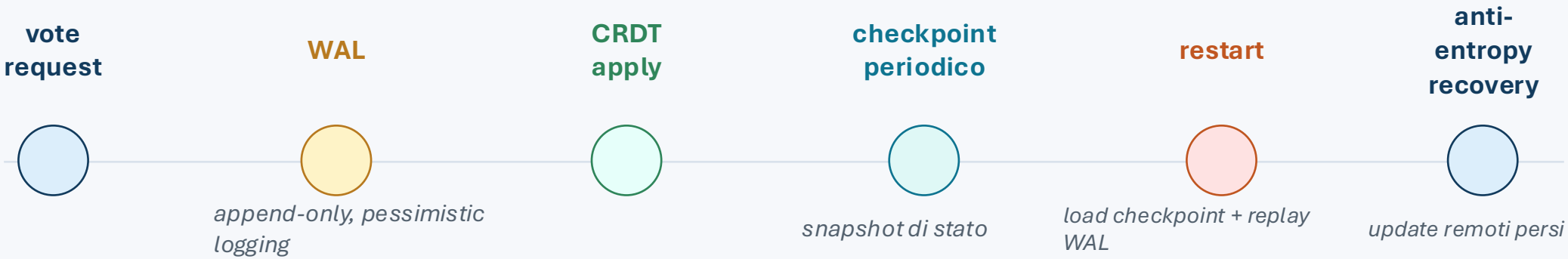
Che cosa implica

- nessuna consistenza forte: letture da repliche diverse possono divergere temporaneamente
- CRDT + merge deterministico: l'ordine degli update non è critico
- modello BASE: Basically Available, Soft state, Eventual consistency
- sotto partizione il sistema privilegia availability rispetto a consistency

AP BASE Eventual consistency

Recovery: locale con WAL/checkpoint, distribuita con anti -entropy

Logging, checkpointing e riallineamento dopo crash o disconnessioni



Cosa guadagno

- durabilità locale degli update confermati
- rollback recovery rapido dopo crash
- heartbeat per failure suspicion

Limite consapevole

- il WAL protegge lo stato locale, non sostituisce il riallineamento distribuito
- gli update remoti persi rientrano dopo via anti-entropy

Scelte progettuali e alternative scartate

Perché CRDT + anti-entropy invece di consenso forte

Opzione	Vantaggi	Perché non scelta / esito
Consenso forte / SMR	ordine globale, consistenza forte	latenza più alta, minore disponibilità, complessità inutile per un polling system
CRDT + replica asincrona	update locali, merge naturale, alta disponibilità	scelta adottata: accetta divergenze temporanee
Logical / vector clocks	tracciano causalità e concorrenza	non necessari: il modello CRDT rende l'ordine non critico
Code mobility	sposta computazione sui nodi	non usata: il progetto muove dati e stato, non codice