Gossip-Based Service Discovery e Failure Detection

Leonardo Polidori (0357314)

Indice

Introduzione: Il Trade-off della Service Discovery

La scoperta dei servizi in un sistema distribuito richiede un equilibrio delicato tra efficienza e resilienza.

Registry Centralizzati

Coerenza semplificata, ma introducono un Single Point Of Failure (SPOF) e colli di bottiglia.

Flooding

Copertura deterministica e bassa latenza. Il traffico di rete cresce in modo esplosivo con la dimensione della rete.

Protocolli Epidemici (Gossip)

*Accettano **copertura probabilistica** in cambio di **costo per nodo quasi costante** (fanout fisso) e maggiore resilienza a perdita/churn.*

Obiettivi di Progetto

Il progetto SDCC (Service Discovery Control Plane) mira a stabilire un approccio bimodale innovativo per la gestione dei servizi distribuiti, concentrandosi su resilienza ed efficienza.

Eliminare SPoF

Rimuovere ogni Single Point of Failure (SPoF) intrinseco ai registry centralizzati, garantendo una maggiore continuità operativa.

Scalabilità Costante

Assicurare che l'overhead per nodo rimanga costante, deduplicazione e regole di arresto mantengono il traffico sotto controllo

Robustezza a Guasti

Offrire una solida tolleranza ai guasti e gestire il churn (l'entrata e l'uscita dinamica dei nodi) con minima interruzione dei servizi.

Latenza e Costo

Trovare un equilibrio ottimale tra bassa latenza nella scoperta dei servizi e un costo di rete controllato.

Fondamenti: Gossip e Rumor Spreading

I protocolli epidemici sono maggiormente usati per la diffusione di stato nei sistemi distribuiti.

Epidemic Gossiping

Scambio periodico di piccoli frammenti di stato time driven con k vicini scelti casualmente (k=log(n)). Varianti usate Push, e Push-Pull.

- *N Copertura tipica circa O(log(n))*
- *Costo per nodo dipende da k*
- *Anti-entropy (on demand e periodico) per riallineare*

Rumor Mongering

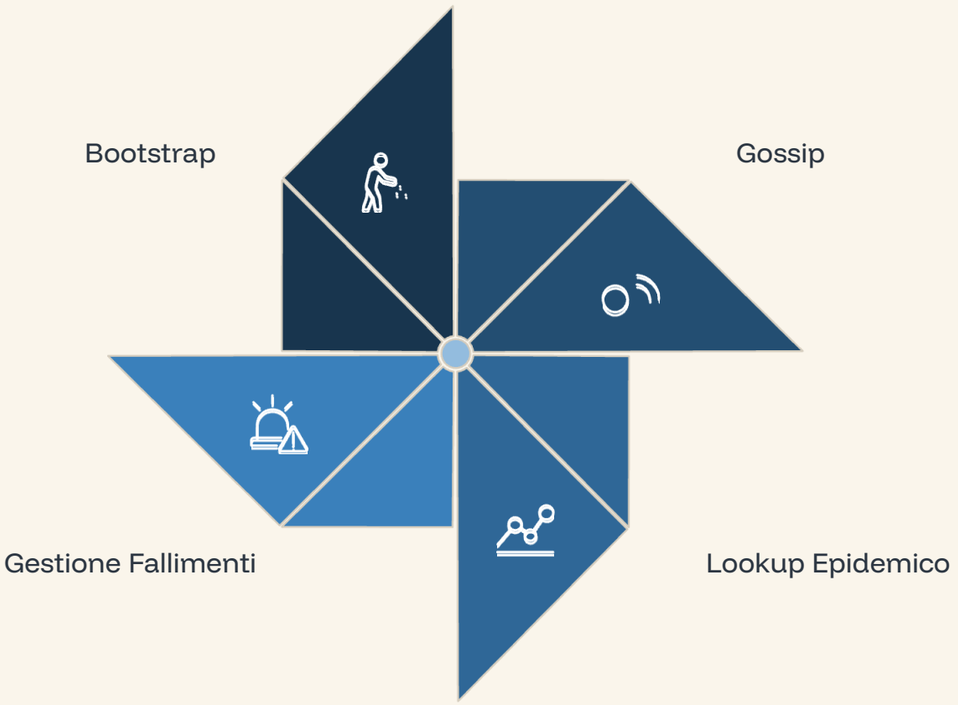
Diffusione probabilistica e limitata di eventi "caldi" (es. Lookup e faildetection).

- ***TTL** (hop massimi): Profondità limitata.*
- ***Fanout (B):** Numero di vicini contattati ad ogni hop.*
- ***Budget (F):** Limite ai reinvii per rumor ID (dedup).*

Il Rumor Mongering taglia la ridondanza rispetto al flooding mantenendo alta la probabilità di raggiungere il target.

Architettura Progetto: Componenti e Flussi

Adottiamo un control plane bimodale con un registry minimo e diffusione epidemica di membership e servizi.



Registry Minimal

*Usato solo per il **bootstrap** iniziale (seed provider), fuori dal data path del lookup (evita SPoF).*

Heartbeat (Light/Full)

Light per liveness; Full per snapshot dei servizi locali e vista peer (innescato da cambio di servizio). Inviati a k=log(n)

Quorum

Quorum dinamico per promuovere nodo Suspect a Dead. Viene applicata una tombstone

Flusso Chiave: Bootstrap

Registry seed-only

Il registry restituisce un solo peer di bootstrap e poi sparisce dal percorso operativo. Un nodo può bootstrappare direttamente da un peer noto

Retry & fallback

Fino a 10 tentativi con 1s di attesa per ottenere almeno un seed. Se la lista è vuota parte isolato.

Inizio gossip

Dopo il primo seed tutto avviene via gossip. Il registry non essendo nel data path, non si occupa della gestione dei servizi e guasti -> niente SPOF.

Flusso Chiave: Gossip

Dettagli sui meccanismi reattivi che garantiscono la funzionalità e la stabilità del sistema.

Gossip light

Gli HB light sono scambi periodici push one hop verso un gruppo di vicini k=log(n). Trasportano solo ID e metadati e servono a mantenere la liveness

Gossip Full

Gli HB Full sono più rari dei light essendo più pesanti. Portano una snapshot dei servizi, della peer-vew e dei metadati. L'invio è forzato quando si aggiunge o rimuove un servizio o in richiesta al repair (on demand o periodica).

Anti-Entropy Periodica

A intervalli regolari si attiva il repair. Si sceglie un target di nodi e si avva uno scambio push-pull per riconciliare le versioni. Accelera la convergenza

Flusso Chiave: Lookup

Lookup Epidemico

*Il client emette un **Rumor di Lookup** (ID univoco, TTL, B, F). I nodi rispondono se conoscono un provider vivo e inoltrano il rumor a B vicini e riducendo il TTL. Ogni nodo fa dedup per reqId (fino a F)*

Coerenza e Robustezza

Si risponde solo se il provider è alive. In caso di miss si attiva una negative cache per evitare tempeste di rumor.

Tuning e comportamento

TTL, B, F regolano il compromesso latenza e traffico. Aumentare TTL aumenta la copertura ma aumenta traffico. Aumentare B (fanout) diffusione piu rapida ma costo piu alto. Aumentare F aumenta resilienza ma più duplicati

Flusso Chiave: Failure Detection

Dettagli sui meccanismi reattivi che garantiscono la funzionalità e la stabilità del sistema.

Rilevazione

Gli HB light aggiornano LastSeen per ogni peer. Se l'intervallo senza HB super SUSPECT_TiMEOUT il peer entra in SUSPECT. Faccio un quickProbe (repairReq) per eveitare falsi positivi

Promozione a Dead

Da Suspect di passa a Dead quando ho un quorum (majority vote) sulla maggioranza dei nodi vivi. Su Dead rimuovo il peer dalle liste e salvo una tombstone

Propagazione

Le transizioni si diffondono con rumor mongering (B,F,TTL). Un msg deead viene scartato se ho visto un HB recente. Le tombstone impediscono revival di messaggi vecchi.

Setup sperimentale

Validazione sperimentale dell'approccio bimodale SDCC, focalizzata su scalabilità, resilienza e prestazioni dei protocolli epidemici.

Topologia

Registry seed-only, 5 nodi (node1,…,node5) con servizi dummy. Client effimero per lookup di servizio. UPD per gossip TCP per registry e utilizzo servizi

Parametri base

- Rumor: B=3, F=2, TTL=3.
- HB light = 3s
- HB full = 9s
- Suspect/Dead = 30s/36s
- Lookup ttl= 3
- Negative cache = 20s
- Repair periodico = false

Metodologia

- Boostrap: contatto registry, ottengo un peer e avvio HB e FaultDetection
- Gossip: HB(light e full) periodici e immediato se aggiunto nuovo servizio o repair req.
- Lookup: client emettere rumor e attende. Nodo risponde se conosce provider vivo altrimenti forwarda a B (ttl--)
- Failure: Aggiorno lastSeen, Suspect su timeout, dead con quorum.

Risultati Sperimentali e Validazione

Validazione sperimentale dell'approccio bimodale per le prestazioni dei protocolli epidemici.

E1 – Baseline

LearnFromHB=on, Repair=off

- *Lookup: risposta direct to origin, latenza bassa*
- *Coerenza: HB full on change, hints via HB light*
- *Robustezza: neg-cache a tempo limista tempeste messaggi*

E2 – HB-learn OFF

LeardFromHB=off, Repair =irrilevante

- *Convergenza servizi solo da lookup/risposte*
- *Repair non efficace (I full non aggiornano)*
- *Più rumore di lookup => trafico alto*

E3 - Crash

Docker kill --signal SIGKILL nodeX

- *Suspect entro Suspect_timeout con quick probe*
- *Dead a quorum dinamico*
- *Propagazione Dead (B,F,TTL).*

Risultati Sperimentali e Validazione

Validazione sperimentale dell'approccio bimodale per le prestazioni dei protocolli epidemici.

E4 – Leave volontario

Docker compose stop nodeX

- *Transizione immediata a Dead+tombstone*
- *Rumor Leave(B,F,TTL)*
- *Quorum non richiesto (evento esplicito)*

E5 – Rientro

Docker compose start nodeX

- *Convergenza servizi solo da lookup/risposte*
- *Repair non efficace (I full non aggiornano)*
- *Più rumore di lookup => trafico alto*

E6 - Join

Via registry o via nodo

- *Seed unico e avvio HB*
- *Convergenza in pochi cicli HB, nessuno SPOF*
- *Lookup resta epidemico.*

Discussione Critica: Trade-off Progettuali

Ogni scelta di design comporta compromessi che richiedono un tuning accurato.

1/3

Negative Cache

*Contiene tempeste su miss, ma rischio **starvation** se il TTL è alto.*

2/3

Parametri Rumor

Copertura rapida con costo per nodo quasi costante. Se TTL/B sono bassi la copertura cala, se alti cresce il traffico

3/3

Anti-Entropy

*Garantisce convergenza, ma introduce **overhead periodico** (mitigato da un intervallo* `T_repair >> T_HB(full)`*).*

> *In sintesi bisogna accettare alcuni trade off come tunare I parametri di rumor per bilanciare velocità e overhead. Anti entropy per bilanciare convergenza e ovehead.*

Conclusioni e Lavori Futuri

Un sistema decentralizzato, resiliente e bilanciato per service discovery e failure detection.

→ Difesa da Attacchi (Sicurezza)

Difesa contro Dos/DDoS, spoofing, MITM e replay attack ***Challenge-Response*** *(contro amplificazione UDP),* ***mTLS****/PKI (contro Spoofing) e* ***Versioni Monotone*** *(contro Replay/Downgrade).*

→ Auto-Tuning

Sviluppo di un meccanismo di auto-tuning di gossip e FD guidato da misure online di ***RTT*** *e* ***perdita*** *di pacchetti per adattare dinamicamente i timeout e i fanout.*

→ Scalabilità e Robustezza

Esecuzione di stress test con un ***elevato numero di nodi*** *(N) per valutare i limiti di scalabilità del design in scenari reali e complessi.*

Grazie per l’attenzione