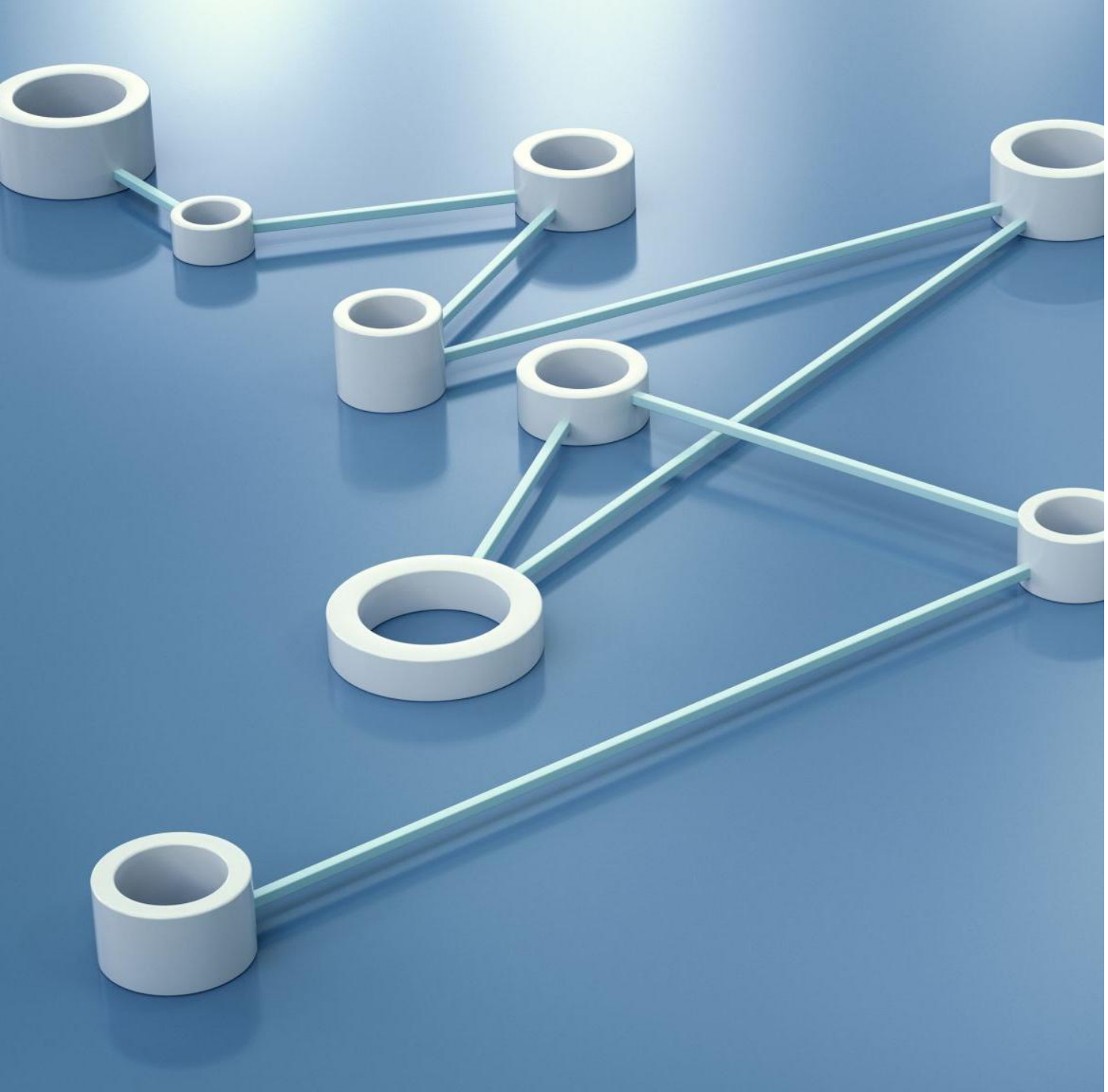


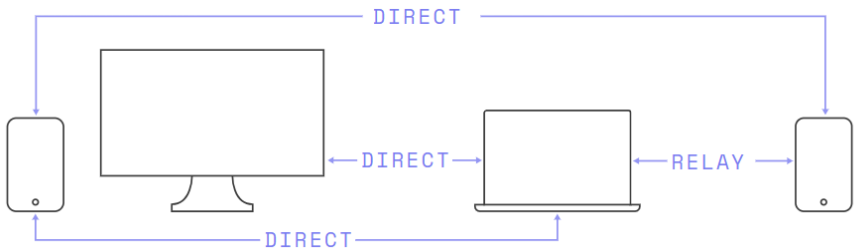
Reliability Testing of iroh-gossip in Realistic Network Environments

ANDREA SEEHUBER

NEW TRENDS FOR LOCAL AND GLOBAL INTERCONNECTS FOR P2P APPLICATIONS

UNIVERSITY OF BASEL, 29.10.2025

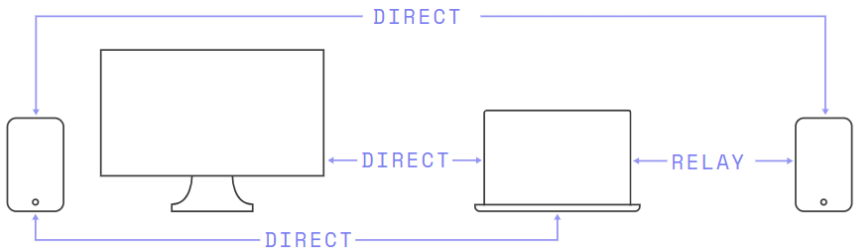
iroh



Recap: What is iroh?

- Library that can be used to establish secure P2P connections
- based on QUIC as network protocol
  - uses UDP as transport protocol
  - can use TLS for cryptographic security
- currently used by DeltaChat, Shaga, DumbPipe (for example)

iroh



Recap: What is iroh?

- **Endpoint**: main API for establishing connections
- **Discovery**: service that resolves EndpointID to a relay URL or a P2P connection (DNS or local)
- **Router**: implements the accept loop and directs incoming connections to the correct protocol (via ALPN)
- uses relay servers to establish secure P2P connections (also fallback if no P2P possible)
- direct P2P connection possible via **Tickets**

Recap: The iroh-gossip

HyParView (Hybrid Partial View)

P2P protocol for managing neighbourhoods in a dynamic network

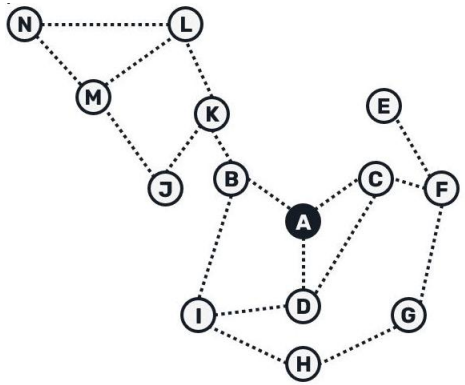
list of peers is devided into subsets: active view and passive view

regularly checks active neighbours and replaces from passive list if no longer active

peers randomly exchange entries from their passive views

Recap: The iroh-gossip

PlumTree (Push-Lazy-Update-Multicast)



tow gropus of neighbours:

- **eager peers:** receive message immediately
- **lazy peers:** only receive hash of the message

advantages:

- **high reliability:** every node ultimately receives every message
- **efficiency:** only a few messages contain the full content
- **self-optimising:** paths with low latency are used preferentially

iroh-gossip testing

What has already been done?

iroh-gossip / tests / sim.rs

Frando fix(hyparview)!: Only add peers to active view after receiving neighb...

Code Blame 134 lines (121 loc) · 3.99 KB

```rust
//! Tests that use the [`iroh_gossip::proto::sim::Simulator`].

use std::{env, fmt, str::FromStr, time::Duration};

use iroh_gossip::proto::{
    sim::{BootstrapMode, LatencyConfig, NetworkConfig, Simulator, SimulatorConfig},
    Config,
};
```

big_hyparview: checks whether all peers have neighbours after bootstrap

big_single_sender: measures delivery rate, duplicates, efficiency

big_multiple_sender: stress test, measures propagation and deduplication

big_burst: each peer sends many messages; measures scalability and performance under high load

iroh-gossip testing

What has already been done?

iroh-gossip / tests / sim.rs

Frando fix(hyparview)!: Only add peers to active view after receiving neighb...

Code Blame 134 lines (121 loc) · 3.99 KB

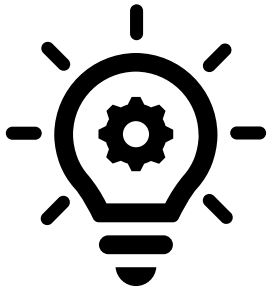
```rust
//! Tests that use the [`iroh_gossip::proto::sim::Simulator`].

use std::{env, fmt, str::FromStr, time::Duration};

use iroh_gossip::proto::{
    sim::{BootstrapMode, LatencyConfig, NetworkConfig, Simulator, SimulatorConfig},
    Config,
};
```

but ...

Tests run entirely within a simulator,
i. e. without a real network!

iroh-gossip testing

What could be done?



Possible questions to answer:

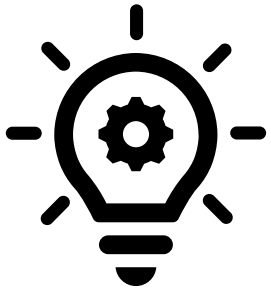
How robust is the gossip against network failures, i. e. packet loss, latency, interruptions?

How much does discovery (direct vs. relay) affect delivery rates?

How does the system converge when peers fail or reconnect?

iroh-gossip testing

What could be done?

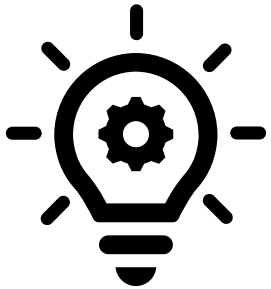


Possible use-cases:

1. Local LAN, Direct Discovery
   - no injection of network disruptions
   - no relay discovery active
2. Relay-Assisted Discovery (NAT Simulation)
   - same as UC1 but alternative relay discovery
3. Relay-Assisted Discovery, Degraded
   - same as UC2 but injection of delay and packet loss
4. Churn and Recovery
   - same as UC1 or UC3 but disconnect/rejoin of peers

iroh-gossip testing

What could be done?



Possible metrics:

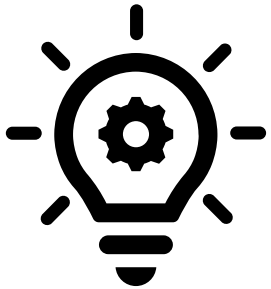
1. **delivery ratio (DR)**: received vs. sent messages
2. **duplicate rate (DuR)**: unique received messages
3. **end-to-end-delay (E2E)**: average delivery time or per message received
4. **convergence time (CT)**: time until all peers receive all messages
5. **peer reachability (PR)**: reachable vs. connected peers
6. **reconnect time (RT)**: time until first received message after reconnect

iroh-gossip testing

Comparison	Constant parameters	Variable	Metrics	Expected trends
UC1 vs. UC2	• no network impairments • same no. of peers, message rate and payload	discovery method	DR, DuR, E2E, CT, PR	relay discovery improves DR and PR; E2E and CT worsens
UC2 vs. UC3	• relay enabled • same no. of peers, message rate and payload	network quality	DR, DuR, E2E, CT, PR	under degraded network: DR decreases, DuR increases, E2E and CT grow, PR remains stable
UC1 vs. UC3	• same no. of peers, message rate and payload	discovery type + network quality	DR, DuR, E2E, CT, PR	largest performance gap expected
UC1 vs. UC4	• no network impairments • same no. of peers, message rate and payload • same discovery type	churn under direct discovery	DR, DuR, CT, RT, PR	during disconnections, DR drops and CT rises; after rejoin, peers recover but measurable RT
UC2 vs. UC4	• no network impairments • same no. of peers, message rate and payload • same discovery type	churn under relay-assisted discovery	DR, DuR, CT, RT, PR	with relay, RT decreases and DR recoveres more quickly compared to direct discovery

iroh-gossip testing

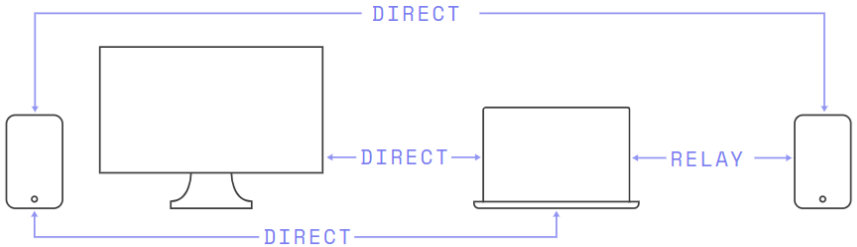
What could be done?



so ...

What really makes sense?

What is feasible?



iroh