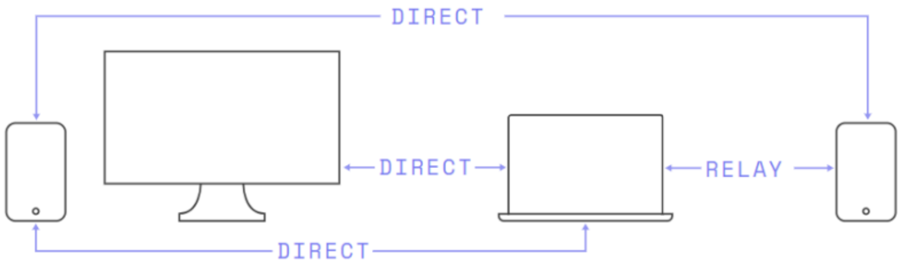
Experimental investigation of the reliability of the iroh-Gossip protocol under realistic network conditions



Report on the seminar "New Trends for Local and Global Interconnects for P2P Applications"

University of Basel, autumn semester 2025/2026

submitted by Andrea Seehuber, 28 December 2025

Table of contents

1 Introduction

1.1 Motivation

Peer-to-peer networks (P2P) play a central role in many modern digital applications. In contrast to classic client-server structures, they distribute tasks, data and communication among many equal participants. Depending on the structure, different forms of such networks can be distinguished:

- Pure, completely **DECENTRALISED P2P NETWORKS** do not require a central server instance. Each node simultaneously assumes the role of a client and a server. A well-known example is the BitTorrent file-sharing protocol, which is used by Facebook, among others (van der Sar, Ernesto, 2010) .
- In addition, there are **SERVER-BASED P2P NETWORKS** in which central servers perform certain tasks, such as coordinating user management, while some data is transferred directly between peers. A prominent example of this is the online service Discord, which uses WebRTC as its communication protocol stack (Vass, Jozsef, 2018) .

The iroh library represents a kind of hybrid technology that enables both fully decentralised and relay server-based communication. It offers a modular network stack whose components include a decentralised broadcast service (iroh-gossip). This is designed to distribute messages efficiently in a dynamic peer network, even when nodes fail, network connections are unstable, or nodes are constantly reconnecting (n0, 2025) . The iroh-gossip protocol has recently been implemented by the instant messaging service Delta Chat, which is available for all common operating systems as a privacy-friendly alternative to WhatsApp (merlinux GmbH, 2025) .

1.2 Objective

The aim of this project is to systematically investigate the behaviour of iroh-gossip. To this end, a dedicated measurement framework was developed that uses Linux network namespaces and netem to simulate realistic network conditions – from stable LAN topologies to artificially increased latencies and packet losses to dynamic churn situations in which individual nodes temporarily leave the network. The analysis aims to show how reliable, fast and robust iroh-gossip is under such conditions and what conclusions can be drawn from this for practical use.

2 Theoretical background

2.1 Epidemic broadcasting

Epidemic or gossip-based broadcasting methods are a class of distributed algorithms based on the propagation dynamics of biological epidemics. Instead of distributing messages via a fixed tree structure or a central coordination instance, dissemination takes place through repeated forwarding between randomly selected neighbours. As a result, information spreads stochastically and with high redundancy throughout the network ( ) and, ideally, reaches all active participants.

The key advantage of epidemic methods is their robustness: since no fixed structure is required, communication remains functional even with high failure rates of individual nodes. This makes gossip approaches attractive for dynamic P2P systems in which nodes frequently join, fail or move. In addition, the method is highly scalable, as the communication effort per node remains independent of the overall size of the system.

However, there is one major disadvantage: the stochastic nature of the process leads to high redundancy. The same message reaches many nodes multiple times, which leads to considerable bandwidth consumption in large networks. For this reason, current variants of gossip broadcasting deal with mechanisms for reducing redundant transmissions without compromising reliability (Leitão, et al., 2007) .

2.2 Neighbourhood management and message distribution

Modern gossip-based broadcast protocols usually consist of a **MEMBERSHIP OR PEER SAMPLING PROTOCOL** that manages the local neighbour view of a node and a **BROADCAST PROTOCOL** that distributes messages based on these neighbour views. The two protocols used by iroh-gossip are described below.

2.2.1 HyParView

HYPARVIEW (Hybrid Partial View) addresses a core weakness of earlier gossip networks: when nodes fail, classic partial view protocols such as Cyclon or Scamp quickly become inconsistent, and the overlay can lose its connectivity. HyParView addresses this problem by dividing the neighbour view into

1. an **ACTIVE VIEW**, which is very small (typically 4–6 entries) and maintains permanent connections to neighbours, and
2. a **PASSIVE VIEW**, which is larger (typically 20–40 entries) and only contains a list of reachable backup nodes.

The active view is used for the actual message transport; the passive view ensures that failed neighbours can be replaced at any time. Nodes are distributed in the passive view via periodic shuffle operations, so that the overlay remains permanently mixed. In their evaluation, Leitão et al. were able to show that HyParView can reconnect the network in just a few rounds, even with 80–90% of nodes failed (Leitão, et al., 2007) .

2.2.2 Plumtree

While HyParView merely determines *which* nodes a node is connected *to*, the Plumtree algorithm defines *how* messages are distributed via these neighbours. The central approach is to combine the advantages of structured and unstructured broadcast methods:

- **EAGER PUSH (TREE COMPONENT):** Upon receiving the first message, each node dynamically builds a tree edge to the neighbour from whom it first received it and actively forwards the message to selected neighbours. This creates an efficient, low-redundancy distribution tree.
- **LAZY PUSH (GOSSIP COMPONENT):** All other neighbours receive only short IHAVE notifications. If they are missing a message, they can actively request it (`GRAFT`). This repairs missing edges if the tree is incomplete or damaged by failures.

This makes Plumtree significantly more efficient than pure gossip, while retaining the robustness that pure tree methods cannot provide (Leitão, et al., 2007) .

2.3 The iroh-gossip protocol in the iroh stack

The **IROH-GOSSIP PROTOCOL** embeds the algorithms described in sections 2.1 and 2.2 into the iroh network stack. iroh itself is a library for P2P communication via QUIC (network protocol at the transport layer) that establishes direct connections between endpoints and automatically falls back to relay servers when necessary (n0, 2025) . The central entry point of iroh is the `endpoint`. An `endpoint` represents a peer in the network, has a cryptographic identity and manages incoming and outgoing QUIC connections. Various application protocols can run in parallel on these connections. The selection is made via ALPN (Application Layer Protocol Negotiation). A `router` receives the incoming connections and forwards them to the appropriate protocol handler (n0, 2025) .

In order for iroh-gossip to build an overlay at all, the endpoints involved must first find each other. This task is not performed by the gossip protocol itself, but by the discovery layer of the endpoint. `Discovery` links the `EndpointId` with specific dialling information, i.e. either a relay URL or directly dialable addresses (IP/port). Each `iroh endpoint` automatically publishes its own address information to the configured discovery services when it starts up. Other endpoints can then look up the current connection data and establish a connection based solely on the `EndpointId`. iroh provides several discovery backends for this purpose, including (The Rust Foundation, 2025) :

- **DNS DISCOVERY** uses a special DNS service in which `EndpointIds` are mapped to address information in TXT entries. Number0 (the iroh community) operates a public instance for this purpose, which is used in the default configuration. From there, iroh attempts to establish a direct connection via hole punching if possible; if this fails, data traffic remains permanently routed via the relay server.
- **LOCAL DISCOVERY** enables endpoints to be found on the same local network without a central service. To do this, iroh uses an mDNS-like mechanism in which endpoints on the LAN announce themselves to each other via multicast.

By default, iroh uses DNS discovery, but can be configured at runtime to use other or multiple discovery mechanisms in parallel.

At the application level, iroh-gossip offers a relatively simple API:

- Gossip groups (swarms) are defined via a `TopicId`.
- To join a swarm, a peer calls `subscribe_and_join(topic_id, bootstrap_peers)`. The `bootstrap_peers` are `iroh EndpointIds`, through which the node first establishes connections; HyParView then takes over the further distribution of messages.
- The application receives a sender/receiver split: the sender distributes the messages to the topic, while the receiver delivers `Event::`Received events as well as structural events such as `NeighborUp` or `NeighborDown`.

The discovery mode that is set ultimately determines how quickly a gossip swarm can be established, whether all peers find bootstrap connections in time, and whether broadcast messages are transported via direct connections or via relays.

3 Experimental setup

The following measurements were performed locally on an ASUS Vivobook S14 Flip running Linux Mint 22.2 (Cinnamon Edition). All experiments were performed fully automatically using a series of shell scripts. The measurements were always averaged over 10 independent test runs per parameter combination to systematically reduce random fluctuations and outliers. A separate, simplified Rust implementation of an iroh-gossip peer was developed for the test series. The aim was to be able to examine the mechanisms provided by iroh-gossip under controlled conditions as transparently as possible and without application-specific side effects.

3.1.1 Varied parameters and measurement targets

In order to investigate different load and environment scenarios, the following parameters were varied:

1. **Discovery mode.** In direct mode, peers connect directly in the LAN, and discovery takes place without a relay. In relay mode, initial contact is made via a relay server; direct connections are only established if hole punching is successful.

   Measurement targets: Influence on join behaviour, message delivery rate and latencies.

2. **Number of peers.** All test runs were performed with 10, 20, 30, 40 and 50 peers in the simulated network. For every 10 peer processes started, one bootstrap node ID was extracted and passed on for the join process.

   Measurement objectives: Investigation of scaling effects on path lengths, neighbourhood sizes, join behaviour and delivery rates with growing overlay.

3. **Network scenario.** Using the Linux kernel tool netem, a 50 ms delay ± 5 ms jitter and 30% packet loss were injected and compared to the undisturbed LAN.

   Measurement objectives: Robustness in terms of latencies, delivery rates and out-of-order events with degraded network quality.

4. **Churn and recovery.** 40 seconds after the start of the test run, randomly selected peers are isolated from the network (churn). The churn rate is 10%, 20% or 30%, with bootstrap peers never being isolated. The peers are reintegrated into the network after 10 seconds.

   Measurement objectives: The protocol's ability to repair a damaged overlay and restore broadcast range is examined.

3.1.2 Test run procedure

The same procedure was used for each configuration:

1. **Set up an isolated test environment via Linux namespaces.** Each peer process runs in its own namespace with a separate network stack, its own routing table and its own virtual

network interface. All peers are connected via a shared virtual Linux bridge, which represents the simulated LAN.

2. **Start of the receiver processes.** The receiver is first started in all peers to ensure that the overlay can be completely established.
3. **Starting the transmitter.** The transmitter process only starts after all receivers are ready.
4. **Sending the test data set.** 2000 messages are sent at a frequency of 10 messages per second and a payload size of 256 bytes. The number and transmission frequency were chosen so that the passive view in the HyParView protocol is shuffled three times during a complete test run. The messages contain a unique test run indicator, a sequence number, a transmitter timestamp and the expected total number of messages.
5. **Recording of all events in structured JSONL form.** The summary, which is mainly used for evaluation, contains the following information, among other things:
   - Delivery rate
   - Number of duplicates
   - Out-of-order events
   - Latency statistics
   - Last delivery hop metrics
   - Average time without active neighbours
   - Average number of active neighbours
   - Join state
6. **Automated termination of peers.** All processes are terminated after a dynamically calculated join timeout to reliably terminate even faulty or incompletely started runs. The timeout safely exceeds the length of a test run in which the maximum number of messages is sent.

4 Results

The log files generated during the experiments were evaluated fully automatically using a self-implemented Python script and the Python library pandas. This script reads the JSON-based summaries generated for each test run, aggregates the relevant metrics over several repetitions, and calculates statistical parameters such as mean values and distributions.

4.1 Network formation and overlay dynamics

4.1.1 Failed test runs

A first key finding emerged during the initial analysis of the initial test runs, which were designed to differentiate between discovery mode and network quality. In these early configurations, regardless of the total number of peer processes started, only one individual bootstrap node ID was passed to all peers. In several scenarios, this led to a high rate of unsuccessful test runs, as the sending peer was never able to join the gossip topic and consequently no messages were sent. In relay mode in particular, it was still observed that the error rates were subject to very high variance depending on the time of measurement. Based on these observations, the bootstrap strategy was adjusted and one bootstrap node ID was passed for every 10 peer processes started. This adjustment led to improved stability of the test runs across all scenarios considered. The results are summarised in Table 1:

Table 1 : Percentage of unsuccessful test runs for one bootstrap node ID transferred per 10 peer processes started. The values per cell refer to the number of peer processes in the respective test run (10, 20, 30, 40 and 50) and are to be understood as rough averages.

	Without network disruption					With network disruption				
Discovery Direct	0	10	10	0	10	0	0	0	0%	0
Discovery Relay	10	10	10	30	30	0	0	0	0	25

The sometimes counterintuitive assignment of use case identifiers in the evaluation script, in the log file directories and in the shell scripts is a result of this adjustment. A detailed description of this is documented in a separate text file in the repository. Since the number of bootstrap node IDs used does not itself represent an examined metric, the use case designations were standardised for the presentation and discussion of the results.

4.1.2 Join behaviour

The **JOIN BEHAVIOUR** describes the proportion of receiver processes started that successfully joined a gossip topic and were therefore able to receive broadcast messages. Only valid test runs are included in the evaluation, i.e. runs in which the sending peer successfully joined the topic. In discovery mode `Direct`, join behaviour is very stable across all scenarios examined: the proportion of receivers that successfully joined is almost constant at 100%, regardless of the total number of peers started and the presence of artificial network disruptions. In `Relay` mode, almost all receivers also successfully join the topic without network disruptions. However, when network disruption is activated, the proportion of receivers that join drops to around 90% with 10 peers in the network and around 80% with 20–50 peers in the network (see Fig. 1).

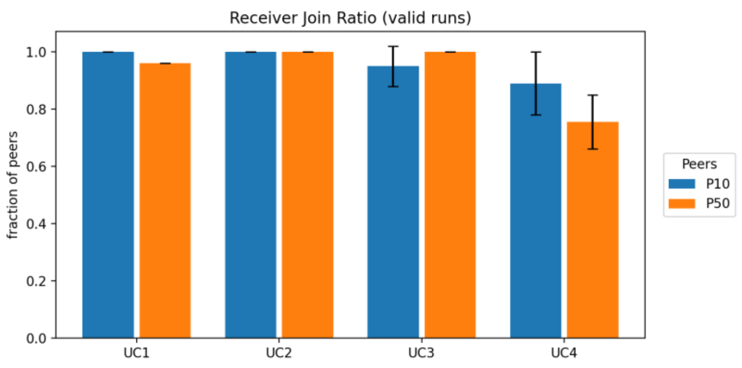


Fig. 1 : Percentage of peer processes that joined the gossip topic and received messages. UC1/UC3: Discovery mode Direct, without or with injected network interference; UC2/UC4: Discovery mode Relay, without or with injected network interference.

4.1.3 Active View and NeighbourUp/Down events

The **ACTIVE VIEW** describes the number of peers with which a node currently maintains an active overlay connection. In the underlying HyParView implementation of iroh, the maximum size of the Active View is limited to five peers (n0, 2025) . The results show that, without network disruption, a slightly higher average number of peers tends to be achieved in Active View in direct mode than in relay mode. This effect is consistent across all scenarios considered. In direct mode, the average size of Active View is slightly lower in larger networks than in smaller networks (see Fig. 2).

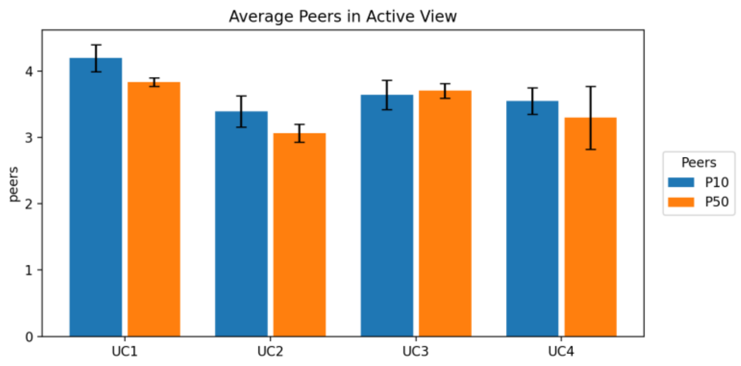


Fig. 2 : Average number of neighbouring nodes in the active view of a peer. UC1/UC3: Discovery mode Direct, without or with injected network disruption; UC2/UC4: Discovery mode Relay, without or with injected network disruption.

NeighborUp and NeighborDown events were also evaluated. These events count how often peers are added to or removed from Active View and thus serve as an indicator of the dynamics and stability of the overlay neighbourhood. In all scenarios considered, the number of such events essentially scales with the network size, but no relevant difference between the two discovery modes can be detected (see Fig. 3).

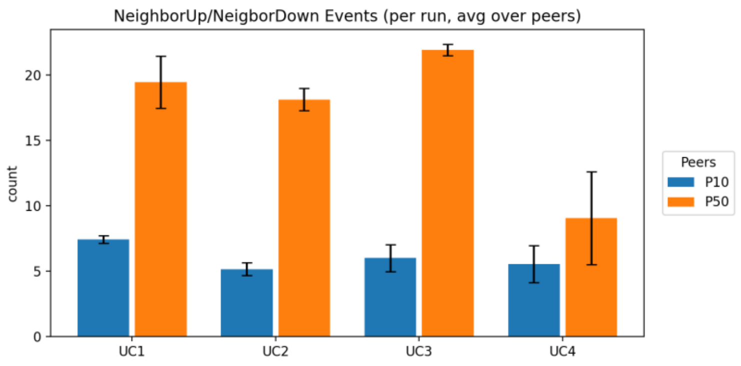


Fig. 3 : Average number of changes to a peer's Active View due to the addition or removal of other nodes in the network. UC1/UC3: Discovery mode Direct, without or with injected network disruption; UC2/UC4: Discovery mode Relay, without or with injected network disruption.

Without injected network interference, most peers have at least one neighbouring node in their Active View throughout the entire duration of a test run. Only in relay mode do isolated cases of peers occur in larger networks that have an empty Active View for 4–5 seconds accumulated over the entire test run. In both discovery modes, the average time during which peers in the network have an empty Active View increases under simulated poor network quality (see Fig.1). However, the variance here is very large, as in relay mode in small networks, for example, there are also individual peers whose Active View is never empty during the entire test run.

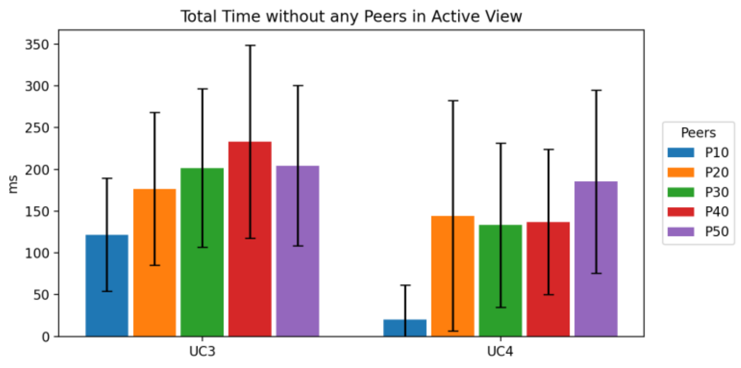


Fig.1 : Total time during a test run that a peer had an empty Active View. The values are averaged across all peers in the network. UC3: Discovery mode Direct; UC4: Discovery mode Relay. Poor network quality was simulated in both scenarios.

4.2 Message transmission

4.2.1 Delivery rate, duplicates and out-of-order messages

Regardless of the discovery mode, the rate of delivered messages in valid test runs is practically 100% throughout, provided that the receivers have joined the topic. No duplicates occurred in any of the test runs evaluated.

Out-of-order events refer to received messages whose sequence number is smaller than that of a previously received message and which therefore do not arrive in the original transmission order. Such events were observed exclusively when network disruption was activated; the results are shown in Fig. 5. In direct mode, an average of around 170 out-of-order messages occur with 10 peers in the network; with 50 peers, this value increases to about twice as much. In relay mode, the number of out-of-order messages is comparatively lower with more peers in the network.

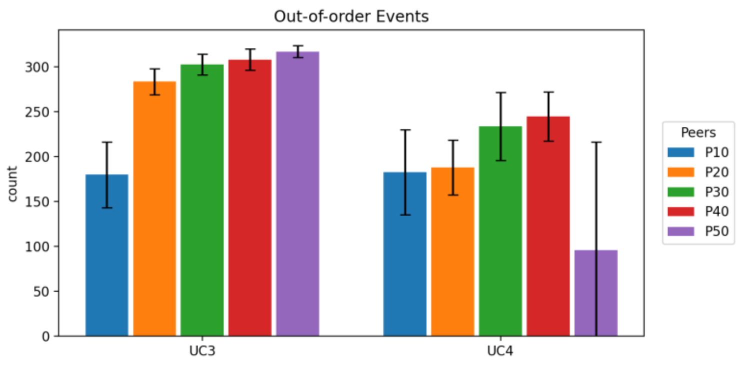


Fig. 5 : Number of messages that were not received in the intended order UC3: Discovery mode Direct; UC4: Discovery mode Relay. Poor network quality was simulated in both scenarios.

4.2.2 Latency and LDH

Latency describes the **end-to-end delay** of a message between the time it is sent and the time it is received at the destination node. Quantiles are used to characterise the latency distribution: the 50th percentile (p50) indicates the median latency, while higher quantiles (p90, p99) and the maximum provide information about the behaviour at the upper end of the distribution and about outliers. In this way, both typical and rare, particularly delayed deliveries can be recorded.

For the use cases without network disruption (UC1 and UC2 in Fig. 6), the latencies are generally at a low and stable level. However, the maximum transmission times in particular scale significantly with the size of the network. Under network disruption (UC3 and UC4 in Fig. 6), latencies increase and, especially at the higher quantiles and maximum values, reach up to 10 times the times measured without network disruption.

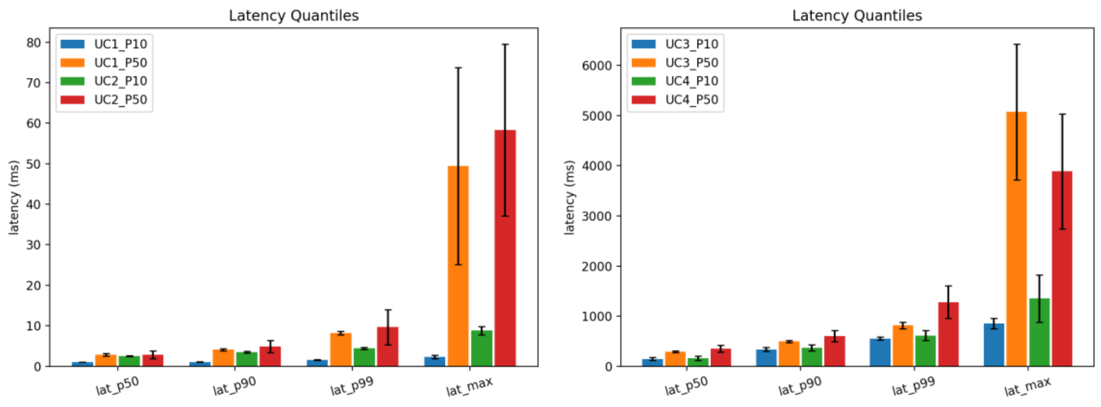


Fig. 6 : End-to-end latencies without (left) and with (right) simulated network disruption. UC1/UC3: Discovery mode Direct; UC2/UC4: Discovery mode Relay. The values shown are for the scenarios with the smallest and largest number of peers in the network (10 and 50, respectively).

The **Last Delivery Hop (LDH) metric** describes the number of overlay forwarding steps that a message passes through before reaching its final recipient. Quantiles were also evaluated here in order to reveal the maximum path length in addition to the typical behaviour (seeFig.2).

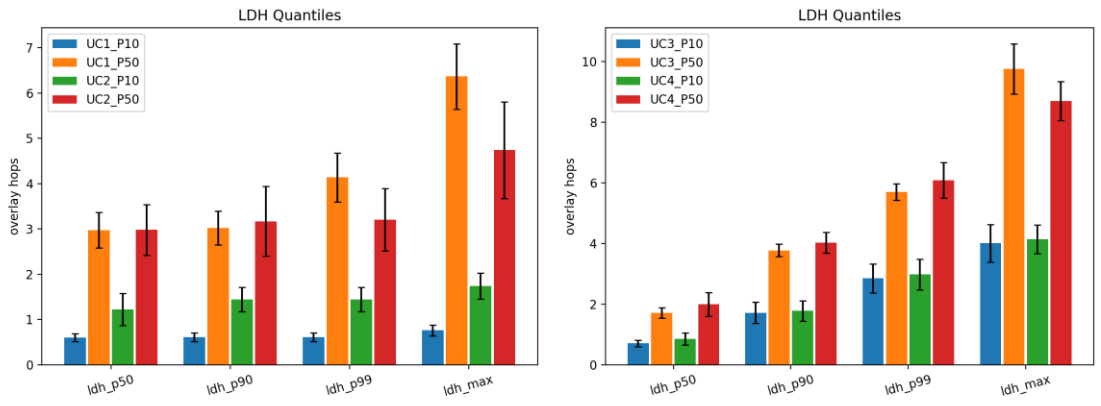


Fig.2 : Average number of last delivery hops without (left) and with (right) simulated network impairment. UC1/UC3: Discovery mode Direct; UC2/UC4 Discovery mode Relay. The values shown are for the scenarios with the smallest and largest number of peers in the network (10 and 50, respectively).

The LDH values are higher in interference-free scenarios (UC1 and UC2 in Fig.2) and in small networks for discovery mode relay than for direct mode, but they even out for larger networks and tend to be lower in relay mode. With simulated network disruption (UC3 and UC4 in Fig.2), very similar values are measured for both discovery modes.

4.3 Effects of churn and recovery

All results described so far refer to scenarios in which the number of peers in the network remains stable during a test run. If a subset of peers is temporarily isolated from the network, this has no significant impact on the following metrics compared to the results presented so far:

- The **DELIVERY RATE** of sent messages remains stable at nearly 100%.
- No **DUPLICATE MESSAGES** are counted.
- The **JOIN BEHAVIOUR** described in 4.1.2 can be reproduced (seeFig. 1).
- The average **NUMBER OF NODES IN** a peer's **ACTIVE VIEW** remains comparable to the results described in 4.1.3 (seeFig. 2). The number of events counted that lead to a change in the active view also does not change significantly (seeFig. 3).
- The path lengths of message transmission (number of **LAST DELIVERY HOPS**) remain essentially unchanged from the results described in 4.2.2 (seeFig.2).

Quasi independent of the discovery mode, but correlating with the percentage of nodes that are isolated for 10 seconds, isolated **OUT-OF-ORDER EVENTS** are now also counted in the undisturbed network (seeFig.3). However, this only affects peers that have been isolated from the network, which explains the high variance in the average values.

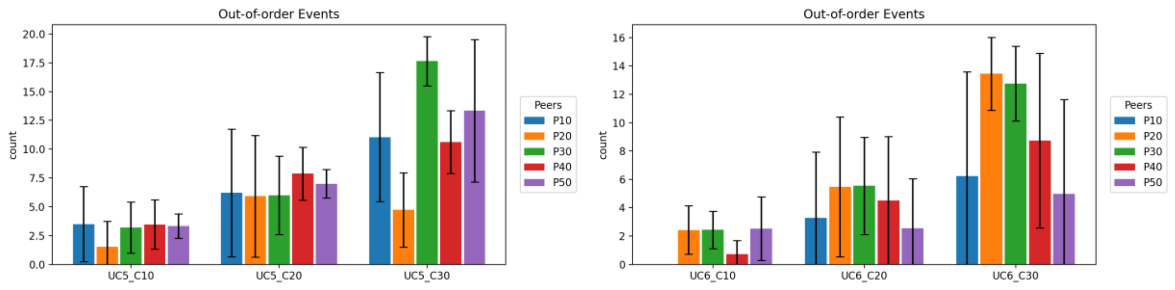


Fig.3 : Number of messages that were not received in the intended order. UC5: Discovery mode Direct; UC6: Discovery mode Relay. Both scenarios are without network disruption. In networks with 10, 20, 30, 40 and 50 peers, 10%, 20% and 30% of the peers were isolated for 10 seconds.

Under poor network quality, the number of out-of-order events increases significantly, but remains in a similar order of magnitude as in the scenarios described in 4.4 (seeFig. 9). All peers are now affected again, with no significant difference between temporarily isolated and non-isolated peers. It is noticeable, however, that in Discovery Mode Direct, the number of messages that are not delivered in the correct order reaches a maximum with a network size of 20 peers and is smallest with a network size of 50 peers (Fig. 9 left). In Discovery mode Relay, the out-of-order events correlate approximately with the network size, but the differences are not significant.

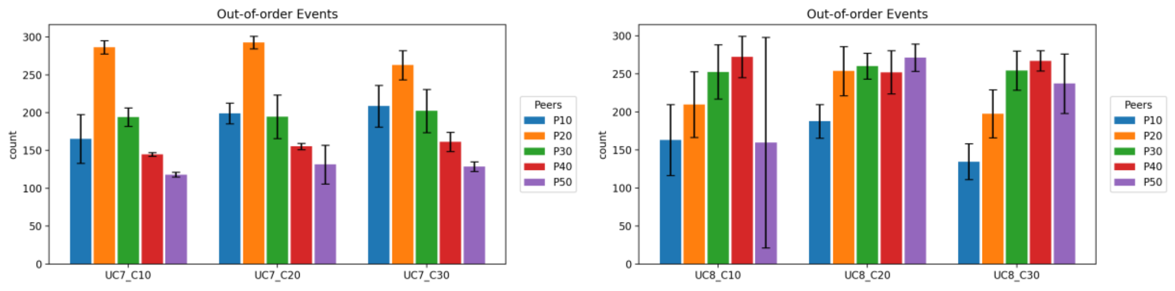


Fig. 9 : Number of messages that were not received in the intended order. UC7: Discovery mode Direct; UC8: Discovery mode Relay. Both scenarios involve simulated network disruption. In all networks, 10%, 20% and 30% of peers were isolated for 10 seconds.

The evaluation of the latency quantiles shows an increase in maximum end-to-end latencies in the churn scenarios. This effect can be observed both in configurations without additional network disruption (Fig.4 left) and in scenarios with artificially introduced delays (Fig.4 right). The lower quantiles, on the other hand, remain in the same order of magnitude as in the scenarios without churn (seeFig. 6).

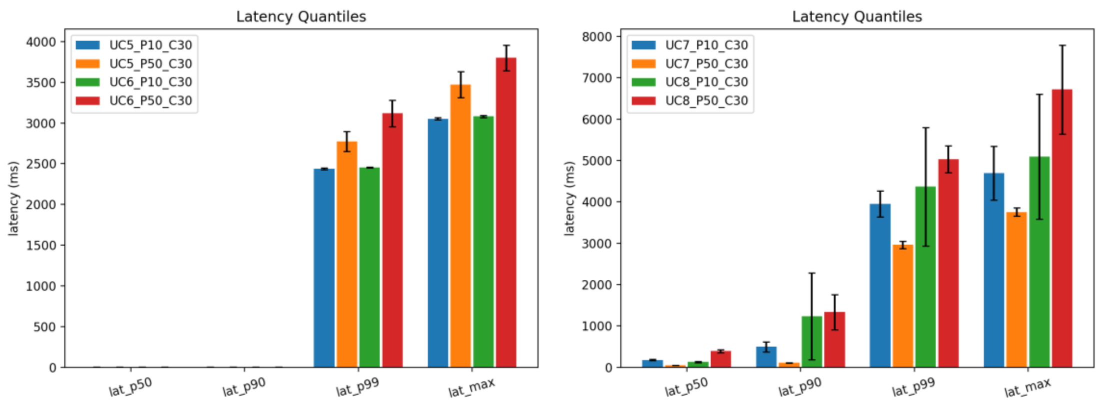


Fig.4 : End-to-end latencies without (left) and with (right) simulated network impairment. UC5/UC7: Discovery mode Direct; UC6/UC8: Discovery mode Relay. The values shown are for the scenarios with the smallest and largest number of peers in the network (10 and 50, respectively). In each case, 30% of the peers in the network were isolated for 10 seconds.

Analysis of the log files of the peers isolated during the churn shows that the maximum latency values recorded there correspond to the duration of the network isolation.

5 Discussion

5.1 Network formation

The observed **JOIN BEHAVIOUR** illustrates the central role of the discovery layer for the stability of the gossip overlay. While reproducible join behaviour was observed across all measurement series in direct mode, relay mode showed marked variability in the number of valid test runs. This variability occurred not only within individual test series, but also between measurements taken on different days and at different locations. In individual cases, no valid test runs could be achieved in relay mode over several consecutive trials. These observations suggest that the join process in relay mode is sensitive to external factors that lie outside the actual gossip protocol. A plausible explanation is the dependence on the availability of the relay server and the quality of the underlying network connection of the executing system (which was always connected to Swisscom's 5G network via a mobile hotspot). However, as the availability of the n0 relay service was not part of the controlled test setup, no clear conclusions can be drawn based on the available data.

Furthermore, it was found that in relay mode, the number of valid test runs stabilises significantly under simulated network disruption, while the proportion of receiver processes that successfully join the gossip topic reproducibly drops to around 80%. This behaviour can be explained by the fact that the join process is time-delayed under poor network quality and thus behaves more deterministically. The observation suggests that relay mode tends to result in a stable but incomplete overlay under unfavourable network conditions.

Another critical issue concerns the need for multiple **BOOTSTRAP NODE IDs** for the stable construction of a gossip swarm. Experiments have shown that passing only a single bootstrap node ID leads to a significantly increased rate of failed test runs, especially in relay mode. This result raises questions about the conceptual role of bootstrap peers, especially in discovery mode Direct. In a local network where peers can discover each other via an mDNS-like multicast mechanism, one would expect that the initial join of a gossip swarm would be possible without explicit bootstrap peers. However, the iroh API enforces the transfer of a vector with at least one NodeID using the `subscribe_and_join()` method. Alternative call sequences in which `subscribe()` and `join()` are used separately do exist, but could not be successfully used for a complete swarm setup within the scope of this work. Neither the available documentation nor the API description on the iroh website clearly specify how a join without explicit bootstrap peers is intended to work.

5.2 Overlay dynamics

Analysis of **THE ACTIVE VIEW** and the **NEIGHBORUP/DOWN EVENTS** shows that the basic overlay structure behaves similarly in both discovery modes. As expected, the time span without active neighbours increases under poor network quality, but without leading to a systematic collapse of the overlay. This indicates that HyParView's self-healing neighbourhood structure is functioning properly, even when connections are temporarily unstable. Even temporarily isolated peers maintain a stable Active View and are reliably reintegrated into the gossip network, at least after the relatively short churn time chosen for this work.

5.3 Message transmission

The nearly consistent **DELIVERY RATE** of 100% in all valid test runs confirms that the combined use of HyParView and Plumtree enables reliable message distribution even in degraded networks and temporary node failures. The complete absence of **DUPLICATES** in all scenarios confirms the effectiveness of the Plumtree approach to reducing redundant transmissions. At the same time, the **OUT-OF-ORDER EVENTS** under network disruption show that iroh-gossip does not enforce strict order at the transport level. This is consistent with the use of QUIC connections and stochastic forwarding in the overlay.

The evaluation of **THE LAST DELIVERY HOP METRICS** shows that the effective path lengths of message transmissions in discovery mode direct and relay mode differ only slightly. Relay mode thus primarily affects the transport layer of individual connections, but not the logical number of forwarding steps in the overlay.

Analysis of **END-TO-END LATENCIES** shows that iroh-gossip exhibits comparatively low and well-controlled delays in stable operation. The lower quantiles hardly differ between the two discovery modes in networks with 40 or 50 peers, suggesting that message forwarding primarily takes place via the peer-to-peer overlay. The observed increase in maximum values with a growing number of peers suggests that rare, heavily delayed deliveries are caused by longer overlay paths or temporary delays in individual connections, without significantly affecting typical latency behaviour. Furthermore, the observations suggest that simulated network disruptions do not lead to a uniform slowdown in message distribution, but rather cause significant delays in individual transmissions. In churn scenarios, it can be seen that the maximum latency values correspond to the duration of network isolation of individual peers. The increased maximum values are therefore primarily due to the delayed delivery of messages after the reconnection of isolated nodes.

6 Conclusion

As part of this work, the iroh-gossip protocol was systematically investigated under controlled but realistic network conditions. The experiments carried out are subject to several limitations: despite the use of Linux network namespaces and netem, real, global networks with high heterogeneity can only be approximated. In addition, the measurements were limited to a maximum network size of 50 peers and were carried out at a fixed message rate with a fixed message size. The effects of very large networks, variable load profiles or long-term churn dynamics could therefore not be investigated. The availability of external relay services was also not part of the test setup and represents a potential disturbance variable. From a methodological point of view, it should be noted that the evaluation was deliberately limited to a simplified test application in order to minimise application-specific influences. Although this increases the comparability of the measurement results, it limits the transferability to complex real-world applications.

In summary, this work shows that iroh-gossip is a powerful gossip-based broadcast protocol whose strengths clearly lie in robust message distribution. At the same time, the results illustrate that discovery and bootstrap strategies have a decisive influence on practical usability and require special attention when developing applications with iroh-gossip. The relay mode, which is officially recommended by the iroh community as the discovery standard, should rather be implemented in combination with the direct mode and as a fallback mechanism for it. In any case, it is advisable to provide multiple bootstrap peers and to explicitly monitor the join process.

7 Bibliography

Leitão, João, Pereira, José, and Rodrigues, Luís. 2007. Epidemic Broadcast Trees. *26th IEEE International Symposium on Reliable Distributed Systems.* 2007, pp. 303-310.

—. 2007. HyParView: a membership protocol for reliable gossip-based broadcast. *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks.* 2007, pp. 419-429.

merlinux GmbH. 2025. Delta Chat is a decentralised and secure messenger. *Delta Chat.* [Online] 2025. [Cited on: 9 December 2025.] https://delta.chat/de/.

n0. 2025. iroh-gossip. [Online] GitHub Repository, 6 November 2025. [Cited on: 9 December 2025.]

—. 2025. p2p that just works. *iroh Computer.* [Online] 23 October 2025. [Cited on: 9 December 2025.] https://www.iroh.computer/.

The Rust Foundation. 2025. iroh - module discovery. [Online] Docs.rs, 6 November 2025. [Cited: 9 December 2025.] https://docs.rs/iroh/latest/iroh/discovery/index.html.

van der Sar, Ernesto. 2010. Facebook Uses BitTorrent, and They Love It. *Torrentfreak.* [Online] June 2010. [Cited: 9 December 2025.] https://torrentfreak.com/facebook-uses-bittorrent-and-they-love-it-100625/.

Vass, Jozsef. 2018. How Discord Handles Two and a Half Million Concurrent Voice Users using WebRTC. *Discord.* [Online] 10 September 2018. [Cited: 9 December 2025.]

8 List of figures