ReP2P Matrix: Decentralized Relays to Improve Reliability and Performance of Peer-to-Peer Matrix

Benjamin Schichtholz
benjamin.schichtholz@kit.edu
Karlsruhe Institute of Technology
Karlsruhe, Germany

Roland Bless
roland.bless@kit.edu
Karlsruhe Institute of Technology
Karlsruhe, Germany

Florian Jacob
florian.jacob@kit.edu
Karlsruhe Institute of Technology
Karlsruhe, Germany

Hannes Hartenstein
hannes.hartenstein@kit.edu
Karlsruhe Institute of Technology
Karlsruhe, Germany

Martina Zitterbart
martina.zitterbart@kit.edu
Karlsruhe Institute of Technology
Karlsruhe, Germany

Abstract

Matrix is a decentralized middleware for low-latency group communication, most renowned for its use in the Element instant messenger. Proposals for peer-to-peer (P2P) Matrix architectures aim to decentralize the current architecture further, which is based on federated servers. These proposals require that the receiver and the originator, or another peer that already successfully received the message, are simultaneously online. We introduce relay-enhanced P2P Matrix (*ReP2P Matrix*) in order to improve message delivery between peers that are online at different times. The design maintains the advantages of P2P Matrix and integrates well into it, e.g., it reuses existing mechanisms for authentication and authorization. Using an extended real-world group messaging traffic dataset, we evaluate ReP2P Matrix by comparing it to P2P Matrix without relays. The results show that relays do not only improve reliability in message delivery, but also increase the share of low delivery latencies by 50 % points in groups with up to 30 members.

CCS Concepts

• **Computer systems organization → Peer-to-peer architectures**; **Client-server architectures**; • **Networks** → *Peer-to-peer protocols*; • **Security and privacy → Network security**.

Keywords

Matrix; Decentralized Systems; Reliability; Asynchronous Delivery; Instant Messaging; Overlay Networks

ACM Reference Format:
Benjamin Schichtholz, Roland Bless, Florian Jacob, Hannes Hartenstein, and Martina Zitterbart. 2024. ReP2P Matrix: Decentralized Relays to Improve Reliability and Performance of Peer-to-Peer Matrix. In *Proceedings of the ACM Conext-2024 Workshop on the Decentralization of the Internet (DIN '24), December 9–12, 2024, Los Angeles, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3694809.3700741

DIN '24, December 9–12, 2024, Los Angeles, CA, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1252-4/24/12
https://doi.org/10.1145/3694809.3700741

1 Introduction

Instant messaging platforms play an important role in today's online communities. Popular platforms such as WhatsApp, Signal, WeChat, Telegram and so on, are logically centralized server-based solutions. They are also closed platforms and allow their providers to gather various user-related metadata (e.g., IP addresses, online times) even if the actual message content is end-to-end encrypted. Matrix [19] is an open decentralized middleware for low-latency group communication, most renowned for its use in the Element instant messenger [11]. Matrix employs a federated architecture in contrast to the centralized platforms. Users take part in the network only via their chosen Matrix homeserver as proxy, and for every chat group, the homeservers of all participating users form a federation of equals.

Nevertheless, even Matrix' federated architecture faces challenges regarding centralization and privacy. In 2019, the authors of [10] have shown that 87 % of Matrix users were concentrated on 1 % of homeservers, resulting in a load centralization towards only few homeservers. Homeservers store a replicated data structure that contains all *events* related to a specific chat room. Events can refer either to a user action (e.g., sending a message) or room state change (e.g., setting a user's privileges). Metadata privacy is a concern in the federated architecture, regarding both information stored explicitly in the replicated data structure (e.g., user IDs) and implicit information that homeservers can obtain by observing users' traffic patterns (e.g., online times). Privacy issues intensify with a higher degree of centralization, because large servers with many users obtain more metadata.

Peer-to-peer (P2P) Matrix architectures [1, 5, 7] have already been proposed in order to address these challenges. In contrast to federated Matrix, users do not rely on homeservers in P2P Matrix, but store the replicated data structure themselves, making it inherently more decentralized. However, the fully decentralized nature of P2P Matrix implies the cost of peers having to be online at the same time with at least one other peer within the same room, in order to exchange events. In an extreme case, a peer wants to send a message before going offline itself, while all other peers are offline. An expectation from federated Matrix is that this message is delivered to any peer as soon as it returns online, whereas in P2P Matrix, peers have to wait until the originator (or another peer that has already received the event) comes online again. Consequently, whether an online peer receives an event, depends on the online state of other peers.

We present relay-enhanced P2P Matrix (*ReP2P Matrix*) that enables peers to receive events even when other peers are offline. ReP2P Matrix preserves the decentralization benefits of P2P Matrix over the federated architecture, such as avoiding single points of failure and centralized storage of user metadata. To evaluate ReP2P Matrix, we extend a real-world messaging dataset with online periods modeled around the sending times and compare ReP2P with P2P Matrix based on this extended dataset. We show that ReP2P Matrix increased the share of delivery latencies under 2 s after returning online by 50 % points. Also, ReP2P Matrix increased the share of successfully delivered events by 28 % points on average.

2 System Model and Requirements

Since our approach enhances P2P Matrix, we define a system model that takes up design decisions and characteristics from existing P2P Matrix architecture proposals [1, 5, 7], such as homeserver-independent user IDs. Second, we specify both functional and qualitative requirements for a relay-enhanced P2P Matrix architecture.

2.1 System Model

We impose as few constraints on potential P2P Matrix architectures as possible in order to avoid dependencies on a particular architecture. In general, P2P Matrix does not employ homeservers, so all peers store their room history and manage the replicated data structure themselves. P2P Matrix allows peers to exchange and disseminate events via a concrete P2P network architecture, from which we abstract. Also, we assume that P2P Matrix provides end-to-end-encryption of event contents, whereas event metadata (e.g., user IDs, timestamps) is not encrypted, as it is the case in federated Matrix. P2P Matrix currently only supports synchronous delivery, i.e., in a two-peer room, both peers must be simultaneously online in order to exchange an event, as shown with event α in Fig. 1. Finally, our system model applies to the context of single Matrix rooms, as our focus is on optimizing the reliability event delivery, which occurs within rooms.

2.2 Functional Requirements

Asynchronous delivery. An event is delivered asynchronously from a sender to a receiver, if (i) the sender sends the event while the receiver is offline, and (ii) after sending, the receiver receives the event after returning online while the sender is offline. Two examples for asynchronous delivery are shown in Fig. 1, where the receiver receives the event β at t_2 and t_3. In the examples of Fig. 1, a relay stores the event β on behalf of the sender and forwards β when the receiver returns online, thereby providing asynchronous delivery.

ASAP delivery An event is delivered as-soon-as-possible (ASAP), if the event is received in the first online period of the receiver after the event is sent. A precondition for ASAP delivery is that the event delivery time must be shorter than the length of the first online period of the receiver after the event has been sent. In Fig. 1, the events received at t_1 and t_2 are delivered ASAP.

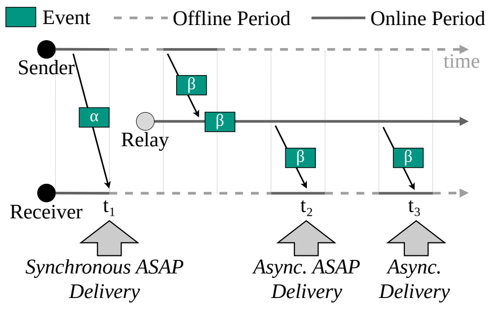


Figure 1: Synchronous vs. ASAP (as-soon-as-possible) vs. Asynchronous delivery in a minimal setting

2.3 Qualitative Requirements

Metadata Privacy. We consider two types of metadata: (i) event metadata, i.e., metadata explicitly stored in events, and (ii) peer traffic metadata, i.e., implicit information that relays can obtain by observing a peer's traffic patterns, such as the frequency of requests made by a peer to a relay. In contrast to homeservers in Federated Matrix, where one user is limited to using a single homeserver, it should be possible to distribute event metadata of a peer's rooms to several distinct relays. Within a room, it should be possible to distribute peer traffic metadata between relays. Additionally, no metadata should be disclosed to non-member peers. These non-member peers possess no room membership and are therefore not authorized to send or read events in a specific room.

Decentralized Relays. Decentralized Relays should not introduce a single point of failure, i.e., peers in P2P Matrix should be able to synchronously exchange events between each other even if no relays are available. Also, the usage of relays is optional, meaning that some peers in a room might use a relay while others do not. Optional relay usage should ensure that relays do not become a requirement for P2P Matrix. The solution should allow for distributing network load among the relays. Peers should be able to discover available relays from a set of different relays and choose freely among them.

3 ReP2P Matrix

Decentralized relays complement P2P Matrix in order to provide asynchronous and ASAP delivery for peers, without re-introducing the drawbacks related to homeservers. Figure 2 shows how events are disseminated within a room. A sending peer, i.e., the originator of the event, sends it to the peer distribution network, where it is distributed among the online peers via its P2P network architecture. At its discretion, the sender also sends the event to a relay from the *relay set*. The relay set of a room is defined as part of the room state, whereby peers can discover them. Only authorized peers (e.g., room admins) can modify the room state and add relays to a room. The receiving relay distributes the event to all other room relays. We assume that the room's relay set size is manageable (e.g., few dozens rather than hundreds). However, for larger relay sets, a more efficient dissemination mechanism among relays (e.g., multicast) may be necessary, but is out of scope for this paper. An asynchronous receiver that was offline during the time of sending

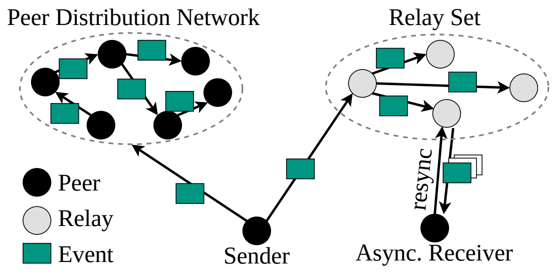


Figure 2: Event dissemination example in a P2P Matrix room with multiple relays

and that now returns online, can then resynchronize (*resync*) its local state with a selected relay, resulting in the original event(s) being delivered asynchronously to the receiving peer.

The design avoids changes to the existing Matrix protocol [20], it merely extends it with the relay-related functionality, such as the *resync* API endpoint. Moreover, peers are not required to use relays, they can proactively choose to profit from asynchronous delivery. Consequently, even if all relays are unavailable, peers can continue to exchange events with simultaneously online peers, i.e., via the peer distribution network.

3.1 Trust Assumptions

Regarding Metadata Privacy (Section 2.3), it is not possible to hide all event metadata from both non-member peers and relays while also providing Asynchronous & ASAP delivery (Section 2.2). Moreover, it has been shown that in instant messaging systems, performance (e.g., throughput, latency) decreases with stronger privacy and security guarantees [6]. We prioritize hiding event metadata from non-member peers, and make stronger trust assumptions about relays. Relays re-use existing Matrix mechanisms to authenticate peers [21], and are equipped with the necessary information to authorize peers (Section 3.2). A trusted relay forgets cached events over time (Section 3.3), so that if the relay is compromised, the attacker can only obtain a subset of the room's event metadata. Therefore, by avoiding metadata accumulation, trusted relays provide a weak form of post-compromise security. We note that event metadata may accumulate at malicious relays who disregard the policy of forgetting cached events over time. However, peers can split up traffic metadata by switching relays over time. For example, if a peer resynchronizes with relay *a* during one time period, and switches to relay *b* in subsequent period, the information about the peer's resynchronization activity (i.e., traffic metadata) is split between these two relays.

3.2 Authorizing Peers and Relays

Relays authorize peers, in order to prevent unauthorized peers (such as non-member peers) from retrieving event metadata or sending events to the relay. The replicated data structure that represents a directed acyclic graph (DAG), stores all room events, including the necessary information to authorize peers. However, because metadata should not accumulate at relays, relays store only a reduced DAG, the *AuthDAG*. The AuthDAG contains only those events relevant for authorization, e.g., membership or permission changes. To authorize a peer's resync request, a relay searches for the peer's room join event in the AuthDAG, and (if found) determines whether this join event itself is authorized within the room. This procedure reuses an existing Matrix authorization mechanism, as described in [22]. The authorization mechanism of relays is therefore also based on eventually consistent partial order without finality and without a consensus algorithm [9].

3.3 Caching Events on Behalf of Peers

Relays store message events in the *event cache*, until either their preemption due to lack of space or a configurable event retention time expires, thus reducing the amount of data stored at the relays, thereby improving post-compromise security. After receiving an event, relays store a tuple consisting of the event and the UTC timestamp of the local clock in the associated room's event cache. We accept best-effort local clocks to avoid the cost of global clock synchronization. Although clock drift can result in occasional missed events on resynchronization, we consider this acceptable given that peers hold the necessary information (i.e., the DAG) to identify and re-request these missing events. Upon receiving *resync* requests from peers, relays return a subset of events from the event cache, containing only events with timestamps higher than the provided *since* timestamp.

In theory, peers could *resync* all missing events based on the total order of cached events established by a relay's UTC timestamps. However, the *since* timestamp allows peers to only approximate the starting point of missing events. This required approximation is a result of relays not storing the complete DAG. Relays only passively cache events, without proactively re-requesting missing events, so that peers perform the partial ordering of events themselves without relying on a third party. This decision is made to enable P2P Matrix to function without relays, thereby preventing relays from becoming a new architectural requirement (Section 2.3) for P2P Matrix. As a result, peers do not necessarily receive a complete chain of events from the relay, but may have to either re-request missing events from other online peers, or retry the *resync* request directed to a relay with an earlier *since* timestamp.

3.4 Benefits over P2P Matrix

A problem for P2P Matrix is that peers only request missing events after having received the event subsequent to the missing event. Consequently, even if a peer has the latest room state and is online, another online peer missing an event is not guaranteed to receive this event. Relays circumvent this issue, because peers can resync with them after returning online, thereby improving the reliability of event delivery.

Moreover, ReP2P Matrix can help in reducing the peer device's energy consumption, which is especially relevant for mobile devices that are frequently used as instant messaging platform. First, if peers can obtain the latest state from relays, they do not have to provide this service themselves to other peers. Second, because events are disseminated between all relays in a room, the number of required connections to obtain the latest room state is reduced, while a relay-less solution would require discovering the peers that are both currently online and hold the latest room state.

Table 1: Comparison of Relay-enhanced P2P approaches from related work and Matrix approaches.
✓ represents a supported property, ✗ represents a property not supported, (✓) represents a partially supported property.

Property \ System	Nostr [4, 14]	SSB [18]	AMP2P [12]	Wesh [2]	Dendrite Relay API [8]	Briar Mailbox [13, 23]	P2P Matrix [1, 5, 7]	Fed. Matrix [20]	ReP2P Matrix
Asynchronous Delivery	✓	✓	✓	✓	✓	✓	✗	✓	✓
Avoid Metadata Accumulation	✓	✗	✗	✗	✗	✗	✓	✗	✓
Relay Selection	✓	✓	n/a	✓	✓	✗	n/a	✓	✓
Relays Authorize Peers	✗	✗	(✓)	✓	(✓)	✓	n/a	✓	✓
Single Point of Failure	none	none	none	none	none	none	none	homeservers	none
Relay Discovery	✓	✓	✗	✓	✗	✓	n/a	✓	✓

4 Related Work: Relay-enhanced P2P

In this section, we compare various relay-enhanced P2P systems providing asynchronous delivery with ReP2P Matrix. We also include P2P Matrix and federated Matrix in the comparison, in order to highlight the differences to ReP2P Matrix. The protocols Nostr [4, 14] and SSB [18] are mostly used as social media applications, in both of which peers can publish messages to relays, and retrieve messages related to a subscribed topic from relays. The Asynchronous Mobile P2P Relay (AMP2P) architecture [12] introduces relays to a mobile P2P network, aiming to provide asynchronous delivery and privacy, while focusing on encryption and authentication between peers and relays. Wesh [2], and Briar [13, 23] are both protocols for secure instant messaging, and provide relaying solutions that require relays to be associated to a user account. The Dendrite Relay API [8] allows peers to send events to a relay that are directed to a single other peer in P2P Matrix. As a result, if all other peers within the same room are to receive an event asynchronously, the Dendrite Relay API requires the sending peer to transmit the event multiple times to the relay, with each event instance directed to an individual peer in the room.

Table 1 compares different solution approaches according to various properties. While the first property, *Asynchronous delivery*, relates to the functional requirements (Section 2.2), the other properties relate to functionality derived from the qualitative requirements (Section 2.3).

Aside from P2P Matrix, all compared systems provide asynchronous delivery through mechanisms similar to those provided by relays in ReP2P Matrix. Federated Matrix provides this functionality at the cost of having federated homeservers, that are a single point of failure from a user perspective, in contrast to the other systems. If a user's homeserver is online, the user cannot participate in any Matrix rooms. Nostr avoids metadata accumulation at relays for private, direct messages (not for public posts) by encrypting metadata, such as user IDs or sending times. While Briar restricts users to a single relay, most of the other systems permit allow users to choose from multiple relays. Relays in Wesh and Briar can authorize other peers for sending or receiving messages by being linked to user accounts. AMP2P authorizes only sending peers (not receiving peers), while relays in the Dendrite Relay API only authorize the receiving peer. Discovering relays is possible by having relays themselves publish the set of available relays (Nostr), or making the relay set part of the group metadata (SSB, Wesh, Briar).

5 Generating Traffic Scenarios

In order to evaluate ReP2P Matrix, we take a real-world WhatsApp group messaging dataset [17], and evaluate only a subset due to resource constraints. The dataset includes 5956 WhatsApp chat traces, that were provided by more than 117 000 users and include groups with up to 252 members. Each trace includes sending times, anonymized user IDs, and the number of characters sent. However, due to different export formats, only 27 % of the traces have timestamps with a granularity of seconds, while the other traces provide a granularity of minutes. The interarrival times (i.e., intervals between two subsequent sending times) of the chats follow a long-tail distribution that fits a beta-prime distribution [17]. This characteristic accounts to both periods of frequent message exchange and long communication pauses.

The group messaging dataset has one major limitation: It does not include on- or offline periods of group members. However, these periods are essential for evaluating asynchronous delivery of messages. To the best of our knowledge, no real-world datasets for online periods in group messaging systems are currently available.

5.1 Deriving Online Periods from Sending Times

To profit from the dataset's traffic characteristics despite missing on-/offline data, we model online periods with the goal of providing a range of different online rates and lengths per chat. Our model establishes online periods around the sending times, by letting the peer be online for a certain time before and after sending. We call the duration between the time of sending and the time of the online state change of a peer the *online margin*. Applying an online margin to all sending times for each peer results in *online periods*, as depicted in Fig. 3. If the time between two subsequent sending times is shorter or equal to the online margin, the online period is extended further. In the example chat course of Fig. 3, such an extended online period is represented by Peer B's second online period. Online periods of different peers may also overlap, e.g., there is an overlap between the online periods of peers A and B shortly after peer A's sending time.

5.2 Interarrival Times

To evaluate a range of different online margins for each chat, we chose multiple online margin lengths. A relevant traffic characteristic to produce online margins are interarrival times (IATs), as these times capture the sending activity of peers, and also because peers can be assumed to be online at the time of sending. We seek

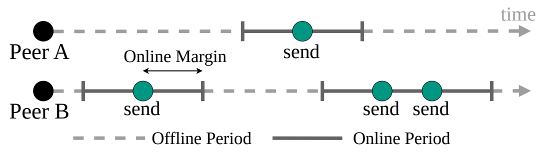


Figure 3: Applying online margins to sending times, resulting in several online periods

to generate a comparable rate of on-/offline periods between the chats, i.e., a low online margin should result in few overlapping online periods for the specific chat. If the same set of fixed online margins was used for a chat with longer IATs and another chat with shorter IATs, the shortest online margin would result in more overlapping periods in the second chat compared to the first chat. We specify five online margins for each chat by extracting its lowest and highest IAT, and exponentially increase the values between these two extremes. As a result, the five online margins are evenly spaced values on a logarithmic scale. We refer to the lowest online margin as *short*, the highest online margin as *long*, and the middle online margin as *medium*. This exponential growth addresses the long-tail distribution of the dataset's IATs, resulting in both bursty periods with short IATs, and communication pauses with long IATs.

An example chat with sending times from the group messaging dataset (chat id: 1796) and online periods generated from the five exponentially increasing online margins is depicted in Fig. 4. This chat illustrates the long-tail distribution of IATs, characterized by a bursty period with frequent message exchanges during the first day, followed by a longer communication pause of approximately 4 to 5 days. Figure 4 also shows that the generated online periods vary in scale, ranging from short durations (as highlighted in the box that "zooms in" on the first day) to longer online periods. Varying the online period ranges enables to assess the relay advantage in ReP2P Matrix towards P2P Matrix as the durations of overlapping online periods increase.

5.3 Selecting Data from the Dataset

The group messaging dataset contains 5956 chat histories. In order to measure temporal differences between sending times within minutes, we select chats with second-granularity timestamps. Evaluating every chat with five different online margins would result in long experiment durations, because we use a real-world proof of concept implementation. Also, to limit the time needed for each experiment, we select chats with up to 500 exchanged messages. We dismiss all chats with more than 30 group members, as the effect of relays is expected to decrease in larger groups, where the probability of peers being simultaneously online is higher. Additionally, chats with up to 30 group members account for 83 % of chats in the dataset [17]. We checked that IATs of chats with this upper-bound of exchanged messages are still representative for the beta prime distribution [17] of the dataset. Also, as the online margins depend on the IATs of the evaluated chat, we expect the results to also hold for longer chats. From the remaining dataset (up to 500 messages), we randomly sample 22 chats. Because each chat trace is applied to both P2P and ReP2P Matrix, and five online margins are applied separately to each chat, we ran 220 experiments in total.

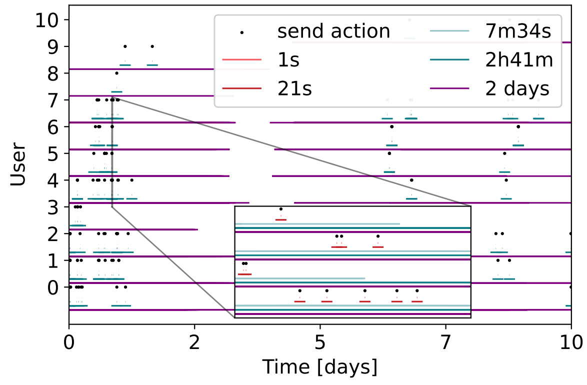


Figure 4: Applying exponentially increasing online margins to a chat from the messaging dataset. For each user, the send action is depicted, with the different online periods below.

6 Evaluation

We compare ReP2P Matrix to pure P2P Matrix without relays, based on the generated traffic scenarios described in the previous section. We provide an automated evaluation setup [15] that allows reproducible experiments for multiple peers and relays, and run our real-world proof-of-concept implementation [16]. Moreover, the setup can be used for experiments on future P2P Matrix architectures.

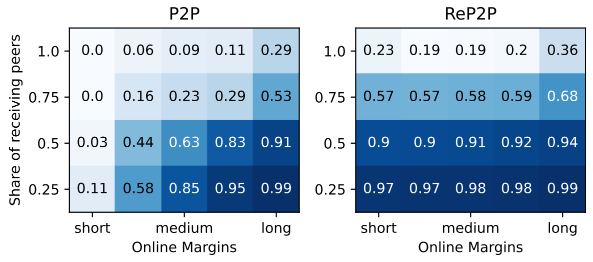
6.1 ReP2P Matrix Implementation

Current P2P Matrix implementations utilize the existing Matrix architecture, by having each peer run a local homeserver. Therefore, protocol changes required to transform federated Matrix into a P2P version are kept to a minimum, as communication between peers in P2P Matrix resembles communication between homeservers in federated Matrix. Existing P2P Matrix demos extend the Dendrite [3] homeserver with P2P functionality, and use various P2P networks [1, 5, 7].

We base the implementation of both peers and relays on the latest P2P Matrix demo, that uses Pinecone [1] as a routing scheme. Because not only peers, but also relays inherit the functionality of Pinecone, relays are also part of Pinecone's overlay topology and can exchange packets with peers or other relays via the same routing scheme.

6.2 Evaluation Setup

The automated evaluation setup allows running traffic scenarios for a configurable number of peers and relays, and evaluating their actions, e.g., sending/receiving events, or online state changes. The P2P network is emulated by running peers and relays in separate Docker containers, which are connected by a network bridge of the host system. As the proof-of-concept builds upon an existing P2P Matrix demo, the peers establish a Pinecone P2P network. The peers can then exchange Matrix events via Pinecone. Peers and relays log their actions during experiments, and the experiment log file is evaluated subsequently. We note that the obtained results are based on artificially generated online periods as explained in Section 5.1 due to lack of real data. Although these online periods model a



	short		medium		long
P2P					
1.0	0.0	0.06	0.09	0.11	0.29
0.75	0.0	0.16	0.23	0.29	0.53
0.5	0.03	0.44	0.63	0.83	0.91
0.25	0.11	0.58	0.85	0.95	0.99
ReP2P					
1.0	0.23	0.19	0.19	0.2	0.36
0.75	0.57	0.57	0.58	0.59	0.68
0.5	0.9	0.9	0.91	0.92	0.94
0.25	0.97	0.97	0.98	0.98	0.99

Figure 5: Comparison of successfully delivered events in P2P and ReP2P Matrix by share of peers and online margin, e.g., the bottom row shows the share of events received by 25 % of peers for the five online margins.

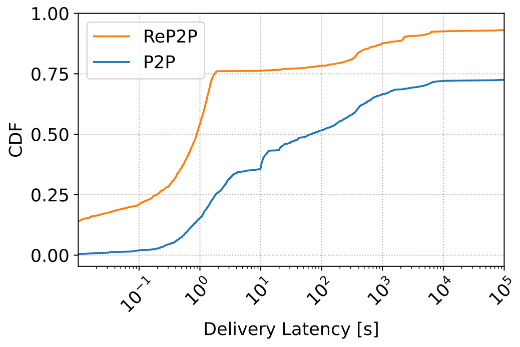


Figure 6: Share of event delivery latencies for first event received after returning online for medium online margins in each chat. These online margins range from 15 s to 3 h.

range of different overlapping online period frequencies, real online periods may substantially differ from the modeled periods, and depend on various factors such as location, cellular connectivity, daily activity, and the like.

6.3 Successful Event Delivery

We evaluate the overall share of delivered events and summarize the results in Figure 5. An event is considered delivered successfully, if a certain share of peers (e.g., 25 %) has received the event (y-axis). The share of delivered events by the share of receiving peers is evaluated for each five online margins (x-axis) per chat. The results show that ReP2P Matrix Matrix increases the share of delivered events by 28 % points on average. For the medium online margin, 63 % of all events are received by half the number of peers in P2P Matrix, while ReP2P Matrix increases the share of delivered events to 91 %. Even for the longest margin, where peers are often online simultaneously, ReP2P Matrix increases the share of delivered events. This accounts to the fact that in P2P Matrix, peers may not receive missing events despite being online simultaneously, as described in Section 3.4. We note that in both P2P and ReP2P Matrix only few events are delivered to all peers. This is because some members in the chat are inactive, rarely send messages and have only limited online periods, which reduces their chances of receiving any messages.

6.4 Event Delivery Latencies

We evaluate the times between a peer returning online and receiving the first event (if any). We name this metric *Event Delivery Latency*. This latency indicates how long it takes for a peer to obtain the first event from other peers/relays after returning online. We collected the event delivery latencies for all selected chats and measured the time from the peer coming online until it receives the first event within the online period. The cumulative distribution of delivery latencies of returning online for the medium-length online margins of all chats is shown in Fig. 6. For peers in ReP2P Matrix, 76 % of the delivery latencies are shorter or equal 2 s, compared to only 26 % for peers in P2P. This shows that relays do not only improve the successful delivery of events, they also reduce the delivery latency of the first events after returning online.

7 Conclusion

In this work, we introduce ReP2P Matrix [16], enabling asynchronous delivery between peers while maintaining the advantages of P2P Matrix over the federated architecture. The designed relay concept reuses existing Matrix authorization mechanisms and allows for using different relays over time so that relays get less information on user's IP addresses and how often peers are active/online. A proof of concept is implemented and evaluated in a reproducible setup, allowing future experiments on P2P Matrix.

We evaluate the performance of ReP2P Matrix compared to P2P Matrix without relays by applying a real-world WhatsApp group traffic dataset. Due to the lack of user on-/offline period datasets, we extend the dataset by deriving ranges of different online periods from the sending times. Our evaluation results show that in groups with up to 30 members, relays increase the number of successfully delivered messages and reduce the event delivery latency for peers coming back online. The advantage over P2P Matrix without relays depends on the peers' online durations and is more pronounced for small chat groups. Future work may consider designing more efficient inter-relay dissemination mechanisms (e.g., multicast).

Acknowledgments

This work was funded by the KiKIT (The Pilot Program for Core-Informatics at the KIT) of the Helmholtz Association.

References

[1] Neil Alexander. 2023. *Pinecone.* https://matrix-org.github.io/pinecone/ Accessed: 2024-08-03.
[2] Berty Technologies. 2023. *Wesh Protocol. Berty Documentation.* https://berty.tech/docs/protocol Accessed: 2024-08-04.
[3] Neil Alexander et al. 2024. *Dendrite (v0.13.8). GitHub Repository.* https://github.com/matrix-org/dendrite/tree/v0.13.8 Accessed: 2024-09-20.
[4] fiatjaf et al. 2024. *Nostr: Basic Protocol Flow Description.* https://github.com/nostr-protocol/nips/blob/master/01.md Accessed: 2024-08-04.
[5] Timothée Floure. 2019. *Experimenting with Matrix Federation over Yggdrasil.* https://www.epfl.ch/labs/dedis/wp-content/uploads/2020/01/report-2019-2-Timothee-Floure-Matrix-federation-over-Yggdrasil.pdf Accessed: 2024-08-03.
[6] Yossi Gilad. 2019. Metadata-private communication for the 99%. *Commun. ACM* 62, no. 9 (2019), 86–93. https://doi.org/10.1145/3338537
[7] Matthew Hodgson. 2020. *The Path to Peer-to-Peer Matrix. FOSDEM Presentation.* https://archive.fosdem.org/2020/schedule/event/dip_p2p_matrix/ Accessed: 2024-08-03.

[8] Devon Hudson. 2023. *Relay Server Architecture. Dendrite GitHub Documentation.* https://github.com/matrix-org/dendrite/blob/v0.13.8/relayapi/ARCHITECTURE.md Accessed: 2024-08-05.

[9] Florian Jacob, Luca Becker, Jan Grashöfer, and Hannes Hartenstein. 2020. Matrix Decomposition: Analysis of an Access Control Approach on Transaction-based DAGs without Finality. In *Proceedings of the 25th ACM Symposium on Access Control Models and Technologies* (Barcelona, Spain) *(SACMAT '20)*. Association for Computing Machinery, New York, NY, USA, 81–92. https://doi.org/10.1145/3381991.3395399

[10] Florian Jacob, Jan Grashöfer, and Hannes Hartenstein. 2019. A Glimpse of the Matrix: Scalability issues of a new message-oriented data synchronization middleware. In *Proceedings of the 20th International Middleware Conference Demos and Posters, Middleware 2019, Davis, CA, USA, December 9-13, 2019*. 5–6. https://doi.org/10.1145/3366627.3368106

[11] New Vector Ltd. 2024. *Element. A sovereign and secure communications platform.* https://element.io Accessed: 2024-10-12.

[12] Yevgeniy Dodis Max Skibinsky. 2015. Asynchronous Mobile Peer-to-peer Relay. (2015). https://s3-us-west-1.amazonaws.com/vault12/crypto_relay.pdf Accessed: 2024-08-04.

[13] Nico Alt. 2023. *Briar Wiki. Briar Project Documentation.* https://code.briarproject.org/briar/briar/-/wikis/home Accessed: 2024-08-04.

[14] Nostr. 2024. *A decentralized social network with a chance of working.* https://nostr.com/ Accessed: 2024-03-09.

[15] Benjamin Schichtholz. 2024. RеP2P Evaluation. GitLab Repository. https://gitlab.kit.edu/kit/tm/telematics/rep2p-matrix/rep2p-experiments.

[16] Benjamin Schichtholz. 2024. RеP2P Peer/Relay Implementation. GitLab Repository. https://gitlab.kit.edu/kit/tm/telematics/rep2p-matrix/rep2p.

[17] Anika Seufert, Fabian Poignée, Michael Seufert, and Tobias Hoßfeld. 2023. Share and Multiply: Modeling Communication and Generated Traffic in Private WhatsApp Groups. *IEEE Access* 11 (2023), 25401–25414. https://doi.org/10.1109/ACCESS.2023.3254913

[18] Dominic Tarr, Erick Lavoie, Aljoscha Meyer, and Christian Tschudin. 2019. Secure Scuttlebutt: An Identity-Centric Protocol for Subjective and Decentralized Applications. In *Proceedings of the 6th ACM Conference on Information-Centric Networking* (Macao, China) *(ICN '19)*. 1–11. https://doi.org/10.1145/3357150.3357396

[19] The Matrix.org Foundation CIC. 2024. *Matrix – An Open Network for Secure, Decentralised Communication.* https://matrix.org/ Accessed: 2024-08-08.

[20] The Matrix.org Foundation CIC. 2024. *Matrix Specification: Server-Server API.* https://spec.matrix.org/v1.11/server-server-api Accessed: 2024-08-06.

[21] The Matrix.org Foundation CIC. 2024. *Matrix Specification: Server-Server API. 3. Authentication.* https://spec.matrix.org/v1.11/server-server-api/#authentication Accessed: 2024-10-07.

[22] The Matrix.org Foundation CIC. 2024. *Matrix Specification: Server-Server API. 5.1.2 Authorization Rules.* https://spec.matrix.org/v1.11/server-server-api/#authentication Accessed: 2024-10-12.

[23] Torsten Grote and Michael Rogers. 2023. *Briar Mailbox released to improve connectivity. Briar Project.* https://briarproject.org/news/2023-briar-mailbox-released/ Accessed: 2024-08-04.