Peer-to-Peer Matrix

Quirin Heiler, Richard von Seck*, Jonas Jelten*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: q.heiler@tum.de, jelten@net.in.tum.de, seck@net.in.tum.de

***Abstract*—The current version of the Matrix network protocol relies heavily on the Domain Name Service and a Public Key Infrastructure, which stands in conflict with its overall decentralised design and makes it prone to censorship. This paper discusses a possible adaption to the Matrix APIs, which allows homeservers to operate in a self organized Peer-to-Peer manner, without the loss of any security assurances or the need for a centralized authority. The introduced approach will be based on the GNU Name System and tries to achieve a proper usability by allowing users to create human meaningful names for known conversation partners.**

***Index Terms*—matrix, p2p, gnu name system**

1. Introduction

The Matrix network tries to provide a generic and openly available service for message based communication. From the launch in 2014 [1], the network size has grown rapidly and is confirmed to have already reached a scale of more than one million users and 2500 homeservers by the year of 2017 [1]. The overall concept has thereby successfully proven to work in a real world, large scale application. However, at the current state of the project, the network is still inherently reliant on the use of the Domain Name Service (DNS) and consequently on a Public Key Infrastructure (PKI) to address and authenticate its homeservers. We believe that the removal of the dependence on these external, central authorities would benefit the system in several ways and comply with the projects core philosophies [2]. The key idea is that, given the ability launch Matrix homeservers in self a contained fashion and without having to rely on or even pay for an external service like DNS, more users will be able to operate their own servers, which would further increase the desired openness and decentralization of the network. Launching a personal homeserver comes with the additional advantage of allowing users to privately host their own conversations, thereby increases the control the user has over his personal data. Ultimately it is to mention that central authorities and DNS in particular [3] leave room for potential censorship, which should also be avoided when developing an open platform for communication.

This paper will propose an adaption to the current version of Matrix, which allows the operation of the underlying federation of homeservers in an open, in a fully decentralized and self controlled Peer-to-Peer (P2P) manner, without the need for a central authority. Additionally, it ensures, that our design fulfils the same standards of security and scalability as the original system, while also maintaining a reasonable degree of usability.

The introductory sections and the above mentioned statics are based on the well maintained documentation on the official matrix.org website [1], [2], [4]–[9]. The article about the Great DNS Wall of China, was published by Graham Lowe, Patrick Winters and Michael L. Marcus [3], it gives an example for the possible risk of DNS censorship. The central resource for this paper and the fundamental bases of the proposed concept was the master thesis of Martin Schanzenbach [10] about the GNU Alternative Name System. The habilitation of Christian Grothoff [11] about the GNUnet System contains the further descriptions of the GNU Name Service (GNS), which is the implementation of GADS, that we used in our design.

The paper will start by giving a basic introduction to the structure and functionalities of the Matrix network in Section 2., before Section 3. will discuss the challenges which have to be overcome when trying to find a suitable substitute for DNS. Section 4. and 5. give a short overview over GNS and present the actual design of our P2P network. The paper is concluded by a short introduction in available alternatives to GNS (Section 6.) and an evaluation (Section 7.) of the proposed solution.

2. Matrix

Matrix is an open standard developed by the Matrix.org Foundation. It defines a set of APIs with the goal to provide a free, globally available and decentralized service with no single point of control, allowing users to exchange persistent data in real time. The security of the system and the user's privacy is ensured through end-to-end encryption. Even though Matrix is still under development, there is already an implementation of the network available. Matrix tries to provide a simple generic interface, applicable to a variety of different use cases like instant messaging, IoT communication or to provide a signalling layer for webRTC based applications [4].

2.1. Basic functionality

The basic service provided by the Matrix network to an outside user is comparable to ordinary chat services. After generating an account at the Matrix homeserver

doi: 10.2313/NET-2019-10-1_08

of choice and acquiring a globally unique ID, users can start to access the system. All communication in Matrix is organized in rooms, which allow their members to publish/retrieve persistent data among/from all other users within. To enhance the user privacy, all room communication may optionally be end-to-end encrypted. Note however, that this communication is not limited to plain text messages. In fact Matrix can be used exchange arbitrary data between its users.

2.2. Network structure

While the current version of a matrix supports the use of webSockets and even includes a CoAP based ultra-low-bandwidth mode [4], we will focus on the default configuration and assume that all messages sent within the matrix network are HTTPS packages with JSON-objects (events) as their payload [4]. Going from there, the actual matrix network is formed between two main entities: Clients and homeservers.

2.2.1. Client Applications. Client [8] applications are important to the network in the sense that they represent the entry point for all user input. Aside from that, clients face a complete black box view of the actual Matrix network. Once connected to the user's homeserver, the further communication does not differ from the usage of contralised, server based web services. In that sense, the client only pushes events to the server as POST messages and stages pre-emptive GET requests to wait for answers. The crucial point is, that different clients in Matrix do never communicate to each other directly, even if they share the same homeserver.

2.2.2. Homeservers. The homogenous web of homeservers [9] is what actually makes up the network. They are responsible for storing their users' profile information and room state. Therefore every homeserver of all members of a room holds its own copy of the current room state. Since clients will only inform their own homeserver about new events, it is also the duty of every server to spread these events across all other servers in the room, while ensuring that every affected server will eventually obtain the exact same room state. This process of merging all incoming events of a room to a globally consistent state is called federation [9] and because it takes a vital role in the design of Matrix, the next section will be dedicated to explain the underlying processes.

2.3. Federation

This section will explain the algorithms behind the process of federation between multiple homeservers with the help of a simple example. We will assume a network layout as depicted in Figure 1.

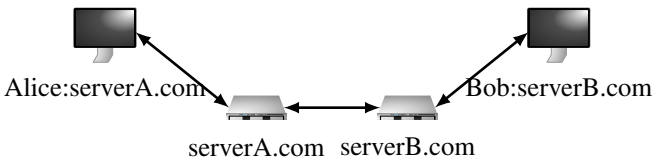


Figure 1: Network layout

The two users Alice:serverA.com (Alice) and Bob:serverB.com (Bob) try to exchange messages with the help of their respective homservers serverA (S_A) and serverB (S_B). The room they are using is assumed to be !roo:serverA.com (roo) (For more information about the naming scheme, see section 2.4). Each server models the room state as a directed, acyclic graph with a single root. We assume that the room was just created, hence the initial room state is empty (Fig. 2):

(a) Room state of S_A (b) Room state of S_B

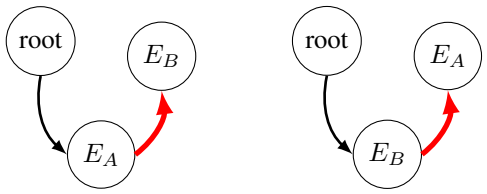
Figure 2: Initial state. Empty rooms.

Alice now pushes an event E_A to her homeserver. S_A examines its state graph and adds an edge from all existing events without an immediate child to the new event. In our scenario this results in exactly one edge from the root to E_A. Simultaneously, Bob pushes an event E_B to his homeserver. S_B reacts analogously to S_A. The resulting room state can be seen in figure 2:

(a) Room state of S_A (b) Room state of S_B

Figure 3: Room state after receiving first packages

Both homeservers now face the task of informing their respective counterpart about the newly received event. If they just forwarded the incoming messages to each other and treated the events received by other servers just like events from their own clients, both would end up with inconsistent versions of the state graph (Fig. 4):



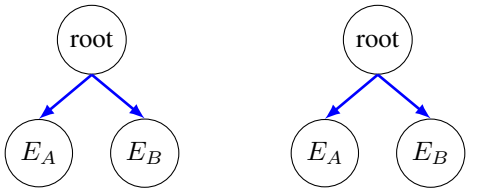
(a) Room state of S_A (b) Room state of S_B

Figure 4: Invalid federation. Inconsistencies marked red.

To circumvent this scenario, the servers do not only inform their counterparts about the existence of the new events, but also about the according parents of said event in the state graph. This will lead to the versions of the state graphs as depicted in figure 5.

Now each server holds a consistent copy of the room state and can pass the newly received events to their clients. However, this approach might not always be sufficient, as two messages in independent branches of the event graph might contain competing events. Imagine the events E_A and E_B in our example would both be attempts to rename the room to different names. To maintain a

doi: 10.2313/NET-2019-10-1_08



(a) Room state of S_A (b) Room state of S_B

Figure 5: Invalid federation. Consistent room states.

consistent room state, these events have to be handled identically at every involved homeserver. Specifically for situations like this, Matrix features so called room versions [5], which define a set of deterministic algorithms to resolve such situations. Different room versions can be applied to every room.

Note that the federation algorithm only ensures eventual consistency, but in no way the stability or immutability of any section of the graph, since new messages might have been delayed on their way through the network, they can reference any node in the state graph as their predecessor [5].

2.4. Discovery

As we already stated out, connections in the Matrix network are only established from client to server or from server to server. Therefore no network entity will ever have to resolve the IP address of a client before being able to build up a new connection. Servers on the other hand need to provide a static identifier as their name. This name can be either a static IP/Port-combination or a domain name with an according DNS Service record [9], [12]. User- and room-IDs are a triplet of three different values, containing: The address of a homeserver responsible for this user/room; A name which is locally unique within the scope of the homeserver; A specifier for the type of the address (i.e. room or user). This structure ensures that you will always be able to deduce the name, and with the help of DNS ultimately the IP-address of the server, responsible for a certain user or room, from the corresponding ID. To illustrate the concrete format, we will again look at the room from the example in section 2.3: !roo:serverA.com. In this case, "!" declares the type of the address and states out that it belongs to a room, while "roo" resembles the specific name of the entity and "serverA.com" is the address of the server where it was first allocated.

2.5. Third Party Integration

A key property of Matrix is that it does not simply provide a substitute for existing messaging platforms, but it creates possibilities to interoperate Matrix with other third party services.

2.5.1. Identity Service. The Matrix Identity Service API [7] allows users to create mappings from third-party identifiers like email addresses or phone numbers to Matrix IDs. These associations are managed by the so called identity servers. Other Matrix users can consult the identity servers to resolve a stored mapping, which allows to lookup a user's unknown Matrix ID from an already known third-party address. To ensure that users can only setup associations from third-party IDs that they actually own, the identity server provides a challenge which can only be solved by the owner of the given ID. In the scenario of creating a mapping from an email address to a matrix ID, the identity server would send an email containing a token to the given address. This token is then required to confirm the association.

2.5.2. Bridging. Matrix provides a whole variety of functionalities for the exchange of messages with third party services (e.g. email, facebook messenger, telegram, what's app, ...). So called Bridges [6] take the role of mediating between Matrix and third parties. Bridges can be designed as simple virtual users (bridge bots) [6], which take all received messages from one service and forward it to the other. More sophisticated approaches like Server-Server-Bridges [6] can be able to interconnect Matrix with other federated protocols (e.g. SMTP, SIP), by taking part in both server federations (matrix & third-party) and take the role of a general gateway between the two services. The overall advantage of Bridging is that it allows the usage of Matrix as a unified interface to communicate with arbitrary internet users, independent of what communication services they use.

3. Challenges

The central idea for our design of a P2P matrix is to keep the overall network structure and communication logics of matrix untouched and to only make minimal changes to the system. To achieve this, our approach will focus specifically on finding a reasonable P2P substitute for the DNS based naming system of Matrix. This section investigates the challenges and limitations, which arise when trying to build a fully decentralized naming system. Note that the considerations in the upcoming paragraph are based on [10] and are only applied to the context of Matrix.

Zooko's trilemma [13] is a hypothesis by Zooko Wilcox-O'Hearn, which suggests that there is always a trade off between three different special properties regarding the name system of a network protocol. To be more precise, these properties are decentralisation, memorability and security. Their meaning will be explained on the example of the current version of matrix:

Memorability. Matrix allows every user to freely pick a desired server name in the form of a web domain. As we expect that most users will prefer to pick simple and concise names, we can assume that these names are easily memorable.

Decentralisation. The name resolution process for server names is based on DNS, which is a fairly decentralised system. However, pure DNS does not impose any security features [10] like cryptographic authentication.

Security. To close this gap of security and to enable users to verify the authenticity of a DNS reply, additional mechanisms like TLS certificates [14] or the DNS Security Extension [15] are required, both of which rely on a Public Key Infrastructure (PKI) to issue, revoke and verify cryptographic certificates. The resulting hierarchy

of certificates, leads to a centralisation of trust [13], which originates from one or multiple trusted third parties. This means that in compliance with Zooko's trilemma, the gain in security is bought by sacrifices in decentralisation.

These considerations lead us to the following conclusion for the design of a P2P based Matrix system: If we want to no longer be reliant on a central, trusted third party, while also keeping Matrix resilient to adversaries, we will have to make sacrifices on the memorability of identifiers in order to increase the decentralisation of the network. However, keeping up a certain degree of memorability will be necessary for the usability of the system. The GNU Alternative Domain System (GADS) [10], is a fully decentralized naming system which was developed on exactly these considerations. The upcoming section will give a brief introduction into the GNU Naming System (GNS), which is a concrete implementation of GADS and part of the GNUnet alternative network stack.

4. GNU Name System

In GNS, every user is identified by a private-public key pair $(K_{priv}^{user}, K_{pub}^{user})$ and manages its own root zone. The ID (fingerprint) of a user's root zone is generated by hashing the public key of a user with SHA256 [16] (BASE32 [17] notation), it is considered globally unique. Zone files are comparable to their DNS counterpart and contain, among others, PKEY-records (reference the fingerprint of another zone file) and A/AAAA-records (for IPv4/IPv6). Each record of a zone file has a locally unique (A and AAAA records with the same name are allowed) name, picked by the owner of the zone. By creating additional key pairs, a single user can create and manage multiple zones. Users may sign their local zone files with the corresponding private key and make them publicly available for the network with the help of a censorship resistant Distributed Hash Table (DHT).

4.1. Name Resolution

The example network in figure 6 will be used to illustrate the address resolution process of GNS:

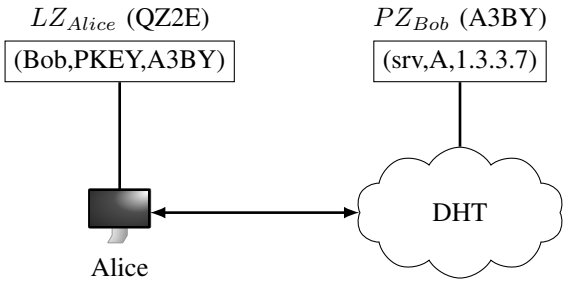


Figure 6: GNS example network state.

Bob wants to use his public zone file PZ_{Bob} with the fingerprint A3BY to advertise his server with the IP address "1.3.3.7". Therefore he creates an A record named "srv" in his zone file and uploads it into the DHT under its fingerprint "A3BY". Alice talked to Bob the other day and now she knows his zone's fingerprint. If Alice wants lookup the address of Bob's server, she has two equivalent options:

4.1.1. .zkey zone. With Bob's fingerprint in hand Alice can directly lookup Bob's server, analogously to DNS lookups by passing "srv.A3BY.zkey" to her GNS resolver. The ".zkey" Top Level Domain (TLD) is used to indicate that she wants to start the name resolution at the root of the global namespace. Thus the resolver will start by looking up Bob's zone file in the DHT, based on the identifier "A3BY" and afterwards retrieves the A record with name the name "srv".

4.1.2. .gnu zone. Unfortunately, the real fingerprint of a zone is way more complex than in our simple 4-digit example and therefore difficult to memorize. Luckily on the other hand, Alice has inserted a PKEY record named "Bob", which contains Bob's fingerprint in her local zone file LZ_{Alice}. She can now tell her resolver that it should start the address resolution process from LZ_{Alice}, by picking the ".gnu" TLD. The full GADS link would therefore be "srv.Bob.gnu". The resolver would now start by searching for entries in LZ_{Alice} named "Bob". Since "Bob" is a PKEY record the resolver will retrieve the contained finger print. The rest of the resolution process will then happen analogously to the ".zkey"-method.

In contrast to ".zkey"-addresses, which are valid globally, ".gnu"-addresses are only valid for a specific user, but at the same time allow the user to organize known own addresses by creating meaningful and easily memorable aliases.

5. P2P Matrix

This section presents our actual concept for establishing a fully decentralized, self organized, GNS based Matrix homeserver network (P2P Matrix). As a prerequisite we will assume that all clients and homeservers take part in the GNUnet system and thereby grant access to the GNS, while also providing bandwidth and storage for the GNS DHT.

5.1. Identification

We found that even in the current version of Matrix, it can be difficult to remember the address of a known user, because you do not just have to provide the actual username (which is possibly rather cryptic), you also need to know the name of the user's homeserver, which usually does not stand in any relation to the user. By introducing IDs that are created from public keys and therefore hard to remember, we expect this problem to get even more immediate. To address this issue, we want to minimize the contact a user has with these cryptographic identifiers and instead encourage the creation of personal aliases (pet names) for our system.

5.1.1. Homeserver IDs. Homeservers in P2P Matrix maintain a public GNS zone (Z_{server}). They are identified by their zone's fingerprint (F_{server}). This zone has to contain at least one A/AAAA record called "matrix", storing the IP address of the server. Every server can therefore be routed globally, by querying "matrix.FINGERPRINT_OF_THE_SERVER.zkey".

doi: 10.2313/NET-2019-10-1_08

5.1.2. Users IDs. When a user allocates a new account at a homeserver H. H will generate a public-private key pair ($K^{user}_{priv}, K^{user}_{pub}$) and publish a new zone file Z_{user} with the according fingerprint F_{user}. Z_{user} initially contains only one PKEY entry, which is called "home" and points to Z_H. Similarly to servers, users can now be globally identified by this fingerprint and their homeserver can be resolved by querying "matrix.home.FINGERPRINT_OF_THE_USER.zkey".

5.1.3. Room IDs. Similarly to user IDs, a room will also be associated with the fingerprint of a zone file containing a single PKEY entry pointing to the zone file of the server, which initially allocated the room (root). To distinguish room zones from user zones, this entry has to be called "root". As a result, the root server of a given room can now be looked up by querying "matrix.root.FINGERPRINT_OF_THE_ROOM.zkey".

5.2. Contact List

As already stated out in the preceding sections, every GNS user maintains at least one local zone file, which can be used to generate human readable mappings to known resources. Of course a user could now create mappings to known contacts in a private zone file at the client side, but since all the data in Matrix is stored at the homeservers and should be available even if the client logs in from another device, this approach would be rather insufficient.

"Remote local zone file". To address this issue, we will expand the Matrix client-server interface, by operations which allow a user to read and manipulate the Z_{user} file, stored at its homeserver. The user can therefore create easy to remember aliases which will then be resolved locally by the homeserver. The following example illustrates this concept (figure 7):

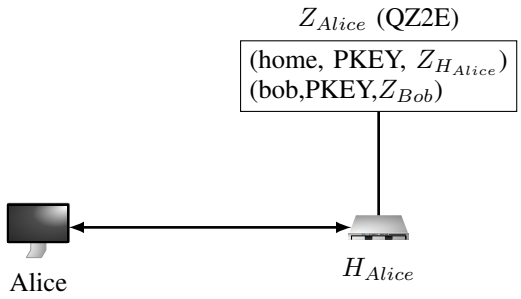


Figure 7: Example remote local zone file

Alice already knows Bob's ID and set a PKEY record named "bob" in her "remote local zone file" Z_{Alice}, located at her homeserver H_{Alice}. From this point on Alice will no longer need to specify Bob's ID when talking to her homeserver. Because the server knows that it is talking to Alice, which is identified by her zone file fingerprint Z_{Alice}, it can resolve the address of Bob's homeserver by looking up "matrix.home.bob.Z_{Alice}.zkey". The GADS resolver will realise, that the zone file Z_{Alice} is already present locally and will not have to resolve this first zone over the network. Bob's ID on the other hand can be read directly from the according PKEY entry in Z_{Alice}.

Note again, that GNS also allows users to create private records in their local zone files, which will not be published. As Alice probably does not want to share all her contacts with the entire network, she would be able to only set private records in Z_{Alice} (excluding the home record). Nevertheless, since the homeserver only performs local lookups on her zone file, this change would not impair the overall functionality.

By the introduction of the "remote local zone file", we enable users to maintain a listing of known IDs and rename them to locally unique and easily memorable aliases, which can be accessed even when accessing the homeserver from different devices. While the above example only mentioned user IDs, the concept works for room IDs analogously.

6. Alternatives to GNS

The concept of introducing a fully decentralized alternative for DNS is not entirely new to the domain of P2P networking. To serve as a bases for upcoming comparisons, this section will introduce two well known P2P alternatives to our GNS based approach.

6.1. TOR

".onion"-addresses are used by the TOR [18] network to advertise hidden services. Conceptually, they are comparable to GADS ".zkey"-addresses, they follow the form "CRYPTOGRAPHIC_IDENTIFIER.onion" and are advertised over a Distributed Hash Table. Note, that each such DHT entry does not contain a full zone file, but only an individual IP address. ".onion"-addresses do not provide a way for users to create human memorable aliases. In that sense, they might be considered a minimal archetype of P2P naming systems.

6.2. Invisible Internet Project

The Invisible Internet Project [19] (I2P) differentiates between two address types. The basic type works similar to the aforementioned ".onion"-addresses, with the only difference that the top level domain is ".i2p" instead. The second type are locally valid aliases, which can be set by the user and are being stored in the so called address book. Aside from being able to manually read and write entries to their own address book, users can also subscribe to public address book files of other users. The latter feature allows users to discover new addresses without ever coming in contact with the underlying cryptic identifiers.

7. Evaluation

This section evaluates the P2P matrix design, established in Section 5, against the initially proclaimed properties of security, usability and scalability.

7.1. Security

The presence of malicious notes in open P2P networks is an unpleasant, yet unavoidable truth. Consequently, it is vital for our system to prove resilience against a variety of different attacks.

doi: 10.2313/NET-2019-10-1_08

7.1.1. Authenticity.

Secure Addresses. Since Matrix can be used to exchange highly confidential data, the naming system has to allow users and servers to safely authenticate their communication partners. This is ensured in the original Matrix with the help of cryptographic certificates, which are issued by an external certification authority. All of the introduced P2P naming systems on the other hand do not require such certificates, since the information necessary for validating cryptographic signatures of a domain owner is already contained in the cryptographically generated domain itself. In that sense, GNS/TOR/I2P addresses can be considered secure by design [10]. To put this into the perspective of the previous considerations on Zooko's trilemma: The described P2P services are sacrificing the memorability of their domains to achieve a high level of security and decentralization. TOR hidden services demand similar authenticity assurances as ordinary websites. The large scale deployment of TOR and its hidden services can therefore be considered a proof of this concept of self certifying addresses.

Fuzzy fingerprinting. The preceding considerations about authenticity in ".onion"-addresses are based on the assumption that every address is bound to a globally unique key pair and can therefore not be claimed by any entity other than the actual owner of said key pair. However, this assumption does no longer hold for partially matching addresses. As a result, it is a practice for attackers to generate ".onion"-addresses with a similar appearance to an actual address [20]. The aim of this attack is to abuse the user's laziness, when comparing the attacker's address to the original and thereby trick the user into inadvertently accessing the attackers address. Notice that this attack becomes increasingly effective if the user frequently accesses the same service and assumes to recognise its correct address by just quickly skimming it.

While an attacker could still try to perform a similar attack in our P2P Matrix, it is expected to be way less effective. A custom pet name system like in GNS or I2P, helps to minimize the risk for users to fall for this attack, since it enables the user to access the given address without the need of constantly supplying its cryptic and easily mistakable identifier, thus avoiding the risk to mistake it with a similarly looking ID [10].

7.1.2. Censorship. The strong authenticity assurances of GNS, combined with the randomized routing algorithm and the redundant storage of entries of its R5N [21], [22] based DHT make it difficult for attackers to manipulate or deny access to other user's zone files [11], [21]. This property makes P2P Matrix resilient to potential censorship by manipulation or denial of its naming system.

7.2. Usability

An important goal of our design was to maintain a high level of usability by minimizing the exposure a user has with the cryptographic GNS IDs, yet providing an understandable naming scheme. As a result, every Matrix entity can be addressed with only a single GNS ID, while the corresponding addresses of an entities homeserver (see sections 5.1.2 & 5.1.3), provide the same information as addresses in the original Matrix, as they: Provide a globally unique identifier for the entity; Resolve to the IP address of the responsible homeserver; Hold type information (i.e. "home" for users/ "root" for rooms). This structure even yields advantages to the current version of Matrix, where users have to memorize pairs of arbitrarily chosen server and user/room IDs. The introduced pet name system allows users to setup easily memorable aliases for already known conversation partners and further enhances the usability.

However, this simplification will not improve the overall discovery of previously unknown users/rooms. Especially the field of verbally exchanging ID information is expected to suffer from the newly introduced, hard to memorize GNS identifiers. The impact of this problem could be reduced by encouraging to exchange public key information with the help of mobile devices and technologies like QR-codes or by using the Matrix Identity Service to create mappings from third party addresses (e.g. email) to Matrix IDs. While the latter option would allow users to reach a user experience comparable to the current version of Matrix, it can not be considered an optimal solution, since P2P Matrix is aimed to function without the dependence on third party services.

7.3. Scalability

An important aspect, which is yet to be covered is the question whether GNS yields the ability to work in real world, large scale environments. R5N is an extension of the Kademlia [23] algorithms and inherits the capabilities of dealing with the dynamics of leaving and joining peers in large scale networks [24] (churn). The R5N routing algorithm allows looking up and depositing zone files in the time complexity $\mathcal{O}(\sqrt{n} * \log n)$ [21](where n is the number of peers in the network), thus manages to maintain efficient lookups, even in large scale networks.

8. Conclusion and Future Work

The central goal of the Matrix network to build a decentralized platform for communication which gives the user maximum control over his personal information align with the benefits which would arise from rebuilding matrix into true P2P system, which does not rely on central authorities. This paper introduced the design for such a P2P matrix, which utilizes the GNU Naming System as a secure and scalable alternative to the Domain Name System to create a simple, hierarchical naming scheme. Even though the approach focused heavily on the use of a pet name system to maintain a high usability, the effects of using "randomly generated" IDs will be noticeable by the users (especially without making use of 3rd party IDs and the Matrix Identity Service), as it will become more difficult to discover new users within the network.

A proposal for a future extension of the introduced design could be to reduce this problem by allowing users to find new contacts within Matrix itself. Following a simple "friends-of-friends" logic it can be assumed that users are likely to interact with the contacts of their

own contacts. Users could be allowed to subscribe to the contact list of other users in the style of I2P address books, in order to automatically accumulate new contacts without relying on cryptographic identifiers at all.

To actually find out how our design works in a real world application and whether it resembles an improvement to the current state of Matrix, it would be the next logical step to actually build and test a minimal prototype. Since both services follow a similar internal structure, it should even be rather simple to create a Server-Server-Bridge between the official version of Matrix and a P2P prototype to run them in full interoperability.

References

[1] "Matrix statistics," https://web.archive.org/web/20190515055721/https://matrix.org/blog/2017/07/07/a-call-to-arms-supporting-matrix/, accessed: 2019-05-15.

[2] "Matrix manifesto," https://web.archive.org/web/20190808012751/http://matrix.org/foundation/, accessed: 2019-08-08.

[3] G. Lowe, P. Winters, and M. L. Marcus, "The great dns wall of china," *MS, New York University*, vol. 21, p. 1, 2007.

[4] "Matrix Main Page," https://web.archive.org/web20190620111226/http://matrix.org/, accessed: 2019-06-20.

[5] "Matrix specification," https://web.archive.org/web/20190611230822/https://matrix.org/docs/spec/, accessed: 2019-06-11.

[6] "Matrix Types of Bridges," https://web.archive.org/web/20190803182830/https://matrix.org/docs/guides/types-of-bridging/, accessed: 2019-06-20.

[7] "Matrix Identity Service," https://matrix.org/docs/spec/identity_service/r0.2.1, accessed: 2019-08-08.

[8] "Matrix client-server-api," https://web.archive.org/web/20190808012751/https://matrix.org/docs/spec/client_server/r0.5.0, accessed: 2019-08-08.

[9] "Matrix server-server-api," https://web.archive.org/web/20190808012750/https://matrix.org/docs/spec/server_server/r0.1.3, accessed: 2019-08-08.

[10] M. Schanzenbach, "Design and implementation of acensorship resistant and fully decentralizedname system," 2012.

[11] C. Grothoff, "The gnunet system," Ph.D. dissertation, 2017.

[12] A. Gulbrandsen and L. Esibov, "A dns rr for specifying the location of services (dns srv)," 2000.

[13] "Zooko's trilemma," https://web.archive.org/web/20040616080110/http://zooko.com/distnames.html, accessed: 2019-08-28.

[14] P. Saint-Andre and J. Hodges, "Representation and verification of domain-based application service identity within internet public key infrastructure using x. 509 (pkix) certificates in the context of transport layer security (tls)." *RFC*, vol. 6125, pp. 1–57, 2011.

[15] M. Larson, D. Massey, S. Rose, R. Arends, and R. Austein, "Dns security introduction and requirements," 2005.

[16] T. Hansen, "Rfc 6234-us secure hash algorithms (sha and sha-based hmac and hkdf)," 2011.

[17] S. Josefsson, "The base16, base32, and base64 data encodings," 2006.

[18] D. McCoy, K. Bauer, D. Grunwald, T. Kohno, and D. Sicker, "Shining light in dark places: Understanding the tor network," in *International symposium on privacy enhancing technologies symposium*. Springer, 2008, pp. 63–76.

[19] F. Astolfi, J. Kroese, and J. Van Oorschot, "I2p-the invisible internet project," *Web Technology Report*, 2015.

[20] "Fuzzy Fingerprints - Attacking Vulnerabilities in the Human Brain," https://github.com/vanhauser-thc/THC-Archive/blob/master/Papers/ffp.pdf.

[21] N. S. Evans and C. Grothoff, "R5n: Randomized recursive routing for restricted-route networks," in *2011 5th International Conference on Network and System Security*. IEEE, 2011, pp. 316–321.

[22] C. Grothoff, "The gnunet dht."

[23] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *International Workshop on Peer-to-Peer Systems*. Springer, 2002, pp. 53–65.

[24] Z. Ou, E. Harjula, O. Kassinen, and M. Ylianttila, "Performance evaluation of a kademlia-based communication-oriented p2p system under churn," *Computer Networks*, vol. 54, no. 5, pp. 689–705, 2010.

doi: 10.2313/NET-2019-10-1_08