On Empowering End Users in Future Networking

Tianyuan Yu
UCLA
Los Angeles, USA
tianyuan@cs.ucla.edu

Xinyu Ma
UCLA
Los Angeles, USA
xinyu.ma@cs.ucla.edu

Lixia Zhang
UCLA
Los Angeles, USA
lixia@cs.ucla.edu

Abstract

In today's Internet, end users communicate largely via cloud-based apps, and user data are stored in cloud servers and controlled by cloud providers. Recent years have witnessed multiple efforts in developing decentralized social apps with various design approaches, although the community at large is yet to fully understand the effectiveness, viability, and limitations of these different designs. In this paper, we make a proposition that a necessary condition of moving towards Internet decentralization is enabling *direct* user-to-user (U2U) communications, and discuss the design choices in several decentralization efforts and identify their limitations. We then articulate why a DNS-derived namespace is the best choice in U2U app developments in general, and use a recently developed decentralized app, *NDN Workspace* (NWS), as an example to show how NWS' use of DNS-derived namespace enables secure U2U communications.

CCS Concepts

• **Networks** → **Network architectures**; • **Security and privacy** → **Network security**.

Keywords

Decentralization, DNS, Applications

ACM Reference Format:
Tianyuan Yu, Xinyu Ma, and Lixia Zhang. 2024. On Empowering End Users in Future Networking. 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, 6 pages. https://doi.org/10.1145/3694809.3700745

1 Introduction

We live in an Internet age where people access Internet everywhere and everyday. According to Forbes Internet Usage Statistics [1], on average, people spend 6.5 hours on the internet every day.

In this paper, we define users as human users and user-owned devices, and define user data as data produced by users or their apps. In its early years, Internet applications largely operated on end user or user-organization provided servers (e.g. email servers, FTP servers), and users had full control over their apps and data. However, the Internet consolidation over recent years has resulted in today's Internet applications, by and large, run in the cloud. Users communicate via apps on cloud platforms, with the resulting user data stored there and their usage controlled by cloud providers.

We believe that the fundamental goal of decentralizing the Internet is to decentralize the control power, so that users can *take (back) the control of their apps and data.* Towards that goal, multiple different approaches have been explored, including regulations [12, 19], decentralized social apps [2, 7, 16, 18], and decentralization through anonymous blockchains (*e.g.,* Ethereum dApps).

In this paper, we make a proposition that a necessary condition to Internet decentralization is to enable *direct and secured User-to-User* (U2U) communications, which can potentially be many-to-many. This approach enables users to run applications on their own devices which communicate directly and securely, eliminating the reliance on any intermediaries whose functions may impact users communications or the exchanges of user produced data. Since all communications must be secured, this means U2U communications must have security and security policies controlled by end users, without outsourcing to third parties.

To support our proposition, we first summarize several existing efforts on decentralized apps and extract a few lessons learned, and articulate why DNS namespace makes the best choice in developing decentralized apps with U2U communications. We then use a decentralized collaborative app, NDN Workspace [22] (NWS), as a proof of evidence to show how it achieves user-controlled U2U communication by assigning DNS names to all end users and user produced data. NWS also shows that such decentralized apps can make use of, and can greatly benefit from, infrastructure provided resources, while having users retaining the full control of their apps and data.

Specifically, this paper makes two contributions. First, we argue that U2U communication is necessary for developing decentralized apps, which in turn is necessary in Internet decentralization. Second, we demonstrate that a DNS-derived namespace is the best namespace choice to enable U2U communications, use NWS to illustrate how it enables data-centric apps with built-in security protection, and articulate how named, secured data enable the integration of networking, storage, and computation.

The rest of the paper is organized as follows. Section 2 described various different designs made by current decentralized social apps, and articulates their limitations. Section 3 elaborates why DNS namespace is the best design choice for U2U in general. Section 4 offers a brief overview of NWS. Section 5 further discusses blockchain-based dencentralization solutions. Section 6 concludes the paper.

2 Decentralized Social Apps

The majority of existing decentralization efforts focus on developing decentralized social networking apps. This section provides a brief survey of several well-known decentralized social apps with two

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/10.1145/3694809.3700745

focuses: how users communicate with each other, and how much control users have over their apps and data.

2.1 ActivityPub and Mastodon

ActivityPub [20] and its best known application Mastodon [9] follow a *node-centric* design approach which led to building interconnected and supposedly independently owned application instances, each is identified by its DNS name. A Mastodon user registers with one of the application instance servers and obtains a name under the server's DNS name. Instead of letting users communicate directly to exchange their posts, Mastodon servers take responsibility to communicate with each other on behalf of their registered users, secure and store user data to be exchanged by HTTP. Users rely on Mastodon servers as intermediaries for identities and secure communications. Therefore, users' control over their data access is subject to their server administrators. The administrator of a server $S1$ can block exchanges with another server $S2$, thereby blocking all communications with the users on $S2$. Although a user can migrate to another server $S3$ when observing $S1$ misbehaviors, the migration leads to user name changes, which has to be handled by the user out-of-band. Furthermore, because intra-server communications are more efficient than inter-server communications, users tend to register with most popular servers [14]. This economy-of-scale effects defeats the design goal of interconnecting large numbers of independent servers for decentralization. At the time of writing, the top 10 popular among the total number of 16256 public servers are hosting 56.93% of the total accounts according to the Mastodon website [18].

2.2 Nostr and Secure Scuttlebutt

Nostr [2] and Secure Scuttlebutt (SSB) [16] use key hashes as user names. This approach makes a user's name self-certifying: one can prove his name ownership by proving the ownership of his private key. However, using key hashes as user names leads to users changing names whenever one needs to changes his cryptographic keys. Although solutions such as [6] exist to securely link the new key to the old one, they add additional complexity and overhead to the system. Moreover, since a user cannot identify others by memorizing their key hashes, the system must provide a binding of each key to a semantic name that human users can recognize. Neither Nostr nor SSB implements a systematic name-key binding solution, therefore the bindings are created and authenticated out-of-band.

Both Nostr and SSB support U2U communications through relays. Users publish their own signed records (called *events* in Nostr and *feeds* in SSB), and subscribe to other users' records and fetch them when they become available. In Nostr, signed records form append-only logs hosted by third-party, isolated *relays* [3]. Since key hashes make a flat namespace, it prevents scalable forwarding among Nostr relays. Therefore, a subscriber must connect to at least one relay that its subscribed publisher is connected to, but there is no instruction on how a publisher should select its connected relays in order to reach its subscribers. As the Nostr creator writes, "there are no guarantees of anything...to use Nostr one must embrace the inherent chaos" [4].

SSB builds a gossip replication network based on social graphs made of followers and followees. SSB users on the same LAN can discover each other via local multicast. Then one fetches the feeds from users that one explicitly follows, and also from users followed by followed users (up to three hops by default). There are also non-federated relays (called *Pub* nodes) on the Internet serving as SSB rendezvous point users. Functionality wise, Pub nodes are regular SSB users with additional functions of being publicly accessible and always online, allowing other users who learns them from out-of-band channels to connect to them to synchronize their feeds. By design, SSB does not specify how Pub nodes authenticate and authorize users. If users want to send private messages, they will directly encrypt the message with recipients' public keys.

Flat and self-certifying usernames enable users to directly control their data, but at the cost of requiring users to handle peer authentication themselves, lacking the ability to express policies, and causing inefficient communication based on application relays.

2.3 AT Protocol and Bluesky

Bluesky [7] is another social app built on the AT Protocol (Authenticated Transfer Protocol), which is a social networking protocol designed to enable user-controlled user identities. Similar to Mastodon, a Bluesky account is affiliated with a third-party custodian server, which is known as Personal Data Servers (PDS) that manage user key pairs and store signed user data. Different from Mastodon which assigns each user a name under the name of a specific server, each Bluesky user can register a name under "`.bksy.social`" for free, and one's data could be named with URIs under one's namespace. Meanwhile, Bluesky is one single application instance, rather than an instance federation (*e.g.,* Mastodon).

However, Bluesky does not directly use semantic user names in communications. Instead, each user is identifier by a *decentralized ID* (DID) [15], in the form of "`did:plc:z72i7hdynmk6r22z27h6tvur`". When creating a user account, Bluesky by default first assigns the user a name under "`.bksy.social`" and generates a key pair for the user. It then creates a *DID document*, a JSON document containing information about the user's public key, DNS name, the URL of PDS (Bluesky hosts a public PDS for user as default) and other information. The user's DID is the SHA256 hash of the *initial* DID document [13] which does not change when the document gets updated later. Bluesky requires a global database to map each user's DID to a DID document that contains the user's information. This mapping service is currently provided by DID directory servers[1] which are controlled and operated by Bluesky Social PBC[2]. A user who already owns a DNS name (*e.g.,*"`foobar.org`") can add his DNS name to the DID document *after* getting his DID. In this case, Bluesky requires the user to prove the DNS name ownership by either serving his DID as a DNS TXT record, or returning the DID in response to a HTTPS request to an appointed URL under the user's DNS name.

Bluesky identifies user data with names under each user's DID, and PDS signs named user data by the key stored in the DID document. For example, a Bluesky signed post produced by user "`did:plc:z72i7hdynmk6r22z27h6tvur`" can be named with URI "`at:`

[1]https://web.plc.directory/
[2]PBC stands of Public Benefit LLC.

/did:plc:z72i.../app.bsky.feed.post/3k43tv4rft22g", with category "app.bsky.feed.post" and content identifier "3k43tv4rft22g". Currently, users from different PDSes can exchange data using Bluesky *Indexer*, an Bluesky infrastructure that crawls all knwon PDSes and notify PDSes for relevant data production. Through the Indexers, Bluesky users (in)directly communicate, control apps and user data. At the time of writing, most of Bluesky Indexers are operated by Blueksy Social PBC.

On the path to regaining user control, Bluesky users directly control their data through owning their data signing keys. However, its user-to-user communications rely on app-specific intermediaries, meaning users can only indirectly control their apps through trustworthy application infrastructures.

2.4 Lessons Learned

An important lesson we learned through examinating the aforementioned decentralized social apps, is the necessity of U2U communications in order to (re)enable user-controlled apps and data. Having users owning their keys would enable user-controlled data, and direct communications would remove application intermediaries that may hold communication control from users.

Another lesson is that user names have to be both self-controlled and semantic. Mastodon names users under their server's DNS names, resulting users not owning the names nor controlling their data's security. Neither naming servers (instead of users) nor naming users by key hashes enables efficient U2U communications. Users need semantic names to allow one to easily identify others and define security policies by names. Flat user identifiers used in Nostr and SSB cannot serve such purposes. In actual usage, users will need to bind keys with semantic names; how to securely provide that binding is an unanswered question in both designs. Bluesky solves the binding problem by identifying users with both DNS names and flat DIDs, and storing the DID-name binding in DID documents, introducing a dependency of centralized directory servers to resolve DID into names.

Based on the foundation of self-controlled, semantic user names, one also needs three additional elements to enable secured, efficient U2U communication in global scale: semantic data naming, securing data directly, and named-based data delivery. Designs without semantic data names (*i.e.,* Nostr, SSB) cannot systematically define security policies on *who* can read or write *what* data. Not directly using semantic names in U2U data delivery (*i.e.,* Bluesky and Mastodon) would create reliance on app-specific infrastructures that are under control of their providers (*e.g.,* DID directory).

3 DNS as a U2U Communication Enabler

The previous section argued that supporting U2U communication is a necessary condition in moving towards Internet decentralization. This section further elaborates that the DNS-derived namespace is the best enabler for U2U communications.

3.1 Why DNS

Current Internet operations rely on two namespaces: DNS and IP address (including both IPv4 and IPv6). Letting users communicate by using DNS names directly brings several advantages in enabling U2U communications.

- The Domain Name System is Internet's namespace used by almost all today's applications. DNS names are semantically meaningful and are directly usable by human users.
- DNS namespace is hierarchical and managed in a decentralized manner [8], and has no foreseeable upper-bound in the number of delegated names[3]. By design, there should be no obstacle for all users to obtain a DNS name, which could be purchased from a DNS registrar[4], or delegated by organizations or user communities[5].
- There exist mature solutions to authenticate DNS names. Web PKI has been widely deployed and used for decades, and client DNS name authentication solutions are being developed [5, 21].
- General Internet users cannot directly communicate using IP addresses. In addition to NAT, Internet connectivity is badly clotted by large number of firewalls with undisclosed filtering policies. Domain names are independent from topological connectivity. Users can use any viable VPN tunneling solutions to create an overlay network where users directly communicate by using DNS names.

Today's DNS system includes two parts: a managed global namespace, and a DNS resolution infrastructure where resolvers and servers communicate via the DNS protocol. Note that using DNS names does not necessarily imply reliance on the DNS resolution infrastructure. Bluesky (Section 2.3) names users by domain names, but only the initial step that authenticates a user as a domain owner rely on DNS infrastructures. After security bootstrapping, Bluesky user data security is independent from it.

3.2 Utilizing DNS Names for U2U Apps

In addition to delegate DNS names to users, we can also delegate DNS names to U2U applications in the following way. A user U can initiate an app instance and delegate to it a DNS name N_a owned by U, and further delegates subdomains under N_a for users joining the app. U can have a self-signed certificate, and issue a certificate to his app, which can further issue certificates to user subdomains. Users can name their data by URI-like names, and sign the data. Note that user certificates are also named, secured data.

In this design, although user and data identities are derived from the DNS namespace, their authentication can be made *independent* from Web PKI or DNS infrastructure if users have other means to authenticate each other. Other than the initiating user needs to obtain unique app names from the DNS system, this design lets the user control the data naming and security protection.

4 NDN Workspace Overview

In this section, we use NDN Workspace (NWS)[6] as an example to illustrate how to use DNS-derived namespace to build a secure U2U app that is fully controlled by end users. NWS is a web-based, decentralized app running over Named Data Networking (NDN) [26] which provides secure, asynchronous, and collaborative document editing and enables users to communicate via any connectivity. It

[3]We acknowledge that RFC 1035 specifies the maximum length of a domain name in implementations.
[4]Cost should not be an issue: short names can be expensive but longer and unpopular names can be cheap
[5]In practice, automating namespace management remains a research topic.
[6]https://ndn-workspace.web.app/

offers users functions similar to that of Overleaf, where users can create projects and invite their collaborators to join; each project makes a *NWS instance* (instance for short). It assumes collaborators have out-of-band trust relations and have already shared public keys with each other before they start collaborating on a document using NWS.

The rest of this section describes how NWS empowers users by using *DNS-derived names* and secured named data. To aid the reader's comprehension, we use an example to illustrate the NWS design.

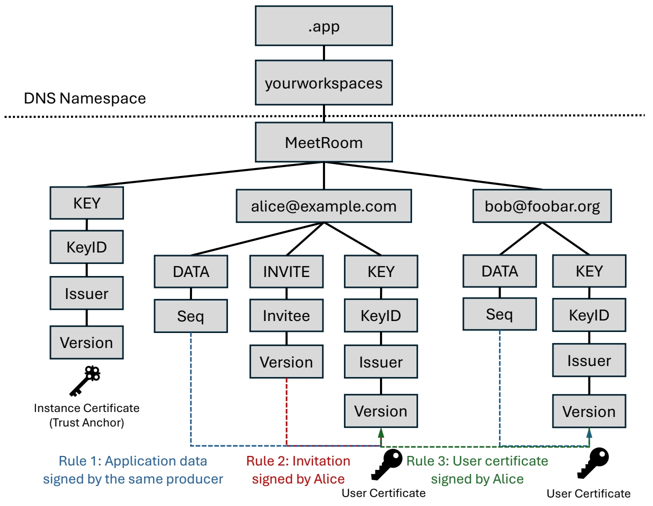

Figure 1: User data are named under each user's name prefix, which has three branches for data that carry document edits, Invitations and certificates.

Bootstrapping users with names: Each instance has a DNS-derived name. For example, Alice owns a name "`yourworkspaces.app`" and opens NWS to initiate an instance "`MeetRoom`" under her DNS name. Receiving the two name components, her NWS browser tab (referred as Alice in the rest of the paper) generates a key pair under "`yourworkspaces.app/MeetRoom`" ("`MeetRoom`" for short) to create "`MeetRoom`" instance. Only the initiator of an instance controls the instance private key.

To identify herself as the "`MeetRoom`" initiator, Alice concatenates the instance name with her email address to derive her username "`MeetRoom/alice@example.com`", generates a new key pair and signs it with "`MeetRoom`" instance self-signed certificate. As results, anyone that receives Alice certificate and trust "`MeetRoom`" instance certificate would authenticate her as the "`MeetRoom`" initiator.

As initiator, Alice invites her collaborator Bob as a new "`MeetRoom`" member. Following the same naming convention of herself, she first names Bob as "`MeetRoom/bob@foobar.org`", then signs a certificate for Bob, and lastly produces an Invitation. It includes "`MeetRoom`" instance certificate, Bob's newly issued certificate, "`MeetRoom`" trust schema (discuss shortly) and a group encryption key. Installing the security components from Invitation, Bob is security bootstrapped [23] into "`MeetRoom`" by getting the instance certificate as trust anchor, his "`MeetRoom`" user certificate and trust schema that Alice has defined[7].

Identifying user data by names: Security bootstrapped users produce named data. NWS names user data under DNS-derived usernames. As depicted in Figure 1, Invitation data follows naming convention "`<username>/INVITE/<invitee>/<version>`". The Invitation that Alice produces to bootstrap Bob has name "`MeetRoom/alice@example.com/INVITE/bob@example.org/v=1`", which means Bob's first invitation from Alice. Data that carry document edits are named as "`<username>/DATA/<seq>`". For instance, "`MeetRoom/bob@foobar.org/DATA/seq=1`" names the first update that Bob makes to "`MeetRoom`"'s documents. Certificates are also named with convention "`/<username>/KEY/<keyid>/<issuer>/<version>`", with "`KEY`" component as the keyword for certificate, "`<keyid>`" specifying the key identifier, "`issuer`" for the certificate issuer, and "`<version>`" for the certificate version. "`MeetRoom/bob@foobar.org/KEY/0x1d4e5/alice/v=1`" names Bob's first certificate that is issued by Alice for key "`0x1d4e5`" in "`MeetRoom`".

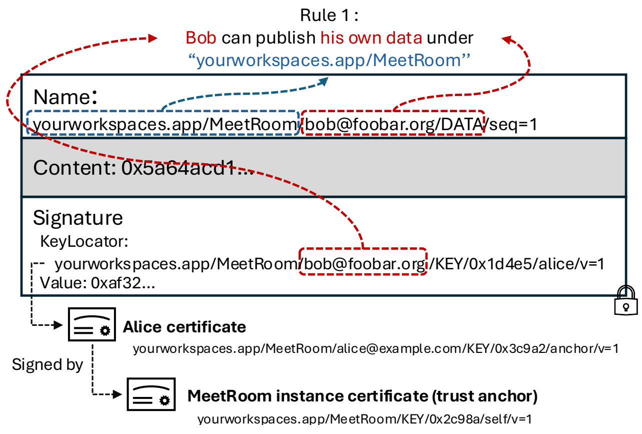
Rule 1 :
Bob can publish his own data under
"yourworkspaces.app/MeetRoom"
Name:
yourworkspaces.app/MeetRoom/bob@foobar.org/DATA/seq=1
Content: 0x5a64acd1...
Signature
KeyLocator:
yourworkspaces.app/MeetRoom/bob@foobar.org/KEY/0x1d4e5/alice/v=1
Value: 0xaf32...
Alice certificate
yourworkspaces.app/MeetRoom/alice@example.com/KEY/0x3c9a2/anchor/v=1
Signed by
MeetRoom instance certificate (trust anchor)
yourworkspaces.app/MeetRoom/KEY/0x2c98a/self/v=1

Figure 2: Bob's Data packet includes its packet name, content, and signature. The signature field includes a key locator, which refers another Data packet from Bob that contains his key.

Securing user data with names: As indicated in Figure 2, each named data is signed and includes a key locator points to the signer. Because all keys and certificates have semantic, DNS-derived names, it is convenient for "`MeetRoom`" initiator (*i.e.,* Alice) to define name-based, *schematized trust rules* [11, 25] that are shown in Figure 1:

(1) "`MeetRoom`" members to sign data under their own names.
(2) "`MeetRoom/alice@example.com`" to sign "`MeetRoom`" Invitations.
(3) "`MeetRoom/alice@example.com`" to issue "`MeetRoom`" user certificates.

Rule 1 outlaws data publication in other's namespace, while Rule 2 and 3 jointly make Alice the only legitimate "`MeetRoom`" inviter. When other "`MeetRoom`" members receive Bob's data, they first use Bob's certificate to verify the data signature, then execute trust schema that Alice defined to check whether Bob's certificate is

[7]For simplicity, this paper only describes one of the two membership models that NWS supports.

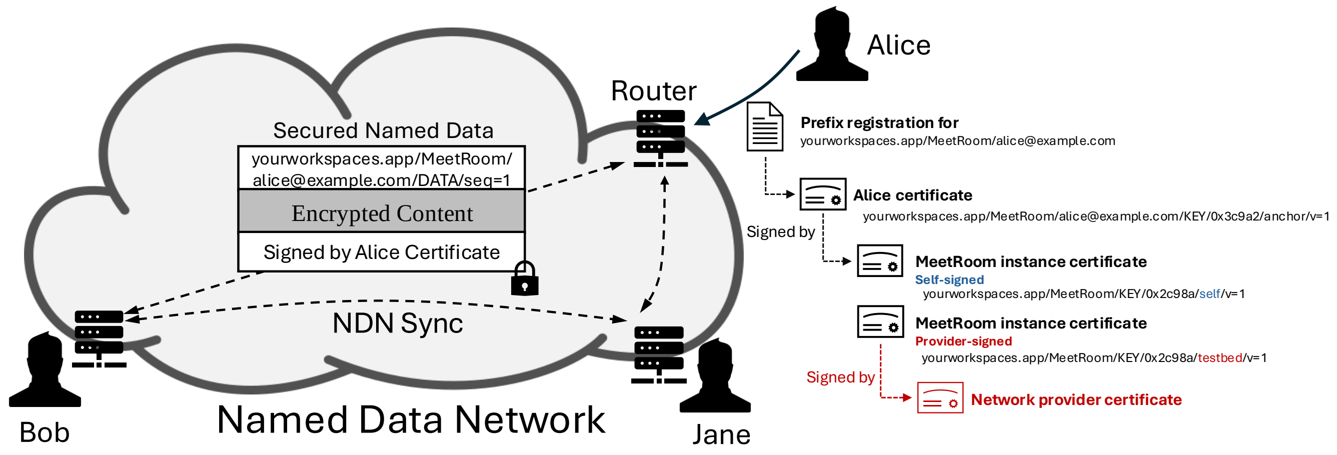
Alice
Router
Secured Named Data
yourworkspaces.app/MeetRoom/
alice@example.com/DATA/seq=1
Encrypted Content
Signed by Alice Certificate
Prefix registration for
yourworkspaces.app/MeetRoom/alice@example.com
Alice certificate
yourworkspaces.app/MeetRoom/alice@example.com/KEY/0x3c9a2/anchor/v=1
Signed by
MeetRoom instance certificate
Self-signed
yourworkspaces.app/MeetRoom/KEY/0x2c98a/self/v=1
MeetRoom instance certificate
Provider-signed
yourworkspaces.app/MeetRoom/KEY/0x2c98a/testbed/v=1
Signed by
Network provider certificate
NDN Sync
Bob
Named Data Network
Jane

Figure 3: A named data network where NWS users directly communicate by names. The edge routers in this network only accepts prefix registration signed by corresponding prefix owner.

authorized to sign this data, and recursively repeat the process for Bob's certificate until reaching the "`MeetRoom`" trust anchor.

U2U communications by names: To enable U2U communications between "`MeetRoom`" users on Internet, one requires an in-operation overlay network that forwards secured user data by names. Besides name-based forwarding, this network also differs from traditional IP network in the following aspects.

(1) Data are only forwarded to the interface(s) that have user interests to them.
(2) Users register name prefixes on edge routers, and edge routers validate a user's name ownership before accepting a prefix registration.
(3) Built-in routing security by producing routing announcements as secured named data.

NDN Testbed [17] provides such a *global, named data network overlay*. It has has about 25 router nodes across four continents, and operated by collaborating universities and research institutes. NDN Testbed operators authenticates users who directly own DNS and certify them via NDNCERT [27], an ACME-like tool that automates domain name authentication and certificate issuance in a named data network. Whoever possesses a Testbed certificate is authorized to register the certified name as a network prefix on NDN Testbed. In order to register prefix "`MeetRoom/alice@example.com`", Alice requests the Testbed certificate issuers to issue a certificate for her instance "`MeetRoom`", and proves her ownership (*e.g.*, DNS challenge, X.509) of "`yourworkspaces.app`" to the issuers. Testbed certificates distinguish from others by naming conventions. The "`<issuer>`" component of those certificates are always "`testbed`", therefore one can infer "`MeetRoom`"'s Testbed certificate name from instance certificate name by names. Note that NDN Testbed is not an Internet equivalent, but a network provider. Other providers can define their own naming conventions in their networks.

Figure 3 illustrates the security workflow when Alice registers prefix to NDN Testbed. She signs the registration with her "`MeetRoom`" user certificate. Receiving the prefix registration with a key locator pointing to Alice certificate, Testbed router traces Alice certificate chain to validates registration and eventually encounters "`MeetRoom`" trust anchor "`MeetRoom/KEY/0x2c98a/self/v=1`". Leveraging aforementioned naming convention, router infers the name of "`MeetRoom`" Testbed certificate and fetches by name. Through validating the certificate chain, router authenticates Alice as the "`MeetRoom/alice@example.com`" owner, and accepts prefix registration. Bob's prefix registration is signed by a similar certificate chain with one depth further, since his certificate is signed by Alice.

After establishing reachability, NWS makes use of NDN Sync [10] to synchronize latest sequence numbers from each "`MeetRoom`" member. Whenever learning the latest sequence numbers from NDN Sync, a "`MeetRoom`" member infers the newly produced data name by naming convention, and fetch secured user data by name.

An important note is that, Testbed certificates are only involved in network prefix registration. It is users who secure user data. NDN Testbed only plays one role in NWS, that is global traffic transit. [22] offers detailed descriptions on how NWS enables users utilizing ad hoc connectivity to build local overlay.

Data-centric app design enabling in-network storage: NWS also demonstrates a great potential of data-centric application design, that is to integrate storage and computation resources with network infrastructure. An in-network storage is as simple as a named service setup by the network provider. It joins applications' Sync groups, and infers the names of user data through naming convention, and fetches all data that are directly secured by users. To further explore this integration, we have developed a distributed, in-network storage called Repo [24], and deployed it on NDN Testbed. Repo joined the Sync groups of several NWS instances that had traffic on NDN Testbed, and enabled fully asynchronous communication among those users. Users are oblivious to Repo participation, since they directly communicate by names, and Repo simply returns secured named data if there is an interest to it. More importantly, data security is fully controlled by users, as opposed to CDNs where one must compromise on sharing X.509 certificates. We argue that in-network computation can be realized in the same spirit.

5 Discussions

Decentralization by blockchain-based solutions: Public (also known as permissionless or anonymous) blockchains are viewed as another approach to decentralization. Instead of empowering users with app and data control, blockchains seek to decentralize *decision* power through voting, which is out of the scope of this paper.

6 Conclusion

We believe that the fundamental goal of Internet decentralization is to decentralize the control power, so that Internet users can take (back) the control of their apps and data in their own hands. In this paper we argued that a necessary condition to Internet decentralization is to enable direct and secure User-to-User (U2U) communications, and that assigning users DNS names can be an effective means to to enable direct and secure U2U communications. We further show, by proof of evidence via a decentralized app, that U2U can be achieved by assigning users DNS-derived names. The use of semantically meaningful DNS-derived names for user and data enables one to build security protection into applications and communications by securing data directly. Reviewing other decentralization efforts also further confirmed our above insights via lessons learned: users should get names instead of their servers; flat names have limitations in supporting U2U; and assigning users DIDs introduces dependency on centralized database services instead of enabling U2U.

We invite the decentralization community at large to examine our conclusions and help identify remaining research challenges in pursuing this new direction towards decentralization.

Acknowledgement

This work was supported in part by National Science Foundation under awards 2019085 and 2126148.

References

[1] 2024. Internet Usage Statistics In 2024. https://www.forbes.com/home-improvement/internet/internet-statistics/ Accessed on June 26, 2024.
[2] fiatjaf. 2024. NIP-01: Basic protocol flow description. https://github.com/nostr-protocol/nips/blob/master/01.md
[3] fiatjaf. 2024. Nostr - Notes and Other Stuff Transmitted by Relays. https://github.com/nostr-protocol/nostr
[4] fiatjaf. 2024. A vision for content discovery and relay usage for basic social-networking in Nostr. https://fiatjaf.com/3f106d31.html
[5] Paul E. Hoffman and Jakob Schlyter. 2012. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698. https://doi.org/10.17487/RFC6698
[6] KERI. 2024. Key Event Receipt Infrastructure. https://keri.one/
[7] Martin Kleppmann, Paul Frazee, Jake Gold, Jay Graber, Daniel Holmgren, Devin Ivy, Jeromy Johnson, Bryan Newbold, and Jaz Volpert. 2024. Bluesky and the AT protocol: Usable decentralized social media. *arXiv preprint arXiv:2402.03239* (2024).
[8] Yekta Kocaogullar, Eric Osterweil, and Lixia Zhang. 2024. Towards a Decentralized Internet Namespace. In *2024 ACM CoNext Conference Workshop on Decentralization of the Internet.*
[9] Mastodon. 2024. *Mastodon.* https://joinmastodon.org/
[10] Philipp Moll, Varun Patil, Lan Wang, and Lixia Zhang. 2022. SoK: The evolution of distributed dataset synchronization solutions in NDN. In *Proceedings of the 9th ACM Conference on Information-Centric Networking* (Osaka, Japan) *(ICN '22).* Association for Computing Machinery, New York, NY, USA, 33–44. https://doi.org/10.1145/3517212.3558092
[11] Kathleen Nichols. 2021. Trust schemas and ICN: key to secure home IoT. In *Proceedings of the 8th ACM Conference on Information-Centric Networking* (Paris, France) *(ICN '21).* Association for Computing Machinery, New York, NY, USA, 95–106. https://doi.org/10.1145/3460417.3482972
[12] Stuart L Pardau. 2018. The california consumer privacy act: Towards a european-style privacy regime in the united states. *J. Tech. L. & Pol'y* 23 (2018), 68.
[13] Bluesky PBC. 2024. DID PLC Method (did:plc). https://github.com/did-method-plc/did-method-plc
[14] Aravindh Raman, Sagar Joglekar, Emiliano De Cristofaro, Nishanth Sastry, and Gareth Tyson. 2019. Challenges in the decentralised web: The mastodon case. In *Proceedings of the internet measurement conference.* 217–229.
[15] Manu Sporny, Dave Longley, Markus Sabadello, Drummond Reed, Orie Steele, and Christopher Allen. 2022. Decentralized Identifiers (DIDs) v1.0. W3C Recommendation. https://www.w3.org/TR/did-core/
[16] 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.* 1–11.
[17] The NDN Team. 2024. NDN Testbed. Online at https://named-data.net/ndn-testbed/.
[18] thekinrar. 2024. *Mastodon instances.* https://instances.social/
[19] Paul Voigt and Axel Von dem Bussche. 2017. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing* 10, 3152676 (2017), 10–5555.
[20] Christopher Webber and Jessica Tallon. 2018. ActivityPub. W3C Recommendation. https://www.w3.org/TR/2018/REC-activitypub-20180123/
[21] Ash Wilson, Shumon Huque, Olle E. Johansson, and Michael Richardson. 2024. *An Architecture for DNS-Bound Client and Sender Identities.* Internet-Draft draft-ietf-dance-architecture-06. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ietf-dance-architecture/06/ Work in Progress.
[22] Tianyuan Yu, Xinyu Ma, Varun Patil, Yekta Kocaogullar, and Lixia Zhang. 2024. Exploring the Design of Collaborative Applications via the Lens of NDN Workspace. *arXiv preprint arXiv:2407.15234* (2024).
[23] Tianyuan Yu, Philipp Moll, Zhiyi Zhang, Alexander Afanasyev, and Lixia Zhang. 2021. Enabling Plug-n-Play in Named Data Networking. In *MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM).* IEEE Press. https://doi.org/10.1109/MILCOM52596.2021.9653033
[24] Tianyuan Yu, Jacob Zhi, Xinyu Ma, Yekta Kocaogullar, Varun Patil, Ryuji Wakikawa, and Lixia Zhang. 2024. Repo: Application Agnostic and Oblivious In-Network Data Store. In *Proceedings of the 2nd Annual IEEE International Conference on Metaverse Computing, Networking, and Applications.*
[25] Yingdi Yu, Alexander Afanasyev, David Clark, kc claffy, Van Jacobson, and Lixia Zhang. 2015. Schematizing Trust in Named Data Networking. In *Proceedings of the 2nd ACM Conference on Information-Centric Networking* (San Francisco, California, USA) *(ACM-ICN '15).* Association for Computing Machinery, New York, NY, USA, 177–186. https://doi.org/10.1145/2810156.2810170
[26] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, KC Claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang. 2014. Named data networking. *ACM SIGCOMM Computer Communication Review* 44, 3 (2014), 66–73.
[27] Zhiyi Zhang, Yingdi Yu, Alex Afanasyev, and Lixia Zhang. 2017. NDN certificate management protocol (NDNCERT). *NDN, Technical Report NDN-0050* (2017).