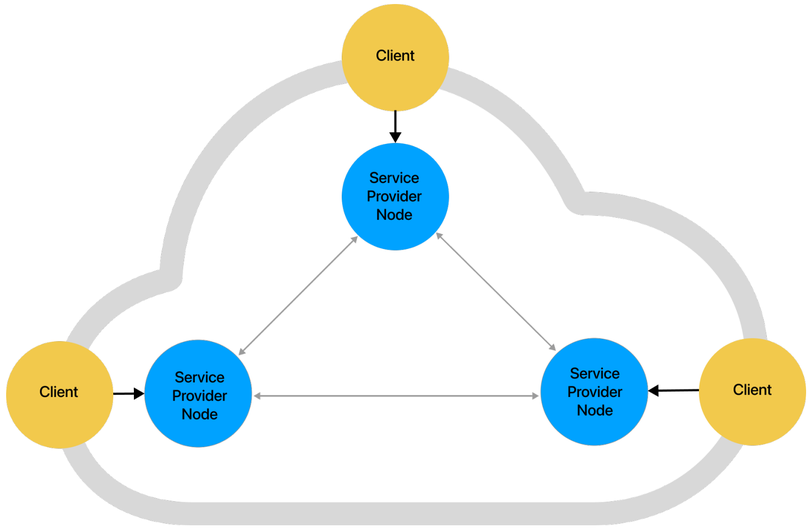
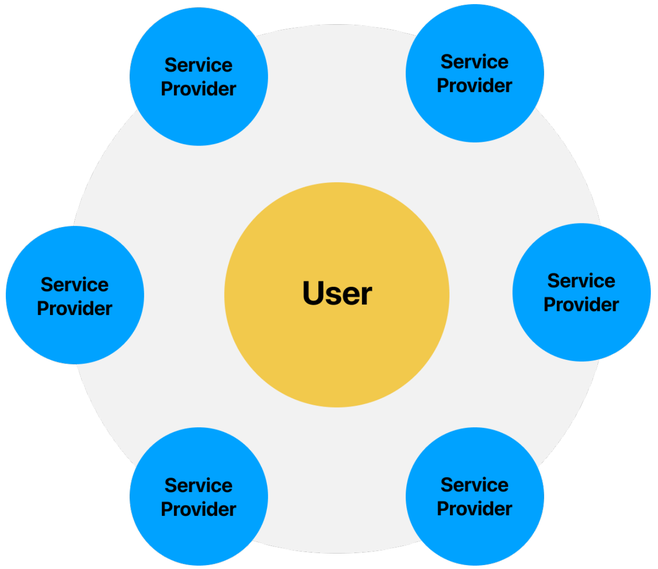
Subnet Tech - Deep Dive

Infrastructure for reimagining digital communications

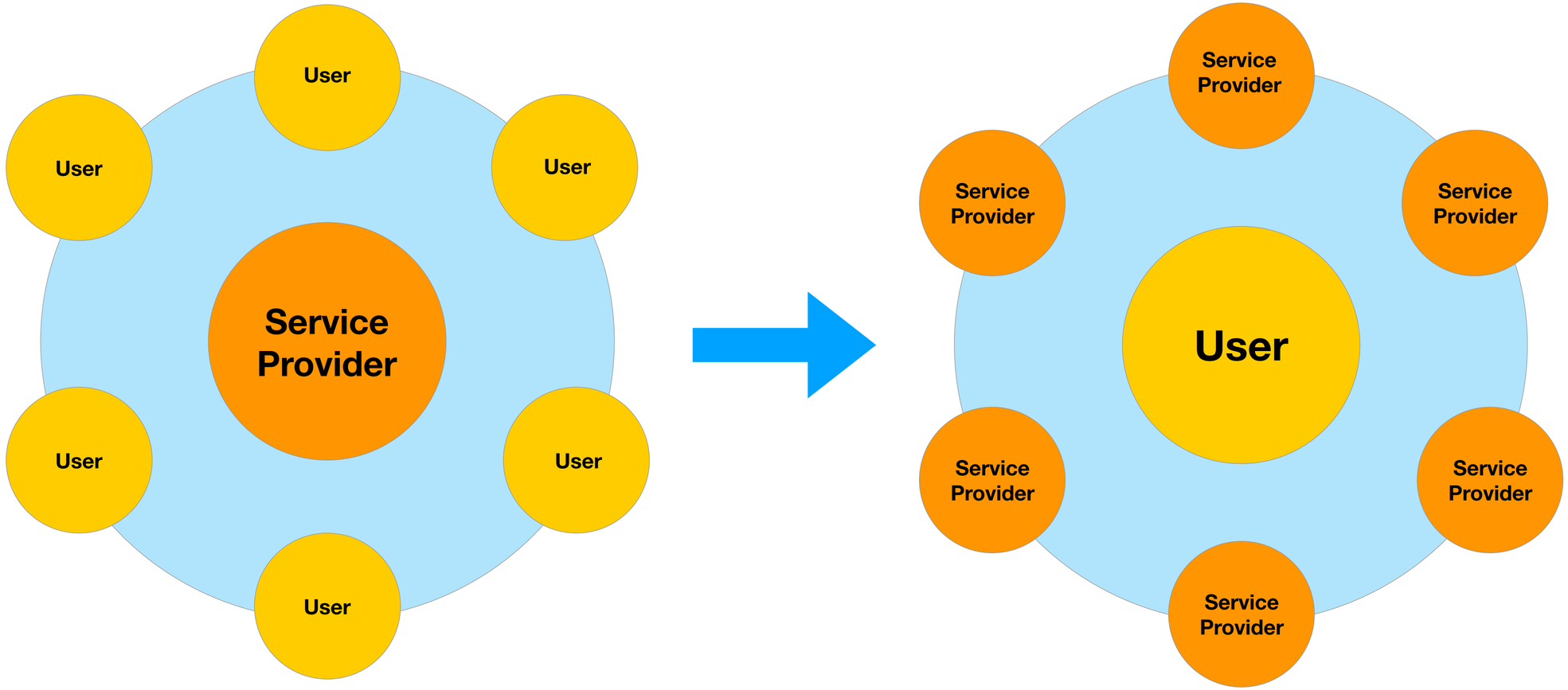




Decentralization via Systematic Invesrions

Web 2.0 Mobile Native		Web 3.0
Everything is free for life.	→	Nothing in life is free. Services fees should be free-market based.
Providers have rights to do almost anything with your data and metadata.	→	No-one has the right to do anything with your data and metadata
You agree to be brainwashed while interacting with content.	→	No promotional or paid content unless you explicitly want to view some.
Trust your provider not to be evil and not to change terms at will.	→	Trust no-one (*), Providers can’t be Evil.
No anonymity. Personal Identification everywhere (to better target ads).	→	Users decide if they want to be anonymous or identifiable.

Decentralization via Architectural Inversion



Web 2.0 / Mobile Native 1.0

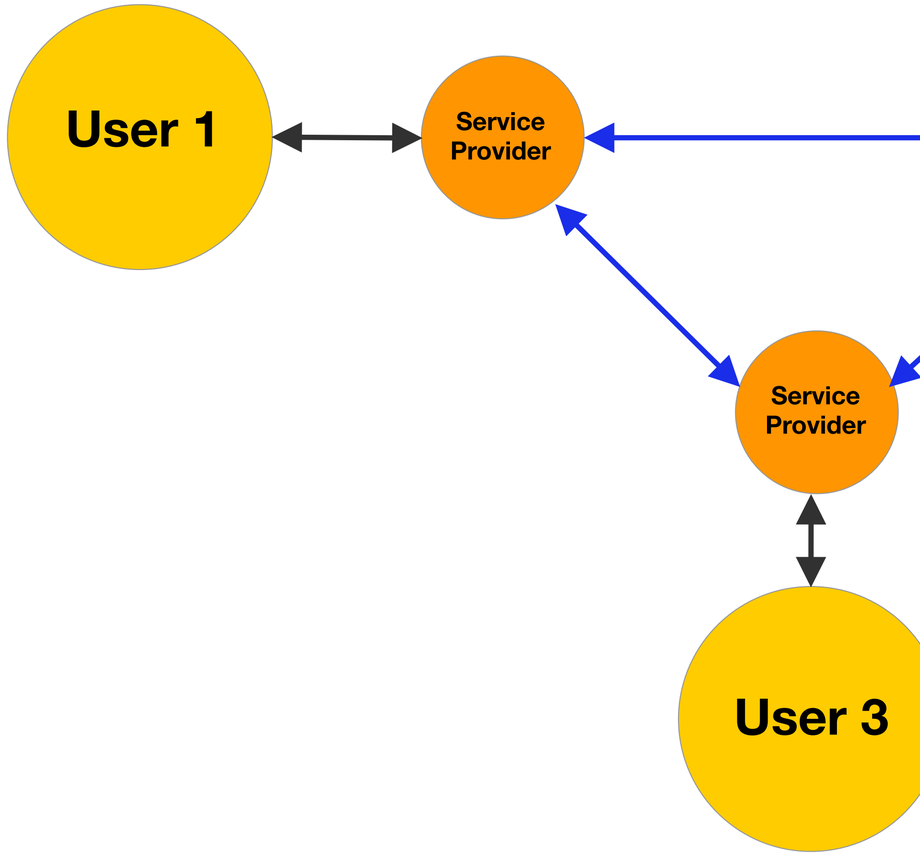
The Digital Middle Ages 2004 - 2019 ?

Web 3.0 / Mobile Native 2.0

Digital Renaissance 2020 ?

Introducing Subnet

- A **highly-opinionated project** designed to facilitate **Upsetting of social media** and digital communications.
- This means following specific system design first-principles when building a network.
- Subnet may appeal to some people and service providers, other systems with different design goals and trade-offs may fit others. As always, there is no silver bullet or a one size fits-all solution…



OVERVIEW

Subnet Vision

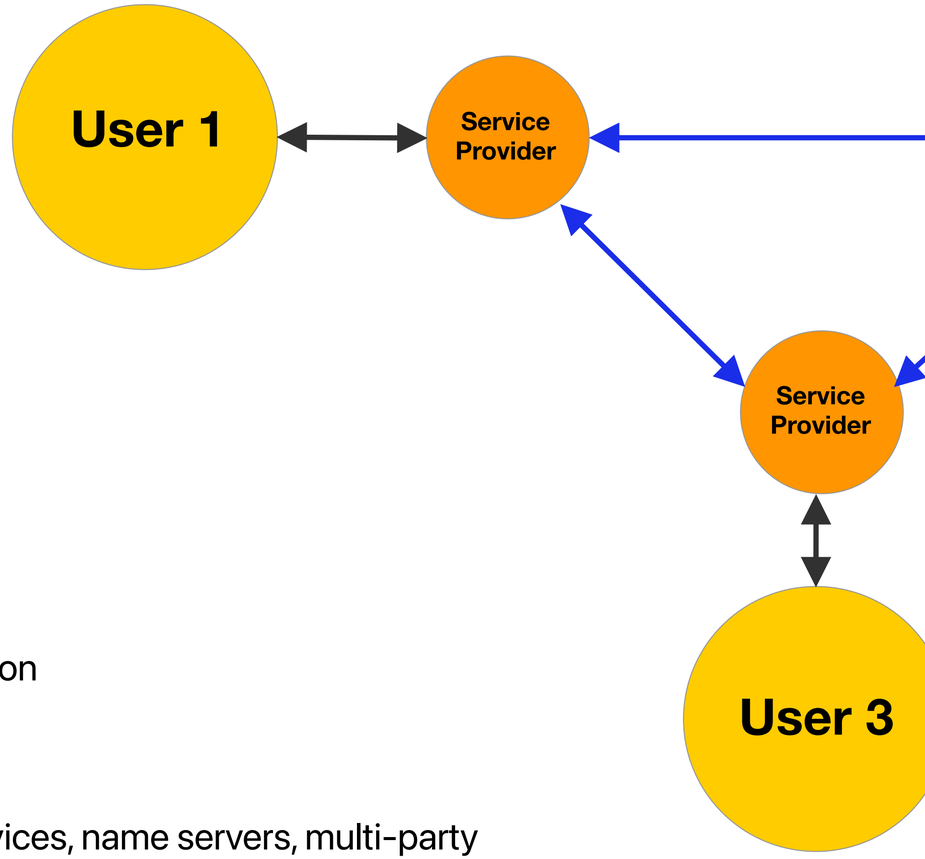
Create **user-centric digital communications apps** built on top of a new kind **decentralized network infrastructure**.

Subnet offers 5 core features:

1. **Instant messaging** - 1:1
2. **Groups** - n:n
3. **Status Updates** - 1:n
4. **Multiple Sources Feed** - n:1
5. **Premium Content** (blog post, image, video, music)

Additional features:

6. **Newsletter -** Premium Status Updates
7. **Premium Groups** - Monthly subscription communities
8. **Limited-Edition Premium Content -** art and fans items
9. **Proofs of Action or Affiliation -** Certification.
10. **Digital Identity** - User-generated based on proofs on proofs of ownership and affiliation
11. **User-to-user** instant payment and all premium purchases.



>> Future net services: decentralized storage, proxy Internet servers, video transcoding services, name servers, multi-party real-time video chat, etc...

DESIGN

Things we have been working on…

1. **Network Architecture.**
2. **Incentive Compatible Network Protocols**: service-to-service, user-to-service…
3. **Nano crypto payments infra** to enable incentive-compatibility and new business models (replacing free) and a performant permissionless ledger to support it.
4. **Robust user experience** on the level of Web 2.0 / Mobile Native 1.0 high convenience standards.
5. **Privacy-First and User-centric protocols and apps.**
6. **Decentralized inverted identities, network discovery** and **messages routing** between these identities.
7. **Permissionless and swappable service providers.**
8. **Banner digital communications apps** - 1:1 messaging, group messaging, private and public status feeds.

DESIGN

User-Centric Network Design Goals

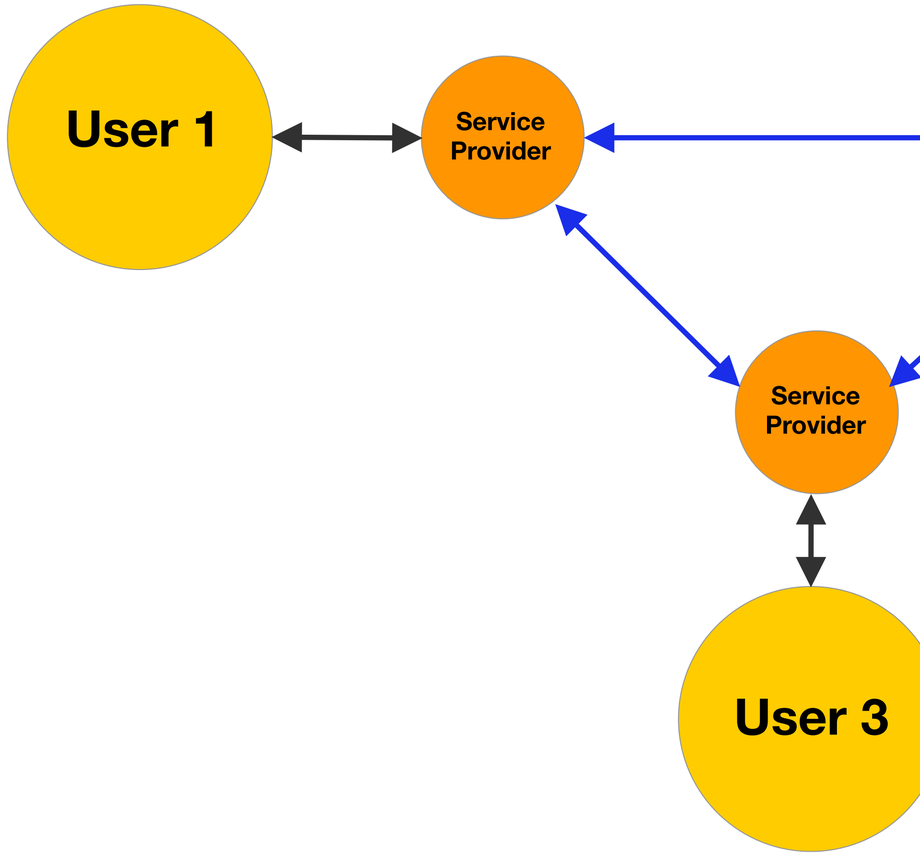
Users should fully control their identity and personal identification

1. **Personal identification** is at the core of web2.0 due to business model constraints.
2. **Anon by default** - only users decide to be anon or personally identifiable, not the service.
3. Users may use **multiple identities without any limitation**.
   Some may be anon, some personally identifiable.
4. **No content censorship by service providers** - users are responsible to moderate their own created social spaces in any way they may see fit.
5. **No censorship possible on using the network capabilities** by anyone in the world.
6. No clear-text user-generated content stored on service providers servers and providers don’t know what content they are routing between users.
7. **Service providers identity is not personally identifiable** to users unless a provider choses to identify itself.

DESIGN

Privacy-first Network Design Goals

1. Service providers can’t access users private data so they can’t use or misuse it in any way.
2. Metadata sharing (hard problem) is bound to service agreement and providers reputation system and is mitigated by seamless provider swapping by users.
3. Users fully control who can access shared data. Modern encryption employed to enforce users controls.
4. Users always own their data even after sharing it with others.
5. Enable users to be as anonymous or identifiable as they want to be.
6. Rely on honest majority of service providers instead of an `honest` monopolistic service provider.



DESIGN

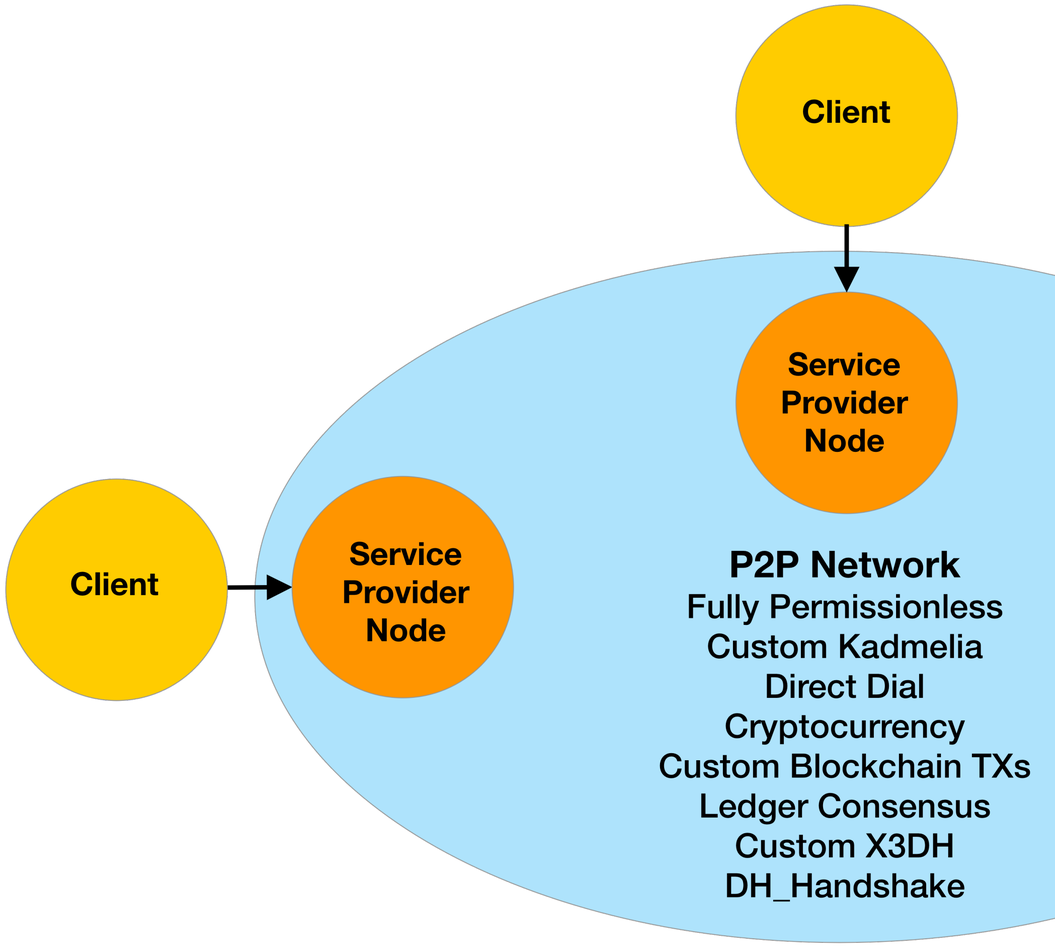
Subnet Design Overview

- **Users** run **clients** (on native mobile or desktop web) and frequency connect to and disconnect from the network.
- **Service providers** run **permissionless full nodes software** on dedicated servers hardware in data centers 24x7.
- Service providers provide clients with **network services** - e.g. instant messaging, groupware, proxy services and Internet storage capabilities which power user apps.
- Providers form a **custom p2p network** over the Internet and maintain a **cryptocurrency ledger** between their servers.
- Providers communicate with each other using standardized and well documented network protocols such as **decentralized discovery, routing and messaging protocols**

DESIGN

System Overview

- **Clients** enable users to create one or more **decentralized identities** which they fully control.
- Clients establish a **contractual relationship** and uses a **service provider** to get network services.
- Clients can **switch to a new different provider** at will at any time.
- Built-in **custom payment channels** capabilities enable **nano-payments** between client and providers.
- **All user data is encrypted** using modern crypto to designated receiver using strong forward and backward secrecy both on wire and on store - **service providers can never read clients data.**



DESIGN

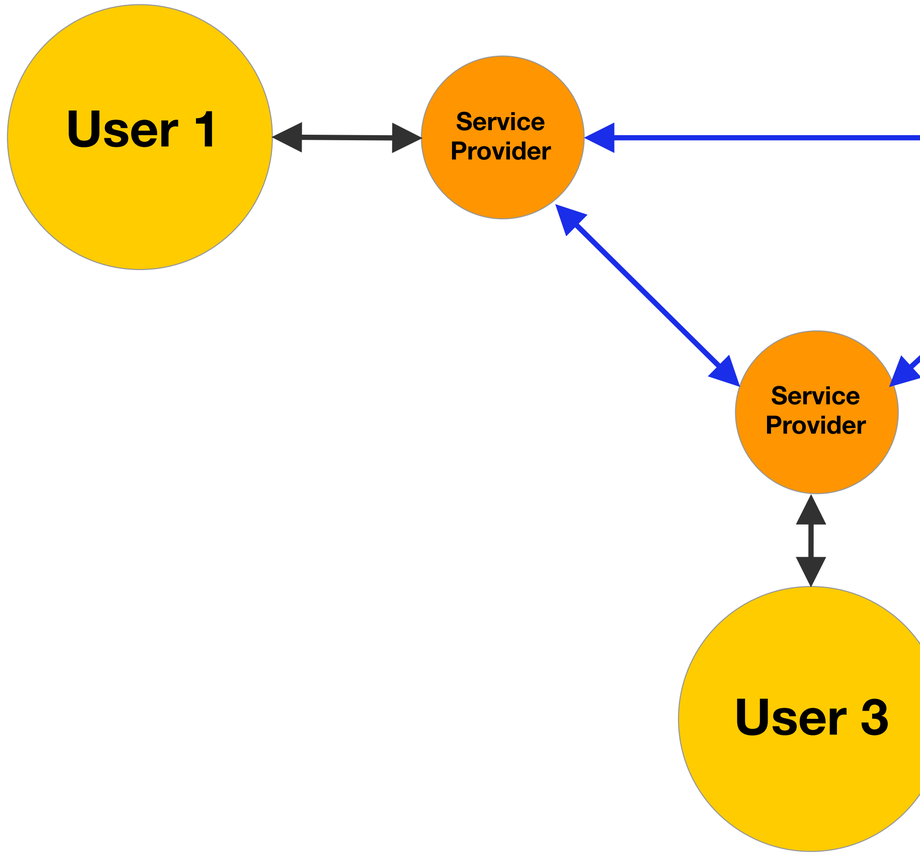
Incentivized Network Protocols Design

1. Client to provider messages always include a nano payment - payments is built into the core protocols.

2. Provider is incentivized to provide a live API, to provide honest results to users, and to lock funds in bi-directional channels with clients.

3. Client always uses its provider for network services and does not communicate directly with providers that it doesn't have a relationship with.

4. Provider to provider messages - a message receiver verifies that requester has recently contributed to the network using ***proofs of useful work*** and drops messages from unverified providers.



DESIGN

Inverted Identities Design

- Public and private parts based on EC crypto - the public key is the public id and the private key proves ownership of the public key by a sentient entity (it can sign).
- Digital signatures to prove attestations - actions, coin holdings, statements, bonds and promises.
- Generalization: **a smart contract based identity** - is configured with crypto key pairs, own cryptocurrency or proves committed resources and has rules about how to modify the pairs and to allocate resources (e.g. DAOs, Smart Wallets). Hardware key used to create a smart identity.
- Cryptocurrency signed slash-able commitments used in network protocols. e.g., payment channels, pricing commitments.
- Reputation is built from provable actions and objective network operational data. e.g. participation in a consensus protocol. Services are delivered to users according to promises.
- May or may not be identifiable to a person or an org - fully at the entity's discretion.

Swappable Service Providers

Crypto Service Agreement

Effective Date
Duration

Feature 1 = 3 coins

Feature 2 = 6 coins

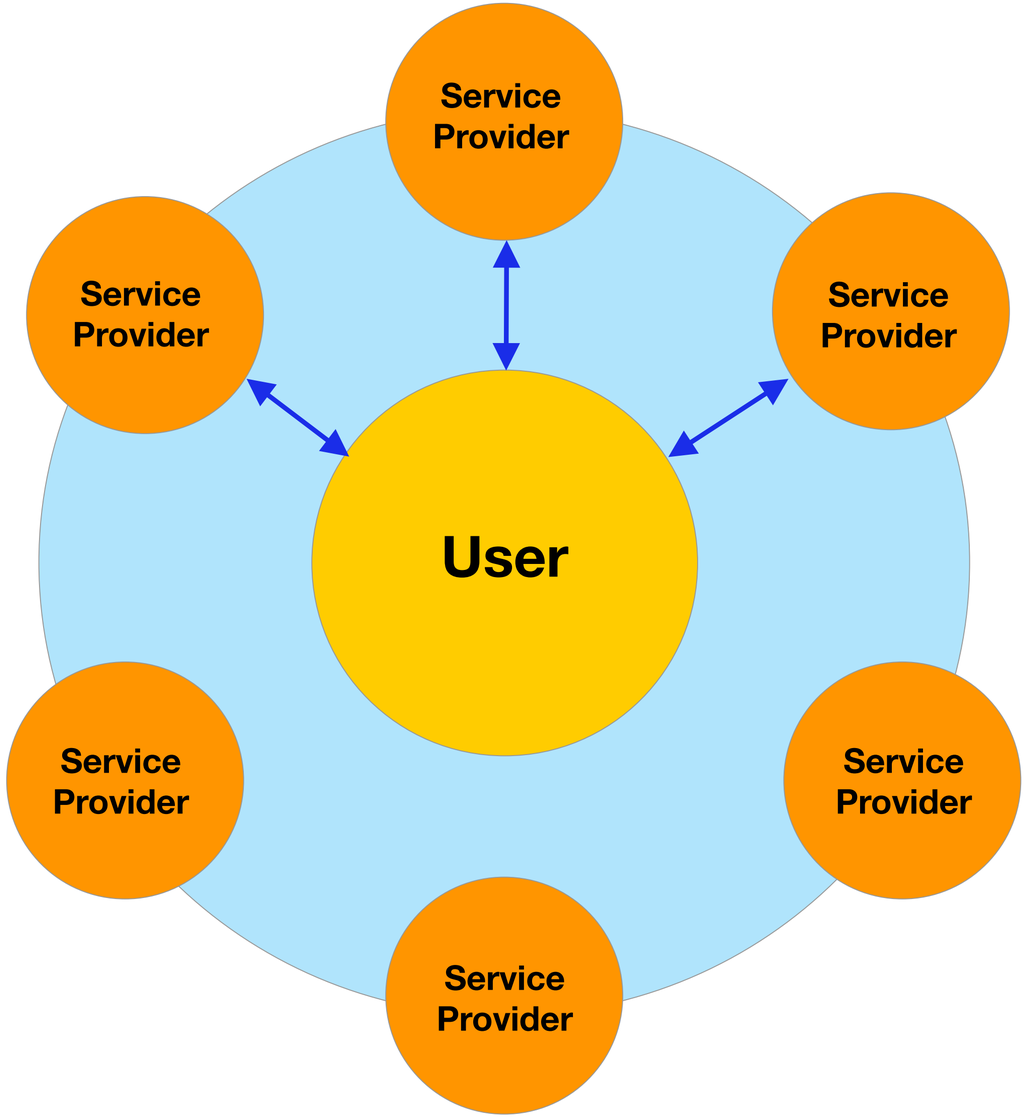
...

Data Ownership

Privacy Guarantees

Coins Bond

Provider User



Provider Swapping

Unconditional & Seamless

Providers Discovery
Open Network Protocols

Incentived Data Migration

Provider Reputation

Historical Operational Data

Users Feedback & Ratings

Objective Fines History

Providers Provisioning Flow

1. ***Discover*** - User discovers candidates - platform-provided random subset of available providers based on user's identity or manual via a provider url.

2. ***Review*** - User reviews providers price list, operational commitments, reputation, operational data.

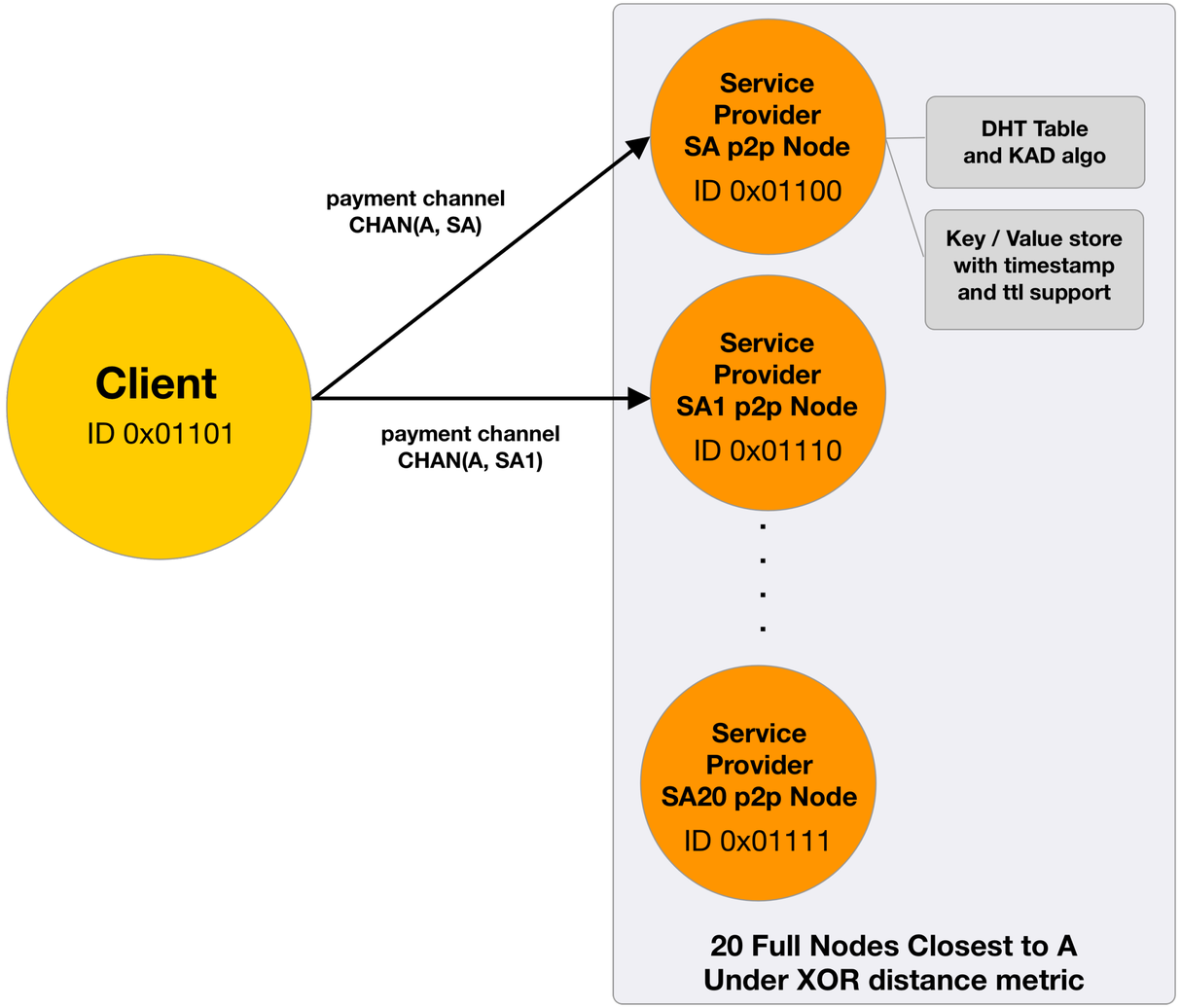
3. ***Provision*** - User selects a provider and commits to it by establishing a two-directional payment channel with it and by locking some coins in the channel.

4. ***Publish*** - B signs X3DH(SB, B) and SB signs and published it to the network (using KAD to ~20 servers closest to B).

5. ***Use*** - Provider is used by the user for network services and app-level protocols. Nano payments are instantly and seemingly conducted over the payment channel.

6. ***Swap*** - User can provision a new provider at any time and close the payment channel it has with the provider.

Decentralized Name Services via X3DH



X3DH(A, SA) BUNDLE

```
{
   T              // Bundle creation timestamp

   IKA            //  A identity hash. e.g. sha512(IKA)
   NA             //  A friendly nickname (optional - defaults to ANON)
   {PKA, ID}      //  A set of DR pre-keys with ids

   IKSA           //  SA identity hash
   {PKSA, ID}     //  A set of DR pre-keys with ids
   DIALSA         //  SA dial-up info bundle

   // Optional: SA1 info…

   SIGNA          // A's signature on all data
   SIGNSA         //  SA's signature on A and SA's data
}
```

Blockchain & Cryptocurrency

1. Blockchain is **just one among several** sets of algorithms, network protocols and data structures running on Subnet to provide its core capabilities. It is going to become a more well understood and mature enabling tech. Think app servers non-sql DBs ~2010…

2. Modern PoS or PoST consensus protocols to eliminate security issue with a small PoW networks.

3. Optimize for the user - built-in payment channels capabilities with use-case specific optimizations (e.g. long half-open state) , w/o a need for turing-complete smart contracts computations.

4. Consensus on a core coin that is paid by users to service providers for providing services.

5. A whole range of new kind of subscription services enabled with nano-payments - e.g. seamless news, music and videos…

Nano Payments

- Automatic and seamless - zero user friction while using apps.
- Cheap - fraction of a US cent. Market-determined prices.
- No transaction fees.
- Key technology for enabling new business model to replace ads.
- Clients hot wallet with spending account funds - low funds security risk.
- Accountable - users should be easily be able to review all payments and to get insights.
- Custom built-in ledger support.
- Optimizations for UX improvements.

Terminology

- **A**, **B** - The clients of 2 users - **User A** and of **User B**.
- **A** is identified by a public key. Its user has the corresponding private key. Same with B. Key pairs are also used for cryptocurrency-ledger coin accounts
- **SA**, **SB** - The service providers of A and of B.
- **CHAN(A, SA)** - a payment channel between A and SA.
- **X3DH(SA, A)** - A custom X3DH bundle with pre-keys for both A and SA, co-signed by A and SA (see X3dH protocol spec). Includes SA net dial-up info.
- **DR** - The double ratchet encryption algorithm. A 2 parties diffie-hellman initiated master, send and receive chains. Forward and backward security. Traditionally used with a centralized server in Telegram, Signal and WhatsApp.
- **KAD** - a modified Kademila algorithm for p2p discovery of X3DH bundles.

p2p Messaging Core

- Each A and its SA on the network, maintain an **open active payment channel** that they established when A chose SA as its service provider.
  This makes nano payments from A to SA (and from SA to A) fast and seamless.
- SA gets and stores messages which are pending delivery to A.
- SA is compensated with nano-payments from A for network services it provides to A.
- SA and A may or many not have an open bidirectional network connection.
- When such a connection is available notifications from SA about messages availability may be pushed to A over it. **A push pattern.**
- When not available A connects to SA periodically and polls for new incoming messages after last polling time t. **A pull pattern.**

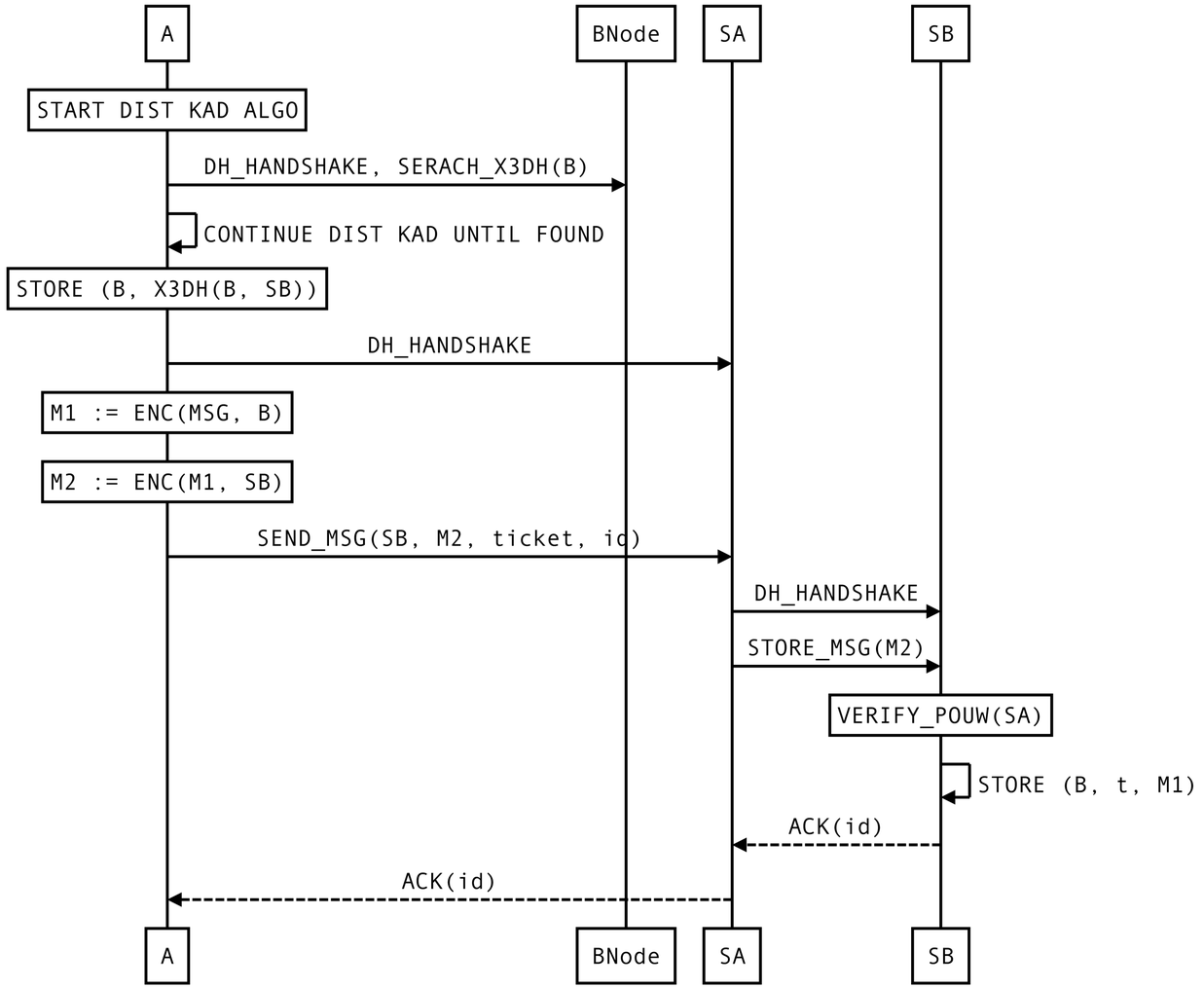
p2p Messaging Core

- To send a message to A, other providers such as SB needs to obtain SA's dial-up and SA's and A's pre-key bundle. They do so by querying the network for **X3DH(A, SA)** using the known **A** ID.
- SA is compensated for storing and routing network messages to A on a **per-message basis** so it is incentivized to do so and to behave according to protocols which employ this messaging scheme.
- **Higher-level protocols exchange messages using this messaging infra.** This enables a whole range of communications apps such as **groupware**, **instant messaging**, **status updates** and more...

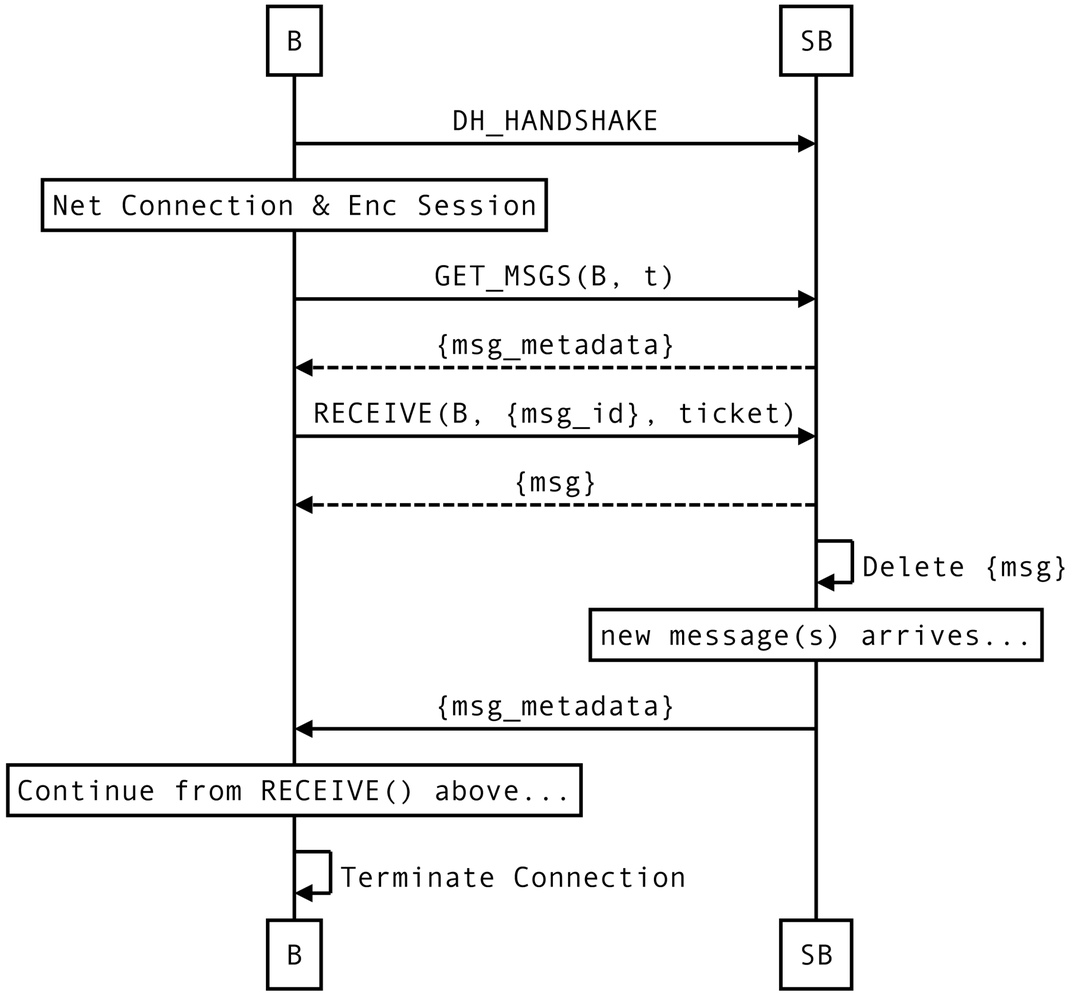
Bootstrapping a 2-party DR session

- Two cases to consider: Case 1. direct net connection between A and B.
  Case 2. no net connection between A and B.
- **Case 1.** Direct X3DH protocol between A and B using the **one-time DR prekey variant** of the **X3DH protocol**. A queries B for a new X3DH bundle and they both use that to establish a session key to bootstrap DR. We call this a **DH handshake**. All messages after bootstrap are DR encrypted.
- **Case 2.** B publishes a signed X3DH bundle to the network and A uses it to bootstrap a DR session between A and B **without a one-time DR prekey**. A generates an ephemeral key pair, creates a session key with it, starts a DR session with the session key, and sends the data B needs to create the same session on its end - **Ephemeral X3dH public key**, **DR public key** and **B's pre-key ID.**
- In both cases A and B maintain a stateful DR session between them and encrypt and decrypt all messages exchanged between them with it **according to DR**.

Sending a Message



Receiving Messages



System Summary

1. Service providers run a set of open p2p protocols between them.
2. One of the protocols is cryptocurrency - mine and maintain ledger.
3. Service providers provide well-defined api end points to clients.
4. Security assumption is honest majority of service providers.
5. Both kind of protocols are incentive compatible - it makes financial sense to not diverge from them and to provide the services to clients.
6. Providers set the prices for the provided api services.
7. Clients create a nano-payment relationship with service provider.
8. The system dynamically finds the optimal price for digital communications.
9. Provider may be swapped out at any time by clients.

Core Apps

Instant Messaging - 1 : 1

Group Messaging - n : n

Public Status Feeds - 1 : world

Private Status Feeds - 1 : n

Onboarding User Experience

1. User install client software (mobile or desktop) and creates first identity.
2. User choses a service provider from several available ones on the platform.
3. User starts using one of the core apps for free using its service provider’s free trial program.
4. User invites friends or family members to groups, join groups, subscribe to feeds, publish status feeds, and can create additional identities - some anonymous, some personally identifiable.
5. When the free trail period ends, users who perceive real value from the apps buys Subnet coin and uses it to compensate service provider.
6. Users pay per use of apps features, using nano-payments to service providers.
7. In the future - users will be able to pay for premium content on the platform with Subnet coins.

Instant Messaging

User A → A → SA → SB → B → User B

1. User A creates a message (text, media, audio, etc....).
2. A Client sends the message to B via SA and SB using the core messaging sending algorithm.
3. B gets the message via SB using the core messaging delivery algorithm.
4. B presents the message on the device it is running on for User B.

Instant Messaging Summary

A → SA → SB → B

- A gets B's **ID** over any digital channel.
- A discovers **X3DH(SB, B)** published by SB on the network using **KAD**.
- A creates a DR session **DR(A, SB)** with SB and a DR session **DR(A, B)** with B (or reuses existing sessions).
- A creates encrypted message for B, **M1 = ENC(MSG, B)** with **DR(B,SB)** where **MSG** is the user set message.
- A creates encrypted message **M2 = ENC(M1, B)** with the **DR(A, SB)**.
- A sends M2 to SA with a **signed nano-payment** on **CHAN(A, SA)**.
- SA sends M2 and **X3DH(A, SA)** to SB.

- SB decrypts **M2** to get **M1** and B using the **DR(A, SB).**
- SB informs B it has a message for it and sends M1's metadata (size, etc...) to B.
- B decides it wants to get **M1** and read **MSG**.
- B sends to SB a request to get **M1** with a nano-payment (based on SB agreement with B and MSG size).
- SB verifies the payment and sends **M1** to B.
- B replies to A using a similar algorithm using **DR(A, B).**

Instant Messaging

Security, Privacy and Anonymity

- SA and SB don’t know the cleartext of A’s message to B - only A and B knows it.
- Messages between A and B (and between any other 2 parties) have strong forward and backward security provided by the double ratchet algorithm.
- SA knows that A wants to send a message to one of SB’s clients but it doesn’t know B’s identity.
- SB knows that SA wants to deliver a message from one of its clients to B, but it doesn’t know A’s identity.

- To conclude that A is messaging with B, SA and SB need to collude with each other.
- The platform’s basic security assumption is **2/3 + honest majority of service providers**.
- This gives us a **~89% darkness guarantee** (nobody knows A is talking with B).
- We can achieve this privacy without having the network complexity involved with darkness.
- There is no personally identifiable information in A and B IDs (If A and B wish to be anon). If the network channels A-SA and B-SB are secure then it is hard for 3rd parties to identify them.

Instant Messaging

Incentive Compatibility

- A is incentivized to nano-pay for message delivery because it wants the message to reach B.
- SA is incentivized to route A's message as it is getting paid for giving A utility, and A will stop using it by swapping to another provider if messages will not be routed to destination according to the protocol.
- SA is incentivized to publish X3DH(A, SA) as it can drive more revenue from A (for example A sending a response to a message from B).
- SB is incentivized to store and let B know about messages it has for it because it is getting paid to deliver messages to B on a per-message basis.
- B is incentivized to pay SB for the message because it wants to read it - it pays for the service it wants.

Group Messaging Design Goals

- **Censorship Free Publishing** - Anyone can create any number of groups, post to any group he's member of and read all messages sent by other group members to a group he's a member of.
- **Privacy** - only group members should be able to send messages to the group and receive messages from the group.
- **Security -** Groups should provide high-degree of both forward and backward secrecy.
- **Anonymity -** The only personally identifying group information should be **voluntarily information shared by group members**. Without such an information the only known information about a group a the public id of its members and the id of its super admin / creator.
- **Manageability -** Each group should a super admin who can invite new users to the group, remove users from the group and accept or reject requests to join the group. Super admin may assign admin right to one more other members. Admins may remove other users from the group.
- **Flexibility** - A group member may leave a group at any time.
- **Incentive Compatibility** - It should be incentive-compatible for service providers to maintain groups and to follow the groups APIs.

Group Messaging

Group Creation

1. A sends to SA a **creation request** for a group with a **nano-payment**.
2. SA creates G's ID and stores group data: **meta-data**, **members list**, group **discoverability**, and **join requests**. A is set as G's super-admin.
3. SA sends creation confirmation and G's ID to A.
4. A can advertise G's ID to people he would like to join G over any digital channel.

Group Messaging

Group Discovery

All group members must be first accepted by A to the group.
If a group is discoverable then anyone with knowledge of A's ID can discover it.
Otherwise, it is up to A to share the group's ID with others so they may request to join the group.

1. B sends a message to SB requesting to **list discoverable groups created by A** and a nano payment.
2. SB locates SA via **X3DH(A, SA)** and queries it for **discoverable groups created by A**.
3. SA returns to SB a list of matching groups which it maintains.
4. SB sends the list to B.

Group Messaging

Joining a Group

1. To join a group **G**, **B** needs to know the **G's ID** as well as its creator **A's ID**.
2. **B** sends a message to **SB** with a nano-payment to join G with an optional intro message to **A**.
3. **SB** receives the message, discovers SA via **X3DH(A, SA)**, and sends to **SA** the join request.
4. **SA** delivers the message to **A** using standard core message delivery algorithm.
5. **A** receives the request, **accepts or rejects it** and sends the response to **SA**.
6. If **SA** receives an **acceptance message** then it adds **B** to the **members list** and removes the B request from the **join requests list**.
7. If **SA** receives a rejection message then it deletes **B** request from the **join requests list.**
8. **SA** sends to **SB** an acceptance or rejection message for **B**.
9. **SB** delivers the message to **B** using standard core message delivery.

Group Messaging

Sending messages

B wants to send a new message to a group G that he's a member of. He knows G's ID and A's ID.

1. B asks SB for a **list of G's current members** and sends a nano payment.
2. SB uses **KAD** to locate **SA dial-up information in X3DH(A,SA)** using A's ID.
3. SB connects to SA and requests the list for B. It also sends **X3DH(B, SB)** so SA may establish a DR session with B.
4. SA verifies that B is a member of G, encrypts the list to B using **X3DH(B, SB)** and sends the response to SB. SB sends the response to B.
5. B decrypts the list and uses **KAD** to obtain **X3DH(C, SC)** for each **member C.**

Group Messaging

Sending a message, cont…

1. B prepares a message **for each member C**, and creates or updates a DR session with C using **X3DH(C, SC)**. The message includes the Group G ID, authoring timestamp and the sender's public ID B.
   For replies, the message includes the ID of the message that the new message replies to.
   Each message also has a unique ID and a timestamp. These allow sequential display of group messages and enables group replies.
2. **B sends each message to C via SB**. The message is routed to SC by SB using the core message delivery patterns.
3. C receives new group messages from SC using the core message delivery patterns.
4. C decrypts the message sent by B using **DR** with B and displayed it to its user.

Group Messaging

Incentives

1. SB is incentivized by B to deliver group messages to it on a per message basis.
2. SA is compensated for storing G messages as it is getting paid by A to deliver the messages to it on a per message basis.
3. SA is indirectly incentivized for giving members lists to members as this is required for posting to the group. SA expects A's will post more to an active group and he will be compensated for these messages.

Note that SA doesn't store any group messages - these messages are sent via the p2p network from sender directly to all group members via SA.

Group Messaging

Administration

1. A can give admin right to any group member of a group he has created to help in managing and moderating tasks.
2. An admin may accept or reject group join requests.
3. An admin may remove members from the group.
4. A may remove the admin right from any group member - he’s the only group super-admin.
5. This is straight forward to implement.

Status Feeds

- Feeds are conceptually similar to groups but are asymmetric while groups are symmetric.
- **Private Status Feeds** - only specific identities approved by the status author can experience the feed's content and strong security and privacy is required.
- **Public Status Feeds** - no approval is need to experience the feed's content, no encryption required, as these feeds are designed to be experienced without any limitations by anyone.
- **Subscriptions** - a way for users to experience an aggregation of status updates from multiple feeds in one convenient information stream.

Public Feeds - Design Goals

1. **Free Speech Enabler** - Anyone should be able to post a public status update without any limitations on what content can be posted content - *absolute free speech*. No moderation.
2. **Censorship-free Publishing** - 3rd parties should not be able to censor an entity (person, organization, or AI) from publishing a status update as long as that person can connect to the Internet and put up the nano-payment in coins required to post.
3. **Censorship-free Access To Information** - Anyone should be able to read public status updates posted by others without any limitations such as a 3rd party censoring content from readers.
4. **Anonymity** - Users should not have to reveal any personal identifying information when publishing or accessing an update, and get good anonymity guarantees from platform.
5. **Orgs and AIs inclusiveness** - There shouldn't be any human verification requirements.

Public Feeds

Creation

1. A sends to SA a **feed F creation message** with a nano payment.
2. SA stores F's data. This includes A's ID, F's ID, **meta-data**, **feed discoverability**, and **subscribers list**.
3. SA returns the unique feed ID for F to A.
4. A may advertise the feed ID to people he would like to join the group on any digital channel.

Public Feeds

Discovery

If a feed is not discoverable then it is up to A to share the feed's ID with others so they may experience its updates.

1. B sends a message to SB requesting to **list discoverable feeds created by A** plus a nano payment.
2. SB locates SA via **X3DH(A, SA)** and queries it for **discoverable feeds created by A**.
3. SA returns to SB a list of matching feeds which it maintains.
4. SB sends the list to B.

Public Feeds

Publishing an update

1. A sends SA a signed time-stamped status update (text, media, etc...) together with a nano-payment.
2. **SA stores the update in local store** and indexes it by feed's ID, A's ID and timestamp.

The update's content is not two-party encrypted nor stored encrypted at rest, as it designed for public consumption w/o any authentication requirements.

Public Feeds

Getting Updates

B client has the **feed F ID** - either it was provided it by its user, or it discovered the feed ID by querying SA for public feeds discoverable by A.

1. B sends to SB **a subscribe message** to F providing **F's ID** and A's ID and a nano payment.
2. SB adds F's ID and A's ID to its store of **public feeds that B is subscribed**.
3. SB (periodically or per B's request) queries SA for n**ew feed items for F** (updates authored after timestamp of the last update that it knows about).
4. SA sends the data to SB.
5. SB sends to B **meta-data about new F items** (item ids, timestamp, byte size, etc...).
6. B sends a message to SB with a nano-payment and the **item IDs it would like to receive**.
7. SB **sends the items** to B.

Public Feeds

Architecture Incentives Discussion

1. SA is incentivized to store and to make available to other service providers (such as SB) A's updates as he's compensated per message by A
2. SB is incentivized to retrieve B's messages from SA, store these messages and make them available to B, as he is compensated by B for these messages.

Private Feeds - Design Goals

1. **Privacy and User Control** - Feed publisher completely controls who has access to his the content of the feed and create as many private feeds as he likes.
2. **Free Speech Enabler** - Anyone should be able to post a private status update without any limitations on what content can be posted content - *absolute free speech*. No moderation.
3. **Censorship-free Publishing** - 3rd parties should not be able to censor an entity (person, organization, or AI) from publishing a private status update as long as that person can connect to the Internet and put up the nano-payment in coins required to post.
4. **Censorship-free Access To Information** - Anyone approved by the feed creator, should be able to read private status updates posted by others without any limitations such as a 3rd party censoring content from readers
5. **Anonymity** - Users should not have to reveal any personal identifying information when publishing or accessing an update, and get good anonymity guarantees from platform.
6. **Orgs and AIs inclusiveness** - There shouldn't be any human verification requirements.

Private Feeds

Private feeds are conceptually very similar to groups and can be implemented similarly to groups with the following important differences:

1. Subscriber must be request to subscribe from the feed creator, and be accepted by the feed creator in order to receive the feed's updates.

2. Feed subscribers can't post to the feed only consume feed items.

3. Feed subscribers can't tell who else is subscribed to the feed (privacy).

4. Feed creator can't assign admin rights to subscribers.

5. Private feed updates appear in user's subscription together with public feed updates. Think Twitter feed of updates from multiple sources.

Building Robust Open Systems

- Standardize network protocols binary spec.
- **Everything is broken** indeed. Aim to build a robust reference full node w/o any locking code to avoid hard to debug dead locks and race conditions.
- Fully distributed team - hand-picked talent from all over the world. Cut all costs associated with an HQ.
- Build a reference client app for one platform (desktop web or mobile native).

Subnet Summary

- Vision to create **user-centric digital communications apps** built on top of a new kind **decentralized network infrastructure**.
- A **highly-opinionated project** that is designed to work in a world where one size doesn't fit all.
- Designed to provide an alternative to **centralized communication apps** and other decentralized emerging platforms that have different core values.
- Focus on **designing the core user-centric incentive-compatible protocols** and on prototyping the protocols.
- Initial **inverted designs for fundamental communication apps** - instant messaging, group messaging and status feeds.
- Aim to build *Subnet* with a remote team of exceptional and passionate creators and builders from around the world - no meta, just building.