Towards a Federated Future

A Decentralized Framework for Developer Services

A technical paper prepared for presentation at SCTE TechExpo24

Christopher Aubut
Principal Engineer II
Charter Communications
christopher.aubut@charter.com

Serafim Sukhenkiy
Principal Engineer II
Charter Communications
serafim.sukhenkiy@charter.com

Vinay Radharam, Charter Communications

Aaron Frank, Charter Communications

Justin Pace, Charter Communications

Andy Dolan, CableLabs

Table of Contents

Title **Page Number**

List of Figures

Title **Page Number**

1. Introduction

The telecommunications industry is at a juncture, united by an emerging push to adopt standardized Application Programming Interface (APIs) that empower the broader developer community to build new services that reach a global audience using the industry's resources. This shift necessitates a new framework to address the diversity of operator platforms. In line with API standardization efforts such as Network-as-a-Service (NaaS) [1] and CAMARA [2], this paper proposes a framework for a federated approach to developer registration, user authorization, and API access. Together, these components might enable operators to monetize API interactions while maintaining operator ownership and control of their exposed network resources.

The framework draws upon the principles of decentralized and federated technologies, focusing on existing communication protocols, federated social media platforms (Fediverse) [3], blockchain governance and service assurance models, and public-key cryptography identity management. It prioritizes solutions that ensure operator autonomy in API exposure to the developer community, optimizing the equitable distribution of industry-derived value by sidestepping the need for centralized control. Considerations for privacy and potential monetization are integrated within the wider context of trust among operators, developers, and users.

Advocating for a transformation of the delivery of developer services, this paper suggests moving toward a decentralized and federated system. This shift is presented as a challenge, calling for a collaborative, industry-wide movement toward open, accessible developer services. Outlining a strategy for implementing this framework as a set of developer services, this paper serves as a vision statement. It invites stakeholders to join in forging a future that fully leverages telecommunications technology, fostering innovation and connectivity on a global scale.

2. Industry Landscape

2.1. API Usage in Telecommunications

The telecommunications industry is increasingly leveraging APIs to enhance interoperability and foster innovation. Major industry bodies such as TMForum [4], Global System for Mobile Communications (GSMA) [5], CAMARA, and the Open Geospatial Consortium (OGC) [6] have been involved in this transformation, establishing guidelines and standards to streamline API usage and integration across various platforms. Despite these efforts, many operators still distribute their offerings through independent systems, leading to non-federated environments [7]. This lack of federation complicates interoperability and often results in fragmented service offerings across different operators. The ongoing challenge for the industry is to move beyond these isolated models towards a more unified approach that can facilitate seamless service delivery and integration.

2.2. Challenges Faced by Operators and Developers

Operators and developers encounter several challenges due to the non-standardized nature of APIs within the telecommunications sector. Different operators often implement APIs with unique features and specifications, forcing developers to navigate a complex landscape of disparate systems. This fragmentation necessitates that developers register with each network operator individually, a process that can be time-consuming and inefficient. In addition to these registration challenges, varying authentication schemes such as basic HTTP authorization, Security Assertion Markup Language (SAML), OAuth, API Keys, and mutual Transport Layer Security (TLS) further complicate the integration process. Developers must adapt to each operator's specific security protocols, adding another layer of complexity for developers seeking to integrate with multiple systems.

2.3. Geopolitical Factors and API Performance

The API landscape in telecommunications is also influenced by geopolitical factors, which can impact operational capabilities. Different regions have varying regulations regarding data privacy, security, and cross-border data flows, affecting how APIs are developed, deployed, and managed. For example, the European Union's General Data Protection Regulation (GDPR) [8] imposes requirements on data handling, influencing how APIs are utilized across European networks compared to other regions. These regulatory differences create additional hurdles for achieving consistent API performance and integration globally. Operators and developers must navigate these complexities to ensure their APIs comply with local laws and perform reliably across diverse geographic areas.

2.4. Communicating API Availability and Benefits

Effectively communicating the availability and benefits of standardized APIs to the developer community remains a challenge for the telecommunications industry. Many developers may not be fully aware of the advantages that standardized telecommunications APIs can offer, such as streamlined integration processes and enhanced service capabilities. Moreover, the technical complexities and varying levels of adoption among operators can make it difficult for developers to understand and leverage these APIs effectively. Efforts to design intent-based APIs help bridge the technical gap [9], but industry partners and network operators need to engage proactively with the broader developer community to highlight the opportunities and simplify the onboarding process for utilizing these APIs.

2.5. Network-as-a-Service (NaaS)

Initiatives like Network-as-a-Service (NaaS) are helpful in driving towards a more unified and accessible API ecosystem within the telecommunications industry. NaaS aims to abstract network resources into a service-oriented model, providing developers with easier access to network capabilities without needing to manage underlying infrastructure complexities [1]. Such initiatives represent steps towards reducing fragmentation in API offerings and enabling a more cohesive developer experience.

2.6. Aggregators and Centralized Control Issues

Despite the progress made by initiatives such as NaaS and CAMARA, the presence of aggregators and issues related to centralized control continue to pose challenges. Aggregators often act as intermediaries between developers and network operators, leading to centralized control over API access and usage. This centralization can potentially stifle innovation and limit the autonomy of operators in managing their network resources. To overcome these challenges, a more decentralized approach could be needed, empowering operators and developers to interact directly, fostering a more open and equitable API ecosystem.

3. Principles of Decentralized and Federated Technologies

3.1. Overview of Decentralized Technologies

Decentralized technologies distribute control and authority away from central entities, which play a role in applications ranging from communication to financial transactions. These technologies could potentially enhance security, privacy, and resilience against attacks or failures. Email, for instance, while often operated through centralized services such as Gmail, fundamentally supports decentralized operations through independent servers and protocols like Simple Mail Transport Protocol (SMTP) [10] and Internet Message Access Protocol (IMAP) [11]. This decentralization allows for greater user control and privacy.

The social media landscape is also seeing decentralization with technologies like ActivityPub [12] and Diaspora [13]. ActivityPub facilitates a decentralized and federated social networking environment across different servers, allowing users to interact across a unified platform despite being on separate servers. Diaspora, a pre-ActivityPub platform, similarly enables users to set up their own community-run servers or "pods," which interact within a federated network, placing greater emphasis on user autonomy and data ownership [13].

Blockchain technologies, known for their role in cryptocurrencies like Bitcoin, utilize a distributed ledger where each block contains a set of transactions and a cryptographically secure link to the previous block, forming an immutable chain. This structure ensures that once data is recorded, it cannot be altered without invalidating all later blocks, thus maintaining data integrity and transparency. Blockchain technologies support not only financial transactions but also decentralized applications and smart contracts, which automate agreements by adding semantics to transactions and standardizing the business logic so that all participants share the same state [14].

Moreover, the Extensible Messaging and Presence Protocol (XMPP) offers a decentralized framework for instant messaging and presence information. Its open standards and extensibility allow for broad applications beyond text messaging, including voice communication and file transfers, supporting a diverse ecosystem of communication tools without central oversight [15].

Together, these technologies illustrate the potential and versatility of decentralized systems across various domains, highlighting their role in enhancing user control, data security, and overall system resilience. Moving forward, the expansion and adoption of these technologies will likely play a role in shaping a more decentralized, secure, and equitable digital landscape.

3.2. Public-Key Cryptography for Identity Management

Public-key cryptography is a cornerstone of decentralized systems, providing several security functions. This form of cryptography utilizes a pair of keys: a public key that may be shared openly and a private key that is kept secret by the owner. This mechanism ensures data protection, identity authentication, and secure communications across distributed networks [16].

Digital signatures play a role in establishing secure connections over the internet as the mechanism to authenticate identities. This is used for protocols like Transport Layer Security (TLS) [17] and Hypertext Transfer Protocol Secure (HTTPs) [18], which use Public Key Infrastructure (PKI) [19] to form a chain of trust. Alternative approaches like OpenPGP (Pretty Good Privacy) perform a similar function but allow for fine-grained trust relationships referred to as a web of trust. Both methods allow creating inferred trust relationships. Trust is verified using digital signatures, where an identity signs information using their private key that can then be verified using their shared public key. Digital signatures help confirm the authenticity of a document or message, verifying that it has not been altered and does indeed come from the stated sender, thus ensuring data integrity and non-repudiation [20].

Encryption is another function of public-key cryptography, safeguarding data so that only individuals with the correct key can decrypt and access the information. This is used for maintaining confidentiality in communications. This method of encryption is referred to as asymmetric cryptography, where each sender encrypts traffic using the receiver's public key so that the receiver may use their private key to decrypt the information. For performance reasons, asymmetric cryptography is often used to exchange a shared secret to use more performant symmetric cryptography algorithms [17].

3.3. Federated Social Media Platforms (Fediverse) - ActivityPub

ActivityPub facilitates a decentralized federation of social media platforms through a protocol that allows users on different servers to interact seamlessly as if they were on a single unified platform. This protocol supports the Fediverse, where content and social interactions are distributed across multiple independent servers, enabling platform interoperability and user connectivity across different social media applications. ActivityPub blends HTTPs with an email-like inbox/outbox design to synchronize state on-demand between nodes [21]. By supporting an over-the-top application layer, many feature-rich implementations have emerged, such as Mastodon, Lemmy, and many others [3].

The decentralized nature of these platforms can lead to challenges, particularly with identity management, as identities are tied to the node used for registering an account, complicating issues of portability and ownership. This mirrors how email accounts are tied to a specific host. Governance mechanisms will also vary per over-the-top application layers, or even per host, based on personal preferences of an instance administrator or through community consensus. For instance, Lemmy's ActivityPub application layer enables instance administrators to disable features such as downvoting [22].

3.4. Blockchain Governance and Service Assurance Models

Blockchain technologies form an immutable ledger of transactions to maintain a shared, secure, and transparent state between applications. It functions as an append-only structure, akin to a stack or list, where once data is appended, it cannot be altered. Every transaction is digitally signed to authenticate the author's identity. Transactions are committed to the ledger in sets, or as the name implies, blocks. Each block is paired with a hash of the block's contents along with contents from previous blocks. Altering the contents of any block would break the chain and be noticeable by all participants, thereby enforcing the immutability of the ledger.

Blockchain networks can be categorized into public, private, and consortium types [23]. Public blockchains are open to anyone and are fully decentralized, while private and consortium blockchains restrict participation to specific members or organizations, offering more controlled environments with varying degrees of decentralization. Private blockchains are maintained by a central governing authority and while the ledger is still immutable and transparent, the governing authority is permitted to commit transactions to modify the stored state without oversight [24]. Consortium blockchains require consensus based on agreed-upon rules used to validate each transaction. Public blockchains operate the same way, with the key difference being that a sizeable set of participants is required to prevent bad actors from taking over the network.

Transactions within these networks are validated through consensus mechanisms such as Proof-of-Work, Proof-of-Stake, and Byzantine Fault Tolerance (BFT) [25]. These validation techniques are used for ensuring the integrity of the blockchain and preventing fraudulent activities. Governance in blockchain involves various models, including hierarchical, democratic, or hybrid approaches, each designed to suit the specific needs and goals of the blockchain community [26]. Since many nodes in the network validate transactions to reach consensus, the system must handle faulty or malicious nodes. As previously mentioned, public blockchains require a sizable and diverse representation of participants to ensure this error margin cannot be exploited. Consortium blockchains instead limit the nodes to stakeholders that share some degree of trust.

4. Framework for Federated Developer Services

This framework draws upon the principles of existing decentralized and federated technologies to propose a secure and efficient method for developer registration, application authorization, and API access. The recommendation is to build a consortium peer-to-peer network between operators using public-key cryptography as the foundational trust and identity model. A private network has the same problem as an aggregator, where a few control the shared state for the many, while a public network requires enough scale such that it would be prohibitive for a malicious actor to take over the network. In a consortium model, the industry maintains ownership and jointly controls the shared state. This framework recommends, but does not require, an immutable ledger, leaving it open for the industry to select the best peer-to-peer technologies to suit its needs.

4.1. Operator Peer-to-Peer Framework

In a consortium, stakeholders operate nodes that form a distributed and federated peer-to-peer network. Nodes exchange information through operations that alter the shared state of the federated network. ActivityPub, as the name implies, calls these activities, whereas blockchain technologies commonly refer to these as transactions, with the latter being the preferred nomenclature for this framework. There is a tight coupling between the exchange and acceptance of transactions between nodes through consensus. Consensus algorithms are discussed in more detail in the following section, with this section focusing on their impact to network characteristics.

While a fully connected mesh of peers presents a high degree of fault tolerance, the complexity of maintaining such a requirement at a global scale is prohibitive when factoring in the number of potential stakeholders. As such, this framework recommends the industry adopt a flexible protocol where nodes are not required to connect to every peer in the consortium. For example, the Sawtooth protocol requires that "all nodes must be directly connected to all other nodes" [27]. In contrast, ActivityPub's inbox/outbox model [12] and blockchain technologies such as CometBFT support a more relaxed topology [28].

Other factors to consider when selecting a protocol include catch-up mechanics for new or temporarily offline nodes. ActivityPub's inbox/outbox model [12] and the immutable ledger of blockchain technologies provide such mechanisms inherently. CometBFT supports periodic snapshots to reduce the time needed to catch up to the rest of the network [29]. Ease of upgrading the network is another consideration. The Matter DCL (Distributed Compliance Ledger) implementation added support at the application layer to democratize deploying new versions [30]. Even if the industry selects a protocol based on an immutable ledger, what data is stored on the ledger becomes subject to right-to-be-forgotten regulations such as General Data Protection Regulation (GDPR) and California Consumer Privacy Act (CCPA). Strategies include storing personal information off-chain using a composed peer-to-peer implementation of two or more protocols, zero-knowledge proofs to prove facts about personal information or using encryption with disposable private keys [31].

Latency is a factor in selecting a protocol to ensure a reactive user experience. Protocols such as ActivityPub and XMPP are designed for low-latency communication by forgoing consensus. This sacrifices data integrity and ordering, which is why they alone are not suitable for critical data but may be useful for adhering to right-to-be-forgotten regulations. Practical Byzantine Fault Tolerance (PBFT) requires two communication phases, or round-trip penalties, with $O(n^2)$ complexity, which is why Sawtooth requires a fully meshed network. CometBFT and HotStuff-2's implementations also require two phases of communication, but with $O(n)$ complexity, vastly reducing the number of messages between peers with the latter supporting optimistic responsiveness [32]

The industry should also consider protocols that support the interconnection of several networks. For example, the Inter-Blockchain Communication (IBC) protocol enables separate blockchains to be linked together [33]. When arranged geographically, regions may be formed to minimize latency during everyday consensus, with the performance penalty of geographically diverse nodes only occurring when shared state must be synced between regions.

4.2. Data Integrity and Transparency Framework

Consensus protocols are a component of this framework to help ensure the integrity and ordering of data. ActivityPub and XMPP use a federated model where each node is responsible for its own state. There is still merit in considering these protocols for non-critical state such as data that is only meaningful between a single operator and developer and/or to assist with right-to-be-forgotten regulations. For the core application layer, this framework recommends using the BFT consensus protocol due to its minimal computational complexity, simplicity, and overall performance. With BFT, a supermajority of greater than 2/3s must agree on the validity and ordering of transactions before committing [26]. Since the original PBFT implementation requires a fully meshed network, it is suggested to consider modern variations that improve performance and minimize peering requirements while also supporting IBC. A chosen implementation should also support weighted consensus, which will be discussed in the following sections.

4.3. Application Layer Framework

The peer-to-peer protocol describes how to exchange data in a federated system, while the consensus algorithm details what data is valid when exchanged. The application layer adds semantics to the data exchanged, which is coupled with the consensus protocol by enabling custom validation and state transformation logic. Lemmy and Mastodon are examples of such semantics built on top of ActivityPub, whereas blockchain technologies use smart contracts to add semantics.

4.3.1. Identity Framework

The basis of this framework's model is using public-key cryptography and digital signatures to ensure the integrity and authenticity of shared state. Every public key is associated with an actor such as a person, organization, group, infrastructure node, or service account. Like the earlier sections, this framework does not advocate for any specific public-key cryptography implementation. That said, considerations should be made for post-quantum (PQ) cryptography as a future-proofing mechanism, which is to say different public keys must be used for signatures and encryption [34].

A basic function of this framework will then require brokering public keys to facilitate nodes authenticating the origin of transactions. Consider a password-based model where a user logs onto a website and a hash of the password is used to authenticate the user in exchange for a session token. It would be impossible to discern the true origin of a transaction in a federated model as any node could claim the hash has been verified without evidence. Using public-key cryptography, the holder of the private key signs their transactions such that every node operator can use the brokered public keys to verify the identity that authored such transactions.

4.3.2. Governance Framework

Building upon the identity model, this framework introduces the concept of roles. The fundamental role of this proposal is a trustee, following the Matter DCL, which holds the responsibility for validating transactions through operating a node and performing interactive voting on governance matters [35]. Trustees should participate in BFT consensus to validate transactions where each trustee is allowed up to one node to prevent voting manipulation. That is, they can run validation functions and broadcast pass/fail results to the consortium until a 2/3s supermajority is reached to commit the transaction. When a 2/3s supermajority is reached for any transaction, each node commits the transaction to its local state using agreed upon business, thereby maintaining the same shared state across nodes. In addition, the trustee role can submit proposals and voting transactions on governance matters. Voting by transaction is a multi-transactional, asynchronous, and interactive process with one trustee making a proposal transaction and the other trustees submitting accept or reject transactions until a customizable threshold is met or the proposal expires.

4.3.3. Network Formation Framework

The initial network launch involves a coordinated ceremony to start accepting transactions. As seen in Figure 1, a minimum of four nodes is recommended to reach a greater than two-thirds supermajority with enough tolerance for a single node failure. Trustees agree to a hard-coded genesis state to start the network, which must include who the trustees are, their public keys, and connectivity requirements for each trustee's validator node such as hostnames, IP addresses, and public keys for mutual TLS authorization.

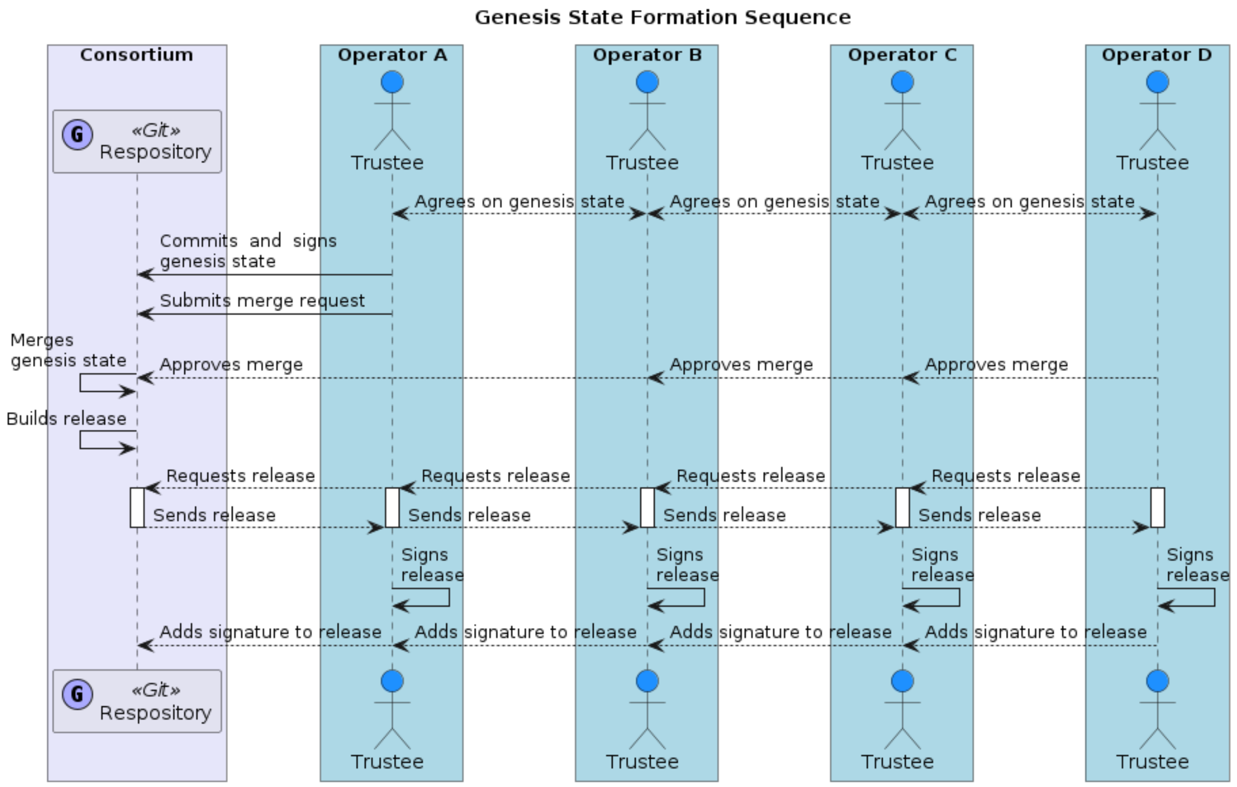

Figure 1: Genesis State Formation Sequence: Step-by-step depiction of the sequence involved in forming the genesis state.

The sequence of steps for a successful launch ceremony, depicted in Figure 2, begins with each trustee starting their node. Nodes will validate the genesis state before establishing a peer-to-peer network with the other nodes in the consortium. Since each node only contains the genesis state, nodes confirm all peers started with the same shared state. So long as greater than two-thirds of the trustee's nodes successfully synchronize, the network will start accepting transactions.

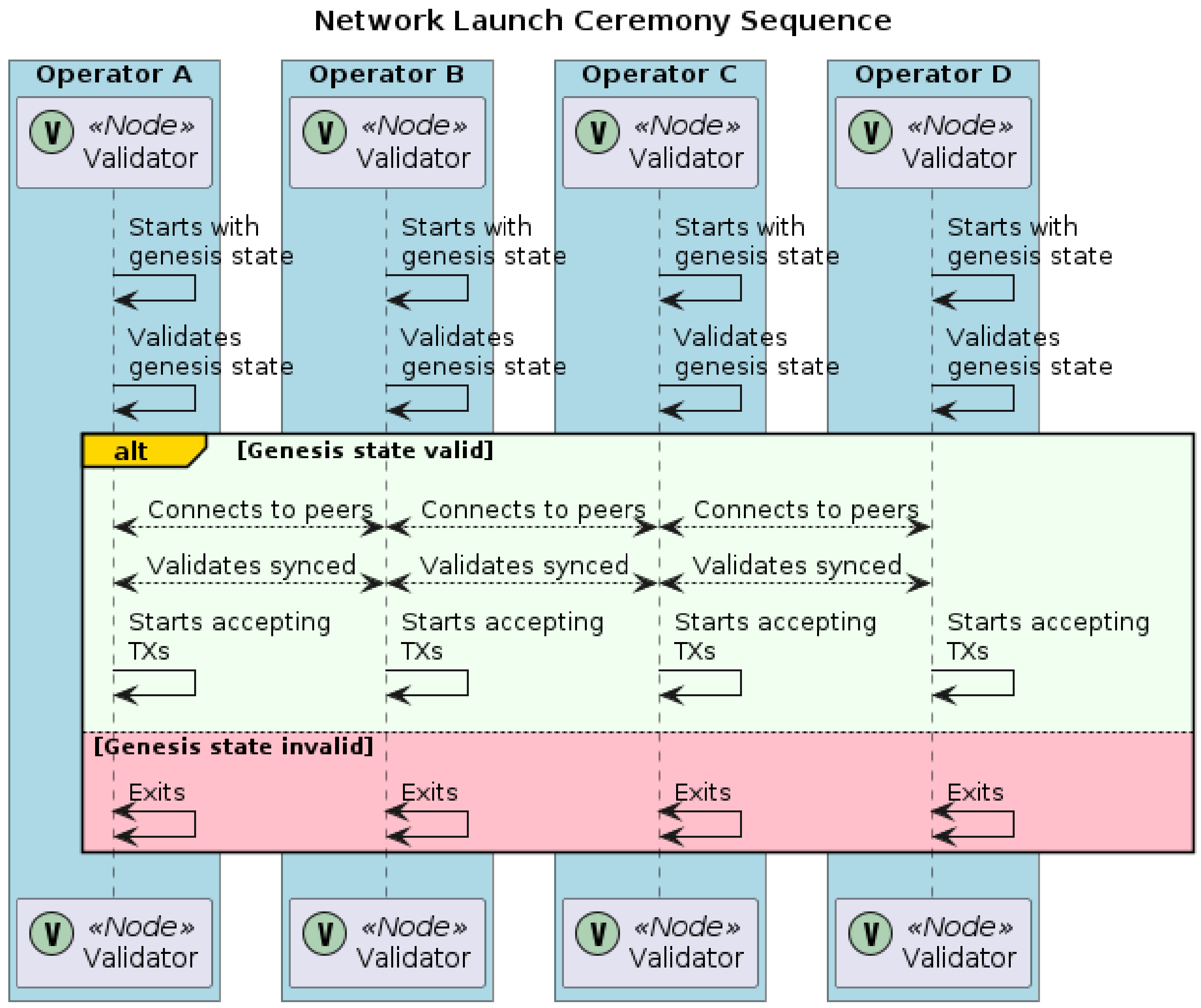

Figure 2: Network Launch Ceremony Sequence: Diagram representing the network launch ceremony, detailing the steps for synchronizing and validating nodes to initiate the federated network.

This initial federated network topology should closely resemble Figure 3. This depiction also includes a user portal for interacting with the network. Each trustee may host such a portal—public or private—as a bridge for submitting transactions to the federated network. Mechanisms for interacting with the network are covered in more detail as the framework unfolds. For now, it is assumed there is a central public user portal with optional private user portals that operators interact with internally.

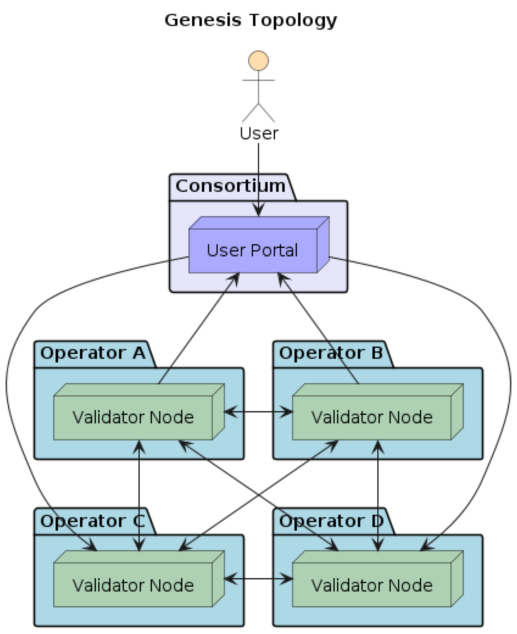

Figure 3: Genesis Topology: Illustration of the initial network topology showing the interconnected nodes at the genesis state of the federation.

The number of trustees grows with each operator or industry partner that joins the consortium. User registration is covered in the next section, but assuming the next trustee has already registered, Figure 4 presents what an interactive voting session should consist of. The user would apply for the role through a portal with the payload signed using their private key. The portal will submit this transaction to a node which then validates and broadcasts the transaction to its peers to do the same. Once all validation checks have passed, the application is committed to updating the shared state of the network. Asynchronously, each trustee must react to these proposals by submitting accept or reject transactions, which in turn must all be validated.

Validation and commit logic are customizable to incorporate a myriad of business logic rules. As seen in Figure 1, trustees may use techniques, such as signing release binaries, to express their verifiable approval of these rules. When nodes are validating the final transaction that pushes voting consensus over its configured threshold, the commit phase executed by each node updates local copies of the shared state by applying the agreed upon business logic. In this case, the trustee role is added to the identity that was proposed and this identity is now able to submit transactions restricted to only trustees. Each node can validate that the new trustee may submit these transactions by inspecting its synchronized copy of the shared state.

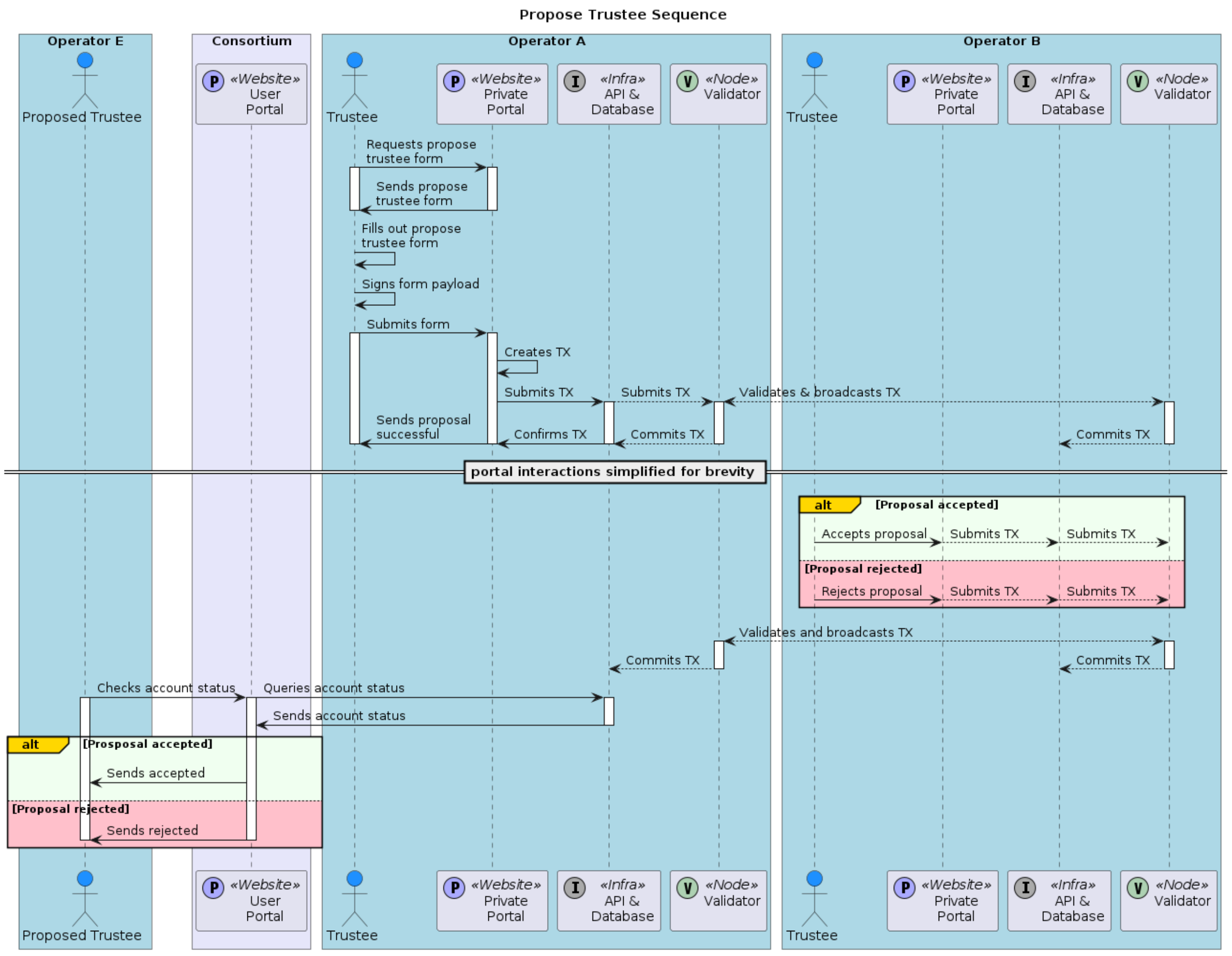

Figure 4: Propose Trustee Sequence: Diagram showing the procedure for proposing a new trustee within the network, including the submission and validation steps.

4.3.4. User Registration Framework

This section proposes the building blocks to handle application registration and API access in a federated system so the industry can maintain developer relationships while retaining operator ownership and control. This framework recommends a flexible authentication mechanism to support existing operator infrastructure that they are in complete control over. This framework recommends OAuth 2.0, which will be used going forward [36], but this framework is flexible to support other mechanisms. Access rights, such as OAuth 2.0 scopes, should be declared at registration time to limit data mining and anti-privacy efforts. Implementations may allow for closed registrations where an interactive voting round is required, like Figure 4, or support open registration, as depicted in Figure 5. With open registration, the flow requires no interactive participation from trustees. The business logic that each node executes is sufficient to automate the entire process in a federated network.

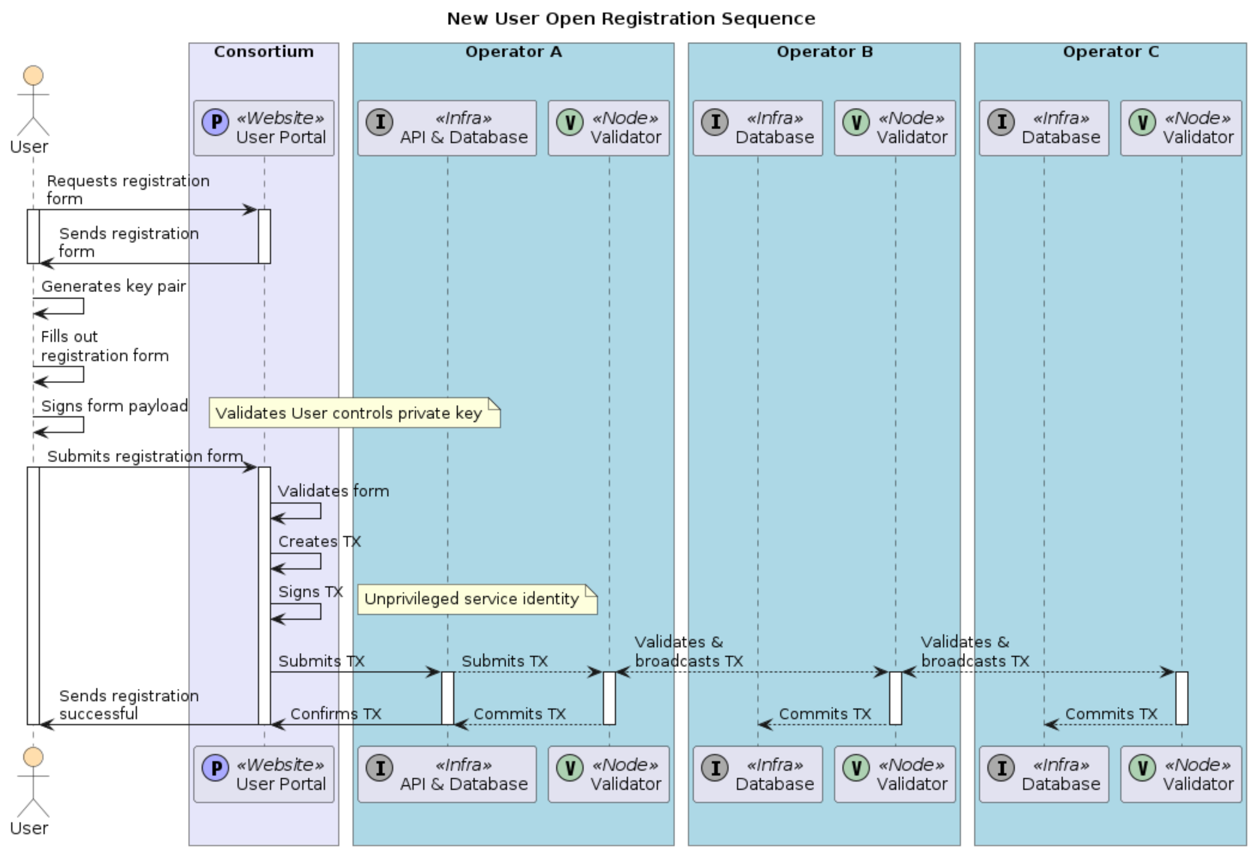

Figure 5: New User Open Registration Sequence: Diagram demonstrating the open registration process for new users, highlighting the steps from user sign-up to successful registration.

Once a user is registered and able to start signing their own transactions, they may now start performing administrative functions, such as forming or joining an organization, setting up contact information, delegating roles, and submitting their application for API access. Figure 6 demonstrates such a sequence where a single developer registers their application, and every operator may approve or reject access to their APIs. This sequence is like interactive voting except for the business logic in the commit. For the commit phase, each operator controls their relationship with the developer which means there are no majority requirements outside of validating transactions.

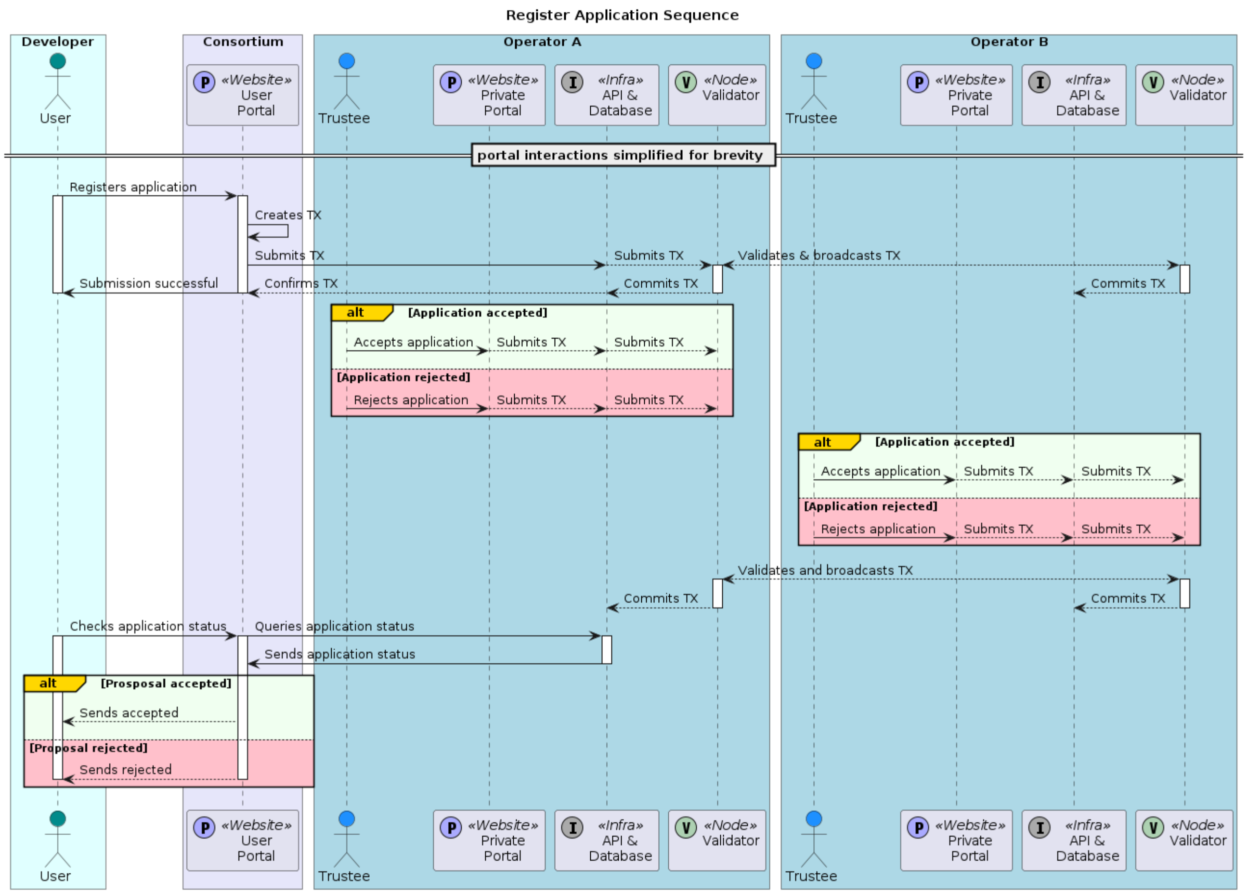

Figure 6: Register Application Sequence: Diagram of actions required for developers to register their applications, illustrating the process from application submission to approval.

When accepting an application, credentials may be broadcast throughout the network such that any node may supply the developer with the necessary credentials. Since these credentials are visible to all nodes, it opens an impersonation attack vector by a malicious or compromised node. Modifying the basic identity structure to include an encryption key can allow secure messaging between an operator and developer. Alternatively, secret credentials may be derived using key encapsulation mechanism (KEM) and key derivation function (KDF) algorithms. Hybrid public-key encryption (HPKE) is an example that abstracts the functionality used by TLS to generate symmetric keys [37].

At the industry's discretion, further augmentation is possible for additional privacy and security. Ring signatures may be used to mask a transaction's originator, where trustees form a ring, and nodes may only verify that one of the private keys in the ring signed the transaction [38]. For instance, if operators do not wish to advertise to the rest of the federated network which developers they have approved or denied, this technique may be used. Successfully employing ring signatures in a consortium requires additional considerations to avoid correlation attack vectors. Trustees may submit these transactions through other nodes or even adopt tunneling technologies which function like Tor or Apple Private Relay for added privacy. This framework also allows for encrypting the recipient such that the federated network is used only as a transport mechanism, but it is not recommended since each identity would need to scan every committed transaction to see if it can be decrypted with their private key.

If completely masking the interaction between developer and operator becomes an industry requirement, out-of-band communication from the main federated network may be used. The front-end portal may be implemented such that it queries each operator's node client-side, but this presents the challenge of implicitly requiring every operator to run a node. An alternative federated protocol, such as XMPP or ActivityPub, may also be considered. These approaches allow for a flexible and secure method for operators to leverage OAuth 2.0 implementations that best suit the industry's needs. Operators are free to add hooks to automate interactions between their internal systems and the federated network to further streamline the process. To promote the expanded footprint greater than any single operator can provide, operators should evolve this model to provide a truly global developer service.

4.3.5. API Access Framework

To facilitate OAuth 2.0 over a distributed and federated network, this framework introduces the concept of an operator list and operator selector to assist with bootstrapping preexisting OAuth 2.0 systems in each operator's infrastructure. The same portal users interact with should be used to access these components. Any node operator may also host a portal to interact with the shared state. It is expected node operators will prefer to use their own portal for interacting with the federated network, which may include custom integrations with their internal infrastructure. That said, it is recommended to have a central portal for developers. Enabling every operator to host their own public portal presents a confusing brand identity and enables a phishing attack vector.

Unlike an aggregator, the portal should exclusively consist of static or read-only assets. For efficiency, it may also host a node that does not participate in consensus but receives transactions in real-time to maintain a local version of the shared state. Trustees should elect a neutral or jointly operated host to serve the public portal. There is not much risk of an entity exerting undue control over the network since all data is controlled and owned by operators. Trustees may at any point decide to relocate the public portal to another hosting platform. In contrast, aggregators maintain the relationship with developers which makes it difficult to change platforms and may impact the equitable distribution of industry-derived value through vendor lock-in when renegotiating contracts.

Returning to the concept of an operator selector, this framework's recommendations support both OAuth 2.0 Client Credential Grant Type (CCGT), or 2-legged authorization to access each operator's resources, as well as Application Code Grant Type (ACGT), or 3-legged authorization to access each operator's subscriber's resources. For CCGT, every developer's app must be permitted to access a list of operators with machine-readable information needed to authorize with each's OAuth 2.0 implementation. Figure 7 represents an example flow using an API call to the central portal, which then proxies the request to a random operator. Alternative approaches, such as the portal containing a read-only copy of the data for fulfilling the request or using a common DNS record to forward the request to any operator in a round-robin manner, are also permissible.

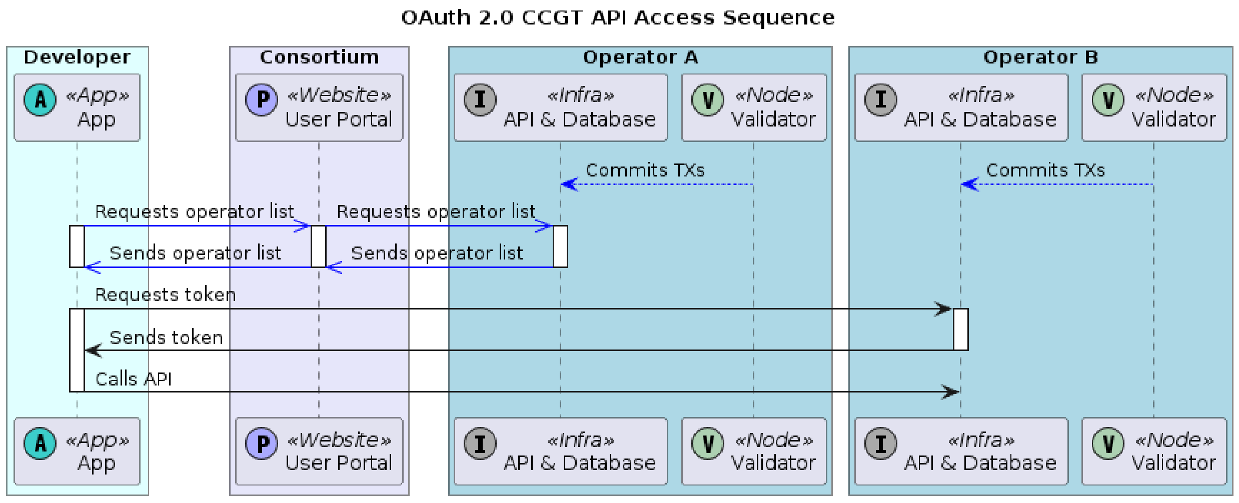

Figure 7: OAuth 2.0 CCGT API Access Sequence: Diagram of the OAuth 2.0 Client Credential Grant Type (CCGT) process, showing how developers gain access to operator resources.

The consortium should provide an SDK for developers to make use of the machine-readable payload. In support of the previous section, this payload may include encrypted credentials or information for obtaining OAuth 2.0 credentials by having the developer's application authenticate with their private key. Other fields may include various OAuth 2.0 endpoints and supported authorization schemes such as client secrets or mutual TLS. Access to the operator list may be authenticated or unauthenticated. Authenticated access is recommended to support including OAuth 2.0 credential exchanges and to filter the operator list to only include operators that have approved the developer's application if the industry decides not to mask which approvals have been given.

This model also extends to 3-legged ACGT authorizations, as depicted in Figure 8, by introducing the operator selector. Expanding upon OAuth 2.0's use of web browser redirects, this flow bootstraps the process by having the user choose their operator, which in turn provides the developer's application with the same payload in the CCGT flow. In both flows, the process is strictly for bootstrapping, and all further interactions are between the developer's application and the operator once complete. This demonstrates a lightweight process where each operator maintains complete control over their resources utilizing existing infrastructure.

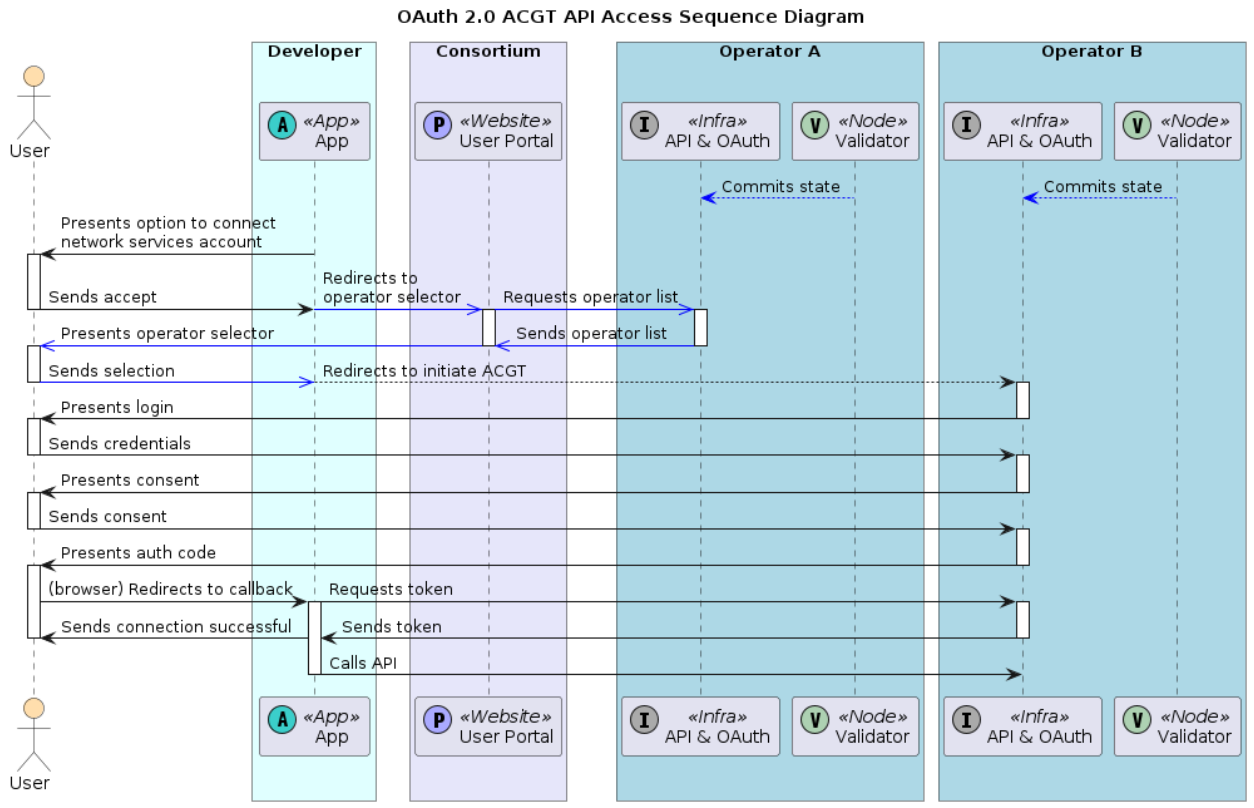

Figure 8: OAuth 2.0 ACGT API Access Sequence: Diagram of the OAuth 2.0 Authorization Code Grant Type (ACGT) process, explaining how developers access subscriber resources through the federated network.

5. Implementation Strategies

The following sections are designed to address challenges and propose additional enhancements for implementing this framework.

5.1. Identity Model Enhancements

Trustees, operators, and developers alike will need to sign their transactions with their private key to authenticate each mutation to the shared state. While users of the portal are likely to come from a technical industry, performing the necessary steps may still present a source of friction. As previously mentioned regarding signing keys and encryption keys, other keys may be attached to a user's identity to streamline the process. This framework recommends implementing modern web browser-enabled public-key cryptography protocols such as WebAuthn. WebAuthn provides a framework for authentication and may be used for creating signatures [39]. Allowing users to connect additional WebAuthn private keys to their identity, such as Passkeys or FIDO2 hardware dongles, should minimize friction for users inexperienced with password-less authentication schemes [40]. It is expressly not advised to perform client-side public-key authentication or encryption using the browser's JavaScript runtime, as it would add unnecessary attack vectors. Browser extensions are another possibility, which is the method the Matter DCL uses, but at the risk of developers not wanting to install the software and impacting adoption. When educating users, it is recommended to direct them to verified end-to-end encrypted (E2EE) solutions to store their WebAuthn private keys. Vendors with solutions that cannot be verified must be assumed to have access to their users' private keys.

5.2. Trust Model Enhancements

Enabling operators to access new revenue streams by offering APIs with a collective footprint is a motivation for this framework. Transparency and cooperative validation using public-key cryptography is the means to maintain trust. Several features, such as organization formation and management, creation and assignment of custom roles and responsibilities, account recovery, compromised keys, and key rotations, are necessary to support the user experience. Implementing these features through transactions on a federated network requires an expanded trust model and identity definition. Two models are recommended: the chain-of-trust and web-of-trust models. An industry-aligned implementation may choose either or both with a hybrid approach.

5.2.1. Chain-of-Trust

In a chain-of-trust model, one or more private keys are chosen as a root authority. Root authorities will sign public keys for other identities they trust. In turn, those identities may become intermediate authorities by signing keys they trust, thereby forming a chain of trust. This model is used by most protocols that verify the validity of DNS records. The Transport Layer Security (TLS) protocol has the server send an X.509 certificate to the client when connections are initiated. Each client has access to a set of certificates for trusted root and intermediate authorities [19]. X.509 certificates contain enough information for the client to walk the chain of signatures to one of its trusted certificate authorities. This sequence of steps verifies that the identity who was issued a DNS record is in control of their private key.

This model complements forming hierarchical organizations where each organization can issue and revoke certificates for its descendants. It solves challenges related to account recovery, compromised keys, and key rotations where parent certificate authorities may revoke and reissue certificates. The Matter DCL uses this method on its federated network for device certification and attestation [41]. There are challenges with this model in that certificate authorities must maintain Private-Key Infrastructure (PKI) or choose another certificate authority. This may also lead to a situation where a few chains hold the authority for the many. Mitigation techniques, such as cross-signing, should be employed when implementing [42].

5.2.2. Web-of-Trust

The web-of-trust model takes the approach of every identity signing the keys they trust. Like the previous model, unidirectional chains may form where A trusts B and B trusts C so A trusts C. These chains connect, forming a web rather than a hierarchy. Additionally, trust relationships are not required to be shared. Using OpenPGP with email is a common application where the original intent was for two individuals to exchange public keys in person to later authenticate each other's exchanged emails [20]. If shared, it is possible to implement business logic for transactions where one identity may delegate permissions to another identity to sign transactions on their behalf to assist with key recovery and compromised keys. Identities may also be able to add backup keys to their own identity to self-serve these processes without trusting another. The challenge with this model is the level of responsibility on users to manage these trust relationships and maintain tight control over their keys, where a lost private key may lead to loss of access to the identity. This can create user overhead and friction when engaging with the network. Additionally, while a web-of-trust can emulate a chain-of-trust, it may require implementing tooling already present in PKI.

5.3. Geopolitical Considerations

To minimize the risk of a single geopolitical entity exerting undue control over the network, enhancements to the framework should allow for weighted consensus and voting powers.

Implementations may group trustees by their geopolitical affiliation or along geographic lines such as the five regions used by the Internet Assigned Numbers Authority (IANA) [43]. Validation and transactional votes are then weighted by the number of trustees in each group such that no single group can obtain a supermajority. Implementations may support subgroups with group-specific weights. A peer-to-peer network that supports a protocol like Inter-Blockchain Communication (IBC) complements such a design.

5.4. Voting Delegations

To minimize the burden of interactive voting participation, an implementation should consider allowing identities to delegate their voting power. This involves designing transactions where one trustee may delegate their vote to another identity under certain conditions. Delegating voting functions to user or service identities such that a trustee's private key may be stored securely offline is one such use case. Another is trustees delegating votes for governance functions to industry partners that maintain authenticated memberships with other operators to reduce the number of trustees required for interactive voting. Combined with group formations in the previous section, implementations should explore allowing different sets of identities to configure their voting structure, such as electing boards or arranging by industry. Weighting must be factored into the design to ensure a healthy representation of participants.

5.5. Monetization

While a detailed monetization model is outside the scope of this document, the framework supports a path forward for a potential worst-case scenario of utilization-based monetization with the requirement that the developer receives a single invoice encapsulating usage from all operators. The following is a basic framework for recording API calls within the shared state. While BFT consensus is generally considered fast, such overhead to every API call would impact the user experience. To mitigate this latency, this framework suggests API clients build and sign API call transactions. When the API client calls the operator's endpoint, they include the signed transaction payload as either a header or query parameter (e.g., https://api.example-operator.com/endpoint?tx=0x123). When an operator receives the request, they validate the signature and asynchronously broadcast the transaction to the federated network.

Operators may fulfill the request before the transaction is committed to eliminate all latency overhead. This may be applied conditionally for trusted developers at the operator's discretion. Pre-processing the API request and holding it until the transaction is committed is another option. For a read request, the operator may hold the response, and for mutations, the operator may perform any reads necessary to process the request and pause mutations until the transaction is committed to minimize latency with less risk. Operators will want to run a node to ensure they have real-time access to the developer's status, such as spend limits (post-pay), balance (pre-pay), and other relevant information. To safeguard developers from delayed transaction processing, transactions should expire if not committed within a certain window. To mask activity from other operators, this may also be combined with ring signatures and encrypted payloads, which is the main feature behind privacy-based cryptocurrencies such as Monero [38]. This provides a viable alternative to aggregators without giving up the customer relationship.

6. Benefits and Challenges

The proposed federated framework for developer registration, user authorization, and API access offers numerous benefits to the telecommunications industry. By providing a unified and consistent API interface, the framework simplifies integration efforts, allowing developers to focus on building innovative services rather than dealing with the complexities of disparate systems. This could accelerate the deployment of new services, enhancing the industry's agility and responsiveness to market demands. The framework might foster a more inclusive ecosystem where developers from various backgrounds can contribute and thrive, potentially driving the telecommunications industry toward a more interconnected and dynamic future.

Privacy and monetization considerations are part of the framework, balancing user data protection with opportunities for operators to generate revenue. By embedding privacy-preserving mechanisms within the federated system, the framework could ensure that user data is handled responsibly and securely. Additionally, monetization strategies could create mutually beneficial arrangements between operators and developers, incentivizing the creation and deployment of high-quality services. This balanced approach builds trust among all stakeholders, maybe ensuring that the framework meets technical and operational requirements while aligning with broader ethical and economic goals.

However, several challenges must be addressed for successful implementation and widespread adoption. The current initiative is limited by the participation of a relatively small number of companies. Without significant stakeholder engagement, the platform could struggle to gain the necessary momentum. Ensuring widespread adoption requires demonstrating long-term benefits through compelling case studies and fostering collaborative efforts across the industry.

The user experience also presents a challenge. Each operator may desire control over their own frontend user portal, leading to fragmentation and increased phishing risks. To mitigate this, implementing standardized front-end interfaces can provide a consistent and secure user experience.

Managing private keys, logins, and recovery processes poses challenges. Traditional public key systems can be complex and cumbersome for users, leading to friction in their adoption. To reduce friction and enhance security, the system should employ modern authorization mechanisms. These measures can improve user convenience and security, ensuring seamless access to the federated system.

Integrating different authorization mechanisms across various platforms can lead to inconsistencies and potential security vulnerabilities. Adopting a unified authorization bootstrapping framework that seamlessly supports multiple mechanisms using machine readable configurations likely ensures interoperability and high security standards. Ensuring that all components of the framework adhere to stringent security protocols is likely crucial to protecting sensitive data and maintaining the integrity of the telecommunications ecosystem.

Addressing these challenges comprehensively is probably crucial for the successful implementation of the proposed federated system. By promoting widespread industry participation and developing secure, user-friendly interfaces and authorization mechanisms, the framework can overcome these hurdles.

7. Conclusion

This framework for implementing a set of federated developer services provides a comprehensive approach to modernizing the telecommunications industry through a decentralized and federated model for developer registration, user authorization, and API access. By leveraging existing communication protocols, federated social media platforms, blockchain governance, and public-key cryptography, this framework could enable operators to maintain autonomy, ensure data security, and promote equitable value distribution.

The GDS framework addresses several industry challenges, including the need for standardized APIs to streamline development and integration processes, thereby enhancing innovation and responsiveness. It also proposes solutions for privacy and monetization, ensuring responsible data handling and creating mutually beneficial relationships between operators and developers.

However, the successful implementation of this framework requires overcoming significant hurdles. The primary challenge is increasing operator participation in its development. Ensuring widespread industry participation is essential to gain the necessary momentum for the GDS framework. Demonstrating the long-term benefits through compelling case studies and fostering collaborative efforts across the industry are likely crucial steps in achieving this goal. Together, the telecommunications industry can further drive global innovation and connectivity, creating a more open and accessible telecommunications landscape. By addressing these areas, the GDS framework can establish a secure, efficient, and collaborative telecommunications ecosystem.

Abbreviations

ACGT	application code grant type
API	application programming interface
BFT	Byzantine fault tolerance
CCGT	client credential grant type
CCPA	California Consumer Privacy Act
DCL	distributed compliance ledger
DNS	Domain Name System
E2EE	end-to-end encryption
FIDO2	Fast IDentity Online 2
GDPR	General Data Protection Regulation
GSMA	Global System for Mobile Communications Association
HPKE	hybrid public-key encryption
HTTPS	Hypertext Transfer Protocol Secure
IANA	Internet Assigned Numbers Authority
KDF	key derivation function
KEM	key encapsulation mechanism
IBC	Inter-Blockchain Communication
IMAP	Internet Message Access Protocol
NaaS	Network-as-a-Service
OGC	Open Geospatial Consortium
P2P	peer-to-peer
PBFT	practical Byzantine fault tolerance
PGP	Pretty Good Privacy
PKI	public-key infrastructure
PQ	post-quantum
RTC	real-time communication
SDK	software development kit
SMTP	Simple Mail Transfer Protocol
TLS	Transport Layer Security
XMPP	Extensible Messaging and Presence Protocol

Bibliography

[1] "NaaS: Network as a Service," CableLabs, 2024. [Online]. Available: https://www.cablelabs.com/technologies/naas. [Accessed 2024].

[2] "CAMARA Project," The Linux Foundation, 2024. [Online]. Available: https://camaraproject.org/. [Accessed 2024].

[3] "Fediverse," 2024. [Online]. Available: https://fediverse.party/. [Accessed 2024].

[4] "TM Forum Introduction," TM Forum, 2024. [Online]. Available: https://www.tmforum.org/. [Accessed 2024].

[5] "GSMA Introduction," GSMA, 2024. [Online]. Available: https://www.gsma.com/. [Accessed 2024].

[6] "Open Geospatial Consortium," Open Geospatial Consortium, 2024. [Online]. Available: https://www.ogc.org/. [Accessed 2024].

[7] A. Hawkes, "Understanding the API economy," Console Connect, 21 8 2023. [Online]. Available: https://blog.consoleconnect.com/understanding-the-api-economy. [Accessed 2024].

[8] "General Data Protection Regulation," European Union, 25 May 2018. [Online]. Available: https://gdpr-info.eu/. [Accessed 2024].

[9] J. Taaffe, "Telstra's Mark Sanders on open architectures, APIs, intent and NaaS," TMForum, 18 9 2023. [Online]. Available: https://inform.tmforum.org/features-and-opinion/telstras-mark-sanders-on-open-architectures-apis-intent-and-naas. [Accessed 2024].

[10] J. Kliensin, "Simple Mail Transfer Protocol RFC," IETF, October 2008. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc5321. [Accessed 2024].

[11] M. Crispin, "Internet Message Access Protocol RFC," IETF, March 2003. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc3501. [Accessed 2024].

[12] "ActivityPub," W3C, 23 1 2018. [Online]. Available: https://www.w3.org/TR/activitypub/. [Accessed 2024].

[13] "Welcome to diaspora*," Diaspora Foundation, 2024. [Online]. Available: https://diasporafoundation.org/. [Accessed 2024].

[14] C. F. T. Commission, "Primer on Smart Contracts," Lab CFTC, 27 11 2018. [Online]. Available: https://www.cftc.gov/sites/default/files/2018-11/LabCFTC_PrimerSmartContracts112718.pdf. [Accessed 2024].

[15] "XMPP Introduction," 2024. [Online]. Available: https://xmpp.org/. [Accessed 2024].

[16] N. Sullivan, "A (Relatively Easy To Understand) Primer on Elliptic Curve Cryptography," Cloudflare, 24 10 2013. [Online]. Available: https://blog.cloudflare.com/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography. [Accessed 2024].

[17] "The Transport Layer Security (TLS) Protocol Version 1.3," IETF, 8 2018. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc8446. [Accessed 2024].

[18] "HTTP Over TLS," IETF, 5 2000. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc2818. [Accessed 2024].

[19] "Internet X.509 Public Key Infrastructure Certificate," IETF, 5 2008. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc5280. [Accessed 2024].

[20] "OpenPGP Message Format," IETF, 11 2007. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc4880. [Accessed 2024].

[21] "Federation," LemmyNet, 21 6 2023. [Online]. Available: https://github.com/LemmyNet/lemmy-docs/blob/32f26b42735aad01ce496e9db8ef58abf0dd36f8/src/administration/federation_getting_started.md. [Accessed 2024].

[22] "Lemmy (commit 5cc798a) [Source Code]," LemmyNet, 27 5 2024. [Online]. Available: https://github.com/LemmyNet/lemmy/blob/5cc798a14694a4bae98af3f56ddc0901139d2c33/crates/api/src/post/like.rs#L35. [Accessed 2024].

[23] "Permissioning Design," Splinter Community, [Online]. Available: https://sawtooth.splinter.dev/docs/1.2/architecture/permissioning_requirement.html#sawtooth-network-scenarios. [Accessed 2024].

[24] E. J. Beyer, "OpenSea's Stolen Item Policy Reveals a Stubborn Problem," NFT Now Media, 18 10 2022. [Online]. Available: https://nftnow.com/features/openseas-stolen-item-policy-reveals-a-stubborn-problem/. [Accessed 2024].

[25] F. Johnand, M. Olehand and C. Luciano, "Consensus Mechanisms In Blockchain: A Deep Dive Into The Different Types," Hacken, 6 4 2023. [Online]. Available: https://hacken.io/discover/consensus-mechanisms/. [Accessed 2024].

[26] L. Daly, "What Is Byzantine Fault Tolerance?," The Motley Fool, 20 November 2023. [Online]. Available: https://www.fool.com/terms/b/byzantine-fault-tolerance/. [Accessed 2024].

[27] "Using PBFT Consensus," Splinter Community, 26 1 2024. [Online]. Available: https://github.com/splintercommunity/sawtooth-docs/blob/main/docs/1.2/pbft/using-pbft-consensus.md. [Accessed 2024].

[28] "Introduction," CometBFT, [Online]. Available: https://docs.cometbft.com/v0.37/introduction/. [Accessed 2024].

[29] "CometBFT Snapshot," CometBFT, 2024. [Online]. Available: https://github.com/cometbft/cometbft/blob/c5dfd20653babac1c06a1b0beb3a84c5d437faf1/spec/abci/abci%2B%2B_basic_concepts.md#state-sync-methods. [Accessed 2024].

[30] "Pool Upgrade How To," Zigbee Alliance, 11 8 2022. [Online]. Available: https://github.com/zigbee-alliance/distributed-compliance-ledger/blob/master/docs/pool-upgrade-how-to.md. [Accessed 2024].

[31] "The Right to be Forgotten Meets the Immutable: A Practical Guide to GDPR-Compliant Blockchain Solutions," Cavath, Sawine & Moore LLP, [Online]. Available: https://www.cravath.com/a/web/636/3898415_1.pdf. [Accessed 2024].

[32] D. Malkhi and K. Nayak, "HotStuff-2: Optimal Two-Phase Responsive BFT," Cryptology ePrint Archive, 2023. [Online]. Available: https://eprint.iacr.org/2023/397.pdf. [Accessed 2024].

[33] "IBC Introduction," Interchain Foundation, 2024. [Online]. Available: https://www.ibcprotocol.dev/. [Accessed 2024].

[34] C. Boutin, "NIST Announces First Four Quantum-Resistant Cryptographic Algorithms," National Institute of Standards and Technology, 5 7 2022. [Online]. Available: https://www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistant-cryptographic-algorithms . [Accessed 2024].

[35] "Matter Distributed Compliance Ledger," Silicon Labs, [Online]. Available: https://docs.silabs.com/matter/2.2.2/matter-dcl/. [Accessed 2024].

[36] "OAuth 2.0 Authorization Framework," 2012. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6749. [Accessed 2024].

[37] "Hybrid Public Key Encryption," IETF, 13 5 2022. [Online]. Available: https://datatracker.ietf.org/doc/rfc9180/. [Accessed 2024].

[38] "Ring Signature," Monero, [Online]. Available: https://www.getmonero.org/resources/moneropedia/ringsignatures.html. [Accessed 2024].

[39] "WebAuthn," WebAuthn.io, 2024. [Online]. Available: https://webauthn.io/. [Accessed 2024].

[40] "Passkeys," FIDO Alliance, [Online]. Available: https://fidoalliance.org/passkeys/. [Accessed 2024].

[41] "Matter PKI Compliance Guide," Amazon, 20 12 2022. [Online]. Available: https://d1.awsstatic.com/whitepapers/compliance/matter-pki-compliance-guide.pdf. [Accessed 2024].

[42] S. Helme, "Cross-Signing and Alternate Trust Paths; How They Work," 22 6 2020. [Online]. Available: https://scotthelme.co.uk/cross-signing-alternate-trust-paths-how-they-work/. [Accessed 2024].

[43] "History of the Regional Internet Registries," APNIC, [Online]. Available: https://www.apnic.net/about-apnic/organization/history-of-apnic/history-of-the-regional-internet-registries/. [Accessed 2024].