ΕΠΑνΕΚ 2014-2020
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΑΝΤΑΓΩΝΙΣΤΙΚΟΤΗΤΑ • ΕΠΙΧΕΙΡΗΜΑΤΙΚΟΤΗΤΑ • ΚΑΙΝΟΤΟΜΙΑ

ΔΡΑΣΗ ΕΘΝΙΚΗΣ ΕΜΒΕΛΕΙΑΣ:
«ΕΡΕΥΝΩ-ΔΗΜΙΟΥΡΓΩ-ΚΑΙΝΟΤΟΜΩ»

re-cent

Ολοκληρωμένη υπηρεσία διαμοιρασμού δικτυακών πόρων για την εξατομικευμένη διανομή ψηφιακού περιεχομένου σε δίκτυα δεδομένων 5ης γενιάς

ΚΩΔΙΚΟΣ ΕΡΓΟΥ Τ1ΕΔΚ-03524

ΠΑΡΑΔΟΤΕΟ Π4.1: Αναφορά τεκμηρίωσης της υπηρεσίας Re-cent διαθέσιμη σε ειδική ιστόσελίδα

Συμμετέχοντες

1. **Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών – Τμήμα Πληροφορικής και Τηλεπικοινωνιών (DIT)**
2. **Fogus Innovations and Services IKE (FOGUS)**
3. **Κέντρο Έρευνας Πανεπιστημίου Πειραιά – Τμήμα Ψηφιακών Συστημάτων (Πα.Πει./ΚΕΠΠ)**
4. **Νέσσος Πληροφορική Α.Ε. (NESSOS)**

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Ταμείο Περιφερειακής Ανάπτυξης
ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ ΥΠΟΥΡΓΕΙΟ ΟΙΚΟΝΟΜΙΑΣ & ΑΝΑΠΤΥΞΗΣ ΕΙΔΙΚΗ ΓΡΑΜΜΑΤΕΙΑ ΕΤΠΑ & ΤΣ
ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ, ΕΡΕΥΝΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΥΔΕ ΕΤΑΚ
ΕΠΑνΕΚ 2014-2020 ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΑΝΤΑΓΩΝΙΣΤΙΚΟΤΗΤΑ ΕΠΙΧΕΙΡΗΜΑΤΙΚΟΤΗΤΑ ΚΑΙΝΟΤΟΜΙΑ
ΕΣΠΑ 2014-2020 ανάπτυξη - εργασία - αλληλεγγύη

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Συμβατική ημ/νία παράδοσης:	Ιούλιος 2021 (Μήνας 36)
Πραγματική ημ/νία παράδοσης:	Ιούλιος 2021
Υπεύθυνος φορέας:	ΝΕΣΣΟΣ
Συμμετέχοντες φορείς:	DIT, ΠΑΠΕΙ, NES, FOGUS
Ασφάλεια:	Δημόσιο
Είδος:	Παραδοτέο
Έκδοση:	2.0

Πληροφορίες εγγράφου

Ημερομηνία έκδοσης: 21 Ιουλίου 2021

Αριθμός σελίδων: 208

Συγγραφείς

Όνομα	Φορέας	Email
Νικόλαος Επίσκοπος	FOGUS	nepiskopos@fogus.gr
Διονύσιος Ξενάκης	FOGUS	dionysis@fogus.gr
Κωνσταντίνος Κεφαλάς	FOGUS	kkefalas@fogus.gr
Χρήστος Κούλης	FOGUS	ckoulis@fogus.gr
Παντελής Πετρογιαννάκης	NESSOS	ppetrogian@nessos.gr
Απόστολος Ζαφείρης	NESSOS	azafeiris@nessos.gr
Νικόλαος Παλλαδινός	NESSOS	npal@nessos.gr
Ευανθία Νικολοπούλου	NESSOS	enikolopoulou@nessos.gr
Αλέξανδρος Κρητικός	NESSOS	akritikos@nessos.gr
Νίκος Πασσάς	DIT	**passas@di.uoa.gr**
Χρήστος Μηλαροκώστας	DIT	en3180005@di.uoa.gr
Ειρήνη Λιώτου	DIT	eliotou@di.uoa.gr
Σταύρος Αναστάσιος Χαρισμιάδης	DIT	anachar@di.uoa.gr
Άννα Ζαμπάτη	DIT	azampati@di.uoa.gr
Μαρία Λιάτσου	DIT	mliatsou@di.uoa.gr
Χρήστος Ξενάκης	ΠΑΠΕΙ	xenakis@unipi.gr
Ηλίας Πολίτης	ΠΑΠΕΙ	ipolitis@ssl-unipi.gr
Κατερίνα Πούπουζα	ΠΑΠΕΙ	kpoupou@unipi.gr
Κωνσταντίνος Λαμπρινουδάκης	ΠΑΠΕΙ	clam@unipi.gr

Ιωάννης Μακροπόδης	**ΠΑΠΕΙ**	gmakropodis@ssl-unipi.gr
Βάϊος Μπολγούρας	**ΠΑΠΕΙ**	vaiosbolgouras@ssl-unipi.gr
Δημήτρης Πατούνης	**ΠΑΠΕΙ**	dpatounis@ssl-unipi.gr
Δημήτρης Κούδας	**ΠΑΠΕΙ**	dkoudas@ssl-unipi.gr

Summary

The new ‘‘status quo’’ created by the emerging integration of Multi-Access Edge Computing (MEC) and Radio Access Network (RAN) infrastructures, combined with the openness of the 5G service market, have set new standards on how the multitude of 5G service domains should be incentivized and coordinated to comply with the performance requirements set on a per session basis. In this paper, we revisit the predominant offline contract-driven mobile data access model enabling users to gain access to the infrastructure of only a few network operators and propose a new blockchain-backed mobile data access model where the key 5G stakeholders can trade, share and consume mobile edge network assets (access to the internet, spectrum, processing, storage, local content etc.) in a fully decentralized, anonymous and highly-robust fashion. Blockchain-backed mobile data access should address critical practical challenges of blockchain decentralization, scalability and security in the context of 5G and Beyond networks.

Accordingly, we provide a meticulous survey of existing solutions in the aforementioned three areas and present a comprehensive holistic framework of protocols to address the key challenges identified, including a delegated Proof-of-Stake (DPoS) for distributed blockchain consensus over 5G and Beyond mobile data networks, a future-proof smart-contract enabled payment relay service enabling instant off-chain payments in a highly robust fashion as well as a hybrid mixing protocol that enables payment relays to act as anonymous payment hubs while addressing the unique challenges opposed by the joint blockchain and network level interaction of the 5G and Beyond service components. A wide range of practical implications and mitigation measures for dishonest operation of the blockchain nodes are investigated and sophisticated yet highly-robust incentive engineering mechanisms are derived. Detailed numerical results also accompany the paper, bringing to light the unique performance trade-offs and valuable design guidelines towards blockchain-backed mobile data access for 5G and Beyond mobile data networks.

In this deliverable, we document the RE-CENT resource trading platform, provide numerical results through computer simulations as well as testing scenarios and results from the RE-CENT testbed setup, as part of the work done in WP4.

The presented work is based on the design of the RE-CENT service in references [81] and [82].

Πίνακας περιεχομένων

Πίνακας εικόνων

Πίνακας πινάκων

1. Introduction

The relentless growth of mobile video traffic, along with the demand for massive connectivity and fully personalized content consumption, have set new standards for mobile data access. According to recent reports [1], by 2025, i) mobile broadband subscriptions will reach 8 billions, ii) connected devices shall exceed 50 billions (including Internet-of-Things - IoT - devices) and iii) mobile data traffic will surpass 160 Exabytes per month. In view of that, the telecommunication industry has put considerable effort towards the consolidation of the 5th generation (5G) of mobile data networks. Firstly, with the release of a standalone version of the New Radio (NR) [2], 3GPP has specified the architectural options towards the evolution of LTE-compliant networks to fully-fledged 5G systems, incorporating forward-thinking radio access technologies, such as multi-GHz spectrum communications, dynamic beamforming and small-sized cellular hotspots. Secondly, via the means of network softwarization and virtualization, the mobile network operators (MNOs) will be in position to dynamically isolate network resources towards serving specific vertical applications with guaranteed quality of service (QoS), or quality of experience (QoE), a.k.a. network slicing [3], or even open up their core network functionality to other third party (OTT) service providers, e.g. authentication, authorization, accounting (AAA).

In parallel, standardization towards the integration of Multi-access Edge Computing (MEC) into the 5G Radio Access Network (RAN) is ongoing. Currently, it aims to enhance network performance and reliability by leveraging edge resources residing within coverage of 5G cellular networks [4]. Dynamic pooling and coupling of edge network resources (including spectrum, storage, processing, and infrastructure) using MEC is key enabler for reduced service creation times, massive edge network connectivity, multiaccess and multi-service support in 5G. Combined with the potential of migrating core network functions to the edge, using network function virtualization (NFV) [5], MEC integration into 5G will enable the numerous 5G key stakeholders (cloud service providers, virtual network providers, content providers, OTT service providers) to create a complex mesh of multi-layered 5G service domains on top of which the myriads of end devices will enjoy fully personalized 5G service consumption with zero-perceived downtime. Although a handful of technical reports and relevant projects have worked towards MEC/RAN integration [6]–[8], fully personalized, anonymous and user-centric mobile data access over joint MEC/RAN infrastructures is in its infancy [9].

Mobile data access is currently carried out through multi-tier, multi-domain and multi-owned RAN islands that span from user-installed access points for private use (e.g. local Wi-Fi

routers) to large-scale mobile data networks enabling nation-wide coverage and world-wide roaming. Multi-tenancy has been shown to improve the utilization of existing network resources and enhance service coverage in demanding scenarios with occasional spatiotemporal peaks [10], e.g. mega events in stadiums. Nonetheless, cellular mobile data access by the end terminals is still governed by prescribed data usage plans that are agreed offline with the ‘‘home’’ MNO in the form of fixed-term contracts. In parallel, non-cellular mobile data access over wireless local area networks (WLANs) is either based on leasing backhaul connectivity to the Internet in the long-term (enabling access to a closed set of users using credentials), or on providing short-term access coupons to the Internet using fiat money payments, or on allowing ‘‘free-of-charge’’ access to the Internet by collecting and processing the users’ personal data,e.g. by enforcing the users to log in a web service with their social media accounts.

Hence, mobile data access in pre-5G systems is static and lacks the flexibility required to support on-the-fly service creation on top of the abundant edge resources coexisting under the numerous 5G service domains. Another critical issue relates to how the key 5G stakeholders (including end users) shall position themselves against the complexity, size and peculiar characteristics of the emerging 5G market. On the one hand, the 5G service providers should comply with the newest regulations and operating standards (e.g. EU general data protection regulation - GDPR), to attain a minimum service coverage over large geographical areas and to serve as anchor points for supporting external OTT services while meeting the QoS, or QoE, requirements set per user. Adding to this, the MNOs have increasingly became responsible for authenticating and charging their subscribers to third party services, using dedicated core network units (e.g. Authentication Server Function - AUSF) that are potential single points of failure and targets of distributed denial of service (DDoS) attacks [11]. 5G service providers with large-scale coverage inevitably offer ‘‘canned’’ (and thus non-personalized) service contracts (e.g. monthly data usage plan at a fixed price) to geographical regions with diverging characteristics (e.g. demographic density, network topology, available technology, user profiles), boosting the risk of increased customer churn and investments of low added value.

On the other hand, fully personalized service consumption dictates end users to share their personal preferences with their service provider(s) in a persistent and typically eponymous fashion (i.e. social media identities, physical network identifiers, service credentials linked to their user identity, and location data) [12]. From a technological viewpoint those features are necessary for seamless service discovery/advertisement, service negotiation and parameterization, pricing and online service optimization (including mobility management and

QoE provisioning). However, the increased awareness raised by the end users on recent technological advancements enabling service decentralization and enhanced user privacy, e.g. blockchain-backed systems and anonymity services [13], [14], urge the 5G service providers to revisit the way they implement their services. Besides, end users are more skeptical on the numerous ways through which the 5G service providers collect and distribute their subscribers' data to third-party brokers [15]. Having familiarized with the concepts of sharing/circular economies, additional skepticism has been raised by the consumers on whether the mobile data market can be regulated by a handful of nation wide MNOs that dominate the worldwide mobile data market.

It readily follows that end-to-end (e2e) service provisioning, continuity, charging, user authentication and data privacy across the plethora of service domains thriving under the 5G umbrella, urge for the design and wide deployment of fully-distributed mobile data access models (and mechanisms) that will enable flexible creation and user-centric service consumption on top of the numerous *network assets* available at the 5G network edge. Service delivery of this type goes beyond the ‘‘mondus operandi’’ of existing systems and standards, necessitating steps forward from the current network-controlled user-assisted service provisioning model dominating the pre-5G service market, i.e. service control by the ‘‘home’’ network. On-the-fly user-driven network assisted mobile service consumption, which is specifically designed to exploit the superior performance features of 5G and Beyond networks, is the future of mobile data access.

In this paper, we provide an in-depth study on how the blockchain technology can be utilized to turn the today's evidently under-organized and vastly heterogeneous mobile data network, where different operators, regional network and content providers, user-installed access points and end terminals, share no interest in improving the networking experience of end users belonging outside their subscriber's whitelist, to a fully decentralized, dynamic and competitive (by consensus) market where different stakeholders have clear incentives to improve mobile data access and content consumption of mobile users falling within their coverage. To this end, we propose and investigate the potential of a new operator-less (in the sense of fixed term contracts signed offline) mobile data access model where the key 5G stakeholders can trade, share and consume mobile edge network assets (access to the internet, spectrum, processing, storage, local content etc.) in a fully decentralized, instantaneous and anonymous fashion. The main vehicle used to achieve this is a specialized crypto-currency platform that we propose, a platform enabling all 5G stakeholders to act both as network asset servers and clients(consumers) by integrating disruptive new blockchain mechanisms that are specifically

designed to effectively support the demanding 5G and Beyond mobile data access use scenario.

The proposed crypto-currency platform can be implemented in the form of two specialized smart contracts (SCs) that remain unchanged for the system lifetime and are designed so as to i) democratize the block validation process, by employing Delegated Proof of Stake (DPoS) over the multi-billion nodes constituting the heterogeneous 5G and Beyond mobile data network infrastructure, ii)scale the transactions capacity of the system to multi-millions transactions per second, enabling support of multi-billion mobile data network devices, and iii) safeguard the anonymity of service peers that jointly operate in both the network and blockchain domains. Our design is not specific to any blockchain framework but requires support of smart contracts (SC) by the blockchain infrastructure. Since the Ethereum (ETH) platform is currently the most popular and well-documented platform enabling the implementation of SCs, in [41] we have chosen to provide a proof-of-concept implementation of the proposed crypto-currency platform using the Parity Aura ETH framework.

To enable a more comprehensive understanding of the concepts and solutions discussed in this work, to the remainder of this paper we focus on the sharing and trading of mobile video content. We choose to do so provided that mobile video will account for over 74% of the total mobile data traffic by 2024 [1]. Besides, other network assets, such as Internet/local connectivity, spectrum, processing, and storage, can be utilized in the process of improving the mobile video delivery service. For example, an asset server can deliver the requested mobile video content by simply providing Internet connectivity towards a free-of-charge video content provider (e.g. YouTube). Alternatively, an asset server can mobilize local storage resources to cache mobile video content during off-peak periods, downscale the pre-cached video content using MEC transcoding upon request and deliver it on-demand with minimum backhaul link usage.

In the sequel, we identify and provide specific protocols to address three main practical challenges towards the smooth integration of blockchain-backed service support into the baseline operation of 5G and Beyond mobile data networks: the consensus protocol, the transactions capacity and the blockchain anonymity challenges. All three challenges are detailed in sections 1.1, 1.2, and 1.3, respectively.

1.1 THE CONSENSUS PROTOCOL CHALLENGE

In the heart of every blockchain system lies the distributed *consensus protocol*, which ensures that all participating nodes share a common view on the transaction history recorded in the

public ledger (i.e. the blockchain) [16], [17]. The distributed consensus protocol specifies message passing across the consensus network, local decision making at each node and the methodology used to append new blocks of transactions to the blockchain (at least 51% should follow the respective blockchain update). Wide acceptance of a blockchain-backed mobile content delivery platform by the key 5G stakeholders, necessitates the implementation of a scalable consensus protocol enabling active engagement of the vast number of 5G service peers (end devices, core network entities, servers, etc.) to the consensus process while taking into consideration their heterogeneous functional capabilities (e.g. limited access to spectrum, processing, storage and energy sources).

The consensus protocol should also incentivize the 5G service peers to engage with both the blockchain maintenance process and the actual 5G service implementation, e.g. by electing block sealers in the blockchain domain and by sharing their available resource pools in the network domain, respectively. Existing consensus protocols typically consider homogeneous capabilities across the consensus network, or assume the formation of clusters with stable connectivity to the Internet (e.g. mining pools in BTC), or delegate blockchain maintenance to a privileged set of consensus nodes that is fixed and a-priori known. On the other hand, mobile content trading in 5G and Beyond networks dictates the design of forward-thinking incentive engineering mechanisms that encourage active engagement of the myriads 5G service components in both the blockchain and network domains (e.g. by striking a good balance between blockchain and network domain reward mechanisms), but also enforce trusted operation of the key blockchain actors through the deployment of credible yet sustainable penalty mechanisms.

1.1.1 Project contribution in the area

In this project, we have developed a distributed consensus protocol that is specifically designed to meet the peculiar characteristics of blockchain-driven mobile video content trading in 5G and Beyond networks. The proposed protocol uses DPoS to authorize a small set of network nodes, which we term as *validators*, seal new blocks for a given time epoch (measured in blocks) in a round-robin fashion. Validators are short-listed and elected on the basis of the largest sum of stakes (coins) locked in their favor (by other network nodes) on a specialized SC that is deployed on-chain: the VSC. Network nodes are incentivized to stake in favor of a candidate validator to i) share a reward that is offered by elected validators (free market emulation), ii) receive a higher priority in their transactions, and iii) enjoy *free-of-charge* (FoC) mobile data from a given set of network nodes. FoC service is provided by network nodes which, instead of staking coins, they promise to offer free service to the supporters (a.k.a. witnesses) of a tagged candidate validator. Honest operation of validators and FoC servers is

enforced by the VSC logic, which employs sophisticated penalty mechanisms exploiting the (locked) funds staked in favor of candidate validators. We detail the proposed DPoS protocol in section 3.2.

1.2 THE TRANSACTIONS THROUGHPUT CHALLENGE

Existing crypto-currency platforms, including the Bitcoin (BTC) [13] and the SC-enabled ETH platform [14], lag the scalability to carry out the transactions volume required by a *pay-per-chunk* mobile video delivery model. According to official Youtube statistics [18], by Q3 of 2020, video consumers generate billions of views on the platform to watch over one billion (1B) hours of video every day, with 70% of the YouTube watch time coming from mobile devices. Similarly, 100 million hours of video are consumed on Facebook every day with 96% of users accessing content through their mobile device [19] (an approximate number of 1.59 billion users per day). Calculated on the basis of only a few of the popular social media platforms like YouTube and Facebook, the total number of *video consumption requests per second* in a world-wide scale for 2020 is estimated to reach 100-200K, highlighting the vast *transactions throughput* (i.e. the number of finalized transactions per second) requirements that a blockchain-backed mobile video trading platform has in 5G and Beyond service scenarios.

This number is in fact a tight lower bound if we consider that i) the average watch time of mobile video consumers is distributed across multiple content delivery platforms, ii) mobile video consumers typically generate multiple video views with shorter video watch times, and iii) mobile video consumers are required to handover across multiple 5G service peers due to user mobility. The combined impact of these effects stresses the need for a blockchain-backed mobile content delivery platform that is capable of supporting multimillion transactions per second (TPS). In comparison, by Q3 2020, widely-used crypto-currency platforms like BTC and ETH currently process an average of up to 7 [20] and 17 [21] TPS, respectively. Besides, even with the use of traditional payment services, support of multi-million peer-to-peer (P2P) payments in fiat currencies is currently impossible [23], [43].

1.2.1 Project contributions in the area

Blockchain-backed content trading over 5G and Beyond mobile data networks urges for the design of highly-scalable blockchain systems that support a multi-million TPS capacity. In this work, we employ DPoS consensus by using a small set of authorized nodes to seal blocks in a round-robin fashion. Adding to the TPS increase following from DPoS consensus, we further develop an innovative payment relay service that effectively combines on-chain and off-chain scaling techniques. Following an epoch-by-epoch approach, we develop a specialized SC (RSC) that is responsible for licensing payment relays and resolving potential disputes

between the relay servers and their clients. To receive a relay license by the RSC, candidate relays should specify a set of operating parameters (max TPS, amount of associated coins by relay clients, promised deadline for updating the balance of payees according to off-chain transactions, relay fees, etc.) and participate in an auction-based scheme that shortlists relays with the highest offers. Depending on the license type they request, relays should lock a minimum amount of funds in the RSC. After doing so, interested payers can establish payment channels to a relay of their preference through the RSC.

Accordingly, the relays receive payment requests from their clients (payers), validate their payment channel balance and issue payments to the designated payees by signing (but not posting) regular on-chain payments. Committed by their operating license, relays should post positive balance updates to payees within the promised deadline (termed as relay delay) and receive legitimate fees. If a relay fails (or refuses) to do so, payees can initiate a dispute resolution mechanism by triggering the RSC. Disregarded payees shall be reimbursed from the funds locked by the relay during the relay licensing phase. The proposed payment relay service not only enables web-based implementation of the relay servers but also enforces decentralized control and honest operation of relays with the minimum amount of on-chain interactions, i.e. using the RSC only for licensing, balance updates and dispute resolution. Section 3.3 details the proposed payment service.

1.3 THE BLOCKCHAIN ANONYMITY CHALLENGE

Although the wide public considers that existing cryptocurrency platforms enable anonymous payments in a fully decentralized fashion, an increasing body of academic studies and analysis tools have revealed that the identity of blockchain users can be exposed using simple deanonymization attacks [23]–[26]. Accordingly, a new market of anonymity-enhancing services has emerged [27]–[35]. The so-called *mixing services* typically implement coin mixing (or tumbling) protocols that are specific to the platform for which they have been designed. In most of the cases, mixing protocols target to the support of *k-anonymity*. Such protocols

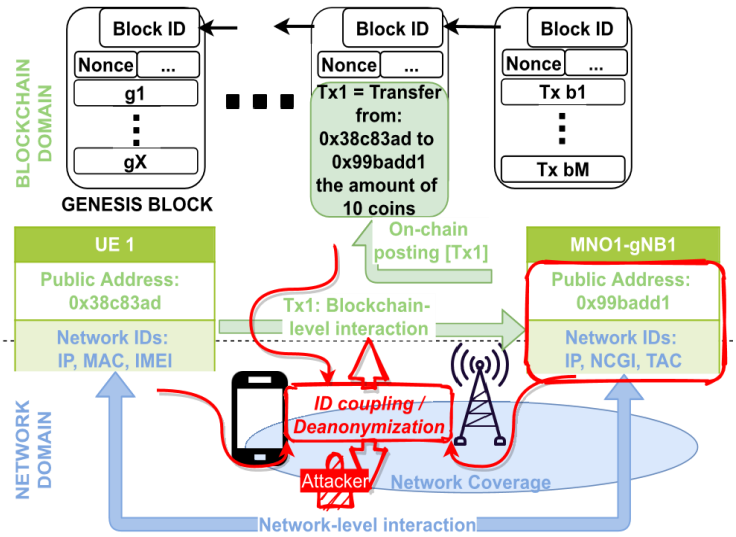

Figure 1: The Network/Blockchain ID coupling problem

take as an input the same (fixed) amount of coins from k individual users (a.k.a. *payers*) and redistribute them to a designated set of output addresses (a.k.a. the *payees*' addresses) in a way that makes it difficult for third parties to link the payment to specific payer-payee pair. Depending on the mixing protocol, different levels of anonymity can be provided. For example, some protocols preserve anonymity from outside observers but enable the mixing server to be infer on the input/output mixing pairs [28].

Preserving user privacy and anonymity in a blockchain backed 5G service ecosystem poses unique challenges that have not been addressed before by the blockchain community. Existing mixing protocols cannot cope with a scenario where the 5G service peers are in physical proximity to deliver/consume mobile data services using both their network-level (MAC, IP, IMEI, CGI, SSID, etc.) and blockchain-level (public addresses) identifiers. Network level interaction combined with online posting of transaction at the blockchain-level (to initiate, continue, or conclude the 5G service delivery) enables potential adversaries to couple the two types of identifiers, opening the door to successful deanonymization attacks (Fig. 1). Accordingly, joint network-level and blockchain-level interaction hinders the effectiveness of fundamental user privacy protection measures, such as the use of fresh blockchain identifiers per new transaction. The *network/blockchain ID coupling problem* is further exacerbated by any event that increases the volume of transactions between the service peers, including handover events due to user mobility, or the employment of micro-payments.

ToR networks [24], [36] and MAC/IP spoofing techniques [37]–[40] also fall short under the ID coupling problem given that: i) ToR networks are designed for fixed data networks (non-wireless) assuming a stable connection to the Internet and cannot scale in view of a wireless service scenario, and ii) not all mobile data users will be in position to effectively hide their

physical network identifiers using such techniques due to their limited resources (e.g. energy, processing, spectrum). Besides, distributed deanonymization attacks can be set up, by closely monitoring many 5G service peers across distant geographical areas and tracking their participation to specific mixing services.

1.3.1 Project contributions in the area

Different from existing approaches, in this paper we leverage *hybrid mixing* according to which, central mixers carry out the actual mixing service in an offline fashion but their honest operation and availability are enforced by a specialized SC in a decentralized fashion. To achieve this, the proposed mixing protocol enables payment relays to act as anonymous payment hubs by extending the operation of Tumblebit in a SC-enabled environment. In more detail, the proposed mixing protocol exploits the payment channels established between network nodes and the payment relay servers (through the RSC) as an equivalent to the on-chain escrow transactions required for the Escrow and the Cash-out phases in Tumblebit, minimizing the on-chain cost necessary for the mixing service.

Founded on the licensing and penalty mechanisms integrated in the RSC logic, the mixing protocol enables on-chain reimbursement of the mixing participants (by the RSC) whenever the mixer goes offline, or acts maliciously, thus, guarantying the *balance* and the *availability* of the mixing service as well. Resistance against Sybil and DoS attacks is established due to the relay fees necessary for implementing off-chain transactions, whereas unlikability at the mixing server is guaranteed through the use of blind signatures and the employment of the generic puzzle solution/puzzle promise protocols of Tumblebit. The proposed mixing protocol not only enables theft prevention, but can also attain instant mixing times for network nodes that are attached to a payment relay. Section 3.4 presents our mixing protocol.

2. MOBILE DATA ACCESS BEYOND 5G

2.1 KEY IDEA AND CONCEPT

In a heterogeneous wireless network, mobile users are interested in consuming video content hosted by servers located in the far Internet. To achieve this, end users utilize different radio access technologies (RATs) to gain access to the available network tiers, having as an entry key pre-cached access credentials provided by their home network operator, e.g. subscriber ID, IMSI, social media account, network keys and passwords. Depending on the RAT and the access rights of end users, mobile data access is governed by i) the coverage provided by the accessible network tiers in the near area, ii) the status of nearby attachment points (in light of the additional load offered by other users and the backhaul connectivity available), and iii) the mobile data usage plan agreed with the home operator per user.

In contrast, the proposed mobile data access model enables end users, access points and cellular base stations to share, trade and consume their network assets (backhaul links, Internet connectivity, cached content, etc.) in real-time and without a-priori service license agreements (SLAs). Under the proposed mobile data access model, which we term as the *REsource sharing model for user-CENTric digital content delivery over beyond 5G mobile data networks (RE-CENT)*, end users (termed as RE-CENT clients) and service providers (termed as RE-CENT servers) can set up on-the-fly service agreements and implement blockchain-backed service charging on a per delivered video chunk basis in line with their current service requirements, coverage, available assets (including local content) and preferences.

At minimum, end users and service providers should hold a blockchain ID (public address) and be capable of assessing the blockchain status, e.g. by querying consensus nodes that are responsible for maintaining the RE-CENT blockchain (transactions propagation, block validations, consensus protocols, etc.). Depending on their operational requirements and functional capacity, RE-CENT nodes undertake specialized roles that we detail in Section 3.1.2, e.g. block validators, payment relays, mixing servers, consensus nodes, witnesses. Even though existing MNOs and large-scale providers will still have a competitive advantage due to their large-scale coverage and reputation, under the RE-CENT model, any network asset holder is enabled to trade under-utilized network assets, complementing (or even competing with) large-scale MNOs in geographical regions with poor service coverage, or non-competitive prices.

Support of the RE-CENT mobile data access model, necessitates the employment of functional and operational enhancements that span both the blockchain and network domains.

In the blockchain domain, the mobile data network should be built on-top of a blockchain-backed platform that enables i) a high degree of *decentralization*, by enabling different roles and levels of engagement to the 5G and Beyond key stakeholders, ii) *scalability*, by supporting a multi-million transactions throughput and iii) *security*, by addressing the blockchain/network ID coupling problem. Necessary blockchain-level enhancements are elaborated in section 3. In the network domain, fully-decentralized and personalized mobile video content consumption beyond 5G, urges the industry and academia to further delve into user-controlled network-assisted procedures, a design approach that has been within scope of large standardization bodies over the past decade, e.g. user-driven network selection and service control by IEEE [62], [63], autonomous cell search by 3GPP [2], [64], [65], MEC/RAN integration by ETSI [7]–[9].

In the sequel, we discuss network-level enhancements that are necessary for blockchain-based design for mobile video content delivery in 5G and Beyond networks. To this end, we decompose the implementation of the proposed service into three phases (Fig. 3): i) service discovery and pairing (section 2.2), ii) service negotiation and parameterization (2.3), and iii) online service management and charging (2.4). The main actions performed during each phase and some key implementation issues are discussed in detail.

At this point, it is important to note that we do not consider that all network functions necessary for mobile video delivery are migrated to the RE-CENT blockchain. Such an approach is not scalable and would insert to the blockchain-backed mobile video delivery process enormous overheads for keeping track of all network-level interactions at a global scale. Thus, service discovery, pairing, negotiation, parameterization and management (including optimization through finetuning of network-level parameters) are still performed by using network-level protocols that are modified accordingly. However, the RE-CENT blockchain-backed platform implements credible service charging, enabling on-the-fly service setup between RE-CENT nodes without a-priori SLAs. Also, SCs are used only for enforcing distributed consensus and honest operation of RE-CENT nodes, such as block validators, payment relays and coin mixers, but not for formalizing SLAs between RE-CENT nodes.

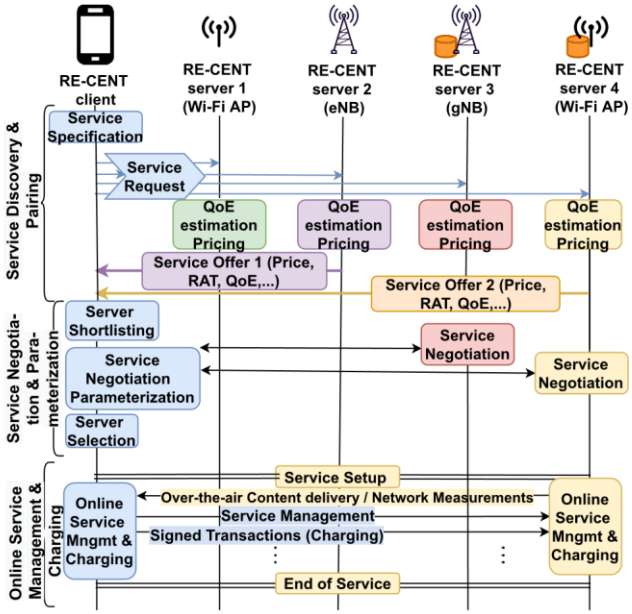

FIGURE 3. RE-CENT service phases.

Figure 2: RE-CENT service domains

2.2 SERVICE DISCOVERY AND PAIRING

Service discovery and pairing is the inextricable prelude for any network-level service. During this phase, RE-CENT clients communicate their service requirements to RE-CENT servers, specifying necessary parameters regarding the target content (e.g. URL, author, keywords) and the target QoE key performance indicators KPIs. RE-CENT servers should advertise their available network asset pools and respond to the service requests with targeted offers, also specifying necessary implementation details (e.g. RAT, resources). Service discovery and pairing can be implemented using *client-driven*, or *server-driven* approaches. Under the client-driven approach, RE-CENT clients shall broadcast their service requests and receive targeted service offers from RE-CENT servers that are interested in serving the tagged request. Under the server-driven approach, RE-CENT servers shall advertise their available asset pools, including locally cached content, data rates for Internet connectivity, tariff list, etc., and RE-CENT clients shall filter service offers to select the most suitable server (service pairing).

Apart from the approach used to communicate service requests and offers, the medium through which this process is implemented plays a key role in the overall robustness of the blockchain-backed mobile trading platform. Service discovery and pairing can be implemented using *network-level*, or *blockchain-level* interactions. Network-level interaction enables the RE-CENT service peers to communicate their requests/offers off-chain, exchanging local network messages over-the-air. On the other hand, blockchain-level interaction enables the RE-CENT service peers to advertise their requests/offers in the public ledger, enabling on-chain service

discovery and pairing. A *mixture* of both methods can take place as well: i) on-chain service requests by RE-CENT clients [32] and off-chain offers by RE-CENT servers, or ii) on-chain advertisements by RE-CENT servers [66] and off-chain service requests by RE-CENT clients.

Although on-chain service discovery and pairing is appealing, due to the transparency and immutability offered when the service data are recorded on-chain, it also comes with enormous on-chain costs and transactions throughput overheads. Taking into consideration i) the large volume of available network assets worldwide (e.g. billions of content), ii) the large number of service peers in a 5G ecosystem (e.g. 50B devices by 2025 [1]), iii) the dynamic nature of the wireless medium and iv) the negative impact of user mobility on the 'freshness' of service requests/offers [67], keeping asset requests/offers up-to-date on a per service peer basis, updating the tariff list on a per asset/server basis, or filtering and extracting service requests/offers from the public ledger, is not scalable with current blockchain-backed systems. Since the mobile video content delivery necessitates physical proximity between the service peers, network-level service discovery and pairing is more practical, mitigating on chain costs and transactions capacity overheads for service control over the public ledger. User-driven service discovery and pairing is also more relevant with the case study under scope, as it enables reactive service control whenever necessary and mitigates the requirement for a large number of messages for the proactive advertisement of generic service offers.

Under the RE-CENT mobile content trading platform, service discovery and pairing is implemented using network level interactions between the service peers in a user-driven fashion. RE-CENT clients broadcast their requests over-the air and RE-CENT servers reply with targeted offers using network-level protocols. The main steps of the RE-CENT service discovery and pairing phase are illustrated in Fig. 2. A discussion of how this process can be integrated in Rel. 16 of the 3GPP 5G system is provided in section 3.1.5.

2.2.1 QoS/QoE-ESTIMATION

RE-CENT clients should be able to specify all parameters of mobile video consumption (Fig. 3), e.g., min/average video bitrate, delay tolerance, packet loss rate, available buffer, target screen resolution. Similarly, the RE-CENT server should be able to estimate its capability to carry out a service request on the basis of the parameters specified by the RE-CENT client and the locally available asset pools (content, spectrum, Internet connectivity, processing and storage capacity, RAT interfaces, etc.). Current literature on QoS/QoE estimation is vast [68]–[72], including a wide range of KPIs and methodologies for QoS/QoE service provisioning at both the client and the server sides. QoS/QoE parameter values specified during this phase, are formalized and adjusted during the negotiation and parameterization phase.

2.2.2 ASSET PRICING

Asset pricing refers to the logic followed by the RE-CENT servers to conclude on the tariff they should offer to RE-CENT clients depending on the received service requests, availability of local network assets and the service offers provided by other servers. RE-CENT servers shall follow their own *asset pricing strategy* and communicate their offers using network-level protocols. Asset pricing should account for the target QoE KPI values specified in the service request message. For example, a higher video bit rate would increase the service cost, e.g. users with larger screen resolutions consume more spectrum resources. The RAT used to implement the last hop of the content delivery service also plays a key role on the pricing strategy, e.g. unlicensed spectrum typically incurs lower costs but lower QoE guarantees. The robustness (supported rates, on-time, jitter, etc.) and cost of backhaul links available for implementing the end-toend content delivery chain should also be considered. If a costly backhaul connection is used to reach distant Internet servers, asset pricing should adapt accordingly. Service costs shall not only account for the cost of utilized spectrum, but should additionally incorporate depreciation and operation costs, e.g. purchase of equipment, energy consumption.

Asset pricing can also take into consideration the current status of the local market. RE-CENT servers can be part of a larger cluster of servers that aim to increase their reputation, or local market share, offering lower prices to attract new users. A higher price can be also requested if the RE-CENT server monopolizes the local asset trading market, or it is widely accepted as trusted. Another important pricing parameter is the availability of requested content in the near area. For example, the server can exploit its local storage resources to fetch popular content in its local cache and lower the price of popular video chunks. In this scenario, the effectiveness of the content placement strategy, which involves content popularity prediction and optimized local storage management, will be clearly a competitive advantage in light of an open 5G and beyond mobile data trading market [73]. Besides, lower prices can be attained if the server is part of a larger content caching and delivery ecosystem that utilizes InformationCentric Networking (ICN) [74].

2.3 SERVICE NEGOTIATION AND PARAMETERIZATION

During this phase, the RE-CENT server employs its own strategy to select the most suitable RE-CENT server (service offer selection in Fig. 3 and interact at the network-level towards service parameterization. Having received the offers of nearby RE-CENT servers, RE-CENT clients shall deploy their own *server selection strategies* to shortlist service offers. During this process, the RE-CENT client should take into consideration criteria regarding i) the price

included in the offer, ii) the RAT options available by the RE-CENT server, iii) the QoE KPI values specified in the offer, iv) the reputation of the RE-CENT server (e.g. servers of large MNOs can be considered as trusted) and v) other service implementation options provided by the server (e.g. support of a given codec, use of minimum encryption). After shortlisting the offers, RE-CENT clients shall negotiate with shortlisted RE-CENT servers important service parameters spanning the entire protocol stack. Firstly, service peers shall conclude on the target QoE KPIs, having as a starting point the original service request/offer messages. Secondly, they should specify the RAT technology to be utilized, the spectrum bands through which the delivery will take place, security parameters regarding the encryption protocol, etc. Thirdly, if a payment relay (or mixing) service is to be utilized, i.e. to reduce on-chain costs necessary to implement service charging, or use micro-payments, service peers should further agree on the payment relay.

To select RE-CENT server(s), RE-CENT clients shall take into consideration network-level parameters specified during the service negotiation phase (mentioned above) as well as blockchain-level parameters, such as the on-chain balance of candidate RE-CENT servers (RE-CENT with increased stake in the system can be considered as more credible), the requested amount of coins for the target service, or other roles assigned to the public address of the server (e.g. FoC server). Multiple servers can be also utilized to meet the service requirements set by the RE-CENT client, e.g. using multisource dynamic adaptive streaming over HTTP. Even though the decision context of the server selection process is enlarged with blockchain-level parameter values, RE-CENT server selection can be implemented using existing protocols optimization tools, e.g. dynamic programming, convex optimization, machine learning and online convex optimization. For example, RE-CENT clients can shortlist RE-CENT servers based on whether they meet the target QoE KPIs (Fig. 3) and select the RE-CENT server requesting the minimum amount of coins. RE-CENT clients that have also acted as validator (or relay) witnesses may choose to prioritize access to FoC servers. Thus, the RE-CENT server selection software shall be implementation-specific, enabling end users to adjust the selection according to their preferences, operational requirements and functional capabilities.

RE-CENT service peers should a-priori specify a *payment timeplan* that will allow progressive charging and delivery of the content delivery service, emulating fair-exchange of assets and payments while enforcing a certain level of trust among the service peers. The payment timeplan should specify both the timing and amount of intermediate payments throughout the entire service lifespan, using a pay-per-video chunk model. An a-priori agreed payment timeplan guarantees that the client will issue the rightful amount of payments for every video

chunk it receives from the server, and vice versa. If the client fails to deliver intermediate payments to the server, the video service delivery shall be interrupted. On the contrary, if the server fails to deliver a video chunk within the agreed time interval, the client can abort the protocol.

Service setup can be formalized by posting the outcome of service negotiation and parameterization on-chain, e.g. using SCs [61]. SCs can be subsequently triggered to resolve potential disputes between the service peers, allowing also other RE-CENT clients to infer on the credibility of RE-CENT servers. Nonetheless, the deployment of a SC on a per session/service peer basis is not scalable, as it will generate an excessive amount of on-chain costs and transactions capacity overheads only for service control, hindering the implementation of the actual service itself. Besides, the RE-CENT peers may choose to abort the service (e.g. YouTube clients typically watch a small part of YouTube videos), or even re-negotiate service parameters on-the-go (e.g. to lower the bitrate and adapt it to the status of the wireless medium),questioning the practical benefits of posting on-chain P2P SLAs. Furthermore, dispute resolution mechanisms that validate the network service status in an on-chain fashion are not easy to implement as they should employ RAT-specific protocols and mechanisms, greatly increasing the complexity of the SC logic and on-chain execution cost.

In our platform, we consider that an a-priori agreement on the key service parameters at the network-level provides sufficient formalization, mitigating unnecessary increase of the transactions throughput towards on-chain service control. However, we also note that the timing and amount of payments agreed in the payment timeplan will play a key role in preserving the credibility and sustainability of the RE-CENT blockchain-backed mobile video service.

2.4 ONLINE SERVICE MANAGEMENT AND CHARGING

During this phase, the client and the server should take all necessary actions to establish, maintain and terminate the mobile video service at the network-level. At the blockchainlevel, the service peers should follow the payment timeplan agreed during the service negotiation and parameterization phase to implement blockchain-backed charging by employing either direct on-chain P2P payments, or off-chain through SC-certified payment relays. The agreed payment timeplan can be tight in case of video service delivery among untrusted peers, or can be implemented by a single transaction when full trust can be assumed. When using payment relays, the relay server shall follow the agreed payment timeplan and act on behalf of the client to issue legitimate payments to the RE-CENT server (section 3.3). Depending on the SC logic, the relay server can update the balance of the server in the public ledger within a prescribed

time period that is acceptable bythe server(relay delay). Ifthe client/serverhave an established payment channel with the relay, the respective amount of payments can be aggregated to further reduce the amount and cost of on-chain payments. Network-level interactions are also necessary to attain service continuity, handle mobility management and deploy QoE-driven service provisioning. Under the RE-CENT mobile data access model, the service management logic is shifted to the client side assuming user-driven network-assisted service provisioning. The client is fully responsible for predicting potential service discontinuity (e.g. either due to unreliable RE-CENT server selection, or due to user mobility), taking mitigation measures whenever necessary and encompassing network measurements provided by the server (Fig. 3).

The employment of user-driven QoE provisioning is also assumed under the RE-CENT mobile data access model. Existing QoE estimation methodologies can be used to this end [12], whereas HTTP adaptive video streaming (HAS) for end-to-end video playback management is another relevant technology that enables end users to adapt video quality based on the availability of network assets at the server side [72], or the channel status [69], [75]. To this end, novel QoE management mechanisms can be utilized to enable sufficient network exposure from MNOs to the RE-CENT clients and video content providers, enabling network-aware video segment selection and caching in the context of HAS. Since service discovery and pairing is implemented using network-level protocols, we consider that an adjustment of the QoE parameters specified in the service negotiation and parameterization phase is handled at the network-level, i.e. by taking necessary corrective measures at the network level, or by allowing the RE-CENT client to abort the service without a blockchain-level penalty. Additional interactions at both the blockchain and network levels might be necessary to implement the logic of the mixing service and mitigate the network/blockchain ID coupling problem (Fig. 1). RE-CENT servers can further create a virtual common pool of network assets that are organized in such a way that allows asset-centric networking (ACN), including service placement, discovery and delivery, moving forward from the host-centric networking model that has dominated IP-based systems over the past decades. ACN shall extend ICN architectures that leverage in-network storage for caching, multiparty communication through replication, and interaction models decoupling senders and receivers [74].

3. BLOCK-CHAIN PROOF CONTENT TRADING BEYOND 5G

3.1 SERVICE ARCHITECTURE, DOMAINS AND ROLES

3.1.1 RE-CENT SERVICE DOMAINS AND FUNCTIONAL SPLIT

We consider a heterogeneous wireless network (HWN) infrastructure that is composed by multiple networking tiers. Network components spanning the network tiers support different RATs and/or have heterogeneous networking

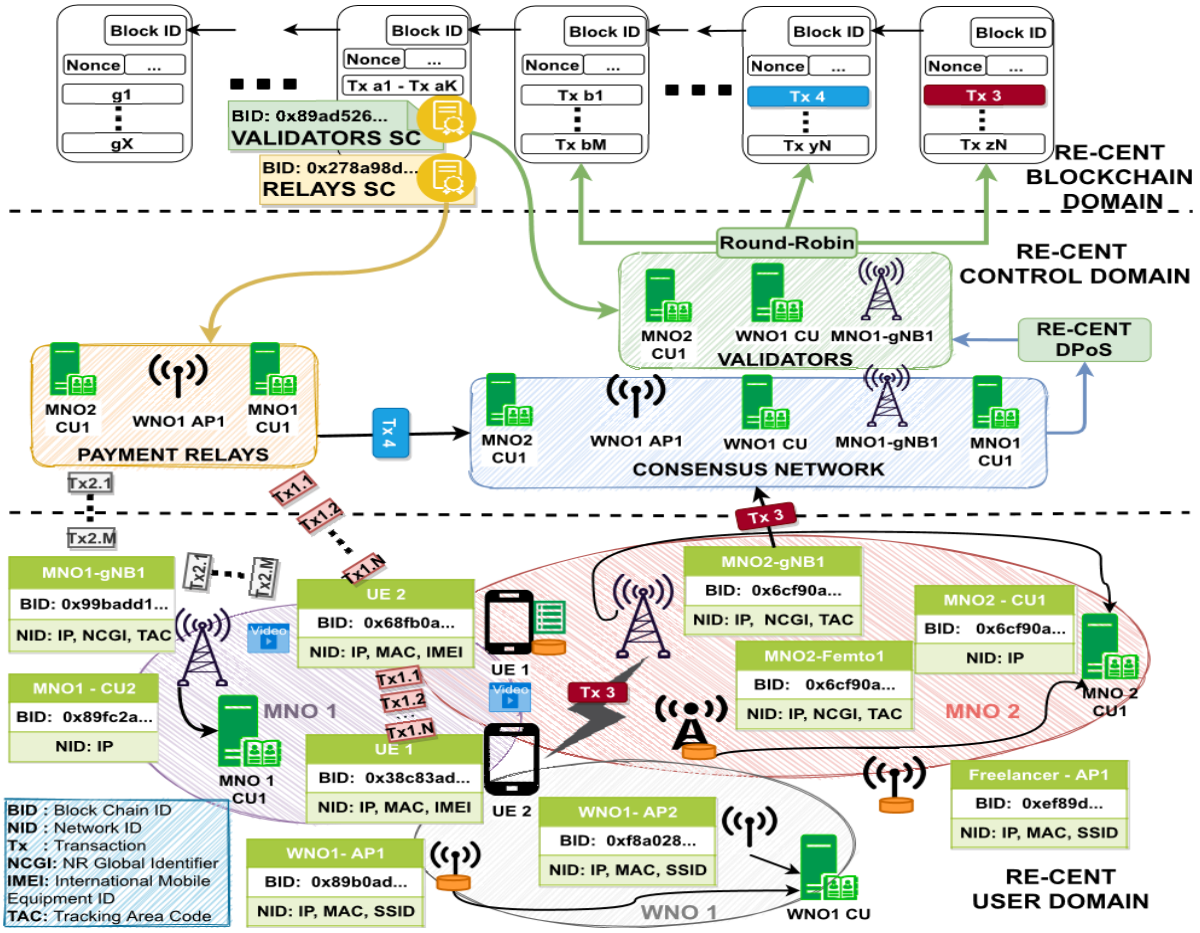

Figure 3: Proposed system architecture

capabilities, e.g. utilize different spectrum bands, host diverse processing and storage capacity. Video consumers are considered to be part of the HWN infrastructure and utilize a number of RAT interfaces to access the different network tiers. On-top of the HWN infrastructure we consider a software architecture that implements the blockchain-enabled RE-CENT content trading platform, which consists of the user, control and blockchain domains (Fig. 3).

The *RE-CENT user domain* is the place where the actual mobile video content service is delivered and the issuing of RE-CENT payments is performed. Network nodes at the RE-CENT user domain run a specialized blockchain-backed software that enables them to dynamically trade available network assets by acting either as network asset clients (consuming network assets and issuing payments), or as network asset servers (sharing their network assets and receiving payments), or as both. Each RE-CENT node is owner of at least one public address (i.e. blockchain ID) that is used to issue/receive payments and interact with the RE-CENT control domain (e.g. for consuming payment relay services). All network-level service phases of Fig. 2 are implemented at the RE-CENT user domain.

Service control of the RE-CENT platform (including charging) is implemented at the *RE-CENT control domain*. At this layer, the RE-CENT nodes undertake different roles towards distributed consensus in line with their functional capabilities and run a specialized software that implements the RE-CENT control protocols. For example, all RE-CENT nodes are considered capable of acting as *witnesses*, staking coins in order to delegate other RE-CENT nodes the role of payment relays, coin mixers and validators for a prescribed time period (Sections 3.2-3.4). However, not all RE-CENT nodes are required to act as full consensus nodes, buffering and propagating new transactions towards block validators, storing and communicating blockchain data to other RE-CENT nodes on-demand, etc, (section 3.2). In the RE-CENT control domain, validators are the only nodes authorized to append blocks in the RE-CENT blockchain, whereas payment relays/coin mixers are the only nodes authorized to aggregate payments/mix coins.

The *RE-CENT blockchain domain* hosts the public ledger structure that records the blockchain-level interactions between RE-CENT nodes. The RE-CENT service control logic is hosted by two specialized SCs: the validators and the relays SCs. The validators SC (VSC) defines all system parameters and functions necessary to implement the RE-CENT DPoS protocol for distributed consensus, also implementing sophisticated reward/penalty mechanisms that enforce honest operation of elected validators. The VSC is designed so as to allow different roles and levels of engagement across the RE-CENT nodes during distributed consensus, further to safeguarding system robustness in the long-term. The relays SC (RSC) defines all parameters and mechanisms necessary to implement credible payment relay and mixing services.

All RE-CENT protocols are designed to enable offline delivery of the mobile video service by local servers while enforcing honest operation of the key RE-CENT actors (validators, payment relays, mixing servers) in a fully decentralized fashion, through the deployment of sophisticated incentive engineering mechanisms that are implemented using on-chain SCs.

This design approach is not only relevant to the problem under scope, i.e. the mobile video delivery service requires physical proximity and offline service consumption, but it also enables minimum interactions with the public ledger; thus, minimum transactions capacity requirements and on-chain costs, ideally only for i) the establishment of payment channels, ii) on-chain dispute resolution between the RE-CENT nodes and actors, and iii) for SC-driven rewards/penalties to dishonest RE-CENT actors. Common basis of all incentive engineering mechanisms is the requirement to timelock to the respective RE-CENT SCs an amount of funds that is proportional to the risks following from a potential dishonest operation of the key RE-CENT actors, an approach that is compatible with the overall PoS-based design of the RE-CENT blockchain platform.

The RE-CENT platform incorporates three innovative blockchain-based protocols for Beyond 5G distributed consensus (section 3.2), payment relay (section 3.3) and coin mixing (section 3.4), all of which are designed to support different roles and levels of engagement in distributed consensus while preserving anonymity and requiring only a low transactions throughput onto the RE-CENT blockchain.

3.1.2 RE-CENT Roles

RE-CENT nodes undertake different roles in light of their functional capacity and desired level of engagement with the RE-CENT service domains. In the sequel, we overview these roles and briefly discuss relevant implementation details.

Validators. High-end RE-CENT nodes that are authorized to seal blocks in a round-robin fashion for a given time epoch (measured in blocks). The number of validators is a system parameter that can be amended in the long-term as soon as the amendment is supported by the majority of validators for a certain number of consecutive epochs.Validators are elected through the RE-CENT DPoS consensus protocol that enables any RE-CENT node to run as candidate validator for a target epoch e, participating in an action-based scheme during the *validators' election epoch* $e-1$. During the election epoch, candidate validators should i) lock a minimum guarantee fund to the VSC, which is used to enforce honest operation of validators through VSC-driven penalties, ii) lock a reward fund that will be shared across RE-CENT nodes that vote (stake) in favor of the candidate (if it gets elected), and iii) a transaction fee that is paid to the validator on a per sealed block basis. Validators with the highest stakes (including their own guarantee fund) get shortlisted and elected on the basis of available validator seats per epoch. If elected, validators receive transactions fees and preserve their role for a given epoch, assuming that they act honestly. Penalty mechanisms and validator

replacement methods are provisioned to safeguard system robustness against inadvertent validator behaviors (section 3.2).

Validator witnesses. RE-CENT nodes that stake in favor of candidate validators for a given epoch. V-witnesses are incentivized to actively participate in the DPoS consensus to: i) share the reward fund offered by the candidate validator (in proportion to their v-witness stakes), ii) receive free service from *FoC servers* supporting the candidate validator, and iii) receive priority in the processing of their transactions. RE-CENT nodes shall employ their own logic towards the selection of a candidate validator and the amount of funds to be staked per candidate validator.

Free-of-charge servers for validators. Special case of v-witnesses that instead of staking an arbitrary amount of coins in favor of a candidate validator, they promise to offer FoC service to v-witnesses of a tagged validator (if elected). FoC servers also timelock funds in favor of a candidate validator in proportion the FoC service they promise to offer. Locked funds are used to incur penalties to FoC servers that fail (or refuse) to deliver the promised service to v-witnesses. FoC servers leverage their network-level capabilities to attract more v-witnesses towards blockchain-level consensus.

Payment relays. High-end RE-CENT nodes that are authorized to act as payment intermediaries enabling instant off-chain payments for a given time epoch. The number of payment relays per epoch varies in line with i) the (estimated) transactions capacity of the blockchain and ii) the type of relay licenses requested by other candidates. Candidate relays lock to the RSC a minimum guarantee fund that is calculated based on i) the number of clients that the relay requests to support, ii) the total amount of coins that can be attached to the relay (using inbound payment channels), and iii) the transactions throughput that the relay promises to spur into the RE-CENT blockchain. Relay licensing follows a similar approach with the validators' election process, enabling RE-CENT nodes to stake funds, or FoC service, in favor a candidate relay. Authorized relays establish payment channels with RE-CENT clients and servers only through the RSC. Payment relays also convert off-chain payments to on-chain balance updates only through the RSC.

Honest relays receive transactions fees on a per off-chain transaction that they process, while they can withdraw their guarantee funds with the expiration of their license. Dishonest relays receive penalties (on their guarantee fund) according to a RSC-driven mechanism that enables disregarded relay clients to trigger on-chain dispute resolution. This process requires i) disregarded clients to submit signed promises (transactions) of the relay and ii) the (reported) relay to submit proofs of its lawful operation. This is possible mainly due to the employment of

a fixed time delay window (measured in blocks) due by which the relay promises to submit new offchain transactions. This parameter is termed as relay delay in the sequel, while it is specified by the payment relay during the licensing epoch and is included in the signed off-chain promises issued by the payment relay. Triggering the RSC for on-chain dispute resolution comes with an on-chain transaction cost that is initially paid by the disregarded relay client, but is fully reimbursed (along with other costs relevant to the dispute) by the guarantee fund of the dishonest relay. The license of relays can be revoked under certain conditions and relay replacement mechanisms are also provisioned. The payment relay service (including the RSC logic) is presented in section 3.3.

Relay witnesses. RE-CENT nodes that stake in favor of candidate payment relays for a given epoch. The selection of a candidate relay and the amount of coins staked in favor of candidate relays is left up to the implementation of the RE-CENT node. R-witnesses are incentivized to actively participate in the relay election to i) share a reward fund offered by candidate relays (if elected) and ii) receive FoC service by servers supporting the tagged payment relay.

Free-of-charge servers for relays. Special case of r-witnesses that support a tagged candidate payment relay for a given epoch. Similar operation to FoC servers for validators. **Payment relay clients**. They are RE-CENT nodes that consume a payment relay service to perform instant off-chain payments at a lower transactions cost. Relay clients select one (or more) payment relays with an active license and establish inbound payment channels on the RSC. To this end, they timelock an arbitrary amount of funds to the RSC and indicate the relay that is authorized to handle its balance.

Mixing servers. Payment relays that also act as mixing servers on the basis of the payment relay license and the RSC mechanisms attached to it, e.g. payment channels established with the RE-CENT clients and servers. The RE-CENT mixing servers implement hybrid mixing, an approach that enables centralized payment relay servers to deploy the mixing service offline; however, enforcing their honest operation in a fully decentralized fashion using the RSC logic for on-chain dispute resolution with disregarded relay clients.

The RE-CENT mixing service extends RSA blinding and puzzle solution/solving protocols of Tumblebit [29] to enable instant fair-exchange of mobile video content and on-chain funds in an anonymous fashion. The RE-CENT mixing protocol is detailed in section 3.4.

Mixing clients. Payment relay clients that additionally consume RE-CENT mixing services provided by the payment relay. Mixing clients pay an additional fee for the use of mixing services, aiming to employ both instant and anonymous off-chain payments.

Full consensus nodes. High-end RE-CENT nodes that are responsible for propagating new transactions through the consensus network, keeping track of blocks issued by validator nodes, storing the full blockchain data and provide them upon request to other RE-CENT nodes. Full consensus nodes are not necessarily part of the network infrastructure, which consumes/delivery mobile video content.

3.1.3 SERVICE FLOW AND CHARGING EXAMPLE

In Fig. 3, the RE-CENT user domain is composed by individual service hotspots (e.g. Freelancer AP1), Wi-Fi Network Operators (WNO) (e.g. WNO1 with the Wi-Fi access points AP1 and AP2 attached to the WNO core unit WNO1 CU), 5G MNOs (e.g. MNO1 with gNB1 that is attached to the MNO1 - core unit 1) and user equipments (UEs) (e.g. UE1 and UE2), which support deviceto-device (D2D) communications. Some RE-CENT nodes utilize local storage resources to employ content caching and instantly deliver mobile video content on demand. The RE-CENT control domain is composed by a subset of the user-domain RE-CENT nodes, which have been additionally engaged in the roles of payment relays (e.g. MNO1-CU1), validators (e.g MNO-gNB1), and full consensus nodes (e.g. WNO1-CU). The RE-CENT blockchain is maintained by full consensus nodes and updated only by elected validators. The VSC and RSC are deployed in the early blocks of the RE-CENT blockchain, enforcing honest operation of validators and payment relays at the RE-CENT control domain.

Moving again to the RE-CENT user domain, UE2 consumes popular video content from two RE-CENT servers: UE1, which uses the 5G base station MNO2-gNB1 to relay the requested content, and MNO1-gNB1, which utilizes its backhaul connectivity to reach the content through the Internet. UE2 is assumed to utilize the payment relay services of MNO1-CU1, enabling instant micro-payments Tx1.1, Tx1.2, ..., Tx1.N with UE1 and Tx2.1, Tx2.2, ..., Tx2.M with MNO1-gNB1. On the contrary, UE1 is assumed to issue a direct on-chain payment Tx3 to MNO2-gNB1 and propagate it to the consensus network directly. However, micro-payments Tx1.1-Tx1.N and Tx2.1-Tx2.M are performed off-chain through the payment relay MNO1-CU1, which subsequently aggregates the respective payments into a single on-chain transaction Tx4. Tx4 indicates as recipient the public address of the RSC and, when processed by validators and posted on-chain, it triggers the RSC logic to update the balance of UE2, UE1 and MNO1-gNB1 on-chain accordingly.

3.1.4 RESOURCE UTILIZATION AND ENERGY-EFFICIENCY ASPECTS

Energy-efficiency of a blockchain-backed system typically comes down to the requirements of the distributed consensus protocol and the size of the consensus network. Popular PoW-based platforms like Bitcoin, consume vast amounts of processing and energy resources due to the participation of myriads of consensus nodes into the puzzle solution process (mining). Through this process, consensus nodes gain an opportunity to seal new blocks and receive block rewards attached to them. Blockchain-backed mobile data access should be sustainable and energy-efficient, enabling also network nodes to adapt the level of their engagement in distributed consensus to their functional capabilities.

The RE-CENT platform is designed to meet the energyefficiency requirements set for 5G and Beyond mobile data networks by employing a DPoS consensus protocol where a very small set of validators seal blocks in a deterministic (round-robin) fashion. Although validators are elected on an epoch-by-epoch basis by RE-CENT nodes, which are provided with clear incentives to do so (section 3.1.2), our system design does not oblige RE-CENT nodes to act as v-witnesses. Even if RE-CENT nodes choose to participate into the DPoS process, they will be only required to sign their stakes (votes) by computing a single hash function and broadcasting this short message to the consensus network. Adding to this, RE-CENT nodes are not required to be actively engaged with the maintenance of the RE-CENT blockchain, by acting as consensus nodes that propagate transactions and keep track of the current blockchain status. Instead, RE-CENT nodes can assess the RE-CENT blockchain status by querying consensus nodes using special calls, e.g. JSON queries to Open Nodes in ETH. Hence, at minimum, a RE-CENT node is only required to i) be holder of a RE-CENT public address and operate a simple wallet application to issue/receive payments, ii) be capable of computing/verifying only a few cryptographic signatures per second (e.g. smartphones can compute thousands of hash signatures per second) and iii) query consensus nodes to assess the RE-CENT blockchain status. At maximum, a RE-CENT node can actively participate in the RE-CENT consensus process (e.g. acting as validator, full consensus node, witness, FoC server), or by being a payment relay that aggregates (or mixes) off-chain payments.

It readily follows that the design of the RE-CENT platform is fully aligned with the heterogeneous nature of a 5G and Beyond mobile data access, enabling network nodes to match their level of engagement with the system in view of their prospects, operational requirements and functional capabilities. The employment of DPoS consensus mitigates unnecessary consumption of computation and energy resources, limiting the number of block sealers to the minimum and generating blocks in a deterministic fashion thus, enabling energy-efficient and sustainable maintenance of the RE-CENT blockchain in the long-term. Besides,

the employment of off-chain payments and their aggregation through the RE-CENT payment relay service substantially reduces the number of transactions (and messages) propagated across the consensus network, keeping the operational requirements of the platform to the minimum (i.e. size of the consensus network, computation and energy consumption).

3.1.5 IMPLEMENTATION ASPECTS

The proposed blockchain-backed payment service aims to revolutionize service charging in 5G and Beyond networks. Different implementations of the proposed payment service can be delivered depending on the architectural and functional capabilities of the mobile data network under scope. In the sequel, we present some ideas on potential implementation under the service-oriented architecture (SOA) of the Release 16 3GPP 5G System (5GS) [76].

The payment relay service shall runs on top-of the standard network protocol stack (PHY, MAC, IP, TCP/UDP) and specifically at the application (APP) layer. RAN nodes are not required to implement the RE-CENT server software and hold a unique public address; instead, the 5GS core can instantiate a single RE-CENT server at the 5GS core network and attach to it many RAN nodes. Alternatively, a separate RE-CENT server instance can be instantiated per network slice. The RE-CENT payment relay server can be implemented as a traditional HTTP server that allows RE-CENT clients and servers to consume its services using RESTful APIs. This approach is in line with the current design of the 5GS core. The RE-CENT client software can also be implemented as a simple HTTP client that is bind to a wallet software, enabling blockchain-level interactions and APP-layer session management (including network selection).

In the 5GS, the RE-CENT server logic can be integrated as part of the network services, taking into consideration the functionality available by the existing 5G core network functions (NFs). The RE-CENT client shall attach to the Access and Mobility Function (AMF) through the RAN nodes. The AMF shall be responsible for negotiating the video delivery - payment timeplan with the RE-CENT client, granting it access to the 5GS and implementing connection management for the entire service lifetime. The AMF shall also determine the Session Management Function (SMF) that is best suited to handle the RE-CENT client session (user plane traffic), while the SMF shall instantiate and subscribe to a Charging Function (CHF) service that shall implement the RE-CENT server software. The CHF service shall be responsible for handling RE-CENT client payments (potentially via the payment relay service) and triggering access authorization/session termination to the SMF/AMF accordingly. Context information on the RE-CENT service can be stored in the form of ‘‘unstructured’’ data in the 5GS using the Unstructured Data Storage Function (UDSF).

At this point, we clarify that RAN and core network nodes that implement the RE-CENT server software are not necessarily full consensus nodes (section 3.1.2). Instead, they are considered capable of assessing the RE-CENT blockchain status through full consensus nodes and issue/receive payments as described in section 3.1.4. In [41], we provide an open-source. NET implementation of the RE-CENT client, server and payment relay server software, which enables interaction across RE-CENT nodes using RESTful interfaces. This software also enables assessment of the RE-CENT blockchain through JSON queries to full consensus nodes. The necessary VSC and RSC logic has been implemented in Solidity assuming the Parity Aura consensus protocol and the ETH platform. Since the RE-CENT service architecture can be instantiated through the deployment of two specialized SCs on-top of any SC-enabled blockchain framework, RE-CENT key and ID management will follow the format of the blockchain framework used for the implementation, e.g. the ETH public address is derived by the last 20 bytes of the SHA3 hash of the user's public key.

3.2 DISTRIBUTED CONSENSUS BEYOND 5G

In this section, we describe all aspects of the RE-CENT DPoS mechanism and the validator smart contract (VSC) logic used to implement it. In section 3.2.1, we specify fixed and adjustable parameters affecting the operation of the VSC, also detailing the mechanisms used to amend adjustable VSC parameters. In section 3.2.2, we describe the DPoS consensus mechanism by detailing how candidate validators, v-witnesses and FoC servers elect validators for a target epoch. In section 3.2.3, we describe the scenario where validators and FoC servers act honestly, whereas in section 3.2.4 we specify the VSC-enforced mechanisms used to award penalties and replace RE-CENT validators that fail to seal blocks on time, or submit invalid blocks.

3.2.1 VSC PARAMETERS AND AMENDMENT MECHANISM

The VSC incorporates fixed and adjustable parameters. Fixed parameters are used to safeguard system robustness against inadvertent (or malicious) behaviors of the validators and are hard-coded to the VSC. Adjustable parameters allow flexible operation of the DPoS consensus protocol in view of the current state of the RE-CENT blockchain and a specific amendment procedure is followed to update their values. Table 1 summarizes the key VSC parameters.

The RE-CENT DPoS mechanism runs on an epoch-byepoch basis, where each *validation epoch* lasts for exactly B_V blocks. A validation epoch is a time period within which a given set of validators is authorized to seal blocks. To get elected for a target epoch e, the validators

participate in an auction-based scheme that starts from the first block of period e−1 and concludes exactly T_V blocks before the beginning of the target epoch e (3.22). Parameter R_V defines the number of newly minted coins generated per block in the first epoch. This value is adapted according to table D_V , which specifies a disinflation rate that is applied on R_V depending on the epoch number. Sealed blocks specifying a different amount of new coins (validator rewards) from this value are considered invalid. This mechanism is used to reduce the number of new coins generated per block in the long-term.

Table 1: VSC parameters for a tagged epoch e.

Parameter	Notation	Type
Epoch duration (in blocks)	B_V	Fixed
Election window deadline (in blocks)	T_V	Fixed
Baseline emission rate (coins per block)	R_V	Fixed
Table of disinflation rates (percentages)	D_V	Fixed
Number of consecutive epochs for amendment	C_V	Fixed
Transaction fee paid for direct payments	c_{min}	Fixed
Current No. of validators	$V[e]$	Adjust.
Max No. of validators	V_{max}	Fixed
Min No. of validators	V_{min}	Fixed
Current baseline penalty for offline validators	$P_{Vo}[e]$	Adjust.
Min baseline penalty for offline validators	P_{Vo}^{min}	Fixed
Current baseline penalty for malicious validators	$P_{Vm}[e]$	Adjust.
Min baseline penalty for malicious validators	P_{Vm}^{min}	Fixed
Minimum stake for validators	$M_V[e]$	Adjust.
Minimum stake for v-witnesses	$W_V[e]$	Adjust.
Free-of-charge service tariff per MB	$f_V[e]$	Adjust.

For example, D_V specifies a disinflation rate equal to 1 for epochs 1 to 999, a rate equal to 0.5 for epochs 1000 to 10.000, and a rate equal to 0 for epochs 10.001 and beyond. R_V and D_V are fixed to create a predictable supply of new coins in the system and discourage validators to vote in favor of a higher block reward. The minimum transaction fee c_{min} is used for direct on-chain payments and is considered to be fixed for the same reason. The value of c_{min} should be tuned so as to enforce the use of payment relay services, allowing the RE-CENT blockchain to scale with the transactions generated by the myriads of RE-CENT service peers. C_V is a fixed parameter defining the number of consecutive blocks that are necessary to amend an adjustable VSC parameter (amendment procedure described below).

The number of validators $V[e]$ plays a key role on the performance of the RE-CENT blockchain. A low number of validators increases the risk for block sealing failures due to inadvertent and malicious behaviors of the validators but enables the system to attain a higher transactions capacity. A large number of validators can safeguard system robustness against failures and dishonest behaviors by the validators, but also reduces the incentives offered to validators towards block sealing(i.e. rewards decreaseproportionally). The VSC enables the validators to amend the value of $V[e]$, keeping it within specific VSC-defined limits (i.e. $[V_{min}, V_{max}]$).

The VSC logic distinguishes between the two scenarios where a validator i) fails to deliver a sealed block on time, e.g. service outage, or ii) seals an invalid block. A lower penalty $P_{Vo}[e]$ is employed in the first scenario, to discourage validators that lack the required functional capacity to perform block sealing, whereas a higher penalty $P_{Vm}[e]$ is employed for the second scenario, to quickly revoke the license of dishonest validators and exclude them from block sealing (section 3.2.4). Both parameters can be amended by the validators but they should be above the prescribed VSC-defined thresholds $P^{min}{}_{Vo}$ and $P^{min}{}_{Vm}$, respectively.

Active participation to the DPoS mechanism, either by setting candidacy as a validator, or by acting as a v-witness, requires on-chain locking of funds to the VSC. The VSC enforces a minimum stake for both candidate validators and v-witnesses, denoted by $M_V[e]$ and $W_V[e]$, respectively, aiming to discourage nodes to act dishonestly. Parameter $f_V[e]$ specifies amount of coins that a FoC server should lock onto the VSC for a given FoC service promise: by locking X coins, the FoC server promises $X/f_V[e]$ MBs per v-witness if the respective candidate validator gets elected.

VSC parameter amendment procedure. Let us now focus on the VSC logic enabling amendment of adjustable VSC parameters (Table 1) by active validators. Note that validators enjoy wide acceptance across the RE-CENT nodes, at least for the period of time that they are elected. Besides, if the RE-CENT nodes are not satisfied by the actions of an elected validator (including an amendment), they can revoke their support by not staking in their favor of them in subsequent epochs. In view of that, active validators are enabled to propose an incremental increase, or decrease, to any adjustable parameter of the VSC. If the respective change is supported by the majority of validators at the end of the epoch, i.e. at least 50%+1 validator, then the VSC increases an amendment counter per adjustable parameter. Amendment counters of parameters that have not been supported for amendment within a given epoch is set to zero. This process continues, until the counter of some VSC parameter reaches to the number of consecutive epochs C_V. Such an event triggers the VSC to amend the respective parameter. The aforementioned logic is equivalent to the amendment of a given adjustable parameter if and only if (iff) this amendment is supported by the majority of validators for at least C_V consecutive epochs.

3.2.2 DPoS FOR VALIDATORS ELECTION

The validators election process for epoch e starts with epoch $e-1$ and concludes T_V blocks before epoch e. In the sequel, we term epoch $e-1$ as the *election epoch* and epoch e as the *target epoch*. A RE-CENT node that wishes to set candidacy as validator for a target epoch e, posts an on-chain transaction having as recipient's address the VSC public address and as

payload: i) an (arbitrary) amount of coins ($> M_V[e]$) that will be used as *guarantee fund* if the validator gets elected and ii) an (arbitrary) amount of coins that will be shared across v-witnesses supporting its candidacy, termed as *v-witness reward*, and iii) *transactions fee* that the validator claims for each transaction it includes per block during epoch e.

RE-CENT nodes that are interested in acting as v-witnesses, or FoC servers shall query full consensus nodes to acquire the list of candidate validators from the VSC. This list not only provides the public addresses of candidate validators but also specifies, among other, the current stakes placed in favor of a given candidate validator (sum of guarantee funds, v-witness stakes and FoC server stakes), v-witnesses supporting the validator (including FoC servers), the promised v-witness reward and the claimed transactions fee. On the one hand, by retrieving the list of FoC servers per candidate, RE-CENT nodes can estimate network-level benefits following from the support of a tagged candidate validator, e.g. to verify if FoC service is promised by a RE-CENT server within proximity. On the other hand, by retrieving the list of v-witnesses, RE-CENT nodes can infer on the reputation of the candidate validator, or evaluate the positive impact of a FoC service promise towards the election of a tagged candidate. Using their own criteria, RE-CENT nodes can support a candidate validator as v-witnesses and FoC servers by timelocking in the VSC an arbitrary amount of coins ($>W_V[e]$) and monitoring the election outcome. Candidate validator and witness votes that do not comply with the minimum stakes $> M_V[e]$ and $W_V[e]$ are ignored by the VSC logic. RE-CENT nodes can form v-witness coalitions by creating a common public address, staking funds using a single on-chain call to the VSC. To this end, they can utilize the payment relay service (that aggregates payments towards a given public address, the VSC in this case), or use ring signatures, hiding their public address while receiving FoC service if the candidate validator gets elected.

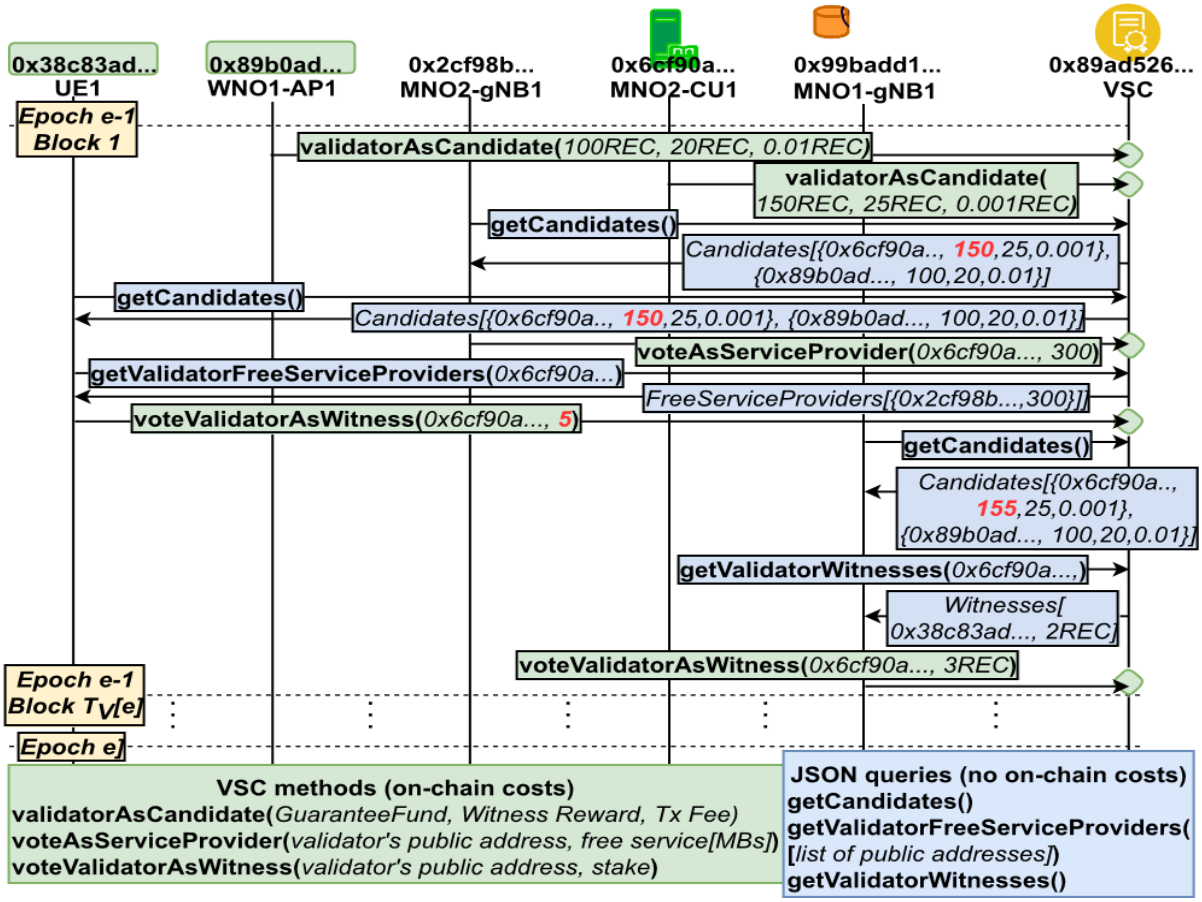

Figure 4: Validators' election mechanism

Every on-chain transaction posted to the VSC changes its state, making it update (or create new) VSC registries and recalculate the shortlist of elected validators (ordering them based on the total stakes placed in their favor). All stakes placed in favor of candidate validator (guarantee funds, witness rewards, v-witness stakes and FoC stakes) are timelocked to the VSC for both the election and the target epochs, independent of the outcome of the election process. This enables the VSC to fast replace active validators during the target epoch (section 3.2.4). The election epoch is concluded $T_V[e]$ blocks before the start of the target epoch, to minimize the probability of having delayed VSC calls related to the validators' election. Nonetheless, validators with higher ranking can always challenge the VSC and replace an active validator having a lower total stake in the VSC (section 3.2.4).

Run-time example. Fig. 4 provides a simple example on how the DPoS protocol is implemented. In Fig. 4, we consider two candidate validators for epoch e, i.e. WNO-AP1 and MNO2-CU1, which place their stakes onto the VSC at the beginning of epoch $e-1$. To this end, they both post an on-chain transaction that triggers the VSC logic to run the method *validatorAsCandidate* with the right input (inserted in the transaction payload). This call not only enables fund locking onto the VSC but also triggers the VSC to shortlist candidate validators based on their total stakes. Using the VSC method *voteAsServiceProvider*, MNO2-gNB1

supports the candidacy of MNO2-CU1 by retrieving the list of candidate validators and promising FoC service proportional to 3 coins. To estimate potential network-level benefits of supporting MNO2-CU1, UE1 retrieves the list of FoC servers attached to the public address of MNO2-CU1 (*freeServiceProviders* method) and stakes in favor of this candidate by locking 5 coins onto the VSC (*voteValidatorAsWitness* method). Accordingly, the VSC recalculates the shortlist of elected validators. After this step, MNO1-gNB1 retrieves the (updated) list of candidate validators and v-witnesses attached to candidate MNO2-CU1, to estimate its reputation (using off-chain methods). Accordingly, MNO1-gNB1 decides to vote the candidate validator MNO2-CU1 by timelocking 3 coins onto the VSC using an on-chain call to the VSC method *voteValidatorAsWitness*.

3.2.3 PROTOCOL OVERVIEW ASSUMING HONEST OPERATION

Let us now focus on the actions performed during an epoch where the validators have been elected (i.e. the target epoch e). During epoch e, RE-CENT nodes identify the list of shortlisted candidate validators by the VSC. Elected validators seal blocks taking turns in a round-robin fashion, receiving legitimate transactions fees (as declared in the VSC) and newly minted coins (in line with the VSC parameters R_V and D_V). Elected validators prioritize transactions of their v-witnesses. At any time during epoch e, v-witnesses can withdraw from the VSC their legitimate v-witness reward share from elected validators. Their request should be proportional to the funds that they have staked in favor of the corresponding validator(s), divided by the total amount of stakes placed in favor of the tagged validator(s). If so, the VSC shall update its state and release the respective amount of funds the balance of the v-witness. If not, the respective VSC call will be rejected by the VSC.

V-witnesses of an elected validator can now receive free service from FoC servers that supported the respective candidate. For every chunk of FoC service they receive, they sign (using their private key) a special type of transaction that is issued by the FoC server in order for the FoC service to continue. This procedure enables FoC servers to prove their honest operation in case of on-chain disputes triggered by the corresponding v-witness (see section 3.2.4). Assuming that every party acts honestly, the epoch concludes without triggering the VSC penalty and validator replacement mechanisms. After the end of epoch e, validators, v-witnesses and FoC servers are enabled to withdraw the (remaining) stakes placed during the respective election epoch $e-1$.

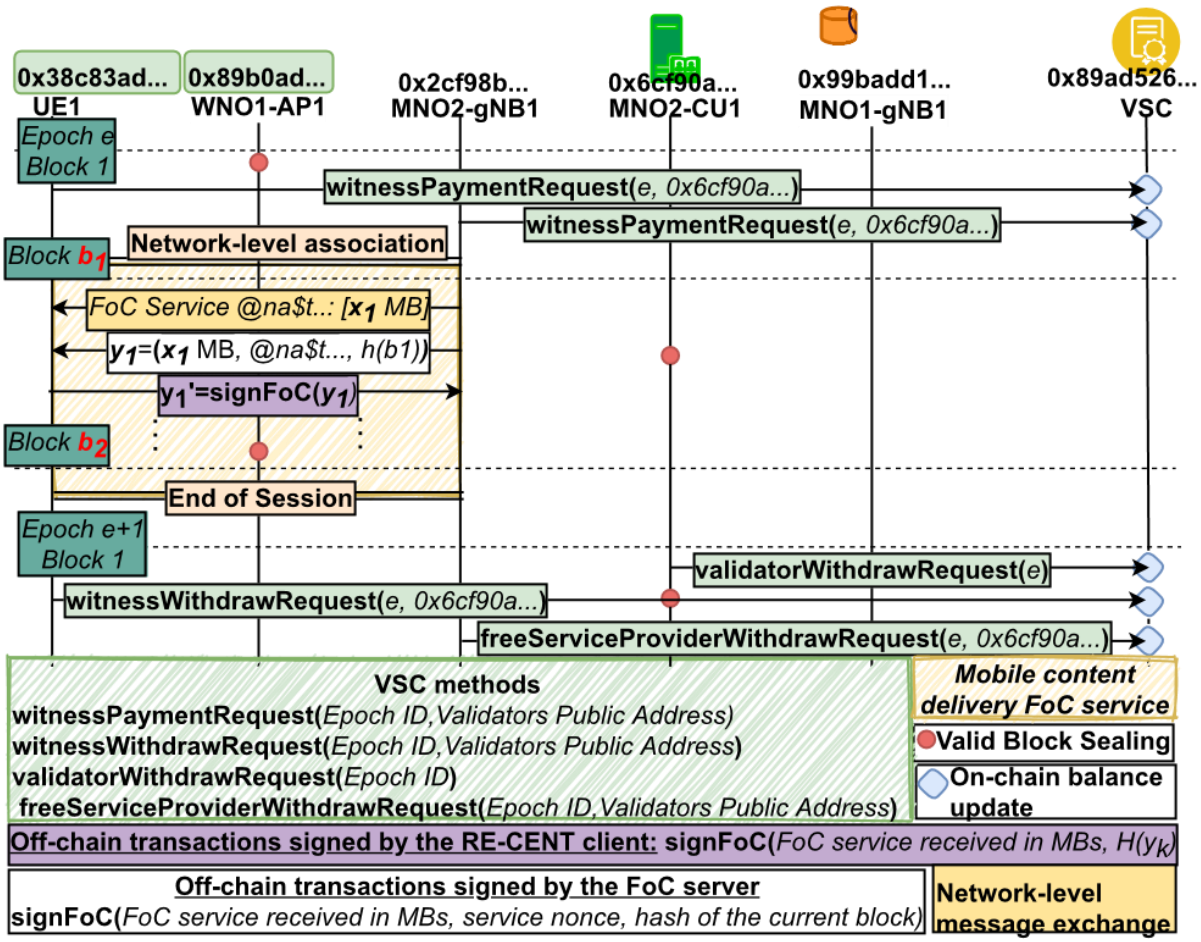

Figure 5: Protocol run time assuming honest operation

Run-time example. Fig. 5 provides an illustrative instance of the aforementioned process, assuming that WNO1-AP1 and MNO2-CU1 have been both elected as validators, MNO2-gNB1 is a FoC server of MNO2-CU1 and UE1 is a v-witness of MNO2-CU1. FoC service proofs are normal transactions indicating in their payload zero coin transfers.

3.2.4 PENALTY AND VALIDATOR REPLACEMENT MECHANISMS

We define two critical events upon block sealing: i)the *missed block* event where a validator fails to seal a block within the time interval defining its turn, and ii) the *invalid block* event where the validator seals an invalid block. The missed block event is very common in distributed systems and can take place for a variety of reasons, e.g. due to a temporary outage of the validator's Internet connection. However, the invalid block event is critical for system robustness and if a persistent behavior of this type is encountered, the malicious validator should be revoked with the authorization to seal further blocks. An invalid block may include a higher block reward for the validator, or an invalid coin transfer resulting in an inappropriate allocation of funds across in the RE-CENT nodes (e.g. double-spending, forged signatures). Since the system is

decentralized, a validation error should be reported to the VSC by the majority of elected validators that govern block sealing in the blockchain-backed system.

To this end, elected validators call special on-chain methods to enable the VSC verify a missed/invalid block event. If the majority (50% + 1) of validators reports as problematic the same validator and the same block, the VSC proceeds with the penalty mechanism attached to the validation error event. If a missed block event has been verified, then the VSC gives a fixed penalty $P_{Vo}[e]$ to the validator that has missed its turn during block sealing. The penalty will be first applied on the guarantee fund of the validator and if the guarantee fund is burn out, the penalty will be applied to the stakes placed in favor of the validator by v-witnesses. If an invalid block event has been verified, the VSC shall invoke to the dishonest validator a penalty that i) increases exponentially with the number of invalid blocks verified for this validator within the target epoch, ii) is proportional to the VSC parameter $P_{Vm}[e]$ and iii) is reversely proportional to the current number of validators $V[e]$ in the system. Assuming that m invalid blocks have been verified by the VSC for epoch e], the penalty scales as follows:

$$Penalty[m, e] = 2^{m-1} \cdot \frac{P_{Vm}[e]}{V[e]} \quad (2)$$

Apart from reducing the guarantee fund and the stakes of dishonest validators, the VSC also enables candidates of the reserve list for the target epoch e, to trigger the replacement of elected validators. Even though elected validators are initially ranked higher, dishonest operation reduces the (active) amount of stakes placed in their favor in case of dishonest operation, potentially making them lower to that of candidate validators that haven't been elected in the first place. Since all stakes placed in favor of candidate validators during the election epoch $e - 1$ are timelocked for both the election and the target epochs, we allow candidate validators belonging to the reserve list to challenge the status quo of elected validators by triggering the VSC to recalculate the shortlist of elected validators. This design approach is fully aligned with the decentralized nature of blockchain systems as it guarantees that a newly elected validator will be aware of its new role, avoiding unnecessary penalties and missed blocks in the system. In the medium-term, validators that belong to the reserve list and have a higher ranking in terms of total amount of remaining stakes locked in the VSC, will challenge the VSC and exercise their right to seal blocks. Since missed/invalid blocks will be ignored by other validators, the transactions capacity of the system will drop by $1/V[e]$ for the specific seal blocking round. Nonetheless, such an event will have a short term impact on the transactions capacity of the blockchain, provided that dishonest validators will be fast replaced

by other validators belonging to the reserve list, due to the exponentially increasing penalties - Eq. (2).

3.2.4.1 RUN-TIME EXAMPLE

In Fig. 6 we provide a simple yet illustrative example of how the penalty and validator replacement mechanisms are implemented under the RE-CENT DPoS consensus protocol. Let us assume that MNO2-CU1, WNO1-AP1 and UE1 run as candidate validators during epoch $e-1$, but only MNO2-CU1 and WNO1-AP1 are elected for the target epoch e. Let us also consider the following critical events to take place:

i) MNO2-CU1 goes offline and misses block sealing for block 81, ii) WNO-AP1 issues an invalid block that grants a higher block reward to itself for block 84, and iii) WNO-AP1 issues an invalid block that tries to reverse its malicious action in block 88. The missed block event is reported by WNO1-AP1 using the VSC method *reportBenign* with arguments i) the public address of the misbehaved validator and ii) the block where the event has took place. Accordingly, the VSC applies a fixed penalty to MNO2-CU1 (assuming 50%+1 validator reports).

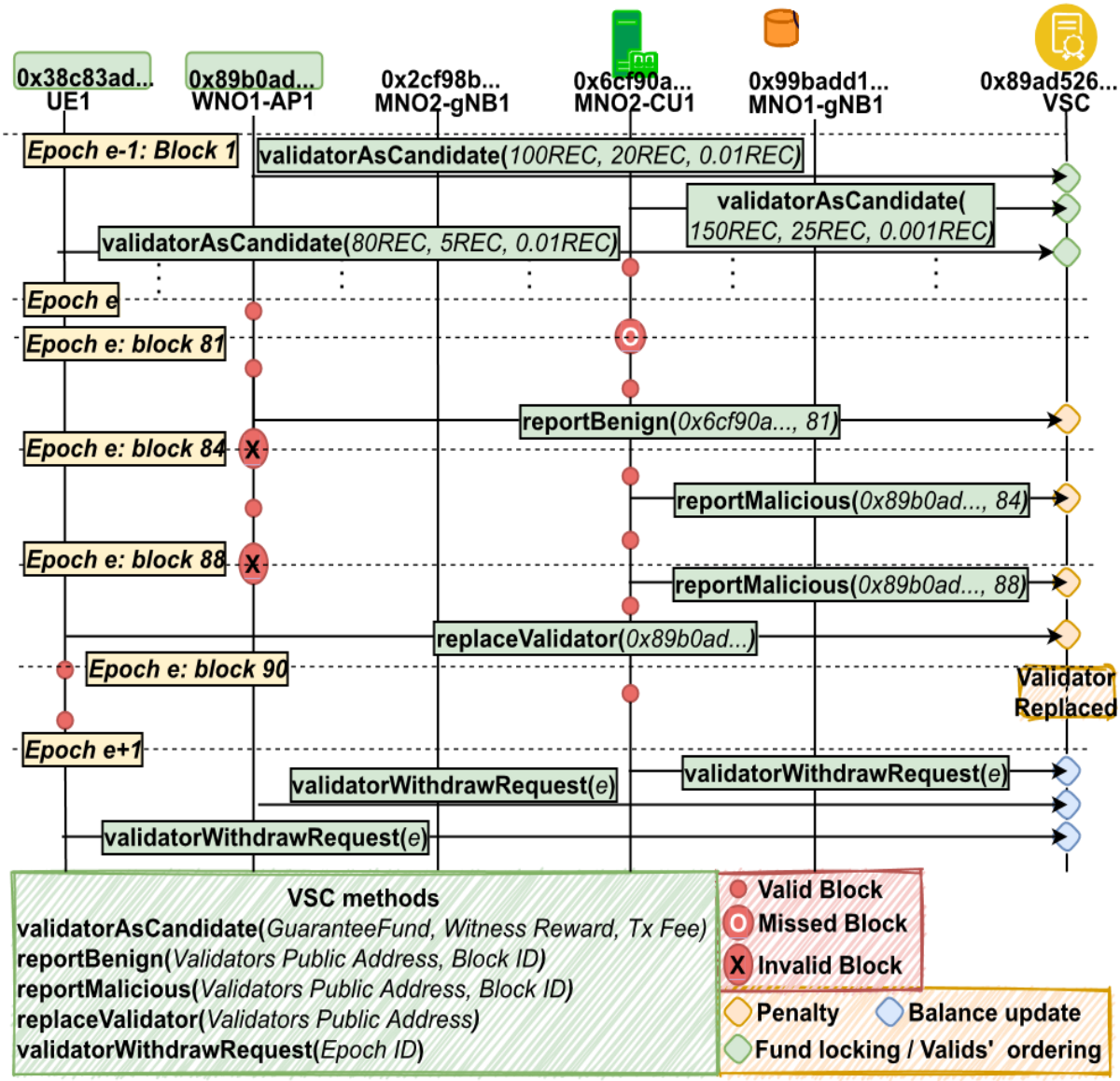

Figure 6: Penalty and validator replacement mechanisms

In parallel, the two events of invalid block sealing by WNO1-AP1 are reported by MNO2-CU1, using the VSC method *reportMalicious* with similar arguments. This report results to the burn out of a notable amount of funds for the validator WNO1-AP1, an event that enables UE1 to challenge the validators' shortlist by triggering the VSC method *replaceValidator*. Assuming that UE1 indicates the active validator with a lower amount of remaining stakes locked in the VSC (as compared to its own stakes), the VSC will recalculate the new shortlist of candidate validators and infer on the fact that UE1 is now a higher-ranking validator as compared to WNO1-AP1. Accordingly, the VSC will emit the validator replacement event to the consensus network and the validators shall ignore all blocks issued by WNO1-AP1, following also blocks sealed by UE1.

3.2.4.2 FOC SERVER PENALTIES

We primarily target to protect FoC servers from dishonest v-witnesses that have received the promised service but trigger the penalty mechanism available for FoC servers by the VSC. To this end, honest FoC servers can submit immutable proofs of their FoC service, discouraging dishonest v-witnesses from triggering the VSC for FoC penalties due to the cost required for this on-chain method call. Protecting v-witnesses from FoC servers that fail (or refuse) to deliver their FoC service, can be implemented by integrating network-level information to the VSC logic. For example, v-witnesses can submit their association attempts with the FoC server to the VSC and the VSC can subsequently incur penalties to the respective FoC server, also reimbursing the v-witness with a number of coins from the FoC server fund. However, this process is protocol-specific and requires different codeline to the VSC logic for different RATs, limiting its practical application. To alleviate this limitation, v-witnesses can incorporate off-chain reputation mechanisms to rate FoC servers and discourage their dishonest operation. The same mechanism can be used to identify dishonest FoC servers, reducing the v-witness interest in supporting a candidate validator due to FoC service.

3.3 ULTRA-HIGH TRANSACTION THROUGHPUT

In this section, we describe the RE-CENT payment relay service and the relay SC logic used to implement it. The payment relay service aims to enable instant off-chain payments across RE-CENT nodes at a lower cost, as compared to direct payments. To achieve this, the RSC logic enables the deployment of a decentralized licensing mechanism that enables RE-CENT nodes to run as candidate payment relays and get elected using a DPoS mechanism that is similar to the one used for validators election. Having received a (payment) relay license, a (payment) relay server is subsequently allowed to establish payment channels with relay clients and servers through the RSC. Payment relays buffer and aggregate off-chain transactions signed by relay clients to reduce the transactions throughput spurred into the RE-CENT blockchain. Payment relays can use their own scheduling strategies to trade-off between off-chain payment rewards and direct costs for on-chain (aggregate) payment posting in the RE-CENT blockchain, always being fully aligned with the operating parameters specified in their relay license. Advanced incentive engineering mechanisms are specified to enforce honest operation of relays while enabling the RE-CENT clients (including both RE-CENT servers and clients) to instantly utilize their assigned resources in a credible fashion. In section 3.4, we discuss the joint deployment of relay payment and coin mixing services.

The remainder of this section is organized as follows. In section 3.3.1, we discuss the key principles that make the proposed payment relay service credible in view of all actors involved.

In section 3.3.2, we overview the key parameters of the RSC. In section 3.3.3, we specify the relay licensing mechanism and discuss how the RSC can estimate the mean transactions capacity of the blockchain system. In section 3.3.4, we overview the baseline operation of the payment relay service assuming honest operation and specify an RSC-driven transactions throughput enforcement mechanism per relay. In section 3.3.5, we detail the penalty mechanisms for dishonest relays and overview the operation of the payment relay service protocol under different scenarios where they key RE-CENT actors of the payment service act dishonestly.

3.3.1 BASIC PRINCIPLES OF THE PAYMENT RELAY SERVICE LOGIC

A RE-CENT node can act as a payment relay iff it receives a relay license from the RSC, following a DPoS mechanism for credible relay licensing. A RE-CENT node can issue off-chain payments using the payment relay service iff it has established a payment channel of sufficient balance to an elected (licensed) payment relay through the RSC. We term the respective payment channel as *client-to-relay* payment channel, or as *inbound payment channel*. A payment relay can issue credible off-chain payment promises to a RE-CENT server only if the payment relay establishes a payment channel of sufficient balance to the respective server by calling the RSC. We term the respective payment channel as *relayto-server* payment channel, or as *outbound payment channel*. A payment relay aggregates payments towards the same recipient and updates the balance of inbound and outbound payment channels only through the use of RSC methods. Disregarded payment relay clients and RE-CENT servers that experience inadvertent (or malicious) payment channel updates by the payment relay server can raise on-chain disputes through the RSC. In the sequel, we discuss the key elements of a credible payment relay service.

A payment relay r accepts an off-chain payment issued by a RE-CENT client c and considers it as credible if i) the RE-CENT client has an inbound payment channel with r (through the RSC), ii) the aggregate balance of off-chain promises from c to r including the amount of coins attached to the new payment request, do not exceed the balance of the inbound payment channel, iii) the RE-CENT client signs transactions to the relay enabling it to release funds from the payment channel (from c to r), iv) the expiration time of the inbound payment channel enables the relay to timely withdraw the respective amount of funds from the RSC, and v) the RSC logic enables credible on-chain dispute resolution between dishonest clients and honest relays on the basis the signed proofs provided by c.

A RE-CENT server s accepts an off-chain payment promise of a payment relay r and considers it as credible if:

i) the relay r is authorized by the RSC to deliver payment relay services for the current epoch, ii) the RSC employs a credible relay licensing mechanism, iii) the relay has established an outbound payment channel towards the RE-CENT server s (through the RSC) iv) the balance of the outbound payment channel is sufficient to support the current and all previous non-finalized payment promises issued by r, v) the relay signs a payment promise specifying the block due by which the outcome of the off-chain payment will be posted on-chain (relay delay), vi) the expiration time of the outbound payment channel is at least D_R blocks larger than the promised relay delay, enabling the server to trigger on-chain dispute resolution and claim its legitimate payment directly from the RSC, vii) the RSC is able to compensate honest RECENT servers submitting signed payment promises issued even by dishonest relays in all possible scenarios.

The RSC is considered as credible by the payment relay service parties if i) the RSC is always online, ii) the RSC can always compensate payment relays that submit signed off-chain payments issued by the RE-CENT clients within the payment channel lifetime, assuming that the payment channel has enough balance (i.e. the relay is the sole entity enabled to withdraw funds from active payment channel established by the RE-CENT nodes given input by r, or c), iii) the RSC enables RE-CENT clients to withdraw the rightful amount of remaining funds upon expiration of their inbound payment channel, iv) the RSC can always compensate RE-CENT servers submitting signed yet undelivered transactions (promises) by the payment relay, and v) the RSC logic employs a credible relay licensing mechanism.

The first credibility requirement for the RSC strongly depends on the availability of decentralized nodes hosting the RE-CENT blockchain. As soon as the RE-CENT blockchain is widely accepted in the community, this requirement can be readily met. The second credibility requirement can be met if the RSC provides the payment relays the methods necessary to withdraw funds from inbound payment channels as soon as they are active. If the payment relays fail (or refuse) to timely withdraw funds from the client's payment channel before its expiration, then the RSC shall ignore signed delayed submission of proofs submitted by the relays. Payment relays are fully responsible for verifying that the inbound payment channel balance is sufficient to implement an off-chain payment.

The third credibility requirement (client compensation) can be met if the RSC logic can compensate all RE-CENT clients attached to a tagged payment relay, in the worst case scenario (WCS) where the payment relay server has withdrawn all founds attached to inbound payment channels acting dishonestly (i.e. without authorization from the clients). This requirement can be met if licensed payment relays timelock to the RSC an equivalent amount of coins with the ones that they are authorized to handle during the licensing process. By

denoting the maximum amount of coins that a tagged payment relay r can handle with *maxCoins*[r], we meet this requirement by enforcing licensed payment relays to timelock to the RSC an amount of *maxCoins*[r] that shall be used for that purpose. We term the respective client compensation fund as the relay's *client mirror fund*.

The fourth credibility requirement (server compensation) can be met if the RSC logic can compensate all RE-CENT servers with an outbound payment channel. On the one hand, the RSC logic shall never allow payment relays to establish outbound payment channels with an aggregate balance greater than that of the current aggregate balance of inbound payment channels. In view of that, the RSC can satisfy the fourth credibility requirement iff during the relay election epoch, each payment relay timelocks to the RSC a minimum of *maxCoins*[r] belonging to it, an amount of coins that will be solely used for compensation of RE-CENT servers in the WCS. We term the respective compensation fund as the relay's *server mirror fund*.

The reservation of funds towards a specific RE-CENT server is completely up to the implementation of the relay and more sophisticated fund allocation techniques can be used to maximize the transactions fees received by payment relays. Payment relays are allowed to attach part of their own funds to the payment relay service (serving also as clients) to increase the capacity of outbound payment channels. Such an

Table 2: RSC parameters for a tagged epoch e

Parameter	Notation	Type
Relay epoch duration (in blocks)	B_R	Fixed
Election window deadline (in blocks)	T_R	Fixed
Relay withdrawal guard interval (in blocks)	G_R	Fixed
Dispute resolution time window (in blocks)	D_R	Fixed
Max relay delay threshold (in blocks)	D_{max}	Fixed
Max transactions fee for off-chain payments	F_{max}	Fixed
Minimum stake for relay	M_R	Fixed
Minimum stake for r-witnesses	W_R	Fixed
Baseline relay penalty (in coins)	p_R	Fixed
Free-of-charge service tariff per MB	f_R	Fixed
Relay monitoring period (in blocks)	k_R	Fixed
Current Transactions Capacity	TC_R^e	Adjust.

Relay license tariff table for max users	T_{users}	Fixed
Relay license tariff table for max coins	T_{coins}	Fixed
Relay license tariff table for max throughput	$T_{throughput}$	Fixed

approach can be used to alleviate occasions where payment relays have miscalculated the reservation of a fixed amount coins to payment channels with RE-CENT servers that are not popular. The fifth credibility requirement can be met if the RSC logic employs smart incentive engineering mechanisms enforcing honest operation of elected relays.

3.3.2 RSC PARAMETERS

Table 2 summarizes the main RSC parameters. Most of them remain fixed; except from the private RSC parameter $TC_R[e]$, which is used to estimate the current transactions capacity of the blockchain system. Similar to the VSC logic, payment relays are assumed to participate in an auction-based scheme to receive a payment relay license for a target epoch e. The duration of each relay epoch is fixed and equal to B_R blocks. The relay licensing epoch should conclude T_R blocks before the beginning of the target epoch. To support payment promises issued close to the end of an epoch, payment relays are enabled to withdraw their funds only after the end of epoch e plus G_R blocks.

When an on-chain dispute resolution is triggered, D_R (<G_R) is used as time window enabling the reported payment relay to submit signed proofs of its honest operation. D_max defines the maximum delay window (in blocks) within which a payment relay can post the outcome of an off-chain transaction on-chain. F_max defines the maximum transactions fee that a payment relay can claim per off-chain payment it processes. M_R and W_R define a minimum stake for candidate relays and r-witnesses, respectively. p_R is used as the basis of calculating exponentially increasing penalties to dishonest payment relays, whereas f_R specifies the coins per MB ratio that FoC servers should timelock (similar to the VSC logic). k_R specifies the interval of blocks within which the RSC measures the transactions posted by a tagged payment relay server, enabling evaluation of the mean transactions throughput per relay. $TC_R[e]$ is adapted by the RSC logic on an epoch-by-epoch basis and is used to conclude on the set of elected payment relays (section 3.3.3).

The RSC also includes three tariff tables used to calculate the minimum guarantee fund that the candidate payment

Table 3: Example of Tariff Tables Tusers, Tcoins, Tthroughput (REC D RE-cent Coin).

	No. of. users	**Fee per user**

Max. no. of Users Tariff Table	1 - 1.000	1 REC
	1.001 - 10.000	0.75 REC
	10.001-100.000	0.5 REC
	100.001-1.000.000	0.25 REC
	1.000.001-	0.1 REC
Max. no. of coins Tariff Table	**No. of. coins**	**Fee per coin**
	1 - 1.000	0.5 RCS
	1.001 - 10.000	0.2 RCS
	10.001-100.000	0.1 RCS
	100.001-1.000.000	0.01 RCS
	1.000.001-	0.001 RCS
Mean Tx throughput Tariff Table	**Tx per block (Tpb)**	**Fee per 0.0001 Tpb**
	0 - 0.0001	10.000 REC
	0.0001 - 0.01	120.000 REC
	0.01-1	150.000 REC
	1 - 100	200.000 REC
	100 -	1.000.000 REC

relays should lock onto the RSC for relay penalty purposes. T_{users} adapts the required guarantee fund in line with the maximum number of RE-CENT clients that the payment relay requests to serve. T_{coins} adapts the required guarantee fund in line with the maximum amount of coins that the payment relay requests to handle. $T_{throughput}$ adapts the required guarantee fund in line with the transactions throughput that the paymentrelayrequeststospurintotheRE-CENTblockchain. In Table 3, we provide an example of how the tariff tables can be structured.

Apart from the RSC parameters mentioned above, the RSC should also keep record of all parameters affecting the online operation of licensed relays. Regarding the operation of licensed relays, the RSC should record per relay r:

i)thetransactionsfee$fee[r]$claimedperoff-chaintransaction, ii) the current $cUsers[r]$ and maximum $maxUsers[r]$ number of relay users allowed to attach to the payment relay, iii) the maximum number of coins $maxCoins[r]$ allowed to attach to the payment relay (through inbound payment channels), iv) the mean transactions throughput $Throughput[r]$ allowed for the payment relay, v) a counter $tc[r]$ measuring the number of submitted transactions for the last period of k_R blocks, vi) the first block $lb[r]$ where the counter of submitted transactions has been updated during the current k_R period (section 3.3.4), vii) the relay delay $relayDelay[r]$ by which the relay

promises to update the balance of RE-CENT nodes on-chain, viii) a counter $d[r]$ of delayed payments reported for the payment relay r, ix) the maximum number of coins $rRSCBalance[r]$ that the RSC can currently handle, x) the remaining amount of coins $rServerMirrorFund[r]$ in the relay's server mirror fund, xi) the remaining amount of coins $rClientMirrorFund[r]$ in the relay's client mirror fund,andxii)theremainingamountofcoins$rPenaltyFund[r]$ in the relay's guarantee fund. The RSC is also required to store information regarding i) the remaining amount of coins $cRSCChannel[r,c]$ locked by a tagged RE-CENT client c in a payment channel established to the relay r, ii) the remaining amount of coins $sRSCChannel[r,s]$ locked by the relay r in the payment channel established to the RE-CENT server v, and iii) the current amount of coins $sRSCBalance[r,s]$ that are released by the relay r and can be withdrawn by the RE-CENT server s.

3.3.3 DPoS PROTOCOL FOR RELAY ELECTION

The payment relay election mechanism is a variant of the validator's election described in section 3.2.2. The payment relay election process is epoch-based with a block duration of B_R blocks. The relay election process for a target epoch e starts with the first block of the election epoch $e-1$ and concludes T_R blocks before epoch e. A RE-CENT node r sets candidacy as a payment relay for epoch e, by posting an onchain transaction that triggers the RSC to this end, including as a payload: i) the total relay stakes $relayFund[r]$ that it timelocks to the RSC, ii) the IP through which relay users can reach the relay server, iii) the name of the relay service, iv) the transactions fee $fee[r]$ claimed by the payment relay, v) the maximum number $maxUsers[r]$ of RE-CENT clients that can be attached to the relay service (inbound payment channels), vi) the maximum number of coins $maxCoins[r]$ that can be attached to the relay, vii) the maximum transactions throughput $Throughput[r]$ that the relay is allowed to spur into the blockchain system, viii) the maximum relay delay $relayDelay[r]$ (in blocks) within which the relay promises to post the outcome of a tagged transaction on-chain (unless differently agreed with the RE-CENT servers), and ix) the r-witness reward $wReward[r]$ that will be shared across r-witnesses placing stakes in favor of the relay's candidacy.

The RSC shall consider the candidacy of the RE-CENT node r as valid only if its relay license request locks a minimum *relay guarantee fund* of $relayFund[r] = 2 \cdot maxCoins[r] + T_p[r]$, where $T_p[r]$ includes the minimum stake calculated in compliance with the tariff tables T_{users},

T_{coins} and $T_{throughput}$. If elected, the amount of $maxCoins[r]$ shall be used only as the *client mirror fund* for the tagged relay r (section 3.3.1), the amount of $maxCoins[r]$ shall be used only as the *server mirror fund* for the tagged relay r (section 3.3.1), while the remaining amount of $T_p[r]$,

hereafter termed as the *relay penalty fund*, shall be used only for invoking penalties to the relay r.

During the relay election epoch $e-1$, the RE-CENT nodes can query full consensus nodes to derive the list of candidate relays and place their stakes accordingly. *R-witnesses* can support candidate relays either by locking an arbitrary amount of funds to the RSC during the election epoch $e-1$ (i.e. those funds can be released at the beginning epoch e), or by promising FoC service (tariff calculated according to f_R). Every stake placed in favor of a candidate relay triggers the RSC logic to recalculate the shortlist of elected relays on the basis of the total amount of funds locked in favor of a candidate relays including the relay's guarantee fund $T[r]$, the r-witness funds and the FoC server funds.

RE-CENT nodes are enabled to additionally request the list of r-witnesses and FoC servers supporting a tagged candidate.

The list of r-witnesses can be useful to RE-CENT servers that aim to better infer on the liability of a candidate relay, or to better estimate the impact of their FoC service on the election outcome. Besides, the list of FoC servers can be useful to RE-CENT clients that participate in the relay election process not only to share the reward promised by a candidate relay but to also utilize the offered FoC services from recognizable FoC servers (using their public address).RE-CENT nodes use their own implementation-dependent logic to stake (or not) in favor of a candidate payment relay, e.g. supporting relays that have acted honestly in previous epochs, or relays with low transactions fees.

Relay election requests received after the end of epoch $e-1$ shallbeignored.RE-CENT nodes shall infer on elected relays by scanning the ordered list of candidate relays (based on their aggregate stakes placed in their favor) and by assuming license to the highest-ranking candidate relays for which the aggregate requested transactions throughput is lower than the (estimated) transactions capacity $TC_R[e]$ of the RE-CENT blockchain (skipping entries violating this criterion). The proposed auction-based mechanism enables full utilization of the actual transactions capacity of the RE-CENT blockchain, as estimated by credible mechanisms employed by the RSC. In the absence of r-witnesses, the relay licensing will be still stake-based compliant according to the guarantee funds submitted by candidate relays.

The key differences between the relay and validator election DPoS mechanisms are summarized as follows: i) payment relays cannot amend the RSC parameters, ii) the RSC uses its own private logic to estimate the transactions capacity $TC_R[e]$, iii) candidate relays that are not elected for the target epoch e can withdraw their funds by the beginning of epoch e, iv)

there is no relay replacement mechanism due to the complexity necessary to handle pending off-chain payments by banned relays, v) r-witnesses and FoC servers can always withdraw their placed stakes by the beginning of epoch e, vi) the guarantee fund locked by candidate payment relays is determined by the parameters of the license that they request according to the RSC tariff tables and vii) the requested license and operation of relays can be fully parameterized in line with the functional capabilities of the relay servers.

At this point it is important to note that large-scale service providers can use a small set of public addresses to support FoC service in large areas, enabling RE-CENT nodes to easily recognize them. Besides, by using a single public address for multiple RAN nodes, the RE-CENT blockchain system can aggregate a larger volume of transactions and implement instant off-chain payments at a very large scale, provided that i) a vast volume of RE-CENT client transactions will be issued towards a very small number of RE-CENT servers in a worldwide scale and ii) payment relays will be required to establish outbound channels to a very limited number of RE-CENT servers (section 3.3.4). Fig. 7 provides an illustrative example of the relay election DPoS mechanism. UE1, WNO-AP1 and MNO1-CU1 deploy RSC calls to officially set their candidacy as payment relays for epoch e. The 5G

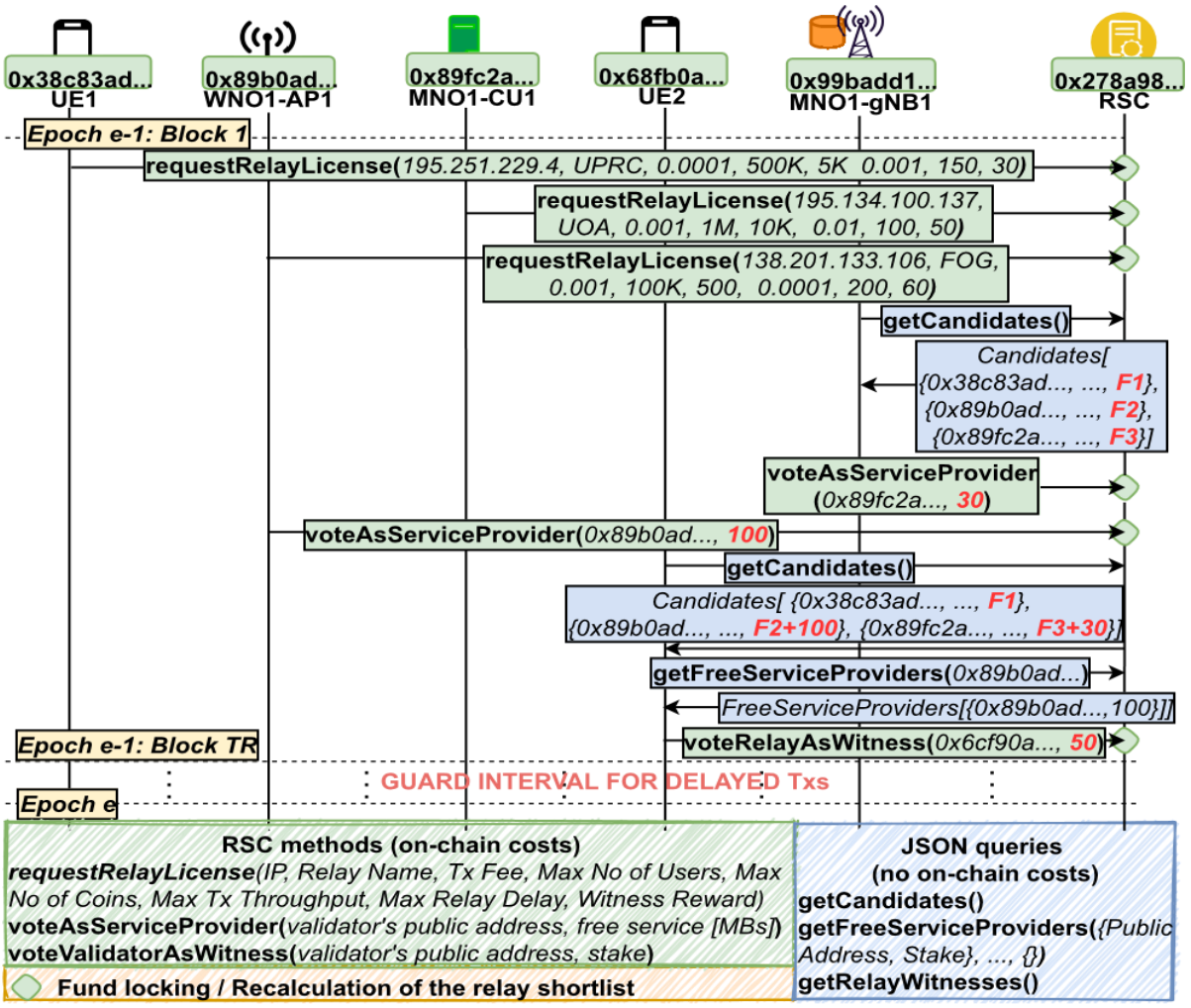

Figure 7: Relay election protocol

base station MNO1-gNB1 subsequently derives the list of candidate relays and votes as a FoC server in favor of the core network unit MNO1-CU1. WNO1-AP1 stakes FoC service in favor of its own candidacy, exploiting its joint network-level (in terms of FoC service) and blockchain-level (in terms of payment relay) capabilities. Assuming that UE2 is in coverage of MNO1, UE2 subsequently identifies the FoC servers supporting MNO1 and votes as an r-witness in favor of the candidate relay MNO1-CU1. The election process completes by the end of epoch e, using a guard interval T_R for delayed RSC calls related to the relay election process.

3.3.4 PROTOCOL OVERVIEW ASSUMING HONEST OPERATION

We now describe how the payment relay service protocol is implemented during a target epoch e, assuming honest operation of RE-CENT servers, clients and payment relays.

3.3.4.1 ACTIONS OF FOC SERVERS, NON-ELECTED PAYMENT RELAYS AND R-WITNESSES

By the beginning of epoch e, r-witnesses of both elected and non-elected payment relays are enabled to withdraw (from the RSC) the full amount of the stakes placed in favor of candidate relays. R-witnesses of elected payment relays can calltheRSCtoadditionallywithdrawtheirlegitimateshareof the r-witness reward. This reward is proportional to the stake placed by the tagged r-witness divided by the total amount of stakes placed in favor of the respective payment relay. Non-elected payment relays can call the RSC to withdraw the timelocked guarantee funds of the election epoch $e-1$. R-witnesses of elected validators may utilize promised FoC services similar to section 3.2.4.

3.3.4.2 ESTABLISHMENT OF PAYMENT CHANNELS

RE-CENT clients use their own logic to select payment relays and establish client-to-relay (inbound) payment channels.

To this end, they call a special RSC method to timelock an arbitrary amount of coins to the RSC, indicating also the payment relay that is authorized to handle them. RE-CENT clients also specify the block due by which the payment channel is in effect, termed as the *expiration block*. Based on the total balance of coins attached to the inbound channels, payment relays subsequently use their own logic to select RE-CENT servers and establish with them outbound payment channels.

The selection of RE-CENT servers can be based on the input provided by the clients (e.g. reactive establishment on a per RE-CENT client request), or on more sophisticated prediction mechanisms using off-chain data (e.g. using data analytics on transactions usage from previous blocks). The aggregate balance of outbound channels should never exceed that of inbound payment channels; if so, the promises issued by the payment relay to RE-CENT servers can be disproportional with the promises received by RE-CENT clients to the relay, compromising the robustness of the payment service. To this end, upon inferring on the balance of outbound payment channels, payment relays should take into consideration both the balance and the expiration time of inbound payment channels. The RSC logic enforces payment relays to conform with this requirement with every outbound payment channel establishment request. If the relay tries to violate this condition, the RSC shall ignore the respective RSC call, making difficult for the payment relay to deliver payments towards RE-CENT servers. Thus, payment relays should deploy smart inbound/outbound payment

channel establishment algorithms to maximize the number of transactions fees they receive in the long-term.

3.3.4.3 SERVICE DISCOVERY, PAIRING AND CHARGING REQUIREMENTS

Using network-level service discovery and pairing protocols, the RE-CENT client shall negotiate the service parameters with the RE-CENT server of its choice and agree on at least the following parameters i) the public addresses that will be used for carrying payments, ii) the payment relay that will be used for instant off-chain payments, iii) the QoE KPI values governing the RE-CENT mobile content delivery, iv)theservicenoncethatshallbeusedtouniquelyidentifythe specific content delivery service per RE-CENT client-server pair and v) a payment timeplan that specifies the timing and amount of payments in full accordance with the size of content chunks delivered per block time. Assuming the payment relay service in the middle, the video session can only start iff i) the payment intermediary is an elected payment relay, ii) the RE-CENT client has established an inbound payment channel of sufficient balance to the payment relay, and iii) the payment relay has established an outbound payment channel of sufficient balance to the specified RE-CENT server.

Service interruption due to insufficient funds in the inbound payment channel is sole responsibility of the RE-CENT client, whereas service interruption due to insufficient funds in the outbound payment channel is sole responsibility of the payment relay. Having insufficient funds in the inbound payment channel is against the interest of the RE-CENT client, which aims to seamlessly consume the mobile video content delivered by the RE-CENT server without interruption (user-driven network-assisted service control). On the other hand, insufficient balance in the outbound payment channel of a tagged RE-CENT server is against the interest of payment relays, which shall lose the opportunity to receive additional off-chain transactions fees.

3.3.4.4 SERVICE IMPLEMENTATION AND CHARGING FLOW

The RE-CENT server delivers to the RE-CENT client the first video chunk as specified in the agreed timeplan and issues an off-chain payment request to the payment relay, specifying the following parameters: i) the public address of the RE-CENT client that should issue the off-chain payment, ii) the amount of coins to be transferred on the off-chain payment, iii) the service nonce and iv) the block due by which the payment relay should update the RE-CENT server’s balance on-chain for the tagged off-chain payment. The reason why we enforce both the RE-CENT server and the payment relay to sign the relay delay value on a per off-chain

payment basis is that this specific value will be used by the RSC to infer on delayed payments in case of an on-chain dispute resolution.

Even though the value of the relay delay is set by the RE-CENT server, the payment relay may reject it if different from the one specified in its license (and stored to the RSC relay registry). If an agreement is reached on the relay delay value specified by the RE-CENT server, the relay signs and forwards a payment request to the RE-CENT client including the following parameters: i) the public address of the RE-CENT server, ii) the amount of coins requested by the RE-CENT client, iii) the signed request for the tagged offchain payment as issued by the RE-CENT server, and iv) the payload of the tagged off-chain payment (which shall enable the client to verify the signature of the RE-CENT server).

Accordingly, the RE-CENT client verifies the signatures/payloads attached to the payment request and issues an off-chain transaction authorizing the payment relay to withdraw the respective amount of coins from the inbound payment channel. The payload of the RE-CENT client transaction includes: i) the public address of the payment relay, ii) the public address of the RE-CENT server, ii) the amount of coins to be released from the inbound payment channel, iii) the relay delay block, iv) the service nonce and the sequence number that shall enable the RE-CENT server to uniquely match the signed transaction to the delivered service chunks. Note that the relay delay block value does not enforce the payment relay to subtract the indicated amount from the respective inbound payment channel; instead, it is a commitment from the payment relay server that it will update the balance of the RE-CENT server due by the specified block.

Both the signed transaction and its payload are forwarded to the payment relay, which performs verifies the signature/payload and buffers the received transaction to its local registry of pending off-chain transactions. This transaction shall also be used as a proof in case of on-chain dispute resolution between the RE-CENT client and the payment relay (section 3.3.5). The payment relay subsequently issues an off-chain payment promise to the RE-CENT server by including the following parameters: i) the public address of the RE-CENT server, ii) the amount of coins to be released from the corresponding outbound payment channel, iii) the service nonce and sequence number of the video chunk, and iv) the relay delay agreed by the payment relay and the RE-CENT server. Signed transactions issued by the payment relay together with their payload are forwarded to the RE-CENT server, which verifies signatures/payloads, and buffers it in its local registry of pending off-chain transactions.

RE-CENT servers are expected to monitor the RE-CENT blockchain to identify whether an off-chain promise has been delivered. If not, payloads and their signed hashes shall be used by

the RE-CENT servers to trigger on-chain dispute resolution.Undersuchoccasions,theRSCshallinvokepenalties to malicious relays and potentially revoke their license as specified in section 3.3.5. In the general case, RE-CENT servers will not report payment relays releasing a larger amount of coins from outbound payment channels to them, while the RSC shall not invoke penalties to payment relays implementing such actions; however, payment relays will not be refunded under such occasions as they are the sole responsible for avoiding payment miscalculations.

The aforementioned process continues until the service delivery session is concluded according to the agreed timeplan, or the RE-CENT peers abort the service. At this point, we note that it is possible for RE-CENT clients to receive a video chunk and not issue the payment to the RE-CENT server. Nonetheless, the negative impact of such unfair behavior by RE-CENT clients can be mitigated by an appropriate adjustment of the agreed payment timeplan. For example, the RE-CENT server may requests higher refunds for earlier video chunks, or even advance payments. Payment relays and RE-CENT servers are also enabled to interrupt the service at any time; however, such an action would be against their interest in receiving transaction and service fees.

3.3.4.5 AGGREGATION OF PAYMENTS

Payment relays shall keep record of pending off-chain transactions and implement aggregate on-chain balance updates using their own vendor-specific logic. Accordingly, payment relays shall release funds from inbound and outbound payment channels taking into consideration the expiration time of inbound channels and the promised relay delay block of off-chain transactions, respectively. In a baseline payment aggregation scenario, the RSC logic would require payment relays to accompany their release fund requests with the full set of signed proofs and payloads. However, since the RSC methods are called through on-chain transactions carrying as payload the necessary input, such an approach is equivalent to direct P2P payments between the RE-CENT service peers. As discussed in section 1, this wouldn’t enable the RE-CENT blockchain to scale and support a multi-million transactions throughput. Besides, payment relays of this type will have to pay a large on-chain cost that is disproportional to the transactions fees received due the deployment of off-chain payments.

In view of that, payment relays are allowed to submit only the aggregate outcome of individual payments without providing the corresponding signed proofs and payloads for verification to the RSC. Instead, the RSC enables RE-CENT payment relays to release funds from inbound and outbound payment channels by submitting only a Merkle tree hash of all pending transactions implemented through the respective balance update. Merkle tree hashes enable

RE-CENT servers and clients to deploy membership verification of individual payments and infer on the honest operation of payment relays comply (correct amount of coins and timely delivery of payments). Given a signed relay promise having a relay delay b, payment relays are enabled to aggregate all pending off-chain transactions that have as recipient the tagged RE-CENT server (including the ones expiring after block b) and all transactions issued by the same public address (RE-CENT server) when acting as RE-CENT client.

In section 3.3.5 we investigate various scenarios where RE-CENT clients, servers and payment relays act dishonestly, specifying necessary on-chain dispute resolution and penalty mechanisms employed by the RSC logic. Payment relays claim their transaction fee in an aggregate fashion as well, in line with the aggregate off-chain payments that they implement through a single RSC call. Transactions fees are paid by RE-CENT clients, which are also the recipients of the actual mobile video content delivery service. Accordingly, payment relays claim aggregate transaction fees only through the RSC method used to update inbound payment channels.

3.3.4.6 FUND WITHDRAWAL FROM PAYMENT CHANNELS

RE-CENT clients are enabled to close inbound payment channels that have expired at any time. RE-CENT servers should never accept payment promises on outbound payment channels that expire earlier than the promised relay delay. Besides, RE-CENT servers should also act proactively and withdraw funds released by payment relays before the expiration of the corresponding payment channel. Provided that payment relays are enabled to withdraw their relay guarantee funds by the end of epoch e plus G_R blocks, we safeguard system robustness against last-minute, delayed, or malicious payment promises by dishonest relays. RE-CENT servers should always withdraw their funds not later than G_R-D_R blocks after the end of a relay epoch, where D_R is the time available for on-chain dispute resolution (section 3.3.5). Payment relays are enabled to reallocate, or completely withdraw, funds from expired outbound payment channels. However, the RSC shall enable refund requests by RE-CENT servers with an expired outbound payment channel and invoke penalties to dishonest relays. The rational behind this design approach is that payment relays are solely responsible for the honest implementation of off-chain payments independent of whether the RE-CENT servers timely withdraw released funds.

3.3.4.7 TRANSACTIONS THROUGHPUT GAINS OF THE PAYMENT RELAY SERVICE

Recall that relays receive transactions fees per off-chain transaction that they implement, based on signed proofs provided by RE-CENT clients. In view of that, payment relays aim to

maximize the number of off-chain payments that are aggregated while minimizing the on-chain costs paid to implement them (updating inbound and outbound payment channels through the RSC). Payment relays should comply with the relay delay blocks promised on a per transaction basis but also respect the transactions throughput specified in their license (to avoid indirect penalties - section 3.3.5). In this fashion, the design of the payment relay service is fully aligned with the blockchain scalability challenge to minimize the number of on-chain transactions in the long-term.

The number of transactions generated by a tagged payment relay is given by i) the number of new inbound payment channels established per block, ii) the number of withdrawals made by RE-CENT servers on their outbound payment channels per block and iii) the number of new outbound channels established per block. Accordingly, depending on the payment aggregation strategy followed by the RE-CENT payment relays, we identify four different transactions throughput performance bounds. The first bound is attained when payment relays aggregate micro-payments between two tagged service peers (client/server), enabling the transactions throughput to scale with the number of new service peers generated per block taking also into consideration the relay delay threshold assumed per micro-payment. The second bound is attained when payment relays aggregate micro-payments of multiple RE-CENT clients towards the same RE-CENT server, enabling the RE-CENT blockchain transactions capacity to scale with the number of new RE-CENT servers per block taking also into consideration the relay delay threshold used on a per micro-payment.

The third performance bound is attained when payment relays keep track of the inbound/outbound balance per RE-CENT service peer (i.e. a RE-CENT client can also act as RE-CENT server, and vice versa), enabling the system to scale with number of non-zero balance updates necessary per block given the relay delay threshold promised per micropayment. This bound is attained when RE-CENT servers choose not to withdraw funds from their outbound payment channels but instead, they authorize the payment relay to transfer released from their outbound payment channels in a corresponding inbound payment channel, enabling the system to scale with the number of new inbound and outbound payment channels generated per block by the payment relay.

If inbound and outbound payment channels are established only once, the last approach allow the payment relay service to attain near-zero transactions capacity of the RE-CENT blockchain, enabling infinite scaling. Such an approach is not far from the cellular MNO reality, provided that large MNOs and content providers may utilize a single public address for RE-CENT service charging, enabling payment relays to minimize the number of outbound

payment channels towards RE-CENT servers and aggregate a large volume of off-chain payments towards a single public address per operator.

3.3.4.8 A RUN-TIME EXAMPLE

Fig. 8 provides an illustrative instance of the payment relay service, assuming honest operation of all actors involved. The example considers that the relay election phase of Fig. 8 has preceded. Aiming to highlight the outcome of the actions deployed by the RSC logic, we provide further details on the values taken from the key RSC parameters (i.e. the RSC state) and the balance updates performed on-chain by the RSC. Let us focus on the operation of the payment relay MNO1-CU1 and assume that the transactions timer of all relays is updated once every two block tags (epoch slot 1 lasts from b_0 to the end of block b_1, etc.). At the early blocks of the target epoch e, the non-elected relay UE1 and the r-witness UE2 uses the *witnessRefund* method to release reward funds (Fig. 7).

UE1 and UE2 establish an inbound payment channel to the payment relay MNO1-CU1, specifying the amount of coins to be locked and the block due by which the payment channel expires. Using network-level service discovery and pairing protocols, UE1 and MNO1-gNB1 initiate a mobile video content delivery session, selecting as a payment relay MNO1-CU1. Accordingly, MNO1-CU1 uses the *relayDeposit* method to establish one outbound payment channel with MNO1-gNB1 of balance 48 coins and one outbound payment channel with UE of balance 40 coins. However, since the aggregate balance of inbound payment channels of MNO1-CU1 equals to 74, the payment relay also transfers from its own public address the excess amount of 14 coins.

The mobile content delivery service begins with the server MNO1-gNB1 delivering video chunks to UE1 according to the QoE KPI values and payment timeplan agreed in the network-level service discovery and pairing phase. UE1 issuesnecessaryoff-chainpaymentstoMNO1-gNB1through the payment relay MNO1-CU1. To this end, the RE-CENT client issues new off-chain payments that aggregate the current to the previous amount of transferred coins, i.e. a new off-chain payment replaces the old one for the same service nonce. The first and the last off-chain payments delivered from MNO1-CU1 to MNO1-gNB1 for this service are assumed to expire due by block b_2 and b_3, respectively.

After the end of the first service between UE1 and MNO1-gNB1, the nodes UE2 and MNO1-gNB1 initiate a new RE-CENT service session using MNO1-CU1 as a payment relay. All signed off-chain payments issued by MNO1-CU1 during this service are assumed to expire due by block b_4. By the end of block b_2, the second video session concludes and MNO1-CU1 updates the outbound payment channel of MNO1-gNB1 by block b_2, including currently

pending off-chain payments of both video sessions. The respective *releaseServerFunds* method call increases the counter measuring the transactions throughput of the respective relay and reduces the balance of attached coins that the MNO1-CU1 can assign to outbound payment channels as necessary. Note that during this call, the payment relay does not claim any transactions fees, which can only be claimed when the inbound payment channels of RE-CENT clients are updated (UE1 and UE2 in this case).

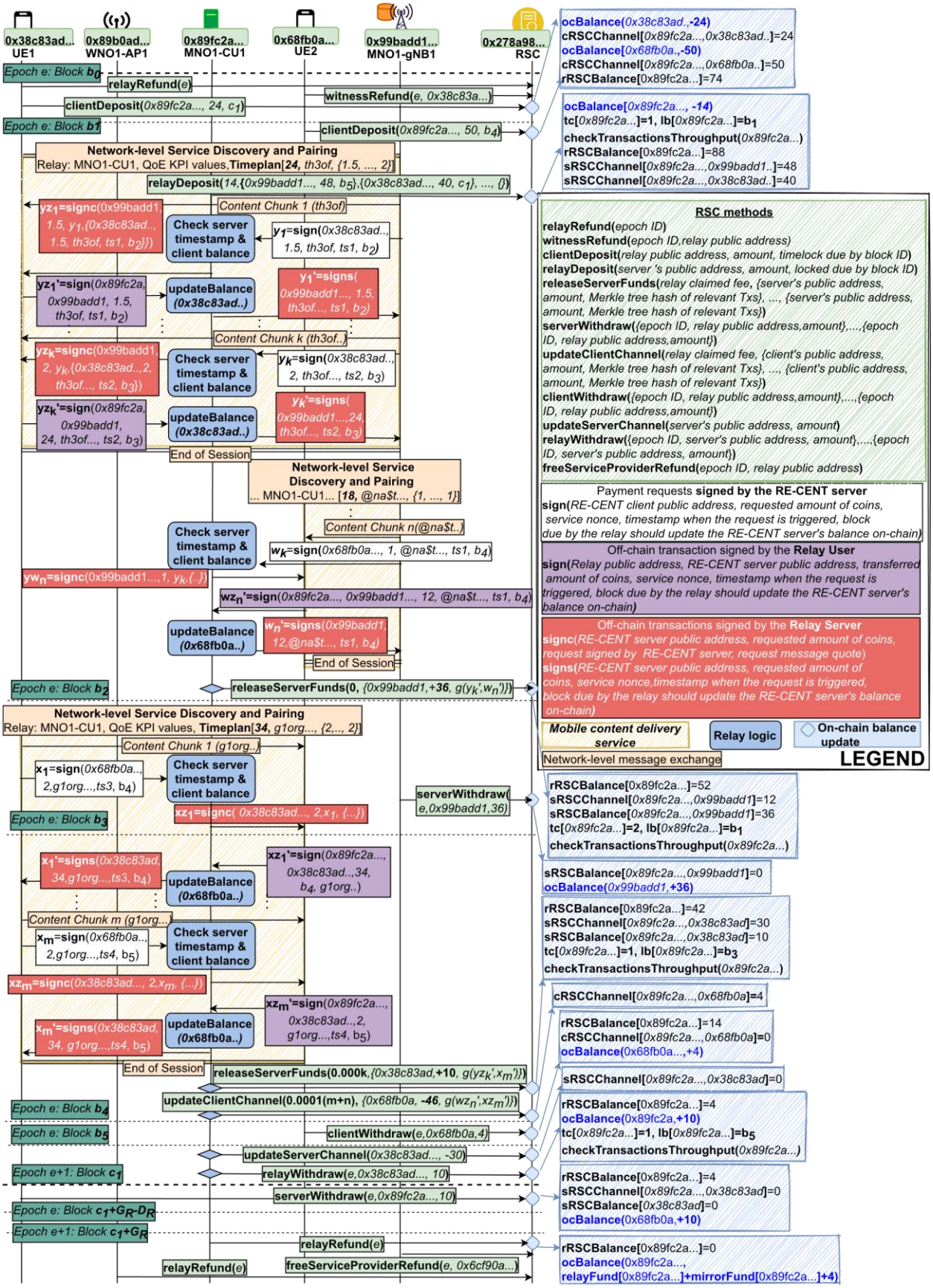

Figure 8: Payment relay service assuming honest operation

Assuming that UE1 has cached the content delivered by MNO1-gNB1 during the first video session, in the sequel we consider that UE1 relays the cached video content to UE2 which has now moved outside the coverage of MNO1-gNB1. Using network-level service discovery and pairing protocols, the two UEs employ D2D (sidelink) communications and agree to use the payment relay services of MNO1-CU1. The video delivery service continues and the first payment promise issued by MNO1-CU1 sets the relay delay due by block b_4. In the mean time, MNO1-gNB1 uses the *serverWithdraw* method to withdraw the funds released in the outbound channel established by MNO1-CU1, a few blocks before the outbound payment channel expires (due by block b_5). The video delivery service between UE1 and UE2 concludes by the end of block b_3, enabling the payment relay MNO1-CU1 to release the rightful amount of coins for the entire video service to UE1 before block b_4. However, MNO1-CU1 identifies that UE1 has spent 24 coins from its inbound payment channel as a RE-CENT client and also acted as RE-CENT server during the third video session with UE2, it aggregates all pending inbound and outbound payments of UE1 to release the difference of 10 coins from the outbound payment channel of UE1.

Since the respective payment implements a signed promise of the RE-CENT client UE1, with this call, the MNO1-CU1 claims transactions fees for off-chain payments implemented by the first video session. To enable UE1 identify its actions but also defend itself in a potential on-chain dispute resolution, the payment relay uses the *releaseServerFunds* call to append the Merkle tree hash of pending transactions where UE1 has acted as client and server. The payment relay can also notify the RE-CENT client offline on which set of transactions are included per Merkle tree hash, enabling easy verification at the client/server side. With (*updateClientChannel*), the payment relay reduces the inbound channel of UE2 by the aggregate cost of both video sessions held with UE1 and MNO1-gNB1 (46coins) and claims the rightful amount off-chain transactions fees.

After the expiration of its inbound payment channel to MNO1-CU1, UE2 withdraws the remaining amount of coins from the RSC. MNO1-CU1 also closes the outbound payment channel to UE1 and withdraws part of its own funds that were locked in the early steps of the epoch e. After the expiration of the inbound payment channel to MNO1-CU1, UE1 also withdraws the remaining amount of funds from the RSC. Following the end of epoch e plus the guard interval of G_R blocks, payment relays and FoC servers withdraw the remaining guarantee funds and stakes. The format of all RSC calls and signed off-chain payments is detailed in Fig. 8.

3.4 RELAY MONITORING AND PENALTY MECHANISMS

The RE-CENT payment relay service is based on the employment of simple yet highly-effective penalty mechanisms that are implemented by the RSC logic and are designed to discourage dishonest operation of RE-CENT clients, servers and payment relays. Every RE-CENT node calling an RSC method is required to pay the on-chain cost of the respective but also the cost necessary for enabling on-chain computations. This is similar to the gas-based execution of SCs on the ETH platform. In case of an on-chain dispute, honest RE-CENT clients and servers shall be reimbursed for this cost by the penalty fund of the dishonest relay. RE-CENT clients and servers are discouraged to act dishonestly, as they will not be reimbursed for the on-chain cost triggering the RSC-enforced dispute resolution. RE-CENT clients and servers consuming payment relay services should always use public addresses that have sufficient balance to trigger the RSC logic in case a payment relay acts dishonestly. Honest relays also pay for the cost of submitting proofs of their legitimate actions, to avoid potential penalties by the RSC on their guarantee funds.

On-chain dispute resolutions concluded in favor of disregarded RE-CENT clients shall enable full reimbursement from the *client mirror fund* of the dishonest payment relay. The capacity of inbound payment channels shall be reduced by the respective amount of misused coins for the entire target epoch(i.e.*maxCoins*[*r*] shall be amended by the RSC accordingly). On-chain dispute resolutions concluded in favor of disregarded RE-CENT servers shall enable full reimbursement of misused coins to them from the *server mirror fund* of the dishonest payment relay. The capacity of outbound payment channels shall be reduced accordingly, provided that its value should always fall short as compared to that of inbound payment channels. Core component of the RSC logic is the employment of a penalty mechanism for delayed payments. This mechanism incurs an exponentially increasing penalty to payment relays that fail to provide proofs of their honest operation. The penalty is proportional to the amount of misused coins X multiplied by the baseline relay penalty $p_R(> 1)$ defined in the RSC, in the power of total delayed payments $d[r]$ verified by the RSC for the tagged relay r. Both parameters are stored and updated by the RSC. The penalty function is given by Eq. 3.

$$Penalty[r] = X \cdot p^{d}{}_{R}{}^{[r]} \quad (3)$$

The RSC logic is designed so as to directly, or indirectly, trigger the mechanism for delayed relay payments for most of the dishonest actions that can be performed by payment relays. Besides, the RSC logic makes impossible for dishonest payment relays to violate their key relay license parameters by design, e.g. exceeding the max number of attached coins, the max number of attached users, or the max transactions throughput. In the sequel, we overview

different scenarios where the payment service parties act dishonestly and discuss how the RSC logic is secured against such operations.

3.4.1 PENALTY MECHANISM FOR DELAYED PAYMENTS

We start our analysis with the scenario where a payment relay *r* fails (or refuses) to timely submit to the RSC the outcome of an off-chain payment issued to RE-CENT server *s*. Let us denote by *b* the relay delay block indicated on the relay's signed promise. Also, let us denote the misused amount of coins with *X* and the on-chain cost paid by the disregarded RE-CENT server with *W*. The disregarded RE-CENT *s* shall use the *reportDelayedPayment* method to trigger the on-chain dispute resolution by submitting to the RSC i) the public address of the dishonest payment relay, ii) the signed proof y^0 and the payload of the delayed off-chain payment, and iii) the necessary cost (gas) required for the RSC to perform onchain computations. Using these values as an input, the RSC shall verify the signature of the payment relay on the payment promise and investigate if the RE-CENT server has an active outbound payment channel with sufficient balance. If not, the RSC shall reduce the amount of misused coins *X* with the balance available to the active payment channel, taking into consideration that the RE-CENT server has either accepted a signed payment relay promise without having the necessary guarantees, or it submits a dispute resolution for a payment that refers to an expired payment channel. In both scenarios, the inconvenience will be the result of faulty operation of the RE-CENT server that has not timely claimed a signed payment promise from the payment relay.

However, if an active outbound payment channel exists between the relay *r* and the server *s*, the RSC shall append a new entry in its local registry of pending on-chain disputes by storing i) the current block time, ii) the (revised) amount of coins claimed by the RE-CENT server, and iii) all input provided by the RE-CENT server upon triggering the onchain dispute. The RSC logic shall be re-triggered again either by the reported payment relay *r*, or by the disregarded RE-CENT server *s*. If the payment relay *r* triggers the RSC on-chain dispute resolution logic and more than D_R blocks have passed since the emit of the *Delayed payment event*, the RSC shall ignore the proofs submitted by the payment relay. If the payment relay triggers the RSC logic and no more than D_R blocks have passed since the emit of the delayed payment event, the payment relay response should include the Merkle tree hash of the RSC call implementing (among others) the respective off-chain transaction and all signed proofs / message payloads of the off-chain transactions implemented with the respective call (identified by the Merkle tree hash).

A key requirement for the second case is that the payment relay also includes sufficient cost (gas) to enable the full execution of actions by the RSC for on-chain dispute resolution (or else the RSC will not be able to conclude on the honest operation of the payment relay). Accordingly, the RSC shall verify the validity of signatures (and payloads) of all the transactions included in the respective set of transactions, verifying whether the reported delayed payment is included, or not. If the amount of coins and signatures comply with the original relay post, the RSC will close the on-chain dispute resolution without proceeding to any further actions. Dishonest relays are not expected to respond to the delayed payment event emitted by the RSC, due to the additional cost required to pay on-chain. However, if the amount of coins and signatures do not comply with the original relay post, the RSC logic shall enable disregarded RE-CENT servers to retrigger the RSC logic and get full reimbursement for both the amount of misused coins verified within this transactions set and the on-chain costs necessary for triggering the RSC.

Accordingly, the RSC shall i) reduce the outbound payment channel balance capacity of the dishonest relay by the amount of misused coins, ii) withdraw the same amount of coins from the server mirror fund, iii) reduce by W coins the penalty fund of the dishonest relay r, iv) reduce the outbound channel of relay r to server s (if not expired) by the amount of misused coins, v) reduce the number of maximum attached coins $maxCoins[r]$ to the relay server by the amount of misused coins, vi) increase the counter $d[r]$ of reported delayed penalties for relay r as necessary, vii) increase the counter $tc[r]$ measuring the transactions throughput of relay r by one and update the last block value $lb[r]$ as necessary, viii) fully reimburse the RE-CENT server s on-chain (misused coins plus W), and ix) invoke a penalty according to Eq. 3 to the RSC penalty fund of the relay r.

The RSC will further check if the penalty fund of relay r is expended. If not, the RSC shall emit a *Relay Penalty Event* to the RE-CENT consensus network enabling other nodes to take further actions (e.g. avoid service from the payment relay). If expended, the RSC shall revoke the license of the relay and emit a *Relay license revoke event*, informing all RE-CENT nodes that the payment relay r is currently banned. RSC calls from banned relays shall be ignored, enabling full refund of RE-CENT clients and servers with pending off-chain transactions from the available client and server mirror funds, respectively. RE-CENT servers are discouraged to request for a higher amount than the one promised by payment relays, provided that i) they cannot forge the signature of the payment relay, ii) the on-chain dispute will conclude against them and iii) they will not be reimbursed for the onchain cost required to trigger the RSC logic.

3.4.2 EXAMPLE OF THE PENALTY MECHANISM FOR DELAYED PAYMENTS

Fig. 9 provides an illustrative example of the RSC penalty mechanism for delayed payments and the states taken by the RSC. Similar to Fig. 8 we assume the implementation of three video sessions between UE1 and MNO1-gNB1, UE2 and MNO-gNB1 as well as UE1 and UE2. A slight change to the block timing is considered in Fig. 8 to enable easier understanding of how the RSC logic implements penalties for delayed payments. Different from Fig. 8, we consider that the payment relay MNO1-CU1 submits only the outcome of the first video chunk payment (y^0_1) to the RSC and fails to submit the outcome of the remaining off-chain transactions referring to the first video session (which now expire by block b_3). After block b_3, MNO1-gNB1 triggers reports a delayed payment from MNO-CU1, by posting to the the on-chain dispute resolution mechanism of the RSC and RSC the public address of the dishonest relay together with the payload and the signature of the delayed payment issued by MNO1-CU1.

The RSC subsequently performs verifies whether an active payment channel exists between MNO1-CU1 and MNO1-gNB1, if the balance of the outbound payment channel is sufficient and if the service nonce of the reported delayed payment has received previous off-chain payments. Accordingly, the RSC emits a Delayed Payment Event to the RE-CENT consensus network, indicating the public address of the reported relay and the signed hash of the delayed payment. MNO-CU1 does not respond to the on-chain dispute resolution call and the dispute concludes D_R blocks after the block due by which the delayed payment has been reported. The RE-CENT server MNO1-gNB1 subsequently triggers the RSC again to claim its legitimate payment along with the on-chain costs paid for triggering the RSC logic (two times). The RSC subsequently identifies that, for the reported service nonce, the payment relay has submitted another offchain payment of 1.5 coin and updates the claimed coins from the RE-CENT server to 22.5 (instead of 24).

Accordingly, it reduces the balance of the outbound payment channel of the respective server by 22.5 coins, updates the current outbound balance of the payment relay MNO1-CU1 and its outbound payment channel capacity by the same amount of coins, while it also updates on-chain the balance of MNO1-gNB1. The RSC further increases the throughput counter of MNO1-CU1, updates the last block throughput registry as necessary, subtracts the respective amount of coins from the server mirror fund, updates the number of delayed payments by one and employs Eq. 3 to reduce the remaining penalty fund of the payment relay MNO1-CU1. The RSC performs all necessary checks to infer on whether the license of the relay should be revoked. This event is not yet triggered and the RSC emits a Relay Penalty Event indicating the public address of the dishonest payment relay.

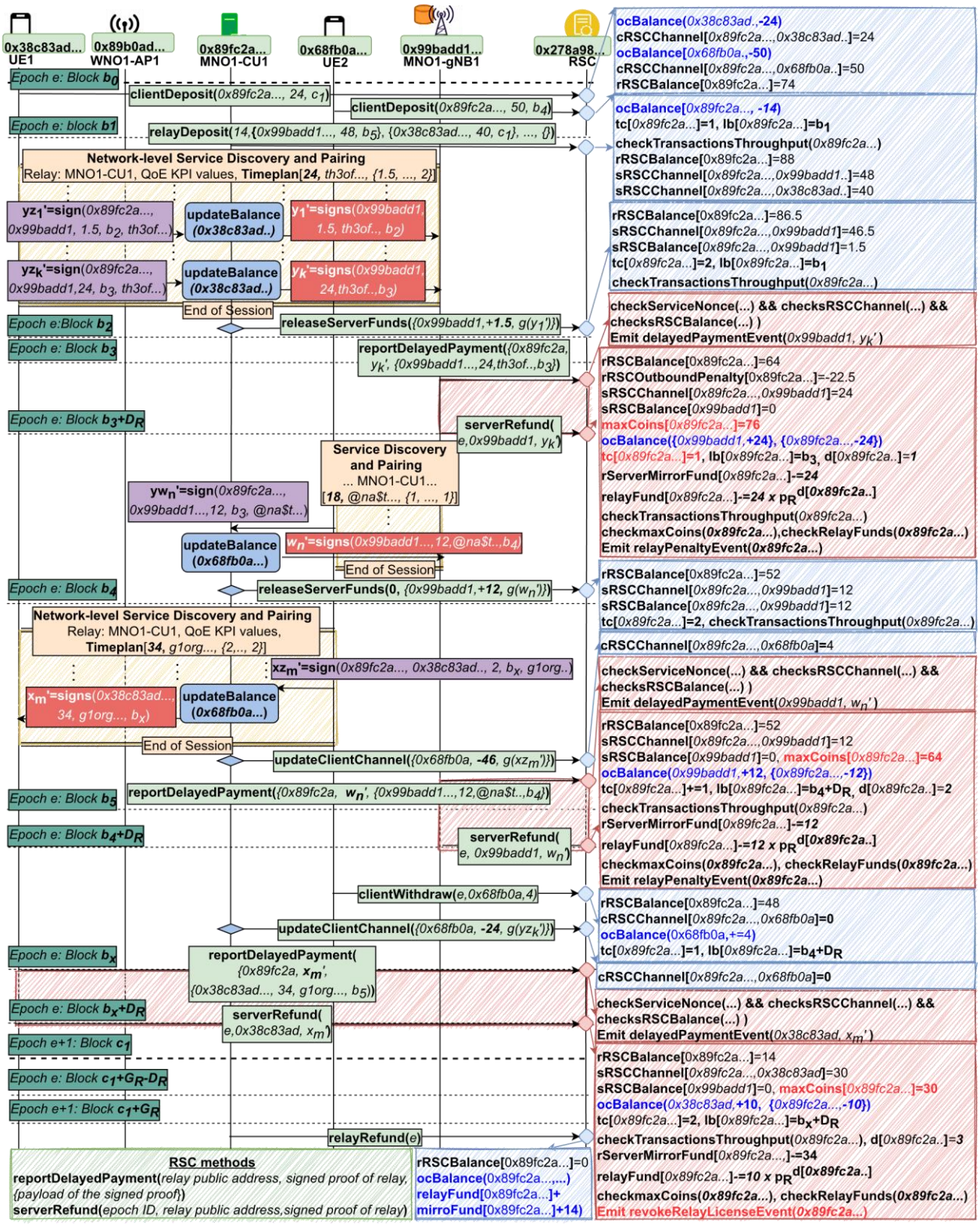

Figure 9: Payment relay service assuming delayed payments

In block b_4, we consider that the RE-CENT server MNO1-gNB1 deploys a dishonest RSC call to trigger again an on-chain dispute resolution with the payment relay MNO1-CU1. However, MNO1-CU1 has been honest and posted within time the reported off-chain payment promise w^0_n (second video session). Thus, it responds to the Delayed Payment Event with all necessary proofs (including Merkle tree hash, signed promise and payload of the promise) to enable the RSC conclude the on-chain dispute in its favor. The dishonest RE-CENT server shall not receive a reimbursement for its actions this time. In block b_6, a new onchain dispute is triggered by UE1, which reports a delayed payment for the MNO1-CU1's signed off-chain promise x_m^0. Once again, the payment relay is unable to submit necessary proofs to the RSC and the dispute concludes in favor of UE1, which receives legitimate refunds and reimbursements by the client mirror and the guarantee fund of MNO1-CU1, respectively. This time the RSC decides to revoke the license of MNO1-CU1 in addition to deploying penalties, also emitting a Relay License Revoke event to the RE-CENT consensus network.

Note that for every delayed payment event that is confirmed, the outbound payment channel balance for the payment relay is reduced by the *rRSCOutboundPenalty*[*r*] value in order to ensure that the remaining funds of the server mirror fund will suffice to refund all potential disregarded servers. Using this parameter, the RSC logic makes sure that the amount of coins attached by the payment relay to the outbound payment channels will always lag the necessary amount of coins as compared to the amount of coins attached to the payment relay from inbound payment channels.

3.4.2.1 UNAUTHORIZED TRANSFER OF FUNDS AND DISHONEST WITHDRAW REQUESTS BY CLIENTS

Let us now focus on the scenario where the payment relay makes an unauthorized transfer of funds to some RE-CENT nodes (i.e. unauthorized withdrawal from an inbound payment channel). To achieve this, the payment relay can either attempt an over-withdraw from an inbound payment channel of an attached RE-CENT client, or transfer the respective amount of funds waiting for some client withdraw operation to exceed the remaining balance of inbound payment channels attached to the dishonest relay.

In the over-withdraw scenario, the RSC logic enables disregarded RE-CENT clients to report the over-withdraw action performed by the payment relay at any time within the relay epoch period where the dishonest payment relay is active. To this end, the disregarded client shall call the *reportOverwithdraw* method available by the RSC, passing to the RSC necessary details of the dishonest payment relay action (i.e. relay's public address, Merkle tree hash and misused amount of coins for the respective transaction). Through this call, the RE-CENT client

shall trigger on-chain dispute resolution that shall conclude after D_R blocks by the RSC. Honest payment relays shall use the *respondOverwithdraw* RSC method to provide i) the RE-CENT client public address, ii) the Merkle tree hash of the disputed transaction, iii) all signed promises issued by the RE-CENT client, and iv) the payload of the signed promise. Dishonest payment relays will be unable to forge a signed transaction by the RE-CENT client, resulting to a dispute resolution against them.

In such occasions, the RE-CENT client shall trigger the RSC logic again to request a full refund of misused coins (*clientRefund* method) and necessary penalties shall be deployed to the dishonest payment relay. In more detail, following a similar approach with delayed payments, the RSC will fully reimburse the RE-CENT client from the relay's mirror fund and reduce the *maxCoins*[*r*] value of the dishonest payment relay *r*. On-chain costs of the RE-CENT server calls and penalties shall be withdrawn by the guarantee fund of the dishonest payment relay. Over-withdraw events shall also increase the delayed payments timer and all parameters related to the throughput measurement of the tagged relay.

Note that dishonest payment relays committing overwithdraws will typically not subtract the respective amount of funds from an attached RE-CENT client using the *updateClientChannel* method, to avoid detection by RE-CENT clients and not trigger the aforementioned on-chain dispute. Instead, they shall transfer the respective amount of funds to the RE-CENT node of their choice without posting an equivalent over-withdraw of funds from a RE-CENT client (i.e. his own public address, or the address of a colluding RE-CENT node). In this scenario, the aggregate balance of the outbound payment channels of the dishonest payment relay will always lag compared to that of inbound payment channels. Recall that the RSC logic shall never allow payment relays to establish outbound payment channels leading to an aggregate outbound channel balance that exceeds the aggregate number of coins attached to them (i.e. through inbound payment channels) minus the balance penalties received by the RSC.

In such a scenario there will be some point where a critical number of RE-CENT clients will request to withdraw funds from inbound payment channels attached to the dishonest relay that expire. Accordingly, the total number of coins attached to the payment relay will lag that of outbound payment channels and the RSC logic will identify unauthorized transfer of funds from the payment relay. Even though the unauthorized transfer cannot be identified, the relay will emit an *RelayOverwithdraw Event* and trigger on-chain dispute resolution between the RE-CENT client performing the client withdraw and the dishonest payment relay. The RE-CENT clients shall then scan the RE-CENT blockchain to identify potential unauthorized payments issued by the dishonest payment relay on their inbound payment channel and trigger the *reportOverwithdraw* method for all dishonest transactions performed by the respective relay.

Dishonest relays will be unable to submit the necessary list of signed RE-CENT client promises to the RSC and the RSC logic will conclude the dispute in favor of the disregarded RE-CENT client. However, if the RE-CENT client acts dishonestly and requests for a higher amount of coins from the ones that it should have, the RSC logic shall reimburse the honest payment relay using the funds remaining to the inbound payment channel of the respective RE-CENT client and shall not reimburse the RE-CENT client for the onchain costs paid to trigger the on-chain dispute. Note that the requirement to submit a target list of dishonest transactions performed by the payment relay, necessitates the RE-CENT clients to pay for a proportional cost to the one necessary for honest payment relays submitting signed proofs of their legitimate withdraw of funds from the client's inbound penalty fund. In this manner, RE-CENT clients are discouraged to act dishonestly and overrun honest payment relays with their dishonest withdrawal requests.

3.4.2.2 EXAMPLE OF UNAUTHORIZED TRANSFER OF FUNDS

In Fig. 10, we provide an illustrative example of an unauthorized payment deployed by the payment relay MNO1-CU1. In this example, we consider that MNO1-CU1 and WNO1-AP1 collude in order to misuse coins attached to the inbound payment channel of MNO1-CU1. To this end, MNO1-CU1 creates an outbound payment channel to the (malicious) server WNO1-AP1 by block b_2 and releases 40 coins to the public address of WNO1-AP1 by providing to the RSC a valid signature for its operation. This action remains unnoticed by both the RSC and the RE-CENT clients until block b_4, where the inbound payment channel of UE2 (attached to MNO1-CU1) expires and the respective RE-CENT clients claims its remaining inbound payment channel funds. Since the inbound payment channel balance of UE2 on MNO1-CU1 goes negative, the RSC can infer on the dishonest operation of MNO1-CU1 and emit a *RelayOverwithdraw* Event to notify the reported payment relay through the RE-CENT consensus network.

Honest payment relays shall respond with all necessary proofs to enable the RSC infer on their honest operation, being partly reimbursed for their on-chain call from the funds remaining at the inbound payment channel of the dishonest RE-CENT client. However, dishonest payment relays (like MNO1-CU1 in Fig. 10) will fail to respond in the on-chain dispute, enabling the RE-CENT client to be fully refunded for its two on-chain RSC calls, further to withdrawing the full amount of funds remaining to the inbound payment channel.

3.4.2.3 TRANSACTIONS THROUGHPUT MONITORING FOR PAYMENT RELAYS

If the transactions throughput of the RE-CENT blockchain system is to be attained within acceptable limits, on-chain balance updates should be subject to rate control of the RSC. To

preserve system scalability, in the sequel we propose a mechanism to measure the transactions throughput offered by payment relays to the RE-CENT blockchain based on the number of fund releases in outbound payment channels (i.e. *releaseServerFunds()* RSC method). Besides, to avoid multiple inbound channel closing requests that will overrun the transactions capacity of a given payment relay, the RSC shall allow RE-CENT clients to perform a single request towards the withdraw of funds from their expired client-torelay payment channel.

The RSC considers that the entire relay epoch is decomposed into *ceil*(B_R/k_R) *epoch slots* within which the number of release fund requests from a tagged payment relay *r* should never exceed the promised max throughput *M*[*r*] specified in its license. Every fund release request using the RSC method *releaseServerFunds*, shall trigger the following RSC-driven rate control mechanism. Assuming that the current block time is *b* and that the epoch has started on block b_0, the RSC shall check if the last recorded counter update in *lb*[*r*] belongs to the current epoch slot, by checking if Ceil(($b - b_0$)/k_R) (current epoch slot) is equal to Ceil((*lb*[*r*] − b_0)/k_R). If the condition is false, the RSC resets the transactions counter to one, *tc*[*r*] = 1, sets the last block registry *lb*[*r*] = *b* to the current block time *b* (to keep track of the last epoch slot where the counter update took place) and performs all computations necessary to deploy on-chain balance update of RE-CENT servers. If the condition is true, the RSC shall increase the counter *tc*[*r*] by one and check if it exceeds the max throughput *M*[*r*] specified by the relay license. If the *M*[*r*] promise is not exceeded, the RSC will perform all computations necessary to deploy the respective release funds operation. If exceeded, the RSC shall ignore the RSC call and reject the release fund operation. Rejecting a release fund call would come as a consequence of the inefficient (or malicious) operation of the payment relay, which has miscalculated the available number of on-chain balance updates it can perform within the current epoch slot.

Although the RSC-enforced rate control mechanism does not involve direct penalties to payment relays that violate their maximum throughput promise, it strongly discourages them from requesting an excess of on-chain balance updates due to the penalties that will follow from the impact of a failed released fund operation (e.g. delayed payments). The RSC-driven rate control mechanism is another measure to enforce payment relays to perform smart payment aggregation, always self-assessing their operation and keeping track of their interactions with the RSC to the minimum. Figs. 8, 9 and 10 provide illustrative examples of how parameters related to the transactions throughput rate control is performed.

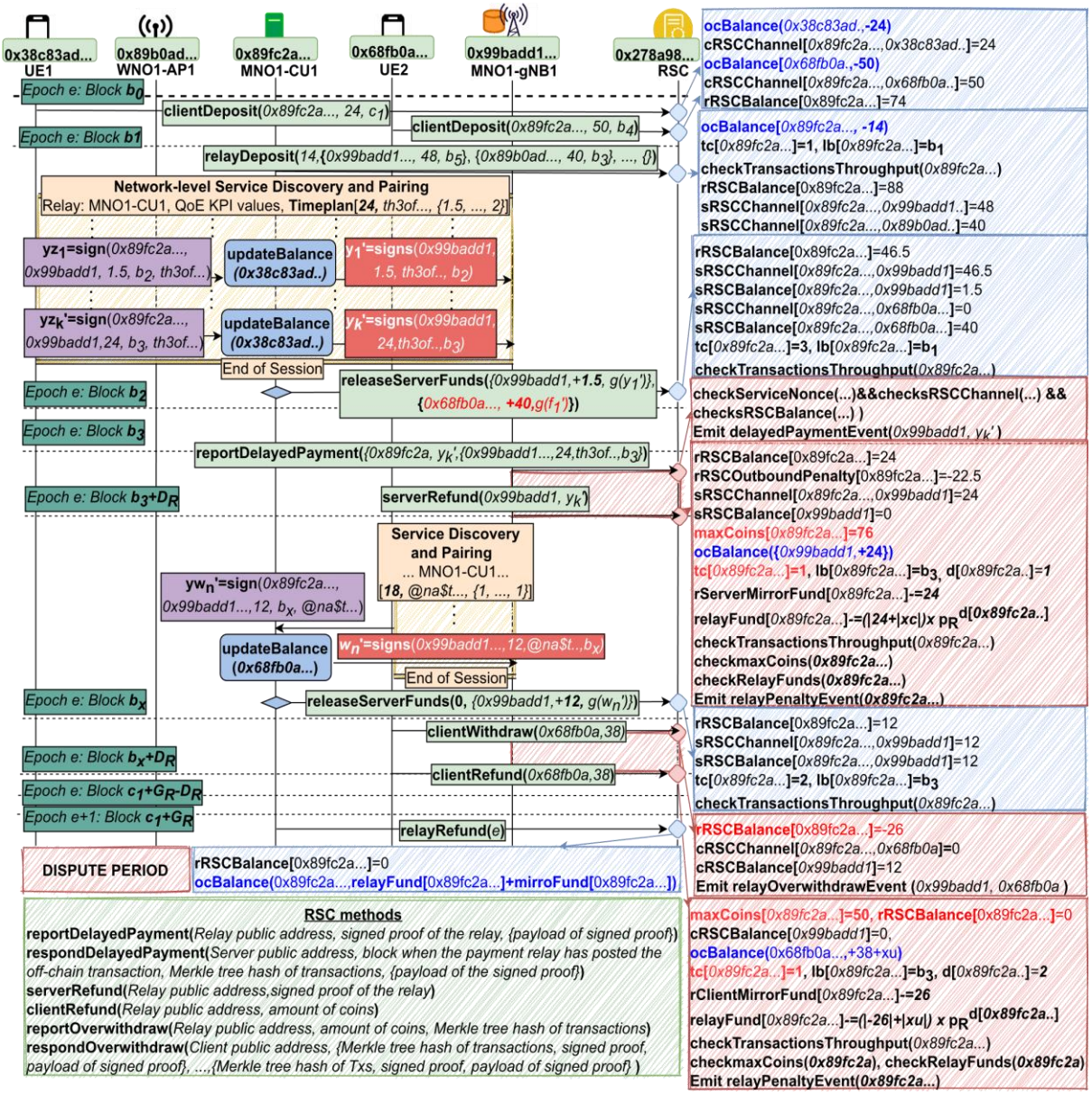

Figure 10: Payment relay service assuming unauthorized payments by dishonest relays

3.4.2.4 DISHONEST OPERATION OF RE-CENT CLIENTS

RE-CENT clients can proceed to the following dishonest actions. Firstly, a RE-CENT client may request to withdraw funds earlier than the expiration time of the inbound payment channel. Such a request should pass through the RSC, which shall ignore it and not allow earlier release of funds. Accordingly, RE-CENT clients are discouraged to perform such a dishonest action due to the on-chain costs paid in vain.

Secondly, a RE-CENT client may request to withdraw more funds than the original amount of coins available in the (expired) inbound payment channel. The RSC logic shall then trigger the on-chain dispute resolution mechanism described in section 3.3.5 for unauthorized payments and shall enable honest payment relays to submit all necessary proofs proving the rightful amount of coins that can be withdrawn by the RE-CENT client. Dishonest RE-CENT clients will not be reimbursed for the on-chain calls triggering the on-chain dispute, whereas they will be also required to reimburse the payment relay from their own inbound payment channel. Besides, dishonest RE-CENT clients requesting for the withdrawal of a higher amount of coins from the ones available in their inbound payment channel, should submit a targeted list of unauthorized transactions, increasing the on-chain cost and making it proportional to the one paid by honest relays for concluding the on-chain dispute.

In case honest payment relays fail to timely update the balance of RE-CENT clients (e.g. they go offline, or act inadvertently),the RSC will fully refund dishonest RE-CENT clients without taking into consideration future signed proofs. Besides, if signed proofs are submitted by payment relays after the expiration of the inbound payment channel, it is impossible to roll back the release of funds towards the RE-CENT client, given that a blockchain always considers the first spending of funds as the legitimate one (to avoid double-spending). In such occasions, the payment relay will be considered to make an unauthorized payment and trigger all the aforementioned mechanisms to itself. Thus, apart from being honest, payment relays should be also efficient and post the aggregate outcome of off-chain payments within the predefined lifetime of active payment channels.

Thirdly, the RSC logic shall completely ignore RE-CENT client requests reporting the withdrawal of a lower amount of coins (than the ones that they have issued) by the payment relays. If a payment relay withdraws a lower amount of coins compared to the one included in signed off-chain payments by RE-CENT clients, the payload of the signed message will not match the signature of the RE-CENT client. In such occasions, RE-CENT clients may trigger on-chain dispute resolution with the payment relay and get full refund on the amount specified in the original payload.

3.4.2.5 DISHONEST OPERATION OF RE-CENT SERVERS

On-chain dispute resolution on non-expired off-chain transactions will be ignored by the RSC, discouraging dishonest RE-CENT server operation of this type (due to the on-chain costs paid to make the RSC call). Dishonest RE-CENT servers triggering on-chain dispute resolution for off-chain transactions that have been posted by the payment relay shall trigger honest payment relays to provide necessary proofs of their operation to the RSC. Accordingly, the RSC shall verify the honest operation of the payment relay and ignore the dishonest RE-CENT server request. Even though such a dishonest operation of RE-CENT servers will enforce honest payment relays to pay unnecessary on-chain costs, we consider that RE-CENT servers are strongly discouraged to act in this fashion, due to the on-chain costs that will be required to pay for an on-chain dispute that will be concluded against them. Besides, even if a high cost is considered for triggering the RSC logic, honest RE-CENT servers shall be fully refunded by the penalty fund of dishonest relays.

3.4.2.6 DISHONEST (OR INADVERTENT) OPERATION OF PAYMENT RELAYS

The transactions throughput monitoring mechanism ensures that payment relays never violate their promised maximum throughput value due to i) the penalty mechanisms for delayed payment events and ii) the on-chain costs required to re-transmit a release fund request that has been ignored by the RSC. Since the establishment of payment channels is implemented on-request to the RSC, the RSC logic will always verify that current amount of RE-CENT clients and coins attach to a tagged payment relay are line with its license parameters. The RSC logic will also reject requests on the establishment of output payment channels that violate the inbound/outbound channel balance. To achieve this, upon an outbound channel establishment request, the RSC logic will always ensure that the expiration block declared for the new outbound payment channel is aligned with that of inbound payment channels, taking into consideration the inbound/outbound payment channel capacity status.

When payment relays over-withdraw funds from inbound payment channels and post the respective over-withdraw claim to the RSC, RE-CENT clients shall notice the claim and shall trigger the on-chain dispute resolution logic described above. RE-CENT clients that fail to notice the over-withdraw claim within the expiration of the inbound payment channel, will request for their legitimate amount of coins to be unlocked from the RSC when the channel expires. At some point, the RSC shall calculate a negative balance for the inbound payment channels of dishonest relays and trigger the mechanisms for unauthorized payments. Honest RE-CENT clients will be fully reimbursed but dishonest payment relays will lose part of their

client mirror and guarantee funds. The same logic will be deployed if payment relays over-withdraw funds from inbound payment channels.

Dishonest payment relays may ignore the service requests of RE-CENT clients with an active inbound payment channel; however, such an operation is against the interest of payment relays that will loose off-chain transactions fees. Even if they choose to follow this approach due to the employment of unauthorized transfers of funds, their dishonest operation will be soon revealed. Payment relays should optimize their decision mechanism for allocating funds in outbound payment channels by giving an efficient answer to the trade-off between minimizing the number of outbound payment channel requests versus having sufficient balance to support all RE-CENT client requests. Besides, RE-CENT clients that are not satisfied by the services offered by payment relays can withdraw their full amount of funds due by the expiration block of the inbound payment channel and avoid consuming services from the respective payment relay.

Payment relays that fail (or refuse) to post the outcome of off-chain payments to RE-CENT servers within the promised relay delay threshold will experience penalties on their server mirror and guarantee fund according to the RSC-driven penalty mechanism specified for delayed payments. Provided that the RSC logic will always align the maximum allowed outbound payment channel balance with the amounts of coins remaining at the server mirror fund, the RE-CENT servers will be fully reimbursed for their services even under the worst case scenario where payment relays completely fail to submit the outcome of all off-chain transactions to the RSC (which can never exceed the maximum number of attached coins specified in the relay license). Payment relays that fail (or refuse) to post the outcome of off-chain payments issued by RE-CENT clients within the lifetime of the respective inbound payment channels, will not be unable to handle the full amount of coins attached to them, reducing thus their outbound channel establishment capability and loosing the opportunity to process more off-chain transactions. Further to this, they will also receive a proportional penalty to their client mirror fund by the RSC, which will always refund RE-CENT client requests according to its current view of the inbound payment channel balance.

3.4.2.7 CALCULATION OF THE MEAN TRANSACTION THROUGHPUT TC_R

Parameter TC_R defines a stopping criterion that enables the RE-CENT nodes to deterministically conclude on the candidate payment relays elected for a target epoch e. This parameter can be adjusted only by the RSC and for epoch $e + 1$ can be calculated based on i) the transaction throughput requested during the relay election epoch e and ii) the counters measuring delayed payments by the end of epoch e. The last counters not only capture

malicious behaviors of payment relays but also depict delayed payments due to the deployment of the relay throughput monitoring mechanism. In our implementation, we have considered that the RSC logic reduces the TC_R parameter by the number of delayed payments recorded during the previous target epoch and if no delayed payments are recorded, it increases TC_R by 10 %.

3.5 ANONYMOUS PAYMENTS

The primary goal of mixing services is to enable a payer *A* to transfer a certain amount of coins *Q* from its public address to the public address of a payee *B*, while making impossible for other blockchain nodes to link the transfer of coins between the two public addresses. The so-called *unlikability* property is critical to preserve user privacy and anonymity in blockchain systems also in view of the RE-CENT mobile content delivery service, where the physical proximity between the service peers and the fact that mobile video is delivered over-the-air enables adversaries to i) link public addresses of the peers to their physical network identifiers (potentially also physical world identities) and ii) assess the critical parameters of the service that has been consumed/delivered (e.g. resolution, preferences, mobility). The aforementioned network/blockchain ID coupling problem has been thoroughly discussed in sections I-C and II-C.

By design, the RE-CENT payment relay service provides a certain level of unlikability between the RE-CENT client/server payments, due to the aggregation of off-chain transactions and the atomic balance update performed on a per RE-CENT client, or server basis. Nonetheless, payment relays still have full information on the payments implemented between two tagged service peers, while third-party adversaries may also link the service pair if the payment relay performs atomic balance updates with the same amount of coins *Q*. Besides, in case of an on-chain dispute, the payment relay server will be required to post both the signatures provided by RE-CENT servers and the respective payloads which, in the baseline scenario of section 3.3, shall enable adversaries to identify the time and amount of delivered mobile video content service. RE-CENT clients can always use fresh public addresses and establish a proportional number of inbound payment channels to the same payment relay service. Similarly, RE-CENT servers can always use fresh public addresses and request for additional outbound payment channels from the same payment relay. Nonetheless, both approaches would further increase the transactions capacity requirements for the RE-CENT blockchain system.

Aiming to efficiently resolve the network/blockchain ID coupling problem and provide cost-efficient anonymity solutions to the RE-CENT blockchain system, both in terms of attaining a low transactions throughput and minimizing the on-chain costs paid for establishing new

payment channels, to the remainder of this section we describe a coin mixing service that builds on-top of the payment relay service. The proposed protocol exploits the puzzle-promise / puzzle solution protocols of [29] and extends their operation in the RE-CENT context. The main idea of the protocol is to exploit the existing inbound and outbound payment channels established by the payment relay service to enable unlikable off-chain payments using blind signatures and fair-exchange of joint network/blockchain-level resources.

The puzzle-promise, the puzzle-solution and the issuing of instant off-chain payments are implemented offline in a centralized fashion by the payment relay servers (i.e. the mixing servers); however, the payment relay servers are enforced to act honestly and never abort the off-chain mixing payment service due to the penalties invoked to them by the RSC logic. Accordingly, the proposed mixing service employs *hybrid mixing*, i.e. centralized mixing implementation together with RSC-driven decentralized control, a concept which, to the best of our knowledge, is firstly discussed in this paper. The remainder of this section is organized as follows. Section 3.4.1 overviews all phases and building blocks necessary to implement the RE-CENT coin mixing service. Section 3.4.2 details the puzzle-promise and puzzle-solution protocols employed by the RE-CENT mixing service. The discussion on the respective protocols is accompanied by a detailed run time examples enabling in-depth understanding of the fundamental building blocks and steps of the proposed hybrid mixing service. Section 3.4.3 investigates how the RSC logic can mitigate dishonest operation of mixing parties and summarizes the modifications necessary to the RSC logic.

3.5.1 RE-CENT COIN MIXING PROTOCOL OVERVIEW

Let us assume that payer *A* and payee *B* utilize the joint mixing and payment relay services of *T* to transfer *Q* coins. The baseline payment relay service requires *A* to establish an inbound payment channel to relay *T* of balance $Y_A \geq Q$ and *T* to establish an outbound payment channel to the payer *B* of balance $Y_B \geq Q$. In the sequel, we consider that the amount of transferred coins *Q* is a denomination of an arbitrary fixed unit *u*, to avoid matching the amount indicated in the coin transfer from *A* and *B* to the puzzles promised by the mixing server *T*. Parameter *u* along with the mixing fee necessary for utilizing the mixing service are assumed to be fixed and specified during the relay election epoch in the RSC.

The *Escrow phase* is implemented with the establishment of the inbound payment channel from *A* to *T* and the outbound payment channel from *B* to *T*. A key requirement for enabling seamless mixing services is that during the entire RE-CENT video delivery service the payer *A* will have an active payment channel to *T* and that the payment relay *T* will have an active payment channel to *B*.

The *Payment phase* can take place any time in between the lifetime of both payment channels and can be performed asynchronously between the (*A,T*) and (*T,B*) service pairs. Within this time, the payer *A* and the payee *B* will perform network-level service discovery, pairing and negotiation to conclude on a payment timeplan of intermediate micro-payments. Different from section 3.3, service peers shall not communicate their agreed timeplan of payments to the mixing server (payment relay). Such information would enable the mixing server to link the timing of puzzle solution requests of *A* to the timing of payment requests from *B*.

Payee *B* (RE-CENT server) can employ the puzzle-promise protocol with the tagged mixing server *T* and acquire an arbitrary number of Y_B puzzles (payment promises) each releasing an amount of *u* coins. To achieve this, we extend the Tumblebit puzzle-promise protocol ([29] - Appendix A) as follows. Let z_q denote puzzle q, where $q = 1,...,Y_B$. Each puzzle z_q is a cryptographically locked off-chain payment signed by *T*, promising the transfer of *u* coins to the public address of *B* due by block b_i. In our protocol, i) *B* can open the cryptographically locked promise iff it receives the puzzle solution of z_q by *A*, ii) *B* can open the cryptographically locked promise z_q if it has already opened all previous cryptographically locked promises $z_1,...,z_{q-1}$ ($q \geq 2$), iii) cryptographically locked promises z_q specify a relay delay b_q that should be a-priori specified by the payee *B* during the puzzle-promise protocol with *T*, and iv) each promise z_q is a payment relay promise including the relay public address, amount of transferred coins $q \cdot u$, unique service nonce and relay delay block b_q.

Recall that the puzzle-promise protocol can be performed any time within the lifetime of the outbound payment channel of *B* (in *T*) and not necessarily in view of a service with a tagged RE-CENT client. However, the relay delay blocks $\mathrm{B} = \{b_1,...,b_{YB}\}$ should be specified a-priori by RE-CENT servers for the entire balance Y_B available to the outbound payment channel, un-pairing future off-chain payments to the RE-CENT server with the actual service delivery towards a specific RE-CENT client. This requirement follows from the fact that cryptographically locked promises z_q are linked together by the Tumblebit puzzle-promise protocol and cannot be issued separately by the mixing server *T*.

According to the payment timeplan agreed between the RE-CENT service peers *A* and *B* (using network-level interactions), *B* shall deliver video content chunks to *A* and request from *A* to solve the respective number of blinded puzzles $\{z^0_q\}$ to continue the video delivery service. Payee *B* shall use RSA to blind the original puzzles z_q before sending them to *A*, making impossible for *T* to link the coin transfer from *A* to *B*. This is possible due to RSA blinding, which enables a solution of the blinded puzzle to be readily used for solving the original puzzle by *B* [29]. Payer *A* (RE-CENT client) and mixing server *T* have interest in solving the puzzle

to continue the video service and receive off-chain payment fees, respectively. *B* shall enable *A* to identify the source of puzzle z_q by providing necessary network and blockchain level identifiers of the mixing server to *A*, i.e. IP and public address. Payee *B* can utilize different mixing servers for the same video service, as soon as *A* has an active payment channel with the respective mixing servers (payment relays). Using the blinded version of puzzles $\{z^0_q\}$ as an input, payer *A* shall interact with the mixing server *T* to perform a fair exchange of puzzle solutions and coins in a completely offchain fashion ([29] - section V.D). Key enabler for this fair-exchange is the use of T_{puzzle} and T_{solve} messages in a similar manner with [29]; However, different from [29], both types of transactions are processed and posted by the RSC logic and not the RE-CENT blockchain itself, whereas an additional mechanism is necessary to mitigated is honest operation of mixing servers that do not timely submit the T_{solve} transaction but post its outcome to the RSC (section 3.4.3).

T_{puzzle} messages are signed by the RE-CENT client *A* and enable the mixing server *T* to withdraw the specified amount of attached coins only under some condition *C*. Accordingly, the mixing server *T* is required to post a new transaction T_{solve} that refers to the signed transaction T_{puzzle} and meets the condition *C*. More details on the puzzle-solution protocol are provided in section 3.4.2.

Having derived the solutions of the blinded puzzles $\{z^0_q\}$ from *T*, payer *A* communicates them to payee *B*. *B* unblinds the solutions provided by *A*, unlocks the cryptographically locked off-chain payment promises of *T* and continues the video service. Different from [29], we consider that *B* does not post the unlocked signed promise of *T* on-chain but instead, it forwards the unlocked signed promise to the mixing server *T*. In this fashion, *T* becomes aware of its obligation to post on-chain the outcome of the respective transaction and is also enabled to aggregate its outcome with other off-chain payments pending for the corresponding outbound payment channel. Both the payee *B* and the mixing server *T* have strong interest in conforming with such an action due to the on-chain costs necessary for direct on-chain post of the respective transaction and the increased transactions fees that will be acquired, respectively. Dishonest operation of mixing parties is investigated in section 3.4.3.

3.5.2 PUZZLE-PROMISE AND PUZZLE-SOLUTION PROTOCOLS

To enable deeper understanding of the proposed hybrid mixing service, in Fig. 11 we provide an illustrative instance depicting the actions performed by all mixing parties. The service starts with the Escrow phase, where inbound and outbound payment channels are established by payee *A* (UE1) and mixing server *T* (MNO1-CU1), respectively. Founded on the outgoing payment channel established by *T* (MNO1-CU1) of balance $Q \cdot u$, payee *B* executes the

puzzle-promise protocol with *T* (MNO1-CU1) off-chain and derives cryptographically locked (and chained together) puzzles promising *Q* individual off-chain payments of size *u*.

3.5.2.1 PUZZLE-PROMISE PROTOCOL

The puzzle promise protocol starts with the payee *B* (MNO1-gNB1) defining a relay delay block timeplan $B = b_1,...,b_Q$ per off-chain payment promise z_q (step 0 performed by MNO1-CU1). Each promise z_q ($q = 1,...,Q$) contains the mixing server's ECDSA signature on a mixing payment transaction (signs2) that instructs the RSC to release $q \cdot u$ coins from the outbound payment channel *T->B* due by block b_q. *B* creates *Q* sets of μ distinct real transactions and η fake transactions, using random pads $\rho_{q,n} \in \{0,1\}^\lambda$ and $r_{q,n} \in \{0,1\}^\lambda$. A real transaction $\psi_q^0{}_{,\mu}$ is derived using the mixing payment format *signs2*, which includes i) the public address of the mixing server, ii) the amount of coins to be transferred, iii) the random pad $\rho_{q,n}$ and iv) the relay delay block promise b_q. A fake transaction q,η is derived using a *fakeFormat* that is a-priori known to both *T* and *B*, using as additional randomness the fake random pad $r_{q,\eta}$. The use of multiple real and fake transactions for a given payment of amount qu coins ensures honest operation of *B* and *T* with probability $1 - \frac{1}{\binom{\mu+\eta}{\mu}}$, by employing *cut-and-choose* methods [29].

Real and fake transactions (steps s1 and s2 in Fig. 11) are hidden under the hashing function $H^0(x)$ (e.g. SHA-256) and their hashed output is shuffled to obtain *Q* sets of (mixed) transactions hashes (step s3) that we denote by $\{\{\theta_{1,1},...,\theta_{1,\mu}\},...,\{\theta_{Q,1},...,\theta_{Q,\mu}\}\}$. The hash h_R of all real transactions hashes in set *R* as well as the hash h_F of all fake transactions hashes in set *F* are also derived. The two values are hidden with the addition of a random 'salt' value $s \in \{0,1\}^\lambda$ that is kept secret by payee *B* (step s4). This step ties together real and fake hashes, making it impossible for the payee *B* to forge them in subsequent steps of the protocol.

Using off-chain network-level messages, payee *B* communicates to *T* the set of shuffled hashes $\{\theta_{q,m}\}$ along with the hash of real and fake transactions hashes h_R and h_F, respectively (step s5). The mixing server *T* (MNO1-CU1) subsequently verifies that the number of transactions matches the balance of the outbound payment channel established by *T* and uses its secret key SK_T to sign each hash $\theta_{q,m}$ (step s6). To hide its signed promises, the mixing server *T* selects a random number $_{q,m}$ for every hashed transaction it receives (step s7), where $q \in \{1,...,Q\}$ and $m \in \mu + \eta$. It further uses this as a secret key to create a proportional amount of RSA puzzles $z_{q,k}$ and promises $c_{q,k}$, which shall enable *B* to unlock the signature $\sigma_{q,k}$ of *T* in the future (using the puzzle solution to $z_{q,k}$ provided by *A*).

The mixing server *T* (MNO1-CU1) subsequently communicates the promises $c_{q,m}$ to payee *B* (MNO1-gNB1) using network-level messages (step s10) and payee *B* reveals to *T* i) the set of real and fake transactions hashes *R* and *F*, and ii) the random pads used to derive them ($\rho_{q,i}$ and $r_{q,j}$) (steps s11 and s12). All communications are performed by employing off-chain network-level messages. The mixing server *T* (MNO1-CU1) is now enabled to validate the hash of hashes of real and fake transactions (and thus verify that they have not been changed by *B*) and verify that the payee *B* has acted honestly during the preparation of real and fake signatures (steps 13, 14, 14a and 14b). Accordingly, it sends to *B* the secrets $_{q,j}$ only for the fake hashes in *F*.

Payee *B* (MNO1-gNB1) can now verify that the mixing server *T* has acted honestly when signing all transactions hashes (real and fake), by verifying that i) the puzzles of fake transactions hashes can decrypted using the secrets $_{q,j}$ for $j \in F$ (step 17a) and ii) the respective ECDSA signatures of *T* are valid (PK_T is the public key of *T* in step 17b). Steps 16-17b conclude the cut-and-choose method, which has by now enabled both parties to verify that the other party acted honestly during the puzzle preparation phase.

The protocol continues to ensure that i) if at least one of the real ($c_{q,i}$,$z_{q,i}$) pairs opens to a valid ECDSA signature $\sigma_{q,i}$ of the mixing server *T*, then just one solution $_i$ with $i \in R$ can be used to open this pair and ii) the solutions of all puzzles $z_1,...,z_j$ should be used to open the j^{th} promise c_j. To meet the first requirement, the mixing server 'chains' together all μ puzzles derived for the single payment of $q \cdot u$ coins by correlating the secrets $_{q,j}$ for $j = 1,..,M$ using a simple quotients technique. To meet the second requirement, we also need to correlate the quotients of the *Q*-th solution to the quotients of the $(Q-1)$-th solution, moving step-by-step recursively to the 1-st solution; thus, running *Q* levels of quotients in parallel. We omit this procedure as it is similar to that described in Appendix A of [29]. *T* calculates the respective quotients and communicates them to *B* (step s19) using network-level messages. The puzzle-promise protocol concludes with payee *B* checking the validity of quotients and aborting the protocol if any check fails (step s20).

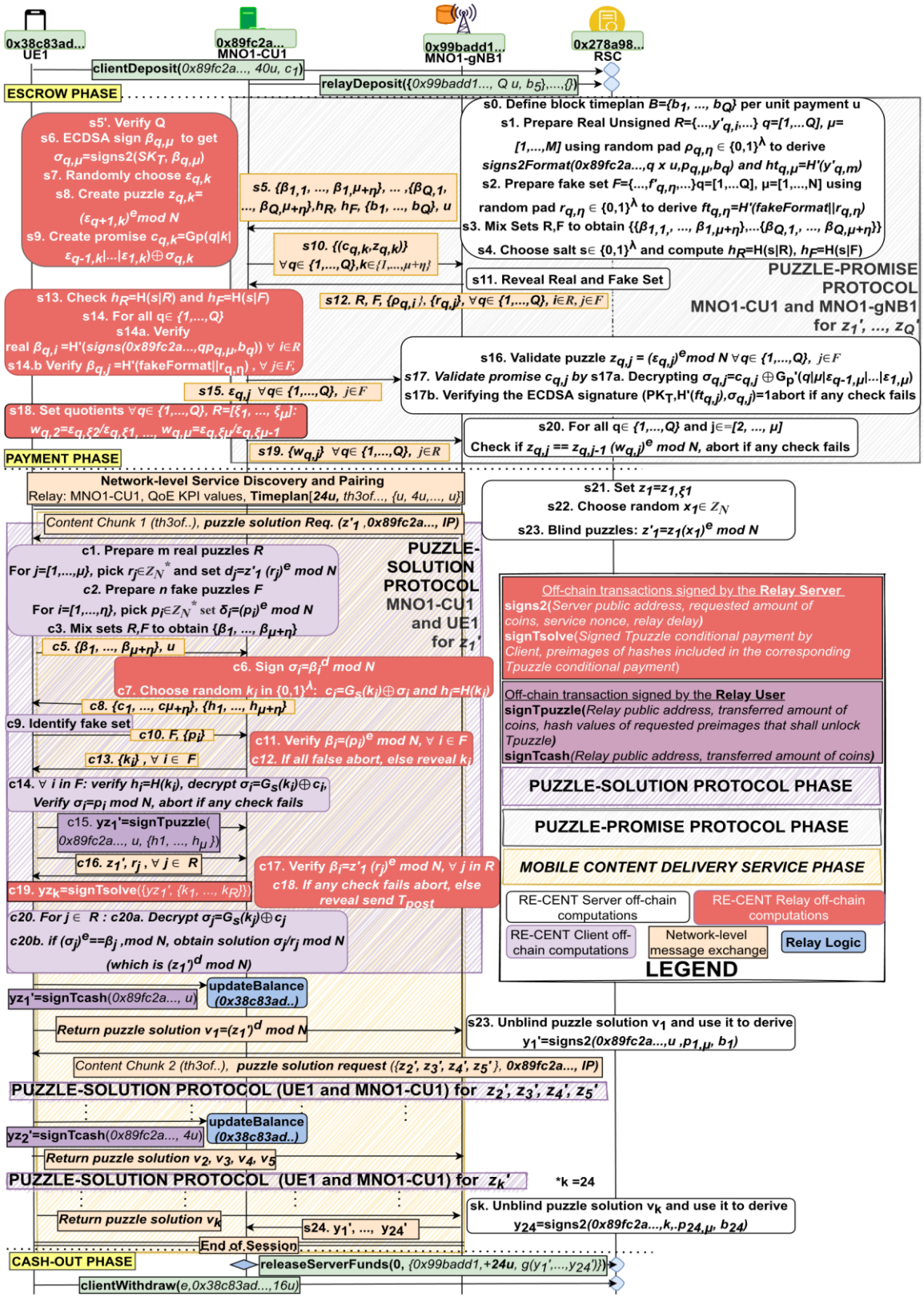

Figure 11: The RE-CENT mixing service over payment relays

3.5.2.2 PUZZLE-SOLUTION PROTOCOL

We now focus on the procedure followed by *A* and *T* to solve the puzzles provided by *B* and to release the respective amount of funds from the inbound payment channel of *A*. Once again, the puzzle-solution protocol uses cut-and-choose to validate honest operation of the mixing server *T* and the payer *A*. Using as an input a tagged blinded puzzle z^0_1, which is provided to *A* by *B* using off-chain network-level messages, payer *A* (UE1) prepares a number μ of real puzzles *R* and a number η of fake puzzles *F* (steps c1-c2). Accordingly, it mixes real and fake puzzles (step c3), and sends them to the mixing server *T* (MNO1-CU1) using off-chain networklevel messages. *T* subsequently signs the received puzzles and uses a proportional number of random secrets k_i to issue cryptographicallylockedpromisesthatsolveboththerealand fake variants of the original puzzle $z^0$1 (steps c6). The secret keys k_i of real puzzles θ_j ($j \in R$), termed as preimages in the sequel, shall enable *A* to obtain the solution to the puzzle z^0_i.

The mixing server *T* hashes the respective secret keys to $h_i = H(k_i)$ (step c7) and sends their hashes together with the derived puzzle solution promises c_i to *A* using network-level messages (step c8). Accordingly, payer *A* (UE1) identifies the fake set *F* (step c9) and sends it together with the random pads ρ_i to *T* (step c10). *T* can now verify that the subset of puzzles submitted by *A* was fake and that *A* has acted honestly during the preparation of puzzles θ_j, $j \in 1,...,\mu + \eta$ (step c11-c12). To allow payer *A* validate honest operation of

T, *T* also sends the preimages k_i used for signing fake puzzles (using its original secret key *d* in step c13). *A* verifies that the respective keys can decrypt the fake puzzle solutions c_i with $i \in F$ and that c_i unlock the signature of σ_i given the random pads p_i of fake puzzles (step c14).

If everything is in place, payer *A* subsequently signs and sends a T_{puzzle} transaction to *T* (step c15), promising to release *u* coins to *T*'s public address if the mixing server provides the corresponding T_{solve} signature revealing the preimages k_i (which were used to cryptographically lock the real solutions of the original puzzle z^0_1 submitted by *A*). At the same time, *A* sends to *T* the original puzzle z^0_1 and the random pads r_j ($j \in R$) (step c16). This piece of information enables *T* to verify that all puzzles unlock the solution to the same puzzle z^0_1 (steps c17-c18). In this step, the mixing server is assured that *A* cannot cheat by acquiring solutions to more puzzles (due to the large number of real puzzles and preimages k_i). Accordingly, *T* issues a T_{solve} transaction (step c19) with reference to the signed T_{puzzle} transaction issued by *A*,

including in its payload all preimages k_i with $i \in R$, which shall enable *A* to obtain the solution to the requested puzzle z^0_1 (steps c20, c20a and c20b).

Note that the off-chain transactions T_{puzzle} and T_{solve} shall be used by the mixing server (payment relay) *T* to prove that it was authorized by payer *A* (RE-CENT client) to withdraw the respective amount of *u* coins from its inbound payment channel. The aforementioned procedure is followed for every new puzzle solution request sent by *B* and can be parallelized if multiple puzzles are sent from *B* to *A*. We point out that the

T_{puzzle} transaction is a conditional authorization from *A* to *T* to withdraw funds under the condition that *T* communicates to *A* (or posts on-chain) the preimages k_i used to derive real puzzles together with T_{solve}.

3.5.2.3 MIXING PROTOCOL TERMINATION

After the completion of the aforementioned puzzle-solution and puzzle-promise protocols, the RE-CENT client has unlocked a set of payment promises $y^0_1,...,y^0_{24}$ that are yet to be implemented by the mixing server (and payment relay) *T*. However, since the mixing server *T* is unaware of which payment promises have been unlocked (due to the use of blinded puzzles by *B* and *A*), protocol termination necessitates the RE-CENT server to communicate to the payment relay *T* the signed promises it has unlocked (step s24). During this step,theRE-CENTservershouldalwaysaccountfortherelay delay block b_q due by which unlocked promises expire.

3.5.3 RSC-ENFORCED MITIGATION OF DISHONEST OPERATION

3.5.3.1 MIXING SERVER DOES NOT POST THE PROMISE ISSUED TO THE PAYEE

Inadvertent (or malicious) operation of the RE-CENT server, or the payment relay, at the final steps of the mixing service is possible. Even though the payment relay signs locked promises to the payees, it is completely unaware on whether these promises will be finally unlocked and should be thus implemented. As a result, mixing payees like *B* (RE-CENT servers) may act dishonestly by not communicating to the payment relay *T* the unlocked promises while claiming the respective refund after the relay delay block expires. This would trigger penalties to the mixing server signing the promise due to the on-chain dispute mechanism provisioned to the RSC for delayed payments. On the other hand, even if the mixing payee *B* (RE-CENT

server) notifies *T* on its obligation to implement the unlocked promise on-chain, the mixing server may act inadvertently (or maliciously), skipping the respective payment. In both scenarios, the RSC will award the payment relay penalties due to delayed payments. To mitigate such events, we exploit the different format of signed promises issued by mixing relays, i.e. they do not include the RE-CENT clients public address, a service nonce and a sequence number, and modify the RSC logic to deploy a different flow and mixture of penalties for on-chain dispute resolution on *delayed mixing payments*.

Let $q \cdot u$ denote the promised amount of coins specified on the disputed mixing payment between the payee *B* and the mixing server *T*. *B* is enabled to trigger on-chain dispute resolution for a delayed mixing payment by posting to the RSC all necessary proofs of the delayed relay payment as in section 3.3.5. The RSC shall verify the signature of *T* and perform the following actions. If the relay delay included in the mixing payment promise has not yet expired, it shall ignore the payees' request for on-chain dispute resolution. If the payee employs such an action, it will not receive reimbursement on the on-chain costs paid to trigger the RSC. If the mixing payment promise has expired, one of the two parties has acted dishonestly and an on-chain dispute on delayed mixing payment will be triggered. If the reported payment relay *T* (and mixing server) *T* submits all proofs necessary to prove its honest operation up to D_R blocks from the beginning of the on-chain dispute, the RSC shall ignore the payee's request and will not reimburse it for the on-chain costs paid to trigger the dispute.

However, if *T* does not provide proofs of its honest operation during the dispute interval D_R, the RSC will employ the following logic to incur penalties to both mixing parties. Payee *B* shall be refunded for only half of the on-chain costs paidfortriggeringtheon-chaindisputeandhalfoftheamount promised by the reported payment relay, i.e. $q \cdot u/2$. On the other hand, the RSC shall: i) reduce by $q \cdot u/2$ coins the server mirror fund of the reported payment relay, ii) subtract half of the on-chain costs paid by the payee to trigger the dispute by the penalty fund of the reported payment relay, and iii) subtract $q \cdot u \cdot fee[r]/2$ from the penalty fund of *T*.

The aforementioned penalty mixture shares the penalties on the on-chain costs and disputed amount of coins, also reducing by 50% the transactions fees received by *T* for its mixing services. Recall that off-chain transactions fees for payment relay services are always paid by the RE-CENT clients and, since the payment relay holds the signed promise issued by the

RE-CENT client (puzzle-solution protocol), it always receives the corresponding off-chain payment. Honest payees will be incentivized to always report dishonest mixing servers iff the gain of reporting the dishonest server is positive.Thus,thesizeofunit*u*shouldbeatleasttwotimeshigher than the cost of triggering the on-chain dispute resolution for delayed mixing payments with RSC.

3.5.3.2 DISHONEST OPERATION OF MIXING SERVERS IN THE PUZZLE-SOLUTION PROTOCOL

Let us now investigate the scenario where *T* refuses to share T_{solve} off-chain with *A* but chooses to charge *A* by releasing the respective amount of funds *u* from the corresponding inbound payment channel of *A*. Then, *A* shall identify that an unauthorized payment of *u* coins has been charged to its inbound payment channel by *T* and trigger the on-chain dispute resolution mechanism for unauthorized payments described in section 3.3.5. Accordingly, within the block interval of D_R blocks, the mixing server will be required to submit the corresponding T_{puzzle} and T_{solve} transactions to the RSC, to prove its honest operation. The on-chain dispute on the unauthorized payment will conclude in favor of the payment relay but the preimages k_i will also appear on-chain for the first time (due to the submission of the corresponding T_{solve} transaction). Note that by this point, the RE-CENT mixing service attains the same properties with Tumblebit, enabling the mixing payee to acquire the solution to the puzzle that it has paid for.

However, the disregarded payer *A* shall use the publicly available k_i images posted by the mixing server after the dispute resolution and employ a fair exchange protocol with the payee *B* in order to acquire the original relay's payment promise to *B* unlocked by puzzle z^0_1 (without revealing the preimages yet). Payer *A* is motivated to participate in this protocol to derive the original payment unlocked with the available preimages and get full reimbursement from the RSC for all of costs following from the dishonest operation of the mixing server. Payee *B* is motivated to participate in this protocol to derive the solution to the original puzzle z^0_1 and thus, the respective payment promise by *T*. If the payment has not expired, the payee *B* will send the unlocked payment promise to the mixing server *T*, which will have to post it in order to avoid penalties on delayed mixing payments. However, if the payment promise has expired, payee *B* shall trigger an on-chain dispute against the respective payment relay, triggering the RSC for delayed mixing payments.

In parallel, Payer *A* shall trigger a new on-chain dispute for *delayed* T_{puzzle} *transactions*, by posting to the RSC i) the public address of the dishonest mixing server and ii) a reference to the transactions where *T* has posted the T_{puzzle} transaction. This type of on-chain dispute will remain open by the end of the relay epoch and conclude only if the payer *A* submits to the RSC a reference to the Merkle tree hash including the original transaction (unlocked with k_i). To this end, payer *A* will keep track of the RE-CENT blockchain to find aggregate payments issued by *T* towards *B*, or on-chain disputes for delayed payments triggered by *B* to *T*, enabling it to use the original transaction and verify the payment relay transaction was posted to the RSC. This can be done by exploiting the Merkle tree hash of each transaction issued by *T* to *B*, or spotting the original transaction due by an on-chain dispute resolution between *B* and *T* for delayed payments. Accordingly,payer*A*shalltriggertheRSCandprovideitwith a reference to the Merkle tree hash that includes the original transaction (unlocked with k_i).

Since *T* is aware of only the blinded version of the respective puzzle, it will impossible to post this transaction before posting T_{solve}. Thus, the RSC will validate that the mixing server *T* has acted dishonestly and fully refund the disregarded payer *A* from the penalty fund of the dishonest relay by reimbursing it for i) the on-chain costs paid for triggering two on-chain disputes, ii) the cost paid to *B* and iii) mixing fee paid to the mixing server *T*, subtracting funds from the remaining penalty fund of the dishonest mixing server (and payment relay) *T*. Using this mechanism, mixing servers can link the corresponding payment made from payer *A* to payee *B*. However, such an action will incur a very high cost (if not license revoking) due to the increased penalties received by

i) the on-chain dispute resolution triggered by the payer *A* for delayed T_{solve} transactions and ii) the on-chain dispute resolution triggered by payee *B* for delayed payments (if the unlocked transaction has expired). The latter event shall happen with a high probability due to the time necessary for resolving on-chain disputes.

3.5.3.3 RSC MODIFICATIONS ENABLING THE PROPOSED MIXING SERVICE

Firstly, payment relays that also provide mixing services for the target relay epoch *e* shall further specify a transactions fee for mixing services. Secondly, the RSC logic should support thenewon-chaindisputeresolutionmethodimplementingthe penalty mechanism for delayed mixing payments. Thirdly, the RSC logic should support the new on-chain dispute resolution method implementing the penalty mechanism for delayed T_{solve} transactions.

4. NUMERICAL RESULTS

In this section, we provide a preliminary study on the performance of fully-distributed blockchain-backed service charging for contract-less mobile video delivery. We choose not to focus on detailed regional network setups but simulate a full-scale worldwide service coverage scenario where a single public ledger is considered to allow worldwide roaming. Besides, blockchain scalability in smaller network setup is straightforward. To this end, we assess the transactions throughput (TPS) of blockchain-backed payments without assuming previously established subscription agreements.

Note that emulating the full protocol stack and network topology of universal fully-distributed blockchain-backed mobile video delivery assuming a large-scale heterogeneous wireless network with billions of UEs and millions of network service points (e.g. base stations, UE network relays, Wi-Fi access points) is impossible with current network simulation tools. Moreover, many different parameters affect the actual performance of the worldwide network of wireless networked systems and service domains, including network density per RAT in specific areas, user mobility patterns, service tariffs and user preferences.

In view of that, with the subsequent simulation campaigns we aim to provide a feasibility study and preliminary assess whether the universal, fully-distributed and blockchain-backed mobile data access is scalable and robust. Accordingly, in specific parts of our simulation model we make simplifying assumptions which, however, enables us to assess the massiveness of blockchain-backed service at a worldwide scale. All simulation models and parameter values under scope are in full compliance with current literature (e.g. video time distribution, session arrival rates per server), drawn from recent reports and studies on how mobile video content delivery will be shaped in future mobile data networks [1], [18], [19], [77]–[81]. Thus, our high-level simulation model accurately adapts its scale, structure and overall behavior to the current and future state-of-play in heterogeneous wireless networking. All models and results have been developed in MATLAB R2019b, which is one of the few tools enabling explicit modeling and tracking of the billions peers and sessions of such a massive large-scale system-level simulation.

4.1 SIMULATION MODEL OVERVIEW

4.1.1 SIMULATION MODEL

We consider a massive in scale multi-tier heterogeneous mobile data network of U UEs and S service nodes (servers). We only focus on UEs that consume mobile video data and utilize a single public address for implementing blockchain-backed payments to this end. Network servers are considered capable of meeting the minimum requirements set for mobile video content delivery and, depending on the simulation scenario under scope, we either consider that each server uses a unique public address for receiving blockchain-backed payments from UEs (*All Servers Scenario*), or assume that each server is attached to a service domain utilizing a common (single) public address to this end (*Cellular MNO scenario*) as in section 3.1. A fixed percentage r (%) of UEs is additionally considered capable of acting as servers, playing the role of *network relays* that provide other UEs with connectivity to the Internet, or locally cached mobile video content. Depending on the scenario, the video-enabled servers are cellular base stations, Wi-Fi access points and UE network relays, reaching to a total of $S + r \cdot U$ (network) servers in the system. Each client UE $u \in \{1,...,U\}$ is assumed to be in coverage of u_s servers that support on-the-fly mobile video content delivery in a contract-less fashion using blockchain-backed payments. In practice, the set of accessible servers per user varies over time depending on the actual network topology and the UE mobility pattern. For tractability, we consider that the number of accessible network servers per UE is normally-distributed with mean s^- and variance $s^-/10$. Using this setup, each UE is assigned with a fixed set of accessible servers (from the total of $S + r \cdot U$ servers). For each new video session, the UE randomly selects one of their accessible servers to implement the blockchain-backed mobile video content delivery service.

On a per second basis, our simulation model randomly (uniformly) selects a number of N UEs that are currently inactive (i.e. no ongoing mobile video session) and for each one of them, it initiates a new mobile video session with one of their u_s accessible servers. We consider parameter N, which captures the number of new mobile video sessions initiated per second in a system-level scale, to be a Poisson-distributed number with mean n. All servers are considered to provide the same service tariff per video request, an assumption that we consider to have a small impact on the transactions throughput of the system. We further assume that a given UE can set up not more than one mobile video service at the same time.

When the list of new service peers is selected, we further assign to each service pair an exponentially-distributed video time V_u of mean size v. The mean video time v is assumed to be common for all new mobile video sessions in the system.

Note that measuring the transactions throughput of the system is irrelevant to the actual rate, volume of bits and RAT with which the server-client link is implemented. Instead, the focus is given on the transactions load produced by each mobile video session in terms of transactions generated per second. In the sequel, we consider that all nodes act honestly.

Each video service pair is assumed to implement a fixed payment time-plan where a single micro-payment is issued once every $m = 60$ seconds (i.e. fixed micro-payment rate). Without loss of generality, we consider that all UEs in the system load to their user balance a fixed amount of coins that enables them to watch exactly b hours of mobile video time and measure the user balance in units of mobile video time. UEs are considered to update their coin balance only when it is fully consumed (zero balance), triggering a direct on-chain coin transfer to their public address under such an event. For the RE-CENT blockchain system we consider that a new block is issued once every $t = 5$ seconds and assume a single RE-CENT payment relay. The RE-CENT payment relay is considered to commit to a fixed relay delay window d for all of its payment promises towards the servers.

4.1.2 SIMULATION MODEL SCENARIOS AND PARAMETER VALUES

Setting the values of a worldwide simulation setup is not an easy task. Nonetheless, in the following we summarize some widely-accepted parameter values that have been reported by credible manufacturers, content providers, and mobile network operators worldwide, aiming to create a realistic basis for setting up the parameter values of our simulation model. According to [1], the number of mobile broadband subscriptions has reached approximately 6.5B by 2021 and is expected to hit a total of 8B by 2026. At the same time, in 2017, the number of cellular base stations world wide has been estimated close to the number of 6.5M physical base stations without including the different sectors [77]. The official number of registered cellular MNOs in GSMA is reported to 750 by the end of 2020, with the typical number of nation-wide MNOs reaching to 4-6 MNOs per country [78]. The 2020 Cisco Annual Internet Report also puts the number of Wi-Fi hotspots by 2021 close to 485M worldwide and estimates a total of 628M Wi-Fi hotspots by the end of 2023 [79]. Based on these numbers, we consider that the total number of mobile broadband users in the system is $U = 6.5B$ UEs, while the total number

of servers (including cellular base stations and Wi-Fi hotspots) is $S = 0.5B$ for the *All Servers* and $S = 1K$ for the *Cellular MNO* scenario.

Official studies on YouTube usage statistics [18], [80] report an average of 1B watching hours daily on YouTube with 70% of watch time being from mobile devices. An average of 40 minutes video is consumed daily by a typical YouTube user, while a total number of unique 2B YouTube users is also reported monthly. On the other hand, usage statistics for Facebook [19] report a daily average of 1.6B mobile active Facebook users that generate more than 4Bvideo views daily on the platform. The typical Facebook user consumes an average of 26 minutes video daily. Taking into consideration that YouTube and Facebook are dominant but other streaming services are largely used for mobile video delivery, e.g. news sites, Netflix, Vimeo, DailyMotion, we fix the worldwide number of new sessions per second to $n = 100K$ (thousands). We further fix the average watching time per server-client session to $v = 1$ hr of watching time. This setup corresponds to a total of 8.64B watching hours of mobile video daily which, assuming a total of 6.5B mobile broadband users globally, gives an average total watching time of 100 minutes per mobile broadband user. This result is inline with existing reports on the daily per-user video watching time[81], while it can also be translated to an average activation ratio per mobile video user of $\frac{100}{6.5B}\frac{K}{}$ daily (i.e. 1.33 video requests per user daily).

Table 4: Simulation model parameters and values

Parameter	Notation	Description
Number of user public addresses	U	6.5B (full-scale)
Number of servers' public addresses	S	All Servers: $S = \frac{0.5B}{6.5B} \cdot U$, Cellular MNO: S=1K
Mean number of accessible servers (public addresses)	$\bar{s}$	All Servers: $\bar{s} = 30$, Cellular MNO: $\bar{s} = 3$
Number of accessible servers (public addresses) per user	u_s	$S \sim \mathcal{N}(\bar{s}, \bar{s}/10)$
UE network relays	r	$0.1 \cdot U$
Mean number of new sessions per sec	n	100K SpS (full-scale) $n = \frac{100K}{6.5B} \cdot U$
Number of new sessions per sec	N	$N \sim Poisson\,(n)$
Mean service time per session	v	60min
Video time per session	V	$V_u \sim Exp(v)$
Block period	t	5 seconds
Micro-payment periodicity	m	60 seconds
Outbound payment channel balance update	o	Once per 120hrs (*PSU*, *SU* and *U* services)
Simulation time	T	10 full system days

In the *All Server* scenario, we consider a total of *U* mobile broadband users and a total of *S* = $\frac{0}{6}\cdot\frac{5}{5}\frac{B}{B}\cdot$ *U* servers, with each server holding a separate public address. In the *Cellular MNO* scenario, we consider a total of *U* mobile broadband users and a total of *S* = 1*K* cellular MNOs, with servers of the same MNO utilizing the same MNO-specific public address. In both scenarios, we consider that each UE holds a unique public address both to issue and receive payments (if also a network relay). Given the enormous number of UEs and servers that should be considered for the full-scale simulation of the system, whenever necessary, we adapt the scale of our simulations in line with the processing resources available; however, keeping the scale of *U*, *S* and *n* in line with the following two main network topology scenarios. To this end, even if a different *U* is used, we consider the number of new sessions per second in the system to scale by $n = \frac{100}{6.5}\frac{K}{B}\cdot U$. The average mobile video watching time is set to *v* = 60 minutes independent on the number of users in the system. Table 4 summarizes the simulation parameter values.

4.1.3 PAYMENT SERVICES UNDER SCOPE

We consider five payment service variants. Service variants NR-DM and NR-P2PMA) emulate current state-of-the-art payment services to enable performance comparisons and bench-marking. Service variants PSU, SU and U emulate three different deployment options for the RE-CENT relay payment service logic.

NR-DM: ***N****o* ***R****elay with* ***D****irect on-chain* ***M****icropayments and user balance updates.* The performance of this service can be viewed as the transactions capacity that a native blockchain-backed system with no payment channels should support to implement contract-less mobile video content delivery. It emulates the performance of any cryptocurrency platform that enforces the UE to directly post P2P micro-payments on-chain, including transactions to update its on-chain coin balance to *b*. The transactions throughput of this service scales with the number of i) micro-payments issued by UEs and ii) personal balance updates performed by UEs.

NR-P2PMA: No ***R****elay with peer-to-peer (****P2P****)* ***M****icropayment* ***A****ggregation per session and user balance updates.* The performance of this service emulates the transactions throughput of any crypto-currency platform that supports UE-to-server P2P payment channels, like BTC and ETH. When a UE starts a new mobile video service session, it sets up a UE-to-server

payment channel by timelocking a sufficient amount of coins on-chain to the address of the server. Upon service reception, the client cryptographically signs off-chain payments and sends them directly to the server, which stores but doesn't post them on-chain. When the video service concludes, or the payment channel is about to expire, the server posts on-chain the last cryptographically signed transaction of the client and closes the UE-to-server payment channel. The transactions throughput under this service scales with the number of i) UE-to-server payment channel establishments, ii) server claims to release funds from the UE-to-server payment channel and iii) personal UE balance updates. Under the assumption of a sufficient UE balance on-chain, this service requires exactly two on-chain transactions per new mobile video session.

*PSU: **P**ositive fund release per server, outbound channel establishment with **S**ervers and inbound channel establishment from **U**Es.* The performance of this service emulates the transactions throughput of the baseline RE-CENT payment service where micro-payments are aggregated on a per server basis according to the relay delay promises promised per transaction. Active UEs establish an inbound payment channel of balance *b* with the RE-CENT relay and update on-chain whenever their balance is consumed. For every service request, the payment relay checks if an existing outbound payment channel is active for the respective server. If yes, the payment relay verifies that the remaining balance in the outbound payment channel is sufficient to support the respective service. If not, it updates (or establishes) a new outbound payment channel with the respective server. The payment relay maintains a per-server registry recording the pending payment promises along with their expiration block.

Accordingly, the payment relay performs an aggregate on-chain release of funds towards servers with non-finalized (pending) off-chain transactions triggered by the ones with a relay delay promise that is about to expire. Under this occasion, the payment relay aggregates all pending per server off-chain transactions available by that time and implements them with a single on-chain RSC call. The balance of inbound payment channels is updated only when they are about to expire (aggregate coin subtractions). The transactions throughput of this service scales with the number of i) inbound payment channel establishments (or updates) performed by the UE, ii) outbound payment channel establishments made by the relay, iii) on-chain withdraws to their outbound payment channel made by servers (once released by relays) and iv) inbound payment channel updates (coin subtractions) performed by the relay.

*SU: outbound channel establishment with **S**ervers and inbound channel establishment from **U**Es.* The performance of this service emulates the transactions throughput of the RE-CENT payment service where all servers choose not to withdraw the funds released by the payment relay but instead, they choose to attach all released funds in their inbound payment channel. This service scenario is similar to the current banking system where payees choose not to directly cash-out every new deposit transferred to their bank accounts, but instead they use their available coin balance for paying services offered by others. This service is similar to the PSU service but it does not perform a positive fund release on a per-server basis. The performance of this service scales with the number of i) inbound payment channel establishments (or updates) performed by the UE, ii) outbound payment channel establishments made by the relay and iii) inbound payment channel updates (coin subtractions) performed by the relay.

*U: inbound channel establishment from **U**Es.* The performance of this service emulates the transactions throughput of the RE-CENT payment service where relays are fully trusted by network servers, thus, omitting the requirement for establishing outbound payment channels. This scenario can be of high practical interest when the payment relay belongs to the same service domain with the network servers, or when a consortium of large mobile network operators has agreed to joint setup a fully trusted payment relay service. The performance of this service scales with the number of inbound payment channel performed by i) the UE (establishment and update) and ii) the relay (for coin subtractions). If UEs can be assumed to timelock a large amount of coins to their inbound payment channel, this service can lead the transactions throughput to zero and enable infinite blockchain scalability.

4.1.4 TPS vs. NUMBER OF USERS

In Fig. 12, we plot the transactions throughput requirements of all payment service variants under the *All Servers* scenario, by increasing the number of users U in the system. Different user balance values b are also considered, with $b = 1$ hr capturing the scenario with UEs having a low coin balance in the system, $b = 24$ hrs being a normal value for the typical UE balance and $b = \infty$ capturing the extreme scenario where UEs load a very large amount of coins to the payment relay service (for the reasons discussed in section 4.1). The transactions throughput of payment service U can also be viewed as the number of transactions generated only for UE balance updates whereas, as discussed in section 4.1, it can be practically reduced to zero

TPS if $b = \infty$ can be assumed. Different relay delay values d are considered for the PSU service. Fig. 12 also plots a flat line Y = 17 TpS to enable us identify the maximum number of users that can be supported per service variant assuming ETH 1.0 as the benchmark platform (Q1 2021 TPS performance [21]).

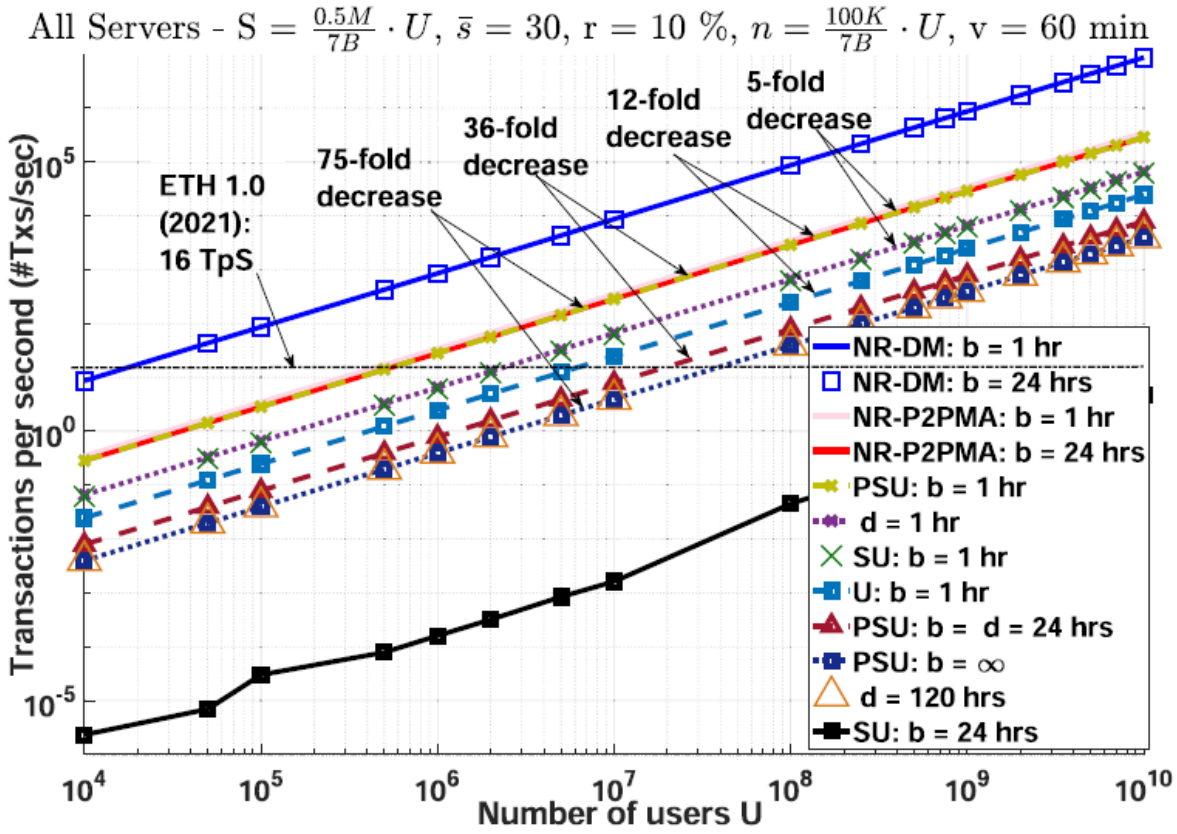

Figure 12: Tps vs. No of users - All server scenario

Fig. 12 illustrates that the transactions throughput of the *NR-DM* payment service remains roughly unaffected by an increase of the UE balance threshold b. This performance trend follows from the fact that the transactions generated for UE balance updates are two and seven orders of magnitude lower than the number of direct on-chain micro-payments for b = 1 hr and b = 24 hrs, respectively. This can be readily verified by observing the performance of the payment service U for b = 1 hr, or b = 24 hrs, and comparing it against that of the *NR-DM* service for the same b values. A similar performance trend is observed for the *NR-P2PMA* service, where an increase of b from 1 to 24 hrs is shown to reduce the transactions throughput requirements by roughly 20%, independent of the number of users U in the system.

Nonetheless, the scale of the respective performance improvement remains roughly the same. As expected, the *NR-DM* payment service requires a comparably lower transactions throughput with respect to the *NR-P2PMA*, due to the aggregation of micro-payments on a per session basis. This performance improvement should be close to half of the average number of micro-payments per session, a parameter that is given by the ratio *v/m* (i.e. close to 30 in our simulations).

4.1.4.1 LOW UE BALANCE CREDIT

For *b* = 1 hr, the RE-CENT *PSU* payment service with a maximum relay delay constraint of *d* = 1 hr attains an improvement of 20% as compared to the baseline *NR-P2PMA* service for the same *b* = 1 and a comparable performance with the *NR-P2PMA* service for *b* = 24. Recall that the *PSU* payment service necessitates two on-chain transactions for establishing the inbound and outbound payment channels as well as two on-chain transactions for i) releasing funds towards the server based on the relay delay *d* and ii) charging the UE client's balance. Nonetheless, since a single inbound payment channel establishment enables aggregation of multiple off-chain payments from a given UE public address and a single outbound payment channel enables aggregation of multiple off-chain payments towards a target server (according to the promised relay delay blocks), the *PSU* service can attain notable performance gains as compared to the *NRP2PMA* service when the relay delay *d* is close to the average video session time *v*.

On the other hand, the transactions throughput requirements of the *SU* and *U* payment services for *b* = 1 hr, are shown to reduce 5-fold as compared to the *NR-P2PMA* service for *b* = 1. This performance improvement is attained due to the fact that under both the *SU* and the *U* payment services, the RE-CENT servers choose not to immediately withdraw their funds using a call to the RSC but instead, they authorize the payment relay to convert off-chain payment promises to an equivalent inbound payment channel balance (section 4.1). This option eliminates the need for on-chain calls to the RSC for fund release on a per server basis, enabling the blockchain-backed payment service to substantially reduce its transaction throughput requirements as compared to both the *PSU* and the *NR-P2PMA* services.

Notably, for *b* = 1 hr, the performance of the *SU* and *U* services is similar, whereas for *b* = 24 hrs their performance gap is very high. This follows from the fact that a very low UE balance

credit (e.g. *b* = 1 hr) results in frequent UE balance updates, making the number of inbound payment channel updates a dominant summand against that of outbound payment channel updates. However, when a larger UE balance credit value is considered (e.g. *b* = 24 hrs), for the given *n* and *v* values, the number of UE balance updates drops rapidly (check *U* for *b* = 24 hrs), turning the number of outbound payment channel balance updates into a dominant summand against that of inbound payment channel updates.

4.1.4.2 MEDIUM UE BALANCE CREDIT

For *b* = 24 hrs, Fig. 12 illustrates that the RE-CENT service variant *PSU* attains a 12-fold reduction of the transactions throughput as compared to the *NR-P2PMA* service, assuming a relay delay of *d* = 24 hrs. The *SU* payment service attains a 75-fold transactions throughput reduction as compared to the *NR-P2PMA* service for the same *b*, i.e. close to two full orders of magnitude improvement, whereas the *U* service attains a significant improvement of roughly five full orders of magnitude as compared to the *NR-P2PMA* service under all *U* values under scope, e.g. for 100M users and *b* = 24hrs the *NR-P2PMA* service requires 2860 Tps while *U* only 0.045 TpS.

4.1.4.3 VERY HIGH UE BALANCE CREDIT

For the extreme scenario where the UEs load a very high amount of funds to their on-chain coin balance, aiming to consume large volumes of mobile video content while paying a very low amount of on-chain transactions fees, we note that i) the performance of the *NR-P2PMA* payment service was measured to be roughly the same for *b* = ∞ and *b* = 24 hrs and ii) the transactions throughput of the *U* payment service is practically zero. In view of that, we have omitted the respective plots from Fig. 12. Notably, increasing the available UE credit *b*, or the relay delay *d*, can substantially reduce the transactions throughput requirements of the *PSU* payment service. Nonetheless, this improvement can reach up to a minimum performance bound, which matches the performance of the *SU* service for the same *b* and *d* values, even if the available UE balance credit is considered to be infinite (*b* = ∞ hrs).

The *PSU* RE-CENT service variant reduces the transactions throughput requirements 36-fold, or roughly two full orders of magnitude, as compared to the *NR-P2PMA* service. For the *SU* service, increasing the UE balance credit from *b* = 24hrs to higher values is shown not to further

reduce the transactions throughput, given that the on-chain transactions generated for personal UE balance updates are very low for b = 24 hrs (check U performance for b = 24 hrs). Hence, even though increasing the UE balance credit b from low to medium values is shown to reduce the transactions throughput requirements of all payment services, especially the ones using the RE-CENT payment relay service, enforcing the end users to have a large balance of coins available onchain will not necessarily reduce the transactions throughput requirements of the blockchain-backed payment service as someone would intuitively expect. As will be shown in the sequel, the value of this b threshold relates to the rate with which the UE consumes its on-chain balance credit. This rate is primarily affected by the n and v values.

4.1.4.4 KEY INSIGHTS AND DESIGN GUIDELINES

Based on the results of Fig. 12, we now draw useful design guidelines related to the performance of blockchain-backed mobile data access in 5G and Beyond networks. First of all, the transactions throughput requirements of all payment services under scope were shown to scale linearly with the number of users U in the system. Thus, for the simulation model and parameter values under scope, the use of additional payment relays (instead of only one) is not expected to increase the transactions throughput requirements for the RE-CENT payment service variants *PSU*, *SU* and *U*. Also, since the impact of U has been shown to be linear for all payment services, a denser network topology assuming a higher U is expected to attain similar performance gains for the RE-CENT payment services.

The RE-CENT service variants (*PSU*, *SU* and *U*) have been shown to better exploit a larger UE balance credit b, in terms of reducing the transactions throughput requirements of blockchain-backed mobile data access, as compared to baseline payment services with direct micro-payments (*NR-DM*) and micro-payment aggregation per session (*NR-P2PMA*). This capability allows the RE-CENT payment service to substantially reduce the transactions throughput requirements of the blockchain-backed system by multiple orders of magnitude as compared to the *NR-P2PMA* service and depending on the values of b and d. Besides, the transactions throughput of the RE-CENT service variants can be limited within predictable performance bounds if a minimum UE balance credit b is set by the RSC for accepting a (UE) client balance update. If the *PSU* service variant is employed, the transactions throughput requirements of the RE-CENT payment service can be reduced by employing a higher relay delay value d. Provided that servers will be always refunded by the servers' mirror fund of

dishonest payment relays (by the RSC), increasing the relay delay *d* will not impact the credibility of the RE-CENT payment service. It will affect only the liquidity of the RE-CENT servers.

Let us now investigate the maximum number of UEs that each payment service can support if we use the popular ETH 1.0 platform as the benchmark for blockchain-backed mobile content delivery in 5G and Beyond mobile data networks. Assuming the employment of direct on-chain payments on a per micro-payment basis (*NR-MD*) the ETH 1.0 system can support up to 20K UEs, whereas the employment of P2P micro-payment aggregation per UE-server session (*NR-P2PMA*) enables the support of up to 500K UEs (*b* = 24 hrs). On the other hand, the number of UEs that can be supported using the *PSU* RE-CENT payment service ranges from 500K to 20M, depending on the values of *b* and *d*, whereas the employment of the RE-CENT payment service with no direct release of funds to the servers (i.e. the *SU* service) can allow up to 40M users in the system. Notably, the use of the RE-CENT payment service *U* variant is shown capable of supporting more than 10B mobile broadband users if a UE balance credit of *b* = 24 hrs can be assumed.

We now identify the transactions throughput requirements of blockchain-backed mobile data access with 100% service penetration in a worldwide scale. Assuming 8B mobile broadband users and a minimum requirement of *b* = 24hrs video balance for enabling UEs enter the system, the *NR-MD* service necessitates a blockchain platform enabling approximately 7M TpS, the *NR-P2PMA* service 286K TpS, the *PSU* service approximately 23K TpS (*d* = 24 hrs), the *SU* service close to 4K TpS, and the *U* service close to 4.8 TpS. Taking into account that the ETH 2.0 platform is envisaged to support up to 100K TpS, we can conclude that the use of the current state-of-the-art strategies *NR-DM* and *NR-P2PMA* make blockchain-backed mobile data access in 5G and Beyond mobile data networks practically infeasible for the years to come. On the other hand, the proposed RE-CENT payment relay service can be readily implemented in the ETH 1.0 platform assuming the *U* service variant, while the most demanding RE-CENT payment service variant *PSU* can curretly support roughly 40M users in the ETH 1.0 platform and the full scale service scenario (8B users) when the ETH 2.0 platform will be delivered (early 2022).

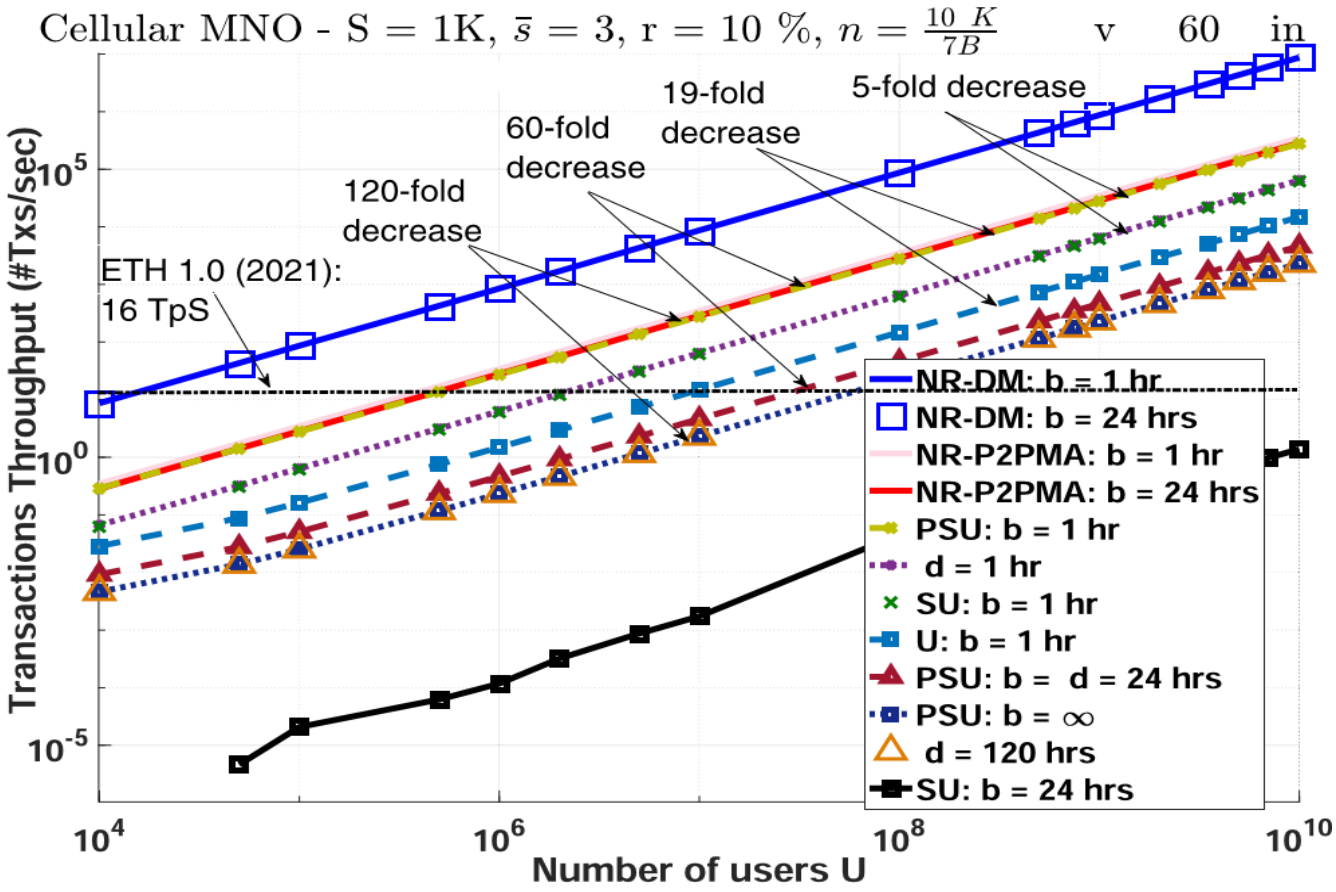

Figure 13: Tps vs. No of users - Cellular MNO scenario

Even if blockchain platforms with even higher transactions capacity capabilities will be made available, i.e. supporting more than 100K TpS and making the baseline no-relay payment services feasible, minimizing the number of onchain payments will still be of high practical interest for the end users. This directly follows from the fact that reducing the number of transactions posted on-chain proportionally reduces the transaction fees paid to implement the service delivery using on-chain payments. Attaining a low transactions throughput proportionally reduces the size of the public ledger, enabling consensus nodes to dedicate a smaller amount of processing, storage and energy resources for blockchain maintenance. Accordingly, the consensus network can be based on a small number of physical nodes, or include many light-weight nodes that consume a comparably lower amount of energy and resources to this end. In view of that, the numerical results of Fig. 12 can also be considered as measure of the performance gains attained by the different payment services in terms of i) the lower transactions fees paid by the end users, or ii) the smaller volume of resources (including energy) consumed for blockchain maintenance.

In Fig. 13, we investigate the impact of *U* on the transactions throughput of the different service variants assuming the *Cellular MNO* network deployment scenario. As depicted in Fig. 13, the cellular MNO scenario enables a higher payment aggregation rate for the RE-CENT payment service variants as compared to the *All Servers* scenario. What is also important to notice is that the key performance trends discussed for the *All Servers* scenario remain unchanged for the *Cellular MNO* scenario as well. Besides, the transactions throughput of the *NR-MD* and the *NR-P2PMA* services remain by default unaffected by *S*. For low UE credit balance values (b = 1 hr), all RE-CENT service variants exhibit a small improvement as compared to their performance under the *AllServers* scenario. However, as the UE credit balance *b* increases (e.g. for b = 24 hrs, or $b = \infty$), we observe a larger performance gap between the *NR-P2PMA* service and the RE-CENT payment service variants *PSU*, *SU* and *U* as compared to the one reported for the *All Servers* scenario. The lower transactions throughput requirements of the RE-CENT service variants under the *Cellular MNO* scenario, follows from their capability to aggregate a higher number of off-chain payments given that UEs will issue payments towards the (RE-CENT server) public address with a higher probability, i.e. a single MNO specific public address is now used for many network servers (RAN base stations).

Under the *Cellular MNO* scenario, the RE-CENT payment relay service variants are shown capable of almost increasing two-fold the number of UEs that the blockchain-backed service can support. For a full scale scenario of $U = 8B$ users and a UE balance credit of b = 24hrs, the RE-CENT service variants are also shown to require an even lower transactions throughput as compared to the *All Servers* scenario, reaching up to 4.5K TpS for the *PSU* service with d = 24 hrs, 1.9K TpS for the *SU* service and 1 TpS for the *U* service. In our simulation campaigns we have recorded similar performance trends for all plots derived for the *All Servers* and the *Cellular MNO* scenarios. For that reason, to the remainder of section 4 we include only the results of the *All Servers* scenario.

4.1.5 TPS vs. RELAY DELAY DEADLINE

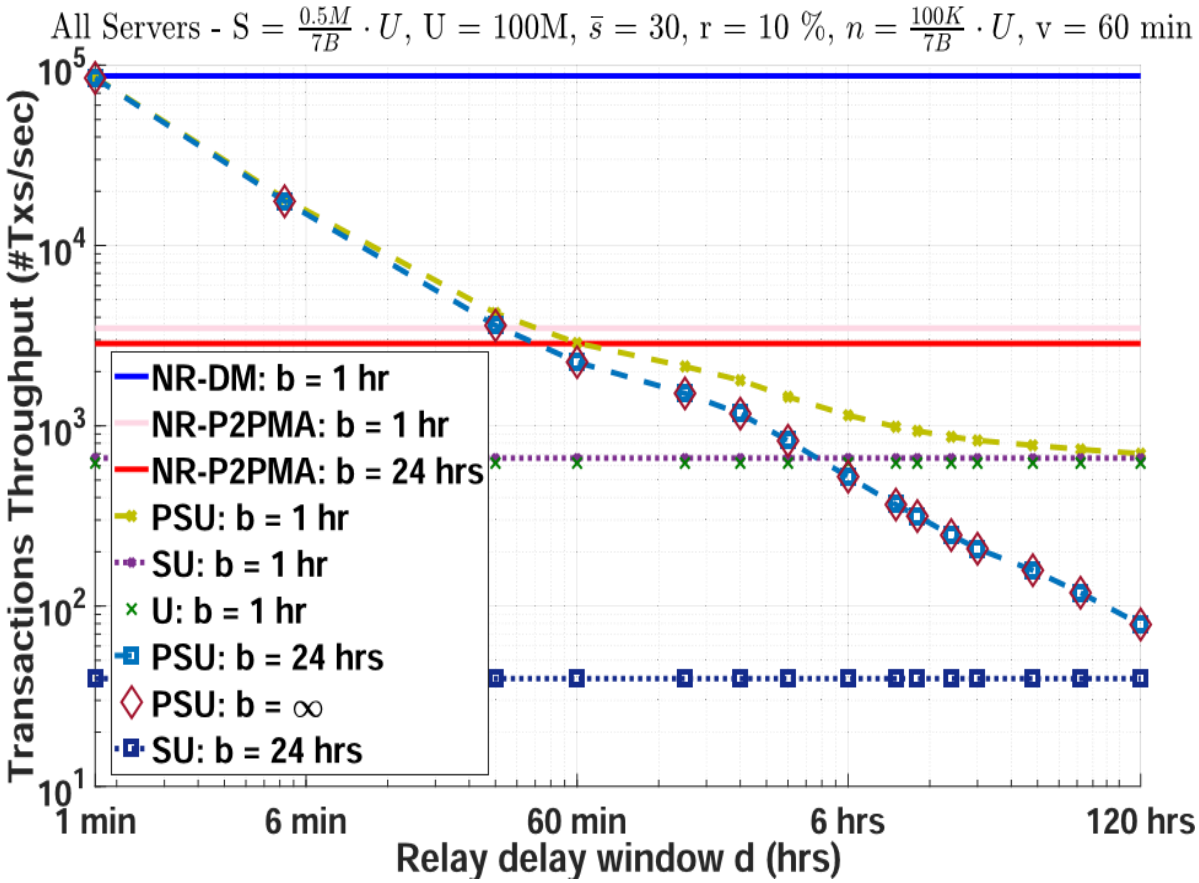

Figure 14: Tps vs. Relay delay (All server scenario).

In Fig. 14, we plot the transactions throughput of the different payment services under scope for the *All Servers* scenario and an increasing relay delay *d*, which ranges from 10 minutes to 120 hrs. Recall that for a given transaction τ, the value of *d* corresponds to the time window size due by which the payment relay is committed to release a given amount of funds towards the server. To sample over multiple network topologies and keep the time/memory complexity of our simulation campaigns within acceptable limits, we have set *U* = 100M (instead of *U* = 7*B*) users. However, we note that similar performance trends are expected for a larger volume of users *U* based on our findings in section 4.2. As expected, the transactions throughput requirements of the baseline *NR-DM* and the *NR-P2PMA* payment services, as well as the *SU* and *U* RE-CENT services, remain unaffected by an increase of *d*. This follows from the fact that only the *PSU* payment service performs on-chain positive server balance updates in line with the relay delay promise *d*.

From Fig. 14, we observe that the transactions throughput of the *PSU* RE-CENT service variant for $d = 1min$ matches that of the *NR-DM* service. This result is expected provided that, under the assumption of nearly instant release of funds, the positive balance updates would be equal to the number of micro-payments at a system-wide scale. However, as the relay delay d increases and approaches the value of the mean video (session) time $v = 60m$, the *PSU* service is shown to attain a similar performance with the *NR-P2PMA* service even for $d < v$. This result can also be explained given that for $d = v$ (= $1hr$), the *PSU* service will be able to aggregate on the average roughly all payments of a video session. Combined with the fact that RE-CENT payment channels are established in a star topology (clients-to-relay and relay-to-servers), it readily follows that in the long-term, the *PSU* service will require a lower transactions throughput as compared to the baseline *NR-P2PMA* service, which requires two on-chain transactions per client-server session: one for client-to-server channel establishment and one for withdrawing funds from the channel (also closing it).

For $b = 1$ hr and $d > 60$ min, we observe that the transactions throughput requirements of the *PSU* service drop fast and reach that of the *SU* service when a relay delay more than $d = 120$ hrs can be tolerated by the servers. On the other hand, for $b = 24$ hrs, the transactions throughput requirements of the *PSU* strategy are also shown to reduce fast with d and reach that of the *SU* strategy for the same b value; however, after a larger relay delay value close to $d = 120$ hrs. What is interesting in Fig. 14, is the existence of a reasonable d value, which is in the order of a few days, that enables the transactions throughput of the *PSU* payment service to scale similarly with the one required for the highly-efficient *SU* payment service (Fig. 12). Besides, as discussed in section 4.2, a higher d value affects the liquidity of servers but not the credibility of the RE-CENT payment service.

It readily follows that the proposed RE-CENT payment relay service provides a flexible framework for fine-tuning the on-chain transactions spurred into the blockchain system towards mobile video delivery in a fully-decentralized and contract-less fashion. Accordingly, the transactions throughput requirements of the system can be adapted in line with i) the transactions capacity governing the underlying blockchain ledger and ii) the anticipated network service type per user (modeled by n and v) and iii) the number of service users in the system (modeled by U). The RE-CENT blockchain platform can achieve this by design and promise a maximum transactions throughput for a given target service type, by integrating into

the RSC logic some minimum thresholds for the values of *b* and *d*, e.g. minimum *d* for relay licensing and minimum *b* for accepting UE balance updates.

4.1.6 TPS vs. UE BALANCE

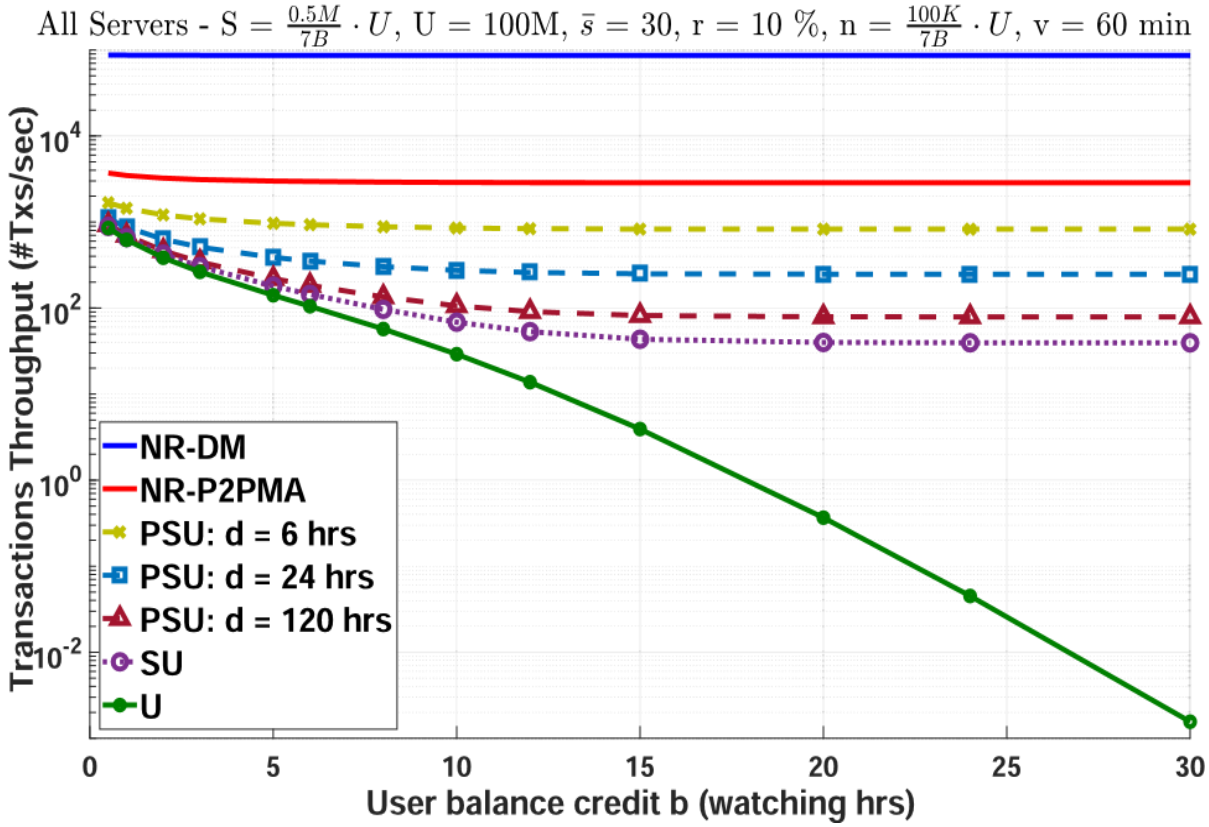

Figure 15: Tps vs. user balance (All server scenario).

In Fig. 15, we investigate the impact of the UE balance credit *b* on the transactions throughput requirements of the different service variants under the *All Servers* scenario. To sample over multiple network instances and keep the time complexity of our simulations within acceptable limits, once again we consider *U* = 100M for our simulations. Recall that increasing the value of *b* results in less frequent on-chain balance updates for each UE, an approach that has been shown to reduce the transactions throughput requirements for all services (Fig. 12).

An increase of the UE balance credit from *b* = 20 mins to *b* = 10 hrs is shown to reduce the transactions throughput of the *NR-P2PMA* payment service by 37%. After this point, no further performance gains are observed for this service with an increase of *b*. Depending on the value of the relay delay *d*, a similar increase of the UE balance credit *b* is shown to reduce the

transactions throughput of the *PSU* RE-CENT service variant as well. As compared to the *NR-P2PMA* service, a 4-fold reduction of the transactions throughput is observed for the *PSU* service if d = 6 hrs and an 8-fold reduction if d = 120 hrs, respectively. Thus, the higher the d value, the higher the performance gains following from an increase of the b value for the *PSU* service as compared to the baseline *NR-P2PMA*. Nonetheless, the transactions throughput of the *SU* service provides a tight performance bound for the *PSU* service, i.e. *PSU* for $d = \infty$ is similar in performance with *SU*, indicating that increasing b and d above a certain threshold would not lead to notable performance gains.

Fig. 15 highlights that, as compared to the baseline no-relay payment services, the RE-CENT service variants can better exploit the availability of additional UE balance credits b and reduce the transactions throughput requirements very fast. When the value of b is approximately 10 to 15 times higher than the average video time v, we also observe a flat performance for all payment services under scope. The RE-CENT service platform can exploit this performance trend and set a minimum UE balance credit b that each UE should load to the payment relay through the RSC. In this manner, the transactions throughput of the RE-CENT payment relay service can be kept below a target threshold, safeguarding the scalability of the blockchain-backed mobile data access model. Besides, UEs are also interested in less frequent on-chain calls to extend their inbound payment channel balance, aiming to reduce the costs attached to this action.

4.1.7 TPS vs. NETWORK RELAY RATIO

In Fig. 16 we investigate the impact of the network relay ratio r on the transactions throughput requirements of all service variants under the *All Servers* scenario. Recall that r is the percentage of UEs that act both as clients and servers under the blockchain-backed mobile video delivery service. To sample over multiple network instances (topologies) and keep the time complexity of our simulations within acceptable limits, we have considered a lower number of users in the system, a parameter that we have set to U = 650K. As expected, the performance of the baseline payment services *NR-DM* and *NR-P2PMA* remains roughly unaffected by an increase on the percentage r of UEs that also act as servers (network relays) in the system. However, the transactions throughput of the *PSU* service variant is shown to increase with the number of UE network relays ($r \cdot U$), provided that the same volume of off-chain payments are now distributed to a larger amount of servers ($S+r \cdot U$) (assuming that n

remains fixed). The average number of off-chain payment promises per RE-CENT server is reduced, limiting the capability of the RE-CENT payment service to aggregate a large volume of payments per server. This effect is more evident for larger *b* and *d* values for the *PSU* service, where a larger performance gap is observed with the *NR-P2PMA* service.

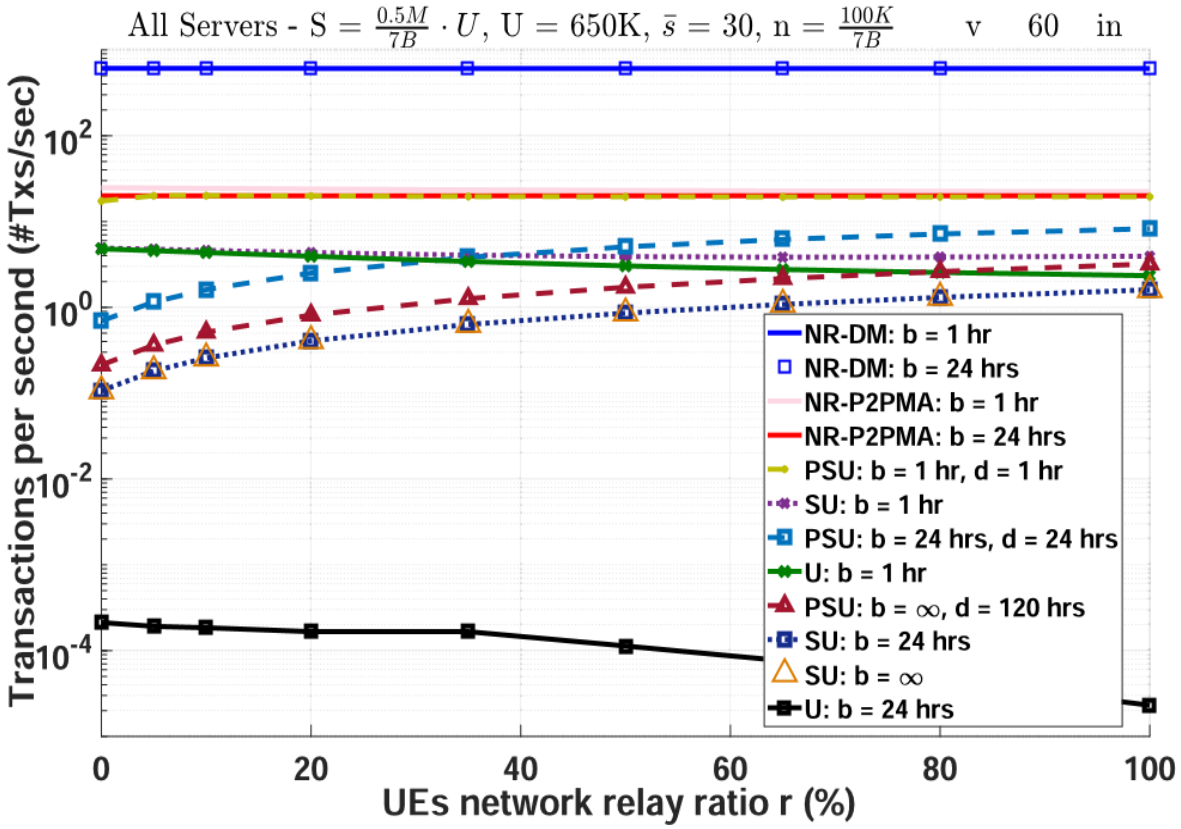

Figure 16: Tps vs. Network relay ratio r (%) (All server scenario).

Besides, a larger amount of (active) RE-CENT servers increases in proportion the number of necessary outbound payment channels. This effect is also the reason why the transactions throughput of the *SU* service variant scales with a similar rate with the *PSU* variant when the available UE balance credit is *b* = 24 hrs. We note that this effect is not evident for the *SU* service and *b* = 1 hr, where the transactions throughput is dominated by the frequent UE balance updates performed (due to the low *b* value). What is also important to note is that both the *SU* and *PSU* payment services are shown to always require a significantly lower transactions throughput as compared to the *NR-P2PMA*, even when the percentage of UE network relays *r* reaches to 100% (i.e. all UEs act as mobile video servers).

Different from all other payment services, the transactions throughput of the *U* payment relay service is reduced for a higher UE network relay ratio *r*. This performance trend follows from the fact that when a UE acts both as a RE-CENT client and as RE-CENT server, it consumes its available UE balance credit with a lower rate due to the funds received from its RE-CENT server mode operation (i.e. the payment relay calculates the difference and subtracts less funds upon client-to-relay balance updates). This result highlights that the RE-CENT payment service variant *U*, which can be employed if a certain level of trust can be assumed (or enforced) between the payment relay service and network servers, is a promising solution for enabling contract-less mobile data access in flat network architectures where nodes act both as service consumers and as service providers with a very high probability (e.g. local P2P networks).

4.1.8 TPS vs. NEW SESSIONS PER SECOND

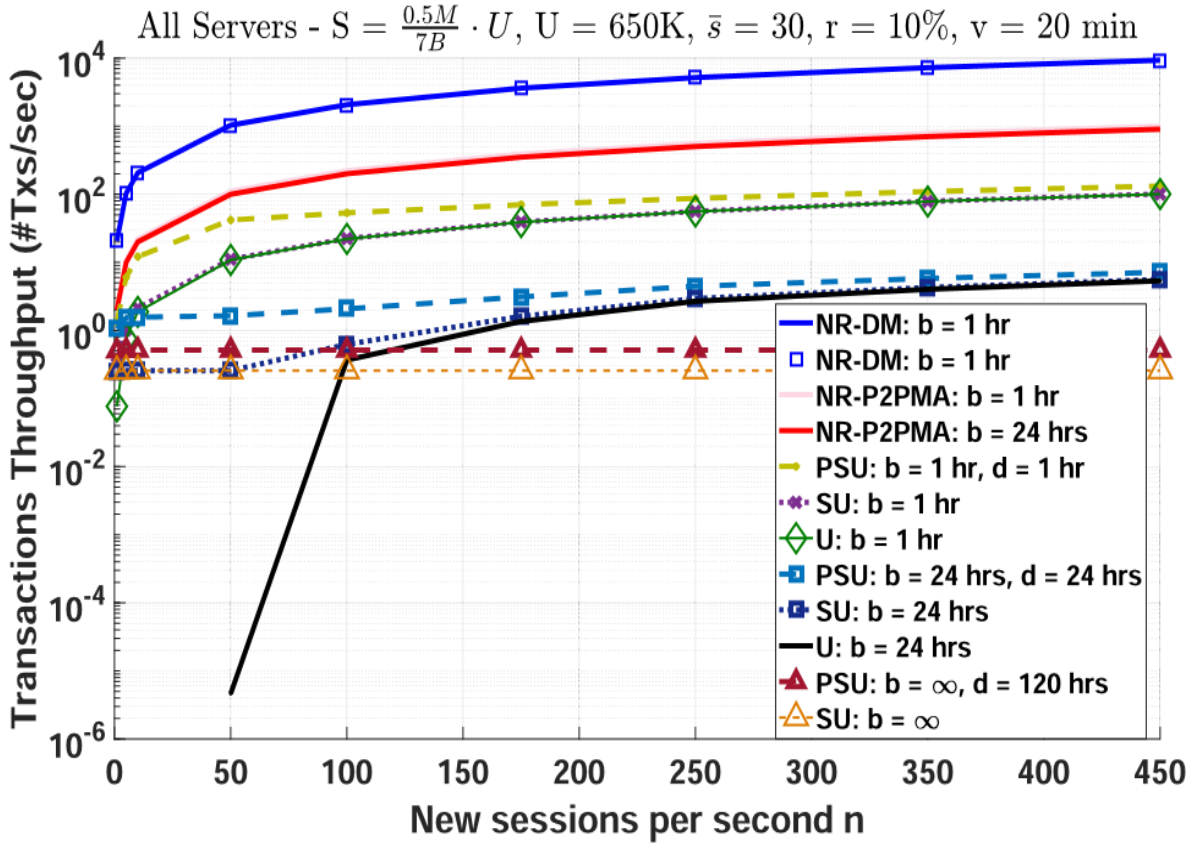

Figure 17: Tps vs. New sessions per second n assuming a mean video time of v D 20 minutes (All server scenario).

In Fig. 17 we investigate the impact of the number of new sessions per second n on the transactions throughput requirements of all service variants under the All Servers Scenario. In this figure, we consider a shorter video time length per session of v = 20 min (instead of v = 1 hr). Then number of users is set to $U = 650K$, whereas the number of servers S is adapted according to $S = \frac{0.5M}{7B} \cdot U$. We also consider that the number of new sessions scales independently of the number of users U in the system, aiming to assess the performance of mobile data networks with a larger volume of new sessions yet with a smaller video session time.

In our simulations, the value of n = 450 results in an active session for almost every UE in the system and thus, a higher n would require each UE to have more than one active mobile video session to a server, an assumption that we don't consider to be valid in our simulation model. Note that the equivalent of having $n = \frac{100K}{7B} \cdot U$ new sessions per second with an v = 1hr video time each (i.e. similar to what we have considered in other plots of section 4), would have given us the benchmark value of n_{650K} = 28 new sessions per second assuming $U = 650K$ users and v = 20 minutes.

As expected, the transactions throughput requirements of the *NR-MD* and the *NR-P2PMA* increase in direct proportion to the number of new sessions per second n. Although a linear increase of the transactions throughput requirements is also observed for the RE-CENT payment relay service variants *PSU*, *SU* and *U*, the rate of this increase is shown to be comparably lower than that of the baseline payment services with no payment relays. Accordingly, the transactions throughput gap between the RE-CENT payment service variants and the baseline services with no relays are shown to increase rapidly with the session load offered to the mobile data network per second (modeled by n). The superior performance of the RE-CENT payment services mainly follow from the one-hop star-topology of payment channels from/to the payment relay (as compared to the mesh P2P topology used by the *NR-P2PMA* service) and the aggregation of multiple off-chain payment promises on a per server basis (given the relay delay deadline d).

Interestingly, even though the performance of the *PSU* service for b = 1 hr and d = 1 hr has been shown to be very closetothatofthe*NR-P2PMA*servicefor$b$ = 1hrinallother plots of section 4, in Fig. 17 we observe that this performance trend is not in effect for the equivalent of n_{650K}

= 28. The main reason is that even though the average daily load per client-server session remains unchanged (i.e. $v = 1hr$ with one session per user is equivalent to $v = 20$ minutes with three sessions per user), the *PSU* performs even better when the average session time is lower (e.g. from 60 to 20 minutes). Accordingly, the payment aggregation implemented by the *PSU* service turns out to be more efficient when the mobile data network serves a larger amount of sessions but with a smaller duration per session. This effect is even more evident at low *n* values and when i) a higher UE balance *b* and ii) a larger relay delay *d* can be considered.

Once again, for $b = 1$ hr we observe that the *U* and the *SU* payment services have similar performance as the result of the very large number of UE balance updates implemented (dominating the server updates). This is also the reason why as *n* increases, the transactions throughput requirements of the *SU* and *U* services tend to be similar to that of the *PSU* service for the same value of $b = 1$ hr (i.e. UE balance updates dominate every other type of on-chain transactions). The same effect is illustrated for $b = 24$ hr between the *SU*, *U* and *PSU* services under high *n* values, where we observe that the transactions throughput requirements of the *U* service to increase rapidly with *n*.

Notably, when a very large UE balance credit *b* can be assumed ($b = \infty$), the performance of the *PSU* and *SU* service variants remains roughly unaffected by an increase to the number of new sessions *n* generated per second. This performance trend reveals that the proposed RE-CENT payment relay service offers significant scalability in network setups where a vast amount of client-to-server sessions are established. Such type of networks can be mobile data networks with high UE mobility (and thus frequent handovers that reduce the client-server session time), or IoT networks where a large volume of short-term (and thus low cost) transmissions are implemented.

5. TESTING OF SOLUTIONS IN THE RECENT TESTBED

In order to test the functionality of the VSC (Validators Smart Contract), RSC (Relayers Smart Contract), and the RE-CENT WebAPI, we'll do a number of tests, using a testbed we've set up including multiple PCs and raspberries.

5.1 Testbed Specifications

The chain will be set up in multiple devices, utilizing many nodes.

The devices are and the nodes running on them are the following:

Table 5: Testbed devices summary

Device	Nodes
PC1	2 validator nodes
Raspberry 3B - No. 1	1 validator node
Raspberry 3B - No. 1	1 regular node

The node software used is **OpenEthereum / Parity**.

The browser extension **Metamask** was used to quickly track the balances of all addresses involved in the testing.

The smart contracts used for the demo were written using **Solidity**. The online solidity editor **Remix IDE** was used.

There were 4 contracts developed:

SafeMath.sol: A simple library with maths functions, used in many cases to assure no integer overflows or illegal computations take place.

RecentBlockchain.sol: A small contract defining the base parameters of the VSC and the RSC. Also includes a few methods allowing users to query chain parameters such as the current epoch, the current block reward, etc.

VSC.sol: The Validators smart contract. Includes functions dictating the validator elections, the parameter amendment procedures. Also includes logic relevant to penalties applied for dishonest - inactive validators. Inherits from RecentBlockchain.sol.

RSC.sol: The Relayers smart contract. Includes functions dictating the relay elections, deposits and withdrawals to and from inbound and outbound payment channels, and transaction aggregation. Also includes functions relevant to dispute resolution between relays and clients / service providers. Inherits from RecentBlockchain.sol.

The WebAPI was developed in **Visual Studio 2019** using **C#.** It's purpose is to provide a simple graphical environment that allows users to call the smart contract functions, and run services that reduce the work that needs to be done manually from them.

5.2 Chain configuration

5.2.1 Genesis block

The basic specs of the RE-CENT chain are defined in the genesis block. Some of the basic parameters are the following:

"name": "Recent" → The name of the blockchain

"engine": "authorityRound" → The AuRA consensus mechanism is used.

"stepDuration": "5" → The block time is 5 seconds. New blocks are verified every 5 seconds.

"blockReward": "10000000000000000000" → The initial block reward in Wei (1 Ether = 10^18 wei). After a duration defined in the genesis file, the block reward will be calculated by the VSC.

"multi": "validators" → The initial validators of the chain. After a duration defined in the genesis file, the validators will be chosen from the VSC.

"minGasLimit": "0x1388" → The minimum allowed amount of gas in a block in hex. Equal to 5000.

"chainID": 12858956 → The ID of the chain. Basically an arbitrarily big number. Chain ID 1 is reserved for the ethereum mainnet.

"gasLimit": "0x1999999" → The maximum allowed amount of gas in a block in hex. Equal to 26843545 (a little lower than 27 million).

"accounts" → A list of accounts with some preallocated coins to their name. These accounts/addresses will be used for testing purposes and are the following:

Table 6: Testbed addresses summary

Address ID / Name	Public address	Balance (in RE-CENT coins)
ServiceProvider1	0x3727b49F8b65049bb25fa7A9dB3Cb98b7A841696	60000
ServiceProvider2	0xc5FD5420FaaaaF902318384dae978aB32dAe81Ee	80000
Relayer1	0x8BBDB937f216210f6C9dFf6d62a6d8A3625E55B9	120000
Relayer2	0x425F533d0c968ad7Db86d613e400d45AaE2E8271	200000
Client1	0x2a4471eFcF34c66B1fb927277dB50Ec6551843Cb	20000

Client2	0xA16c5233ed101Fd82ca6B8985AE217C77980dC82	30000
Client3	0xAfC0D03D14AA3F3d9F8050702dA516c4f44A7F38	9000
Client4	0xaD5D19398b21dC4551BB838BeE29D9C15e4aff21	70000
TestNode1	0x65ef4ca71bc88faab65f8962ca70cfca0be8637c	100000
TestNode2	0xc4afcfefc3ebd20f13de87715e164ea788c602df	250000
TestNode3	0x7c44c05941218e16f3d72e8357ac9a92e3c66e15	300000
TestNode4	0xc7b93b13ba9c334c54442858553722084d33ddaf	500000
TestNode5	0x2be0b1778a60bdbafbb48a5f4be709f76ced84d1	180000

In this chain, 1 ether = 1 RE-CENT coin.

5.2.2 Node information

Table 7: Node information summary

Node name / Device	Node address	Mining address	Network / RPC / Websockets ports
Node 1 / PC1	FILL LATER!	FILL LATER!	30300 / 8545 / 9545
Node 2 / PC1	FILL LATER!	FILL LATER!	30301 / 8546 / 9546
Node 3 / R3B 1	FILL LATER!	FILL LATER!	30300 / 8545 / 9545
Node 4 / R3B 2	FILL LATER!	----------	30300 / 8545 / 9545

Validator nodes are responsible for verifying blocks and continuing the blockchain. Validator nodes require a valid mining address and need to be whitelisted by the genesis block or the VSC in order to verify blocks.

Regular nodes are consensus nodes keeping track of the blockchain. They are not allowed to verify blocks, but they can serve as watchdogs of the blockchain for upper-level applications.

In order to verify blocks or simply keep track of the blockchains, the nodes must be constantly online.

5.2.3 VSC-RSC Parametrization

The VSC and RSC parameters would be different in a production-level deployment, but for testing purposes some of the parameters were tweaked so results can be acquired faster. The tables below includes most basic VSC - RSC parameters and their base values in the testbed.

5.2.3.1 VSC parameters

Table 8: VSC parameters

Parameter	Notation	Value	Type
Epoch duration (blocks)	B_V	360	Fixed
Election window deadline (blocks)	T_V	30	Fixed
Baseline emission rate (coins per block)	R_V	1000	Fixed
Number of consecutive epochs for amendment	C_V	3	Fixed
Current No. of validators	$V\|e\|$	----------	Adjustable
Max No. of validators	V_{MAX}	3	Fixed
Min No. of validators	V_{MIN}	3	Fixed
Current baseline penalty for offline validators	$P_{VO}\|e\|$	2% of total stake	Adjustable
Min baseline penalty for offline validators	P_{V0}^{MIN}	2% of total stake	Fixed
Current baseline penalty for malicious validators	$P_{VM}\|e\|$	25% of total stake	Adjustable
Min baseline penalty for malicious validators	P_{VM}^{MIN}	25% of total stake	Fixed
Minimum validator stake (coins)	$M_V\|e\|$	200	Adjustable
Minimum v-witness stake (coins)	$W_V\|e\|$	10	Adjustable
FoC service tariff per MB	$f_V\|e\|$	0.1	Adjustable

The coin emission disinflation parameter (D_V) is defined as follows:

Table 9: Deflationary table

Epoch intervals	Percentage of initial emission rate
1-5	100%
6-10	20%
11 and onwards	0%

5.2.4 RSC parameters

Table 10: RSC parameters

Parameter	Notation	Value	Type
Relay epoch duration (blocks)	B_R	360	Fixed
Election window deadline (blocks)	T_R	60	Fixed
Relay withdrawal guard interval (blocks)	G_R	60	Fixed
Dispute resolution time window (blocks)	D_R	50	Fixed
Max relay delay threshold (in blocks)	D_{MAX}	180	Fixed
Max transactions fee for off-chain payments (coins)	F_{MAX}	5	Fixed
Minimum stake for r-witnesses (coins)	W_R	5	Fixed
Baseline relay penalty (%)	p_R	2	Fixed
Free of charge service tariff per MB	f_R	0.1	Fixed
Relay monitoring period (in blocks)	k_R	60	Fixed
Current transactions capacity (Txs per epoch)	$TC_R\|e\|$	80	Adjustable
Relay license tariff table for max users	T_{users}	(see RE-CENT paper)	Fixed

Relay license tariff table for max coins	T_{coins}	(see RE-CENT paper)	Fixed
Relay licence tariff table for max throughput	$T_{throughput}$	(see RE-CENT paper)	Fixed

The minimum relay stake (M_R) is defined by the tariff tables (T_{users}, T_{coins}, $T_{throughput}$) found in the RE-CENT paper.

5.3 Scenarios

In order to test the functionality of the Validators and Relayer smart contracts, we need to run our testbed under a slew of different scenarios checking all the functions of each smart contract.

Although there are tons of possibilities, we identify **7** different scenarios, 3 for the VSC and 4 for the RSC. Each scenario tests a different functionality of the smart contract in question.

For more efficient testing, we use the WebAPI and the Metamask browser extension, so we can initiate on-chain transactions and trigger contract methods, query the state of the blockchain through JSON requests, and keep track of all relevant account balances.

5.3.1 Scenario 1 – Validator election

This scenario checks the “Validator election” functionality of the VSC. We first assume that we’re on epoch *e* of the RE-CENT blockchain. The validators for the next epoch (*e+1*) must be decided through the election mechanism of the VSC. Any address can register as a candidate validator, but only open nodes can verify blocks and participate in the consensus. If a candidate validator wins the election, he must run an open mining node with a mining address equal to the address he declared his candidacy with. Excluding the validator candidates, v-witnesses and FoC servers will also participate in the consensus mechanism by staking in favour of their preferred candidates.

For Scenarios 1, 2 and 3, we’re utilizing 5 different public addresses - *TestNode1*, *TestNode2*, *TestNode3*, *TestNode4*, *TestNode5* (see table above). These are their initial balances:

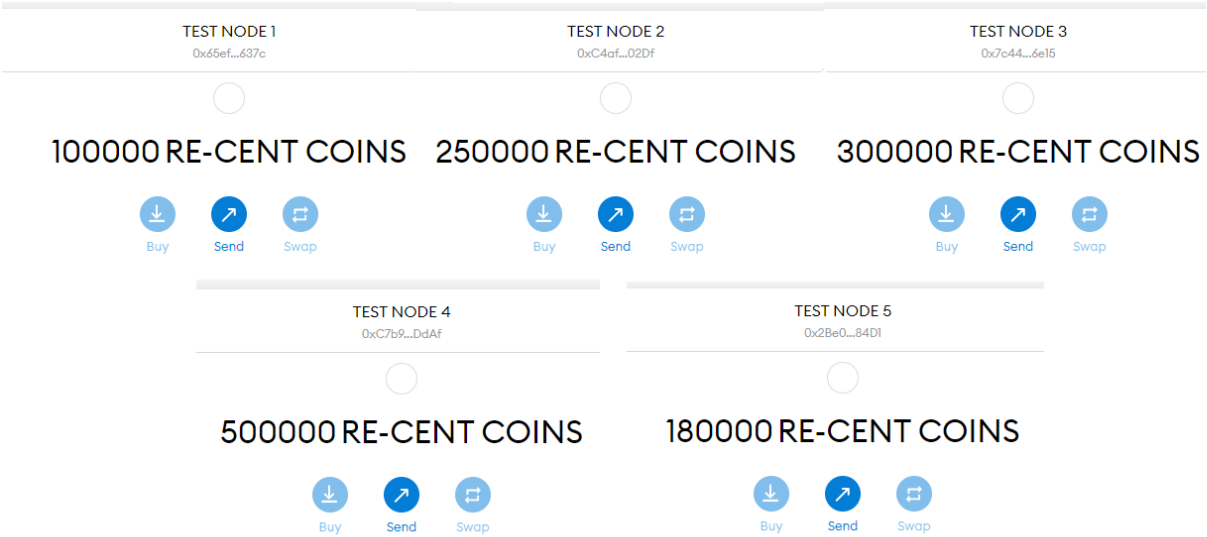

Figure 18: VSC scenarios relevant public address balances

5.3.1.1 Scenario summary

For this scenario, the following steps will be followed:

1.) Acquisition of the list of validator candidates for the next epoch.

2.) TestNode1 declares candidacy in the validator elections.

3.) TestNode2 declares candidacy in the validator elections.

4.) TestNode3 declares candidacy in the validator elections.

5.) Acquisition of the current state of the validator shortlist for the next epoch.

6.) TestNode3 supports TestNode1 as a v-witness.

7.) TestNode4 declares candidacy in the validator elections.

8.) TestNode5 supports TestNode1 as a v-witness.

9.) TestNode5 supports TestNode3 as a FoC service provider.

10.) Acquisition of the final state of the validator shortlist before the epoch changes.

11.) TestNode1 verifies the validator set change at the start of the epoch.

12.) Review the new validators.

13.) TestNode3 and TestNode5 claim their witness reward from TestNode1.

14.) After the epoch is over, all the test nodes withdraw their validator, v-witness and foc-server stakes.

5.3.1.2 Runtime

In order to track the chain parameters and the VSC and RSC variables, we use the WebAPI. For the balance tracking of our test nodes, we can use Metamask.

We assume we're in a random epoch, for example, epoch 2.

A random test node, in our case TestNode1, decides that he wants to participate in the validator election for the next epoch, epoch 3. Firstly, he wants to look at the candidates for the next epoch, to estimate whether he has chances of winning the election. He calls the **"ValidatorElections/Candidates/{epoch}"** method of the API, which calls the **"getCandidates"** VSC method, and allows him to see the candidates for epoch 3:

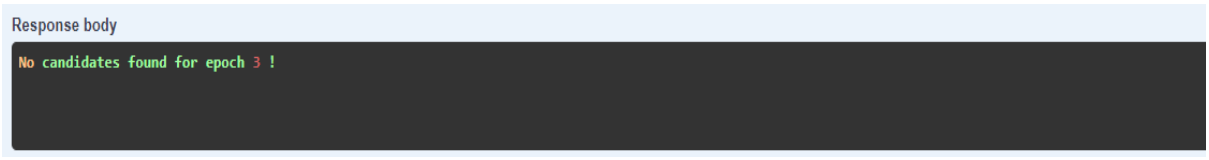

Figure 19: Scenario 1 - Initial candidate list

TestNode1 notices that there are no candidates, therefore he decides to register as a candidate, using the WebAPI method **"ValidatorElections/ValidatorAsCandidate"** which calls the **"validatorAsCandidate"** VSC method. As arguments to this method, he must provide his validator stake and his witness reward stake. The witness reward stake is a sum that will be distributed to the v-witnesses of this validator, based on their witness stake, in case the validator wins the election. TestNode1 will stake 500 RE-CENT coins for his candidacy, and 300 RE-CENT coins for his v-witnesses. His total stake is the sum of these 2 values.

Figure 20: Scenario 1 - Declaring validator candidacy

TestNode1 declares his candidacy, and subsequently calls the **"ValidatorElections/Candidates/{epoch}/Leaders"**, an WebAPI method that allows users to track the current validator shortlist for any epoch. TestNode1 verifies that his candidacy was declared, and currently he has the largest stake in the election. As things stand, he will be a validator in the next epoch.

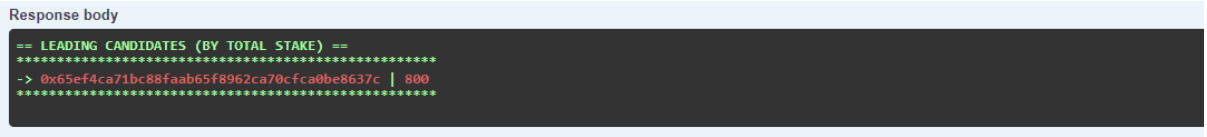

Figure 21: Scenario 1 - Validator shortlist (1 candidacy)

TestNode2 also decides that he wants to participate in the validator election. He declares his candidacy in a similar manner to TestNode1. TestNode2 stakes 1500 RE-CENT coins in favour of his candidacy, but doesn't stake any coins for his witnesses.

TestNode3 also wants to participate in the validator election, but he can only stake 200 coins for his candidacy and 100 coins for his witnesses. He then looks up the validator shortlist for epoch 3, to see if he's in the top 3 that will win the election:

Response body

```
== LEADING CANDIDATES (BY TOTAL STAKE) ==
*****************************************************
-> 0xc4afcfefc3ebd20f13de87715e164ea788c602df | 1500
-> 0x65ef4ca71bc88faab65f8962ca70cfca0be8637c | 800
-> 0x7c44c05941218e16f3d72e8357ac9a92e3c66e15 | 300
*****************************************************
```

Figure 22: Scenario 1 - Validator shortlist (after 3 candidacies)

TestNode3 notices that TestNode1's public address is a candidate. Maybe he had previous dealings with this node, so he decides to look up more details about this candidacy. He uses the **"ValidatorElections/Candidates/{epoch}/{address}/Info"** API method, which calls several VSC functions and gets a complete overview of any candidate, including his validator stake, his witness reward stake, his v-witnesses, his free-of-charge service providers, etc.

Response body

```
Candidate address: 0x65ef4CA71Bc88faAb65f8962ca70CFCa0be8637c
-------------------------------------------------------------------------
Total stake: 800 RE-CENT coins
Validator stake: 500 RE-CENT coins
Witness reward fund: 300 RE-CENT coins

Total witnesses stake: 0 RE-CENT coins
V-Witnesses:
-----------------------------------------
No V-witnesses found.

Total free service provider free MBs: 0
Free service providers:
-----------------------------------------
No FoC servers found.

Currently within validator shortlist: YES
```

Figure 23: Scenario 1 - TestNode1 candidate info (no witnesses)

TestNode3 notices that TestNode1 has a very high witness reward fund, but no v-witnesses. Which means that if he supports TestNode1's candidacy, he'll earn the entirety of the witness reward fund.

He decides to support TestNode1 as a v-witness, using the **"ValidatorElections/SupportCandidateAsWitness"** API method, which calls the **"voteValidatorAsWitness"** VSC method. He will stake 100 coins in favour of TestNode1.

Figure 24: Scenario 1 - Staking in favour of candidate validator as witness

He then looks up TestNode1's candidacy info again. TestNode3 is registered as a v-witness for TestNode1, and his stake is increased accordingly:

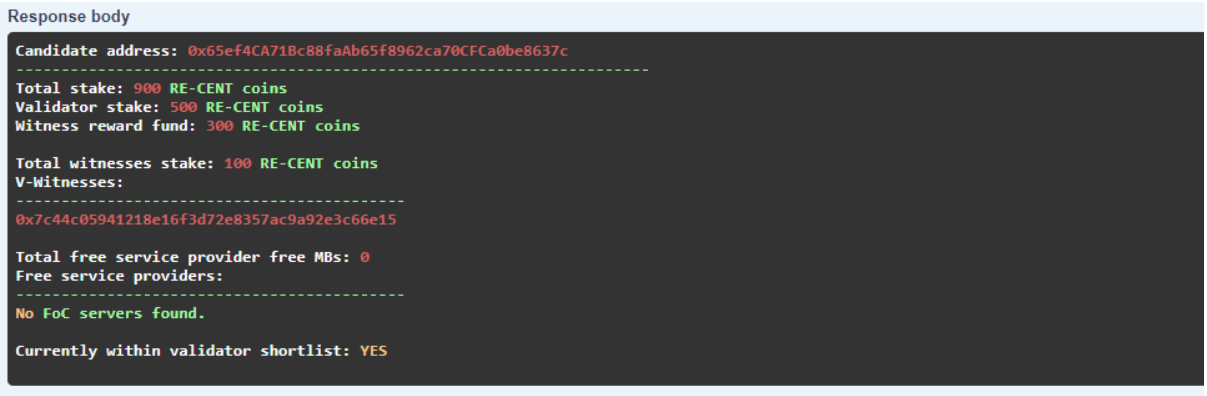

Figure 25:Scenario 1 - TestNode1 candidate info (1 witness)

TestNode4 also declares his candidacy, with 75 coins for his v-witnesses and 250 coins for his candidacy. He looks up the shortlist and realizes that he's in the top 3, and as things stand he'll be a validator in the next epoch. TestNode3 is now 4th, and since the maximum numbers of validators in this example is 3, his stake isn't high enough to be a validator in this epoch.

Response body

```
== LEADING CANDIDATES (BY TOTAL STAKE) ==
*****************************************************
-> 0xc4afcfefc3ebd20f13de87715e164ea788c602df | 1500
-> 0x65ef4ca71bc88faab65f8962ca70cfca0be8637c | 900
-> 0xc7b93b13ba9c334c54442858553722084d33ddaf | 325
-> 0x7c44c05941218e16f3d72e8357ac9a92e3c66e15 | 300
*****************************************************
```

Figure 26: Scenario 1 - Validator shortlist (4 candidacies)

The last participant in this scenario is TestNode5, who is both a client and a service provider in the network. He decides to support candidates that will yield him high reward funds. He calls the **"ValidatorElections/Candidates/{epoch}/Leaders/WitnessRewards"** API method, which ignores validator stakes and only provides info about witness reward funds.

Response body

```
== LEADING CANDIDATES (BY WITNESS REWARD FUND) ==
*****************************************************
-> 0x65ef4ca71bc88faab65f8962ca70cfca0be8637c | 300
-> 0x7c44c05941218e16f3d72e8357ac9a92e3c66e15 | 100
-> 0xc7b93b13ba9c334c54442858553722084d33ddaf | 75
-> 0xc4afcfefc3ebd20f13de87715e164ea788c602df | 0
*****************************************************
```

Figure 27: Scenario 1 - Validator witness reward funds - sorted in descending order

TestNode5 decides to stake 50 coins in favour of TestNode1 due to his high rewards, but he also decides to support TestNode3's candidacy as a free service provider. To this end, he calls the **"ValidatorElections/SupportCandidateAsFoCServer"** API method, which in turn calls the **"voteValidatorAsServiceProvider"** VSC method. He decides to provide free service of 900 Mbs to the witnesses of TestNode3, if they request it.

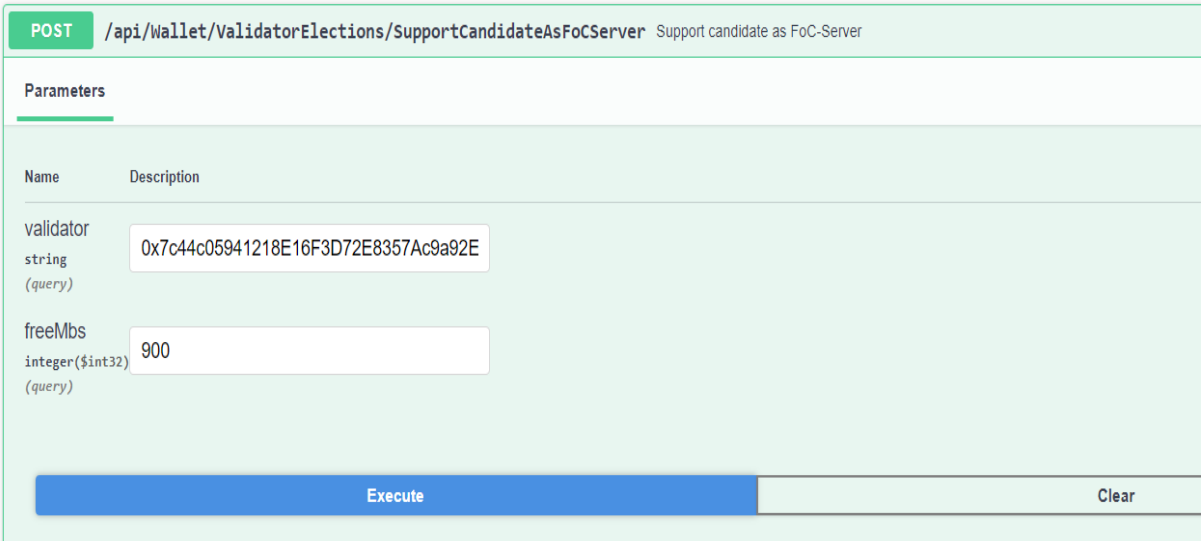

Figure 28: Scenario 1 - Staking in favour of candidate validator as FoC service provider

Since the MB to RE-CENT coin analogy is 10 Mbs for each coin, TestNode3's total stake is now increased by 90 coins. TestNode3 is now again in the top 3 stakes for this election epoch:

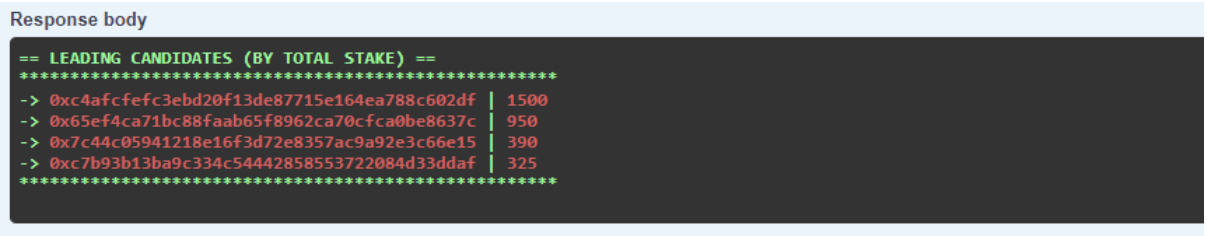

Figure 29: Scenario 1 - Validator shortlist (final)

No further votes or candidacies are cast in this election epoch. The election period is over, and the chain has progressed to epoch 3. In order for the new validator set to take over and start verifying blocks, any of the winning validators of the previous election must first call a specific VSC method to do so. TestNode1 looks up the shortlist and realizes that he's one of the election winners. In order to declare to the nodes that he's won the election, he must call the **"verifyValidatorSetChange"** function of the VSC. He can do this through the **"Validators/Functions/ChangeValidators"** function in the API. The nodes are now aware of this change:

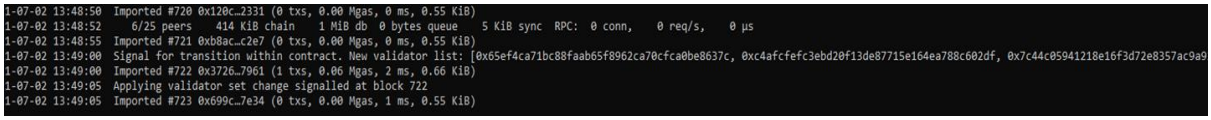

Figure 30: Scenario 1 - Validator set change communicated to network nodes

Assuming all winning validator election have their mining nodes up and running, they can now start verifying blocks and earning block rewards and transaction fees. The API function **“Validators/{epoch}”** finds the validators for this epoch:

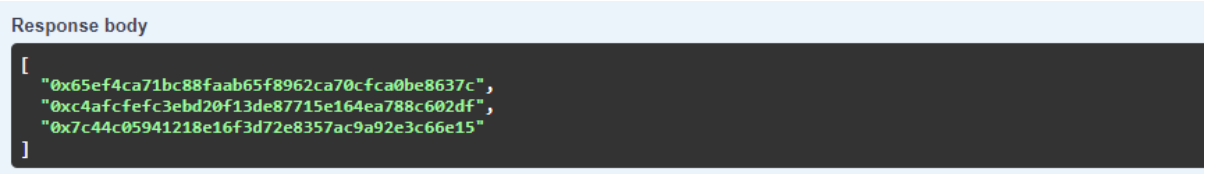

Figure 31: Scenario 1 - Final list of validators for epoch 3 (TestNodes1,2,3)

TestNode3 and TestNode5 see that TestNode1 has won the election, therefore they can claim their rewards as valid v-witnesses. They call the **“vWitnessPaymentRequest”** VSC function through the API, and each is given a share of the reward. Since TestNode3 staked double the coins in comparison to TestNode5 in favour of TestNode1, he’s getting double the reward (200 coins in comparison to 100 coins).

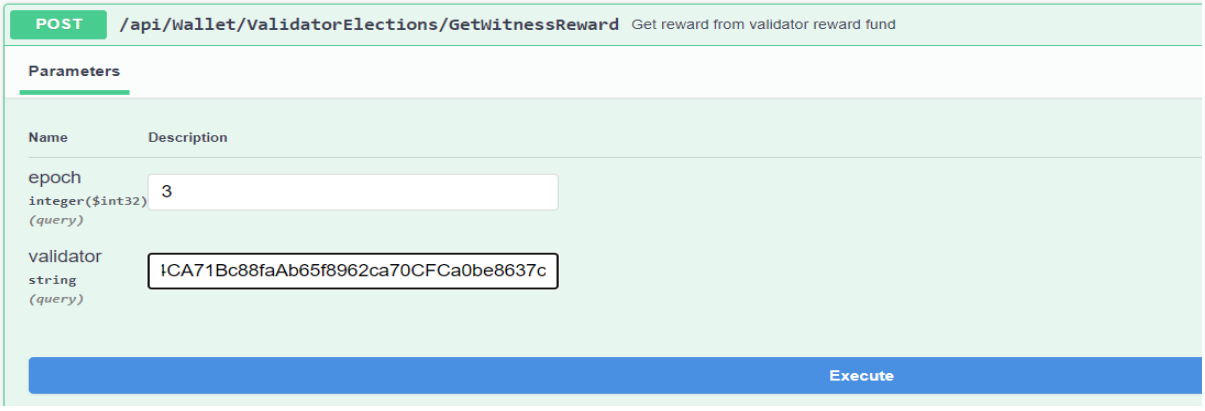

Figure 32: Scenario 1 - Getting witness rewards from elected validators

We assume the winning validators verify all their blocks in time, and no penalties are applied to any of them.

Epoch 3 is now over, which means everyone involved in the election for that epoch can now withdraw their stakes. Through the **"ValidatorElections/WithdrawValidatorStake"**, **"ValidatorElections/WithdrawWitnessStake"** and **"ValidatorElections/WithdrawFoCServerStake"** VSC methods, validators, v-witnesses and free service providers can withdraw their stakes. The API makes this easy for them, by providing the appropriate methods.

The final balances of all nodes involved in this scenario are the following:

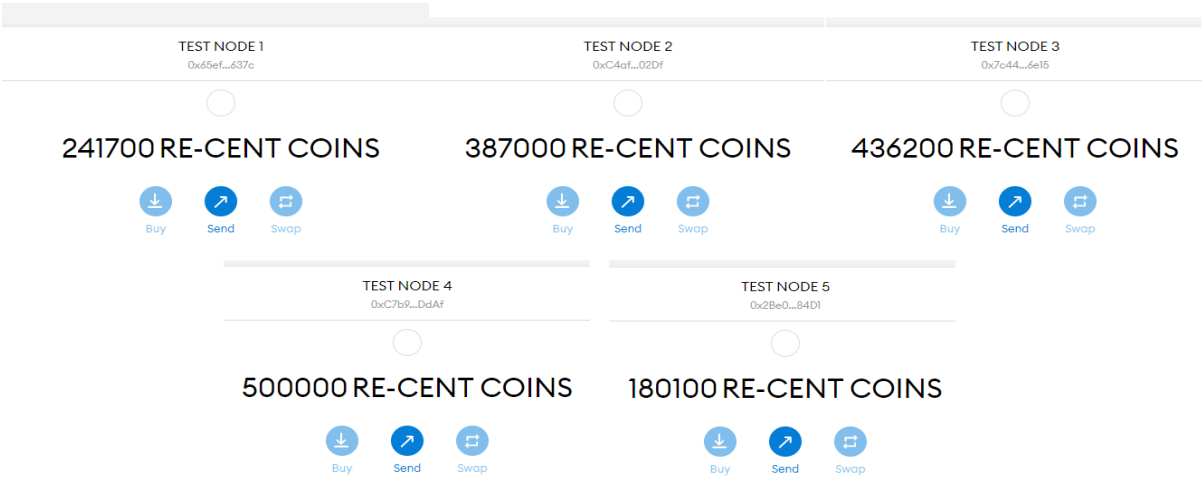

Figure 33: Scenario 1 - Final balances

We notice that TestNodes1, 2 and 3 have increased their balance significantly, due to verifying many blocks each. Since the block reward for epoch 3 was 1000 RE-CENT coins, the increase is analogous to this value. However, excluding block rewards, TestNode1 is 300 coins down because his witness reward fund was claimed, and TestNode3 is 200 coins up because he claimed 2/3rds of TestNode1's witness reward fun. TestNode5 has increased his balance by 100 - this increase is equal to his reward for supporting TestNode1. Finally, TestNode4's balance has remained the same.

5.3.2 Scenario 2 – Validator benign penalties / replacement mechanism

This scenario is a variation of the previous scenario - we assume that one of the winning candidates in the validator election (for example, TestNode3) for one reason or another has suddenly gone offline, and is no longer validating blocks. The other 2 online validators pick up on his inactivity and report him to the VSC. The VSC decides whether the report is valid based on the number of reports it has received about the offline validator. If the reports are over half of the validators in this epoch, then the VSC deals the appropriate penalty ($P_{VO}|e|$) to the offline validator.

In this scenario, after TestNode3 is offline for enough blocks, there suddenly comes a moment when TestNode4's balance becomes larger than his. TestNode4 didn't win the previous election because his stake was below TestNode3's. At this moment though, TestNode4's stake is higher, so the VSC gives him the ability to replace TestNode3 and join the active validators set. We'll first track the stake of TestNode3 and its decrease for his inactivity, then we'll activate the VSC replacement mechanism when the time is right and find the new validator set for this epoch.

5.3.2.1 Scenario summary

For this scenario, the following steps will be followed:

1.) Acquisition of the stakes of the current list of validators for the running epoch.

2.) Validator TestNode3 goes offline.

3.) Rest of the validators report TestNode3.

4.) TestNode3's validator stake is rapidly slashed due to benign reports from the rest of the validators.

5.) When TestNode3's stake drops below TestNode4's, TestNode4 initiates the validator replacement procedure.

6.) Review new validator set.

5.3.2.2 Runtime

In order to track the chain parameters and the VSC and RSC variables, we use the WebAPI. For the balance tracking of our test nodes, we can use Metamask.

We assume we're in a random epoch, for example, epoch 5. TestNode1, TestNode2, TestNode3 and TestNode4 participated in the validator elections, with TestNode4 missing out in the end and ranking 4th in the validator shortlist.

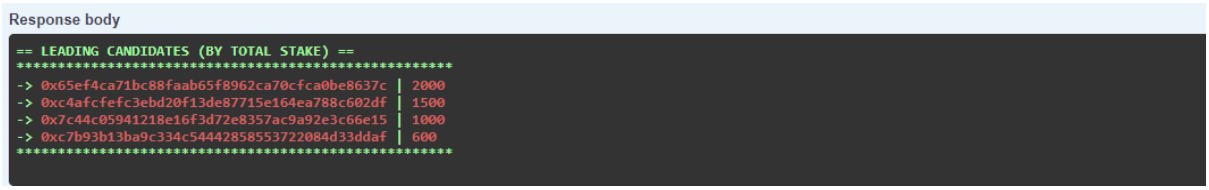

Figure 34: Scenario 2 - Initial validator shortlist

Response body

```
[
  "0x65ef4ca71bc88faab65f8962ca70cfca0be8637c",
  "0xc4afcfefc3ebd20f13de87715e164ea788c602df",
  "0x7c44c05941218e16f3d72e8357ac9a92e3c66e15"
]
```

The target epoch for the election has started, and TestNode1, TestNode2 and TestNode3 are now the active validators sealing blocks.

TestNode4's balance at this time is seen below:

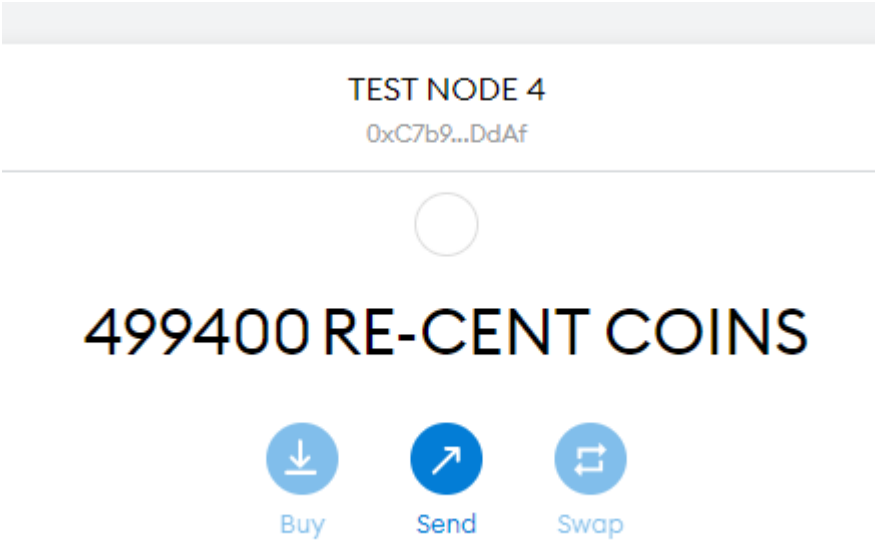

Figure 35: Scenario 2 - TestNode4 balance before validator replacement

At some point during the epoch, one of the active validators, TestNode3, goes offline. This means that he can no longer verify blocks, even though he's an elected validator. As far as the chain is concerned, this means that blocks are getting verified slower, as TestNode3 cannot seal blocks when it's his turn in the authority round. Less blocks means less transactional throughput. Therefore, TestNode3's inactivity is actively harming the chain.

Active validators can help punish offline nodes by utilizing the **"reportBenign"** VSC function. They can do this automatically, as long as they have their node software running. We assume honesty from the remaining validators, who report TestNode3 for being offline.

```
2021-07-02 17:27:45  Benign report for validator 0x7c44…6e15 at block 1487
2021-07-02 17:27:45  Imported #1487 0x51c7…7730 (0 txs, 0.00 Mgas, 6 ms, 0.55 KiB)
2021-07-02 17:27:50  Transaction mined (hash 0xc65ba6aa5645d5982655d6cd5f0edc3eb0072a529439fce316bda8cdf7dfb6a4)
2021-07-02 17:27:50  Imported #1488 0x6970…51be (3 txs, 0.17 Mgas, 8 ms, 1.06 KiB)
2021-07-02 17:27:55     3/25 peers    180 KiB chain    2 MiB db  0 bytes queue    8 KiB sync  RPC:  0 conn,    0 req/s,    0 µs
2021-07-02 17:28:00  Benign report for validator 0x7c44…6e15 at block 1489
2021-07-02 17:28:00  Imported #1489 0xe1ed…deff (0 txs, 0.00 Mgas, 7 ms, 0.55 KiB)
2021-07-02 17:28:05  Imported #1490 0x2db5…3453 (3 txs, 0.15 Mgas, 9 ms, 1.06 KiB)
2021-07-02 17:28:05  Transaction mined (hash 0x15fede969850862fc8940d67821ceb62b0ff9af5041ada42606bf059a90dfede)
2021-07-02 17:28:15  Benign report for validator 0x7c44…6e15 at block 1491
2021-07-02 17:28:15  Imported #1491 0x39fa…8723 (0 txs, 0.00 Mgas, 6 ms, 0.55 KiB)
2021-07-02 17:28:20  Imported #1492 0x942e…387b (3 txs, 0.15 Mgas, 9 ms, 1.06 KiB)
2021-07-02 17:28:20  Transaction mined (hash 0xef83b93f85024b8d3c4c5f847c1bd06d00fcb4428197304b74bd9eb6d6b10bcc)
2021-07-02 17:28:25     3/25 peers    182 KiB chain    2 MiB db  0 bytes queue    8 KiB sync  RPC:  0 conn,    0 req/s,    0 µs
2021-07-02 17:28:30  Benign report for validator 0x7c44…6e15 at block 1493
2021-07-02 17:28:30  Imported #1493 0xfd38…ff53 (0 txs, 0.00 Mgas, 7 ms, 0.55 KiB)
2021-07-02 17:28:35  Transaction mined (hash 0x144f1a0e83a0b986701236b9b75c5ae995d80767402849b1c0331f843cf0185f)
2021-07-02 17:28:35  Imported #1494 0x0f9d…946e (3 txs, 0.15 Mgas, 0 ms, 1.06 KiB)
```

Figure 36: Scenario 2 - Benign reports for offline validator TestNode3

Due to them being the majority, their reports are approved by the VSC, which punishes the offline validator TestNode3 with for every block he should have sealed. The size of the penalty depends on the $P_{VO}|e|$ parameter.

Response body

```
== LEADING CANDIDATES (BY TOTAL STAKE) ==
*****************************************************
-> 0x65ef4ca71bc88faab65f8962ca70cfca0be8637c | 2000
-> 0xc4afcfefc3ebd20f13de87715e164ea788c602df | 1500
-> 0x7c44c05941218e16f3d72e8357ac9a92e3c66e15 | 936
-> 0xc7b93b13ba9c334c54442858553722084d33ddaf | 600
*****************************************************
```

Figure 37: Scenario 2 - TestNode3's stake slashed due to inactivity

At some point, due to the penalties applied on TestNode3, TestNode4 notices that his stake has surpassed that of TestNode3.

Response body

```
== LEADING CANDIDATES (BY TOTAL STAKE) ==
*****************************************************
-> 0x65ef4ca71bc88faab65f8962ca70cfca0be8637c | 2000
-> 0xc4afcfefc3ebd20f13de87715e164ea788c602df | 1500
-> 0xc7b93b13ba9c334c54442858553722084d33ddaf | 600
-> 0x7c44c05941218e16f3d72e8357ac9a92e3c66e15 | 584
*****************************************************
```

Figure 38: Scenario 2 - TestNode4's stake surpasses that of TestNode3

Therefore, since he's now 3rd on the validator shortlist, he can now trigger TestNode3's replacement process. He can do this by using the **"Validators/Functions/ReplaceValidator"**

API function, which calls the **“replaceValidator”** VSC method.

Figure 39: Scenario 2 - Validator replacement method

With this method call, TestNode4 publicizes to the network that his stake is now larger than TestNode3’s, therefore he can take TestNode3’s place in the validator shortlist. The validator set change is communicated to the network:

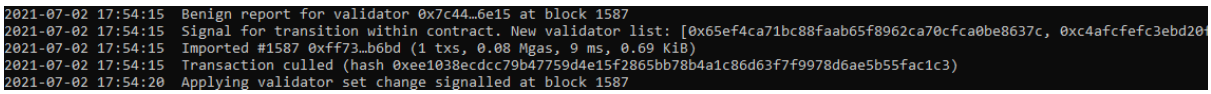

Figure 40: Scenario 2 - TestNode4 signals new validator set to the rest of the network

And TestNode4 is now an active validator:

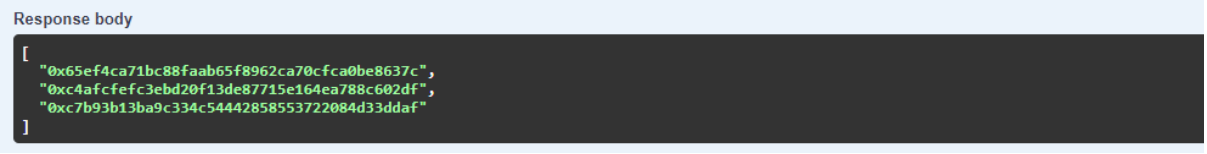

Figure 41: Scenario 2 - New validator set

Which means that he is now able to reap rewards from sealing blocks:

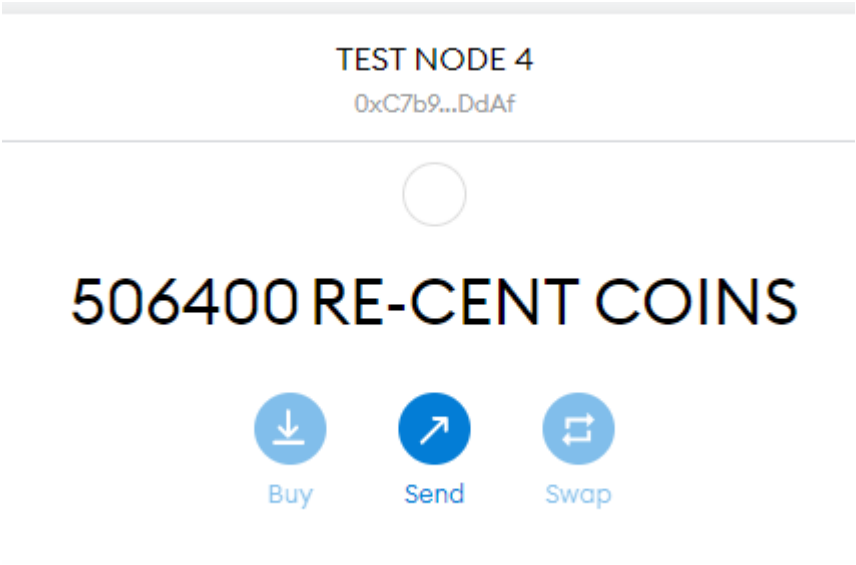

Figure 42: Scenario 2 - TestNode4's balance after validator replacement and block sealing for some time

The replacement process was successful.

5.3.3 Scenario 3 - Parameter amendment functionality:

This scenario checks the functionality of the parameter amendment mechanism.

Validators of each epoch have the ability to vote for the increase or decrease for a few select adjustable parameters (for example, the minimum validator stake $M_V|e|$). For a parameter amendment to be successful, it needs to be voted for several epochs consecutively (C_V) by the majority of validators. If the proposal fails to pass for a single epoch, then it must be voted again for C_V epochs in a row. For simplicity, we'll omit the validator election mechanism in this scenario and assume that TestNode1, TestNode2 and TestNode3 win the validator election in every epoch.

5.3.3.1 Scenario summary

For this scenario, the following steps will be followed:

1.) Retrieve chain and chain parameter summary.

2.) Epoch E: TestNode1 and TestNode2 vote to increase the minimum validator stake.

3.) Epoch E+1: TestNode1 and TestNode2 vote to increase the minimum validator stake.

4.) Epoch E+2: TestNode1 and TestNode2 vote to increase the minimum validator stake.

5.) Review the parameter change.

5.3.3.2 Runtime

In order to track the chain parameters and the VSC and RSC variables, we use the WebAPI. The current epoch is epoch E.

In this scenario, we assume that in an off-chain discussion forum, there are talks of increasing the minimum validator stake so only the most dedicated network participants are allowed to become validators. Since this proposal is very popular, the validators of the current epoch decide to vote in favour of increasing the minimum validator stake ($M_V|e|$). For amendments to pass a majority of the validators is needed, therefore in our case, 2 out of 3.

We can check the current minimum validator stake, by using the **"Info/Summary"** function in the API, which includes information about all relevant chain parameters.

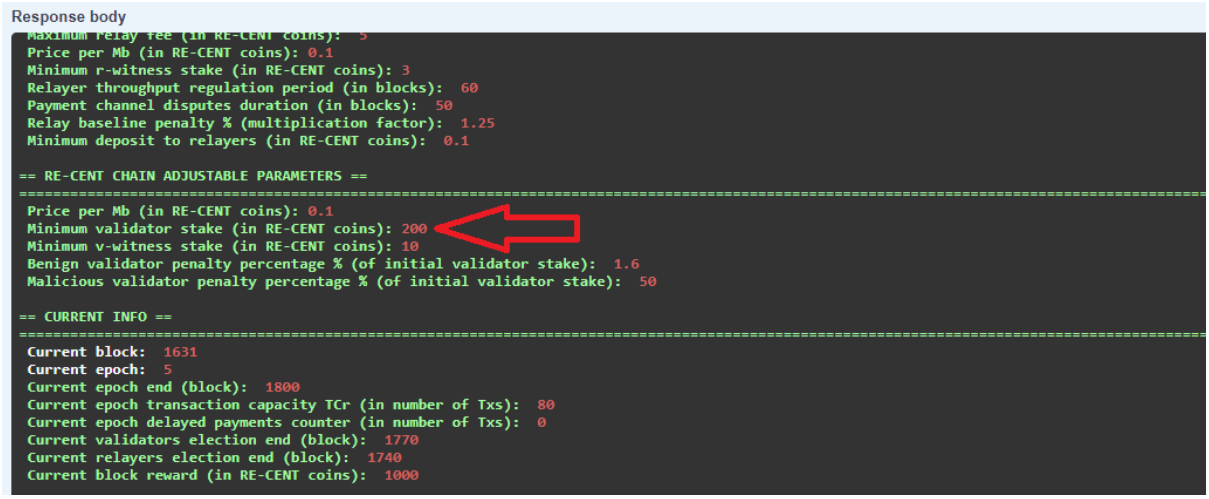

Figure 43: Scenario 3 - Initial minimum validator stake

TestNode1 uses the **"Validators/Functions/Amendments/AmendParameter"** function of the API, which calls the **"amendParameter"** VSC function. Validators can choose to increase one of the adjustable parameters or decrease them. The nature of the change depends on the parameter. For the minimum validator stake though an increase corresponds to a doubling, and a decrease to the halving of the parameter's value.

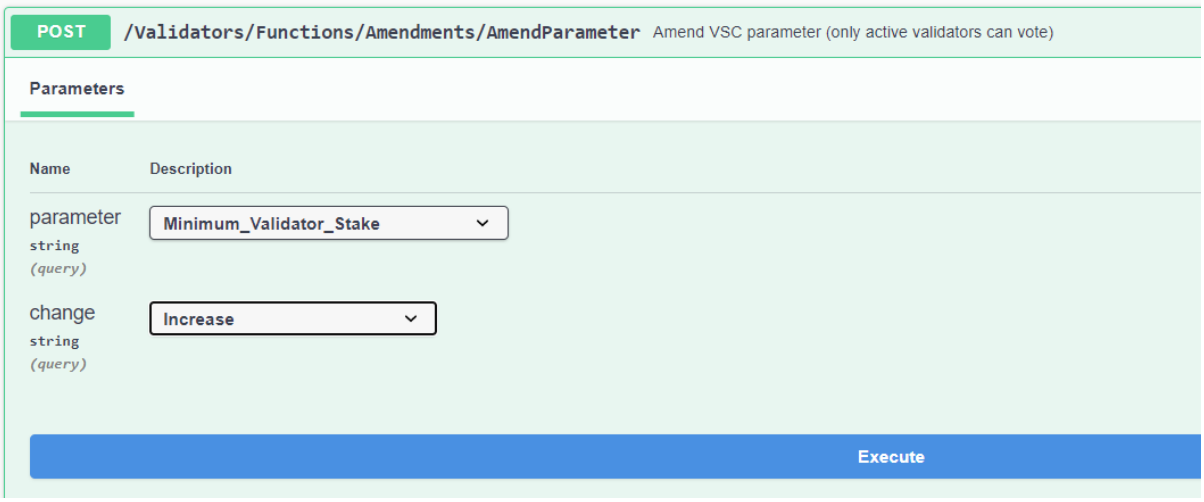

Figure 44: Scenario 3 - Voting for parameter amendment

TestNode2 then decides to also vote in favour of this amendment. He then uses the **"/Validators/Functions/Amendments/GetInfo"** API function to check if the proposed changed has been voted in favour of in this epoch.

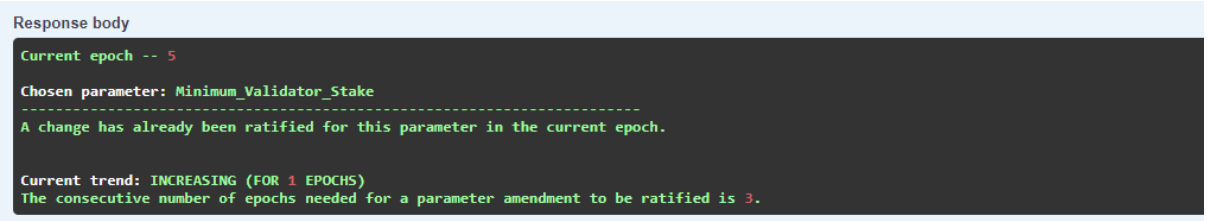

Figure 45: Scenario 3 - Amendment voted (1st epoch)

Since C_V is equal to 3 epochs, this procedure must continue for 2 additional epochs for the parameter change to take place.

In epoch E+1, testNode2 and testNode3 vote in favour of increasing $M_V|e|$. The amendment has now been voted for 2 epochs in a row, therefore it must be voted for an additional epoch:

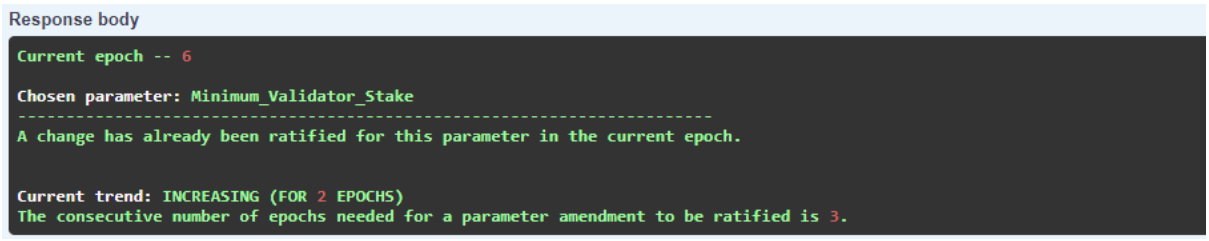

Figure 46: Scenario 3 - Amendment voted (2nd epoch)

In epoch E+2, testNode1 and testNode2 vote in favour of the amendment. This means that the prerequisite of 3 consecutive epochs is reached. The parameter amendment is effective immediately, and is now stored on-chain:

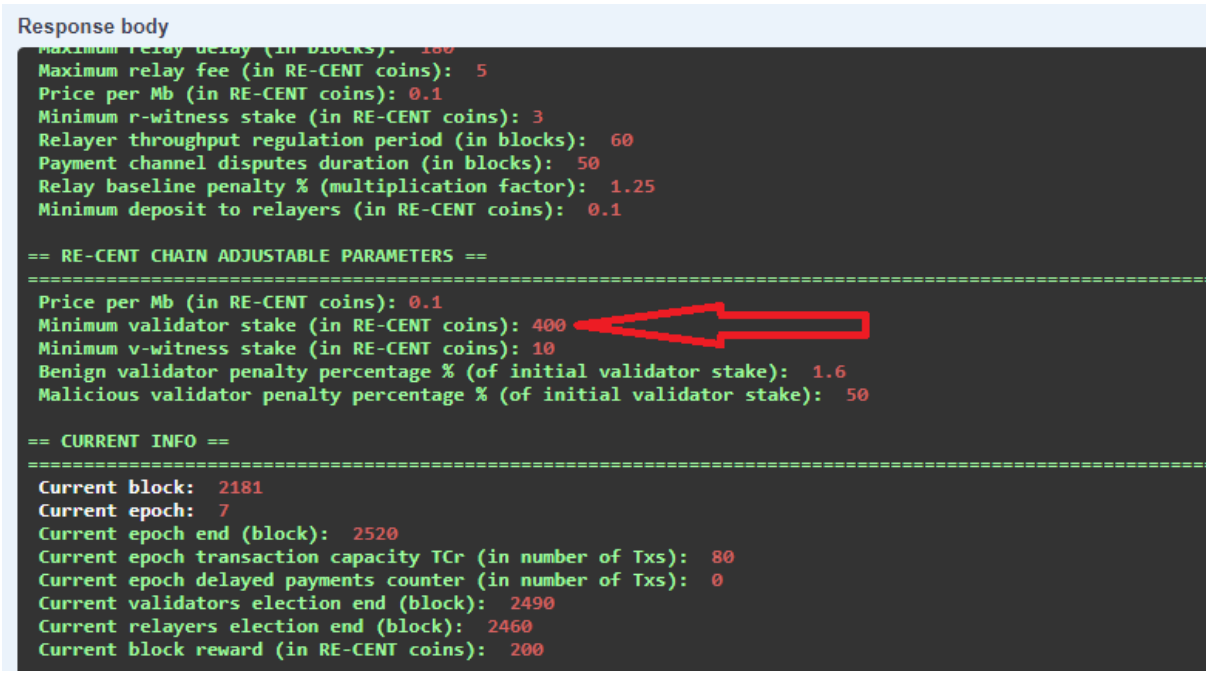

Figure 47: Scenario 3 - Change ratified (minimum validator stake doubled)

The parameter amendment procedure was a success.

5.3.4 Scenario 4 - Relay elections

This scenario examines the functionality of the relay election mechanism in the RSC. We first assume that we're on epoch *e* of the RE-CENT blockchain. The relayers for the next epoch (*e+1*) must be decided through the election mechanism of the RSC. Any address can register

as a candidate relayer, however in order to fulfil their relay duties, relayers must run the relay service (as defined in the API) locally.

The relay election is a bit more complex than the validator election. Candidates must also declare their relay specs while registering as candidates, including parameters such as their relay delay, their fees, etc., alongside their stake. The minimum required stake for a candidate is calculated via a tariff table (see paper) and depends on the maximum user, coin and transaction throughput availability a candidate will set. Once the voting period is over, the candidates must be shorted based on their stake. For this, they must perform an on-chain transaction to register in the final relay shortlist for the next epoch. Once the epoch is over, the RSC calculates the winners based on the top stakers and their maximum transaction throughput. Overall, candidates with a maximum transaction throughput less than the transaction capacity of the chain (which is calculated on an epoch-by-epoch basis) are chosen as valid relayers. However, candidates with a maximum transaction throughput above the remaining capacity are ignored, regardless of their stake. In the end, the chosen candidates must have a combined maximum transaction throughput less or equal to the transaction capacity of the chain.

Excluding the relay candidates, r-witnesses and FoC servers will also participate in the relay election mechanism by staking in favour of their preferred candidates.

For Scenarios 4, 5, 6 and 7, we're utilizing 5 different public addresses - *ServiceProvider1, Relayer1, Relayer2, Client1, Client2* (see table above). These are their initial balances:

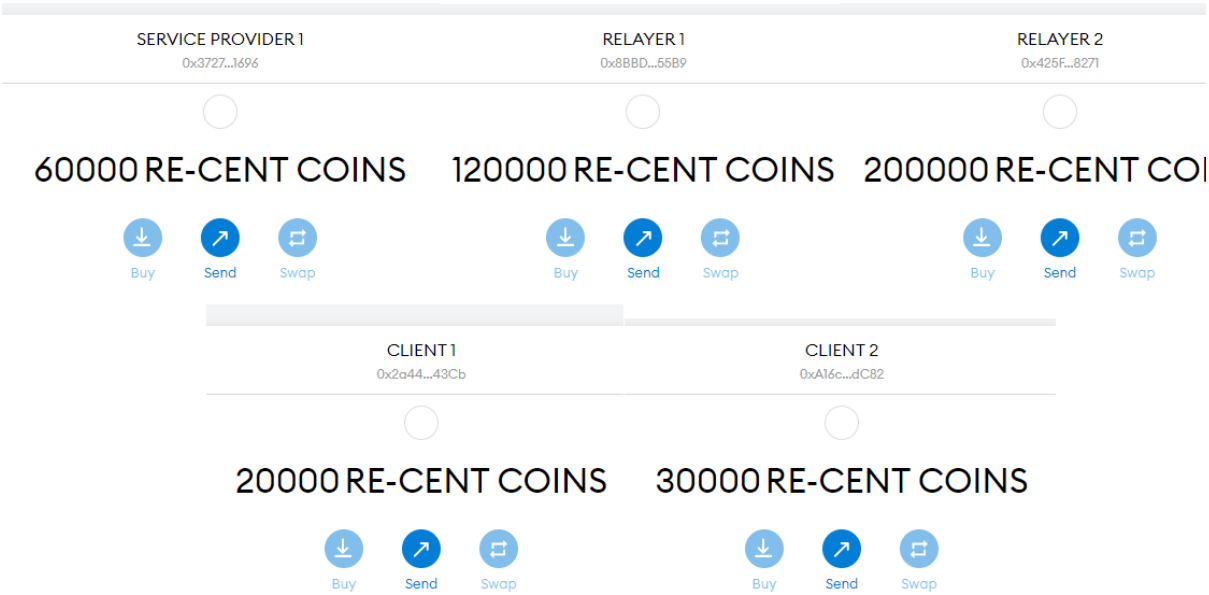

Figure 48: RSC scenarios relevant public address balances

5.3.4.1 Scenario summary

For this scenario, the following steps will be followed:

1.) Retrieve chain and chain parameter summary.

2.) Relayer1 declares candidacy for relay election of the next epoch.

3.) Relayer2 declares candidacy for relay election of the next epoch.

4.) Client1 declares candidacy for relay election of the next epoch.

5.) Client2 supports Relayer2 as an r-witness.

6.) ServiceProvider1 supports Relayer1 as a free-of-charge service provider.

7.) After the voting period is over, Relayer1 registers his accumulated stake to the final relay shortlist.

8.) Relayer2 and Client1 also register their accumulated stakes from the election period to the final relay shortlist.

9.) Acquisition of the final relay shortlist before the epoch change.

10.) Relayer1 verifies the relayer set change. The new relayers receive a license based on their maximum transaction capacity and the system transaction capacity.

11.) Review the new relayer set.

12.) Client2 claims his witness reward from Relayer2.

13.) Client2 reviews and compares the new relayers.

14.) Client2 withdraws his r-witness stake from Relayer2.

15.) ServiceProvider1 withdraws his foc-server stake from Relayer1.

16.) After the epoch has passed, Relayer1, Relayer2 and Client1 withdraw their relayer stakes.

5.3.4.2 Runtime

In order to track the chain parameters and the VSC and RSC variables, we use the WebAPI. For the balance tracking of our test nodes, we can use Metamask.

We assume we’re in a random epoch, for example, epoch 21.

A random public address - network participant, for example, Relayer1, wants to acquire a

relay license for the next epoch. His first step is to check the blockchain's transaction capacity. This is because in order to make a better offer that will yield him a higher chance of success, his maximum transactional throughput must be less than the capacity of the chain. After, the election system benefits relay candidates with low maximum transactional throughputs and high stakes. To get the current blockchain capacity, Relayer1 uses the **"Info/Summary"** API method. We can also see that there are no active relayers in the system currently.

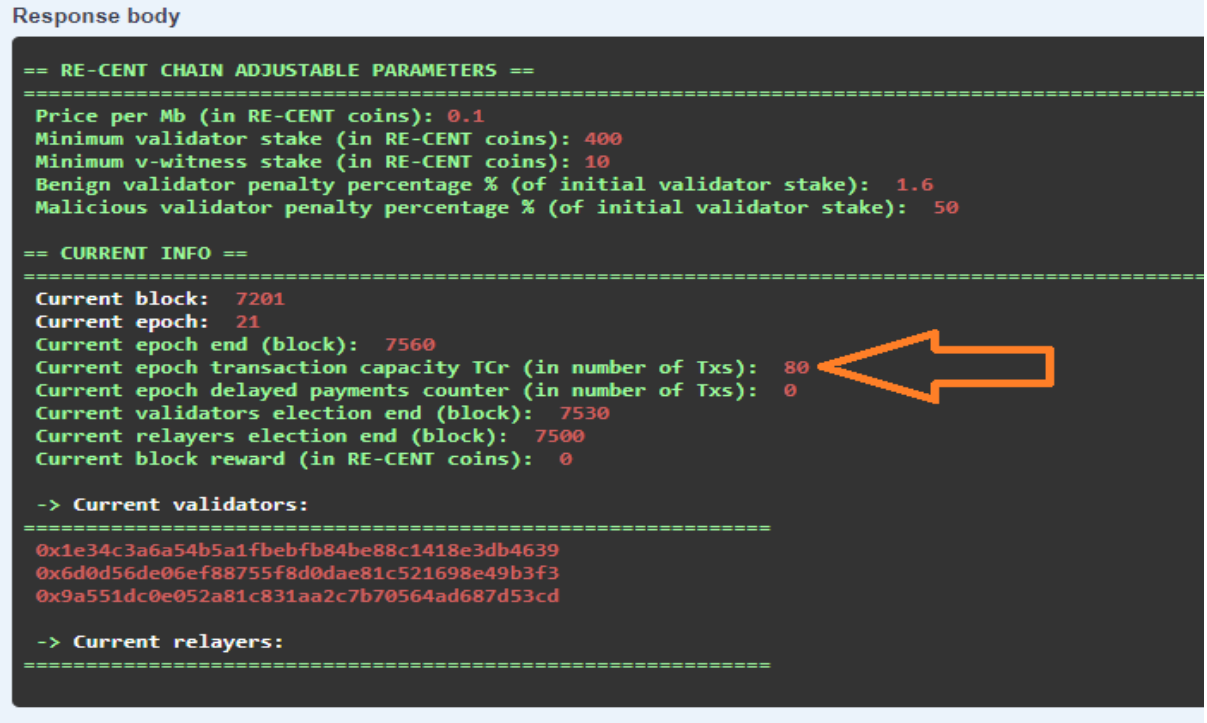

Figure 49: Scenario 4 - Transaction capacity in the network

Relayer1 then decides to declare his candidacy. This is done through the **"RelayerElections/RelayerAsCandidate"** API method, which calls the **"relayerAsCandidate"** RSC method. This method has many arguments – the candidate relay must declare first declare his stake. The RSC has a complex minimum stake calculation mechanism for relays, based on a tariff table which increases the minimum stake when the maximum number of users, coins, or transactions the relay allows. Before declaring his candidacy, the candidate relayer can calculate his minimum stake through the **"RelayerMinimumFundRequired"** API method, which calls the **"getFundRequiredForRelayer"** RSC method. Relayer1 estimates that he can cover for a maximum of 10 users, a total fund capacity of 250 coins, and a total transactional throughput up to 50 on-chain transactions per regulation epoch.

Response body

```
For relay specs:
Max users = 10
Max coins = 250 RE-CENT coins
Maximum transaction throughput (per regulation period) = 50 transactions

Minimum relayer stake is 10635 RE-CENT coins.
```

Figure 50: Scenario 4 - Getting the required minimum stake for set properties

Once he has calculated his minimum stake, he can declare his candidacy. Apart from his relayer stake and his witness reward fund, the relayer must also declare his name, the IP address where his relay service will be running, the maximum amount of users he'll allow funds for, the maximum amount of coins he'll allow in all his inbound payment channels, and the maximum transactional throughput he'll allow in a regulation epoch. Additionally, he must declare his relay delay and relay fees. The relay delay is the interval before which the relay must update the on-chain balance of a service provider involved in a transaction. A high relay delay allows for better transaction aggregation and less on-chain updates. The relay fee is essentially a constant fee on each micropayment. It's the relay's reward for forwarding transactions and requests between service providers and clients. Both the relay delay and the relay fee have maximum values, which can be looked up via the **"Info/Summary"** function.

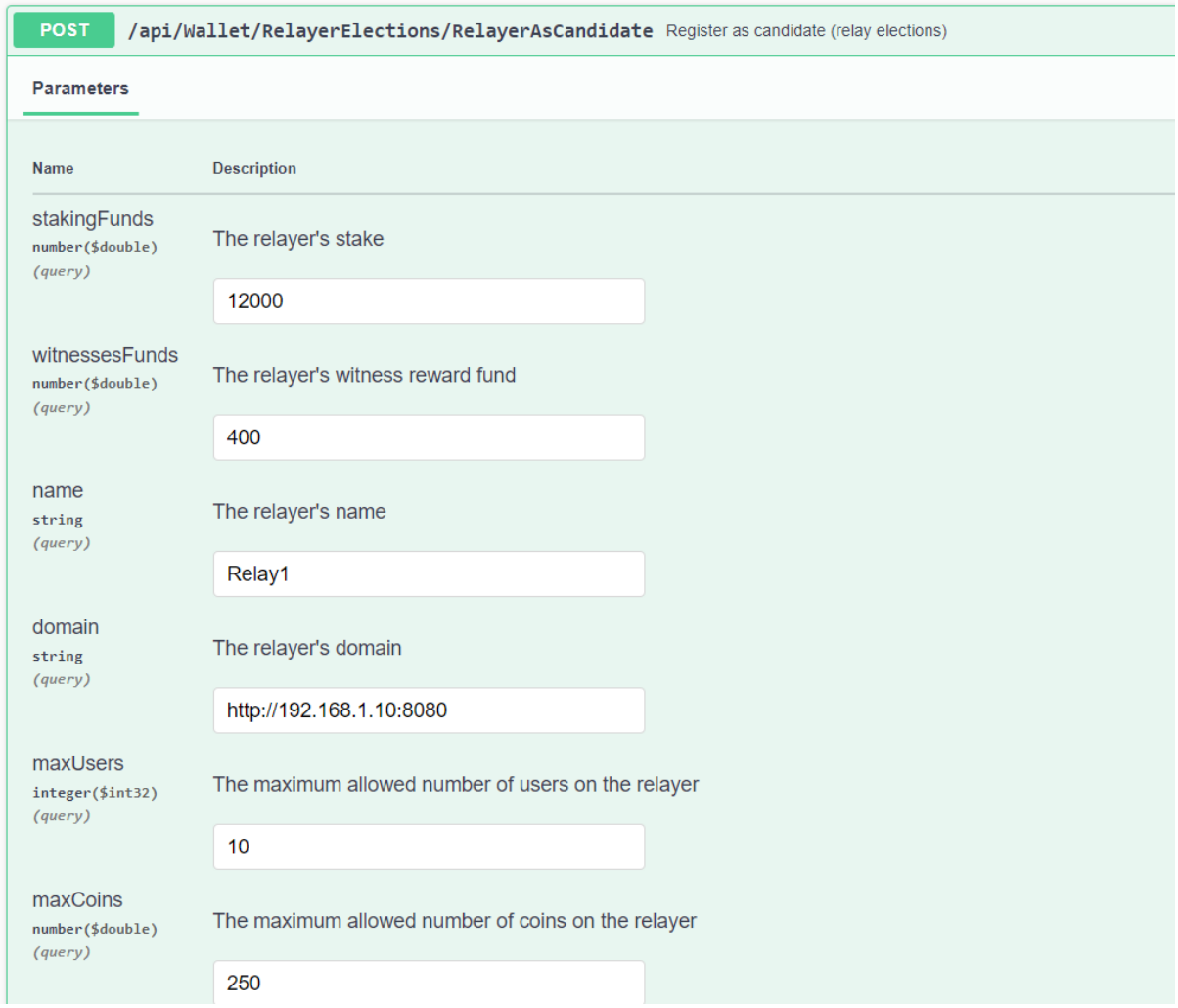

Figure 51: Scenario 4 - Declaring relayer candidacy

Relayer1 can then call the **"RelayerElections/Candidates/{epoch}/{address}/Info"** API function to verify the details of his candidacy:

Response body

```
Candidate address: 0x8BBDB937f216210f6C9dFf6d62a6d8A3625E55B9
--------------------------------------------------------------------------
Total stake: 12400 RE-CENT coins
Relayer stake: 12000 RE-CENT coins
Witness reward fund: 400 RE-CENT coins

Total witnesses stake: 0 RE-CENT coins
V-Witnesses:
----------------------------------------
No V-witnesses found.

Total free service provider free MBs: 0
Free service providers:
----------------------------------------
No FoC servers found.


~~~~~~~~~~~~~~ RELAY SPECS ~~~~~~~~~~~~~~
Relay name = Relay1
Relay domain = http://192.168.1.10:8080
Max users = 10
Max coins = 250 RE-CENT coins
Max gas throughput per subepoch = 6250000 gas
Relay delay = 100 blocks
Relay fee per transaction = 0.002 RE-CENT coins
Server mirror fund = 250 RE-CENT coins
Client mirror fund = 250 RE-CENT coins
Remaining penalty funds = 11500 RE-CENT coins
Total stake = 12000 RE-CENT coins
```

Figure 52: Scenario 4 - Relayer1 candidacy info

Relayer2 also chooses to participate in the election. His preferred maximum allowed of users, coins, and transactions is lower than Relayer1's, therefore he can afford to stake less money in favour of his candidacy. His candidacy details are the following:

Response body

```
Candidate address: 0x425F533d0c968ad7Db86d613e400d45AaE2E8271
------------------------------------------------------------------------
Total stake: 1500 RE-CENT coins
Relayer stake: 1000 RE-CENT coins
Witness reward fund: 500 RE-CENT coins

Total witnesses stake: 0 RE-CENT coins
V-Witnesses:
-----------------------------------------
No V-witnesses found.

Total free service provider free MBs: 0
Free service providers:
-----------------------------------------
No FoC servers found.


~~~~~~~~~~~~~~ RELAY SPECS ~~~~~~~~~~~~~~
Relay name = Relay2
Relay domain = http://192.168.1.10:8081
Max users = 4
Max coins = 80 RE-CENT coins
Max gas throughput per subepoch = 3750000 gas
Relay delay = 120 blocks
Relay fee per transaction = 0.01 RE-CENT coins
Server mirror fund = 80 RE-CENT coins
Client mirror fund = 80 RE-CENT coins
Remaining penalty funds = 840 RE-CENT coins
Total stake = 1000 RE-CENT coins
```

Figure 53: Scenario 4 - Relayer2 candidacy info

Client1 also declares his candidacy, using the following specs and stakes:

Response body

```
Candidate address: 0x2a4471eFcF34c66B1fb927277dB50Ec6551843Cb
------------------------------------------------------------------------
Total stake: 1600 RE-CENT coins
Relayer stake: 1500 RE-CENT coins
Witness reward fund: 100 RE-CENT coins

Total witnesses stake: 0 RE-CENT coins
V-Witnesses:
----------------------------------------
No V-witnesses found.

Total free service provider free MBs: 0
Free service providers:
----------------------------------------
No FoC servers found.


~~~~~~~~~~~~~~~ RELAY SPECS ~~~~~~~~~~~~~~~
Relay name = Relay3
Relay domain = http://192.168.1.10:8084
Max users = 3
Max coins = 50 RE-CENT coins
Max gas throughput per subepoch = 3750000 gas
Relay delay = 120 blocks
Relay fee per transaction = 0.01 RE-CENT coins
Server mirror fund = 50 RE-CENT coins
Client mirror fund = 50 RE-CENT coins
Remaining penalty funds = 1400 RE-CENT coins
Total stake = 1500 RE-CENT coins
```

Figure 54: Scenario 4 - Client1 candidacy info

Client2 on the other hand wants to support a candidate with a high witness reward fund. He uses the **"RelayerElections/Candidates/{epoch}/Leaders/WitnessRewards"** API function and notices that Relayer2 has the highest witness reward stake:

Response body

```
== LEADING CANDIDATES (BY WITNESS REWARD FUND) ==
**************************************************
-> 0x425f533d0c968ad7db86d613e400d45aae2e8271 | 500
-> 0x8bbdb937f216210f6c9dff6d62a6d8a3625e55b9 | 400
-> 0x2a4471efcf34c66b1fb927277db50ec6551843cb | 100
**************************************************
```

Figure 55: Scenario 4 - Candidates sorted by witness reward

Therefore, he stakes as a witness in Relayer2's favour, using the **"RelayerElections/SupportCandidateAsWitness"** API function which in turn calls the **"voteRelayerAsWitness"** RSC function:

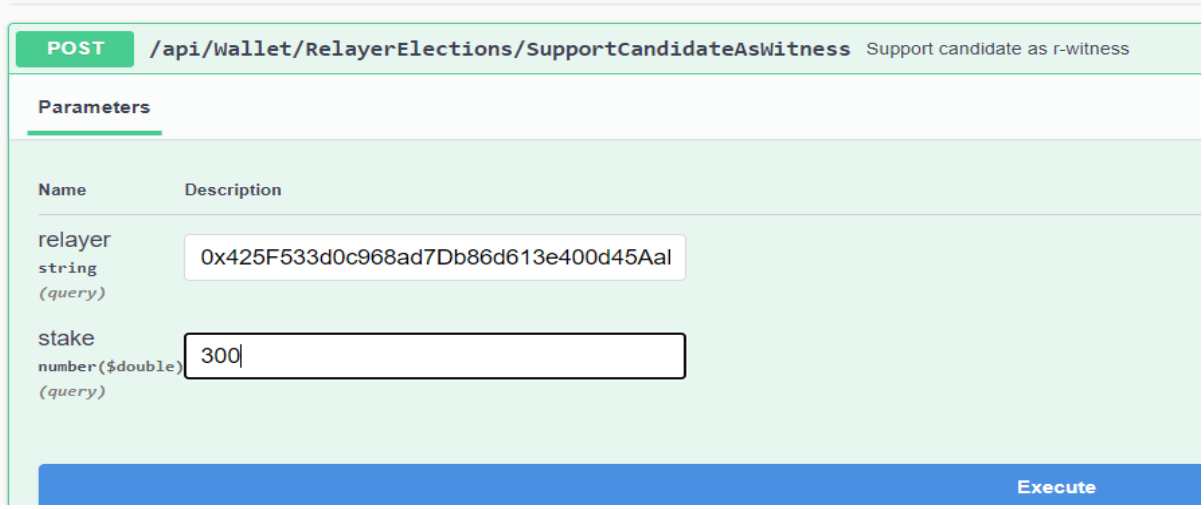

Figure 56: Scenario 4 - Staking in favour of candidate as witness

ServiceProvider1 decides to support the candidacy of Relayer1 for personal reasons, by providing free Mbs to potential witnesses. He uses the **"RelayerElections/SupportCandidateAsFoCServer"** API function (**"voteRelayerAsServiceProvider"** RSC function):

Figure 57: Scenario 4 - Staking in favour of candidate as FoC service provider

Using the **"RelayerElections/Candidates/{epoch}/Leaders"** API function, we can look at the current standings as far as the relay election is concerned. These will remain until the end of the voting period in the current epoch:

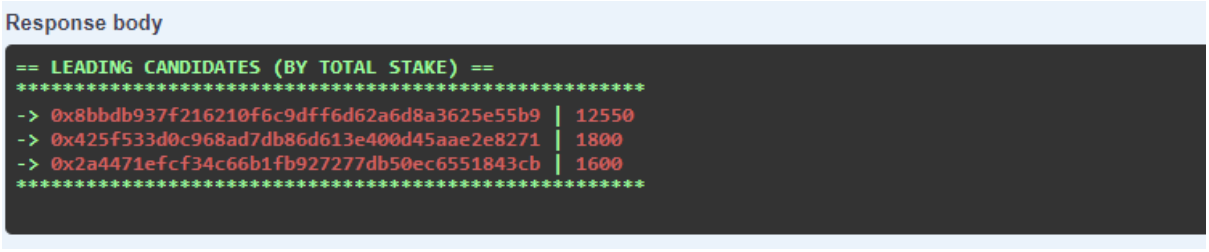

Figure 58: Scenario 4 - Total stakes by candidate

Eventually, the voting period in the epoch passes. Once this happens, each relay must register his total stake (from witnesses, FoC servers, etc.) to the RSC. The RSC will then place him in the final *Relay Shortlist*. The relay shortlist is a list where wall candidate relayers are shorted by their total stake. Based on this shortlist, the new relayers for the next epoch will be chosen, when this epoch passes.

Each of the candidate relays from the previous epoch must register to the final shortlist using the **"RelayerElections/Functions/RegisterToShortlist"** API function, which calls the **"requestLicense"** RSC function. In some cases, if a candidate has seen that he won't be among the first stakers, because many candidates with higher stakes have already registered ahead of him or because his maximum transactional throughput won't be covered by the chain's capacity, he can omit registering his total stake altogether and save the gas costs of the transaction.

Figure 59: Scenario 4 - Registering to final relay shortlist

The final shortlist is the following (can be acquired via the **"RelayerElections/Candidates/{epoch}/FinalShortlist"** API function):

Response body

```
Relay selection shortlist for epoch 22:
============================================================
Total applicants = 3 relay candidates
Transaction capacity for RE-CENT chain = 80 transactions per epoch

-------------------------------------------------------------------------------------------------------------------
|               Candidate address               |||  Stake (in coins)  |||   Max transaction throughput (transactions per epoch)  |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
|   0x8bbdb937f216210f6c9dff6d62a6d8a3625e55b9    |||             12550 |||                                                50 |
|   0x425f533d0c968ad7db86d613e400d45aae2e8271    |||              1800 |||                                                30 |
|   0x2a4471efcf34c66b1fb927277db50ec6551843cb    |||              1600 |||                                                30 |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Current relay election winners:
=======================================
0x8bbdb937f216210f6c9dff6d62a6d8a3625e55b9
0x425f533d0c968ad7db86d613e400d45aae2e8271
```

Figure 60: Scenario 4 - Final relay shortlist for epoch 22, plus extra details

We have now entered epoch 22. In order for the new relay shortlist to be calculated, any participant in the network must call the **"verifyRelayerSetChange"** RSC method (easily done through the **"RelayerElections/Functions/ChangeRelayers"** API method).

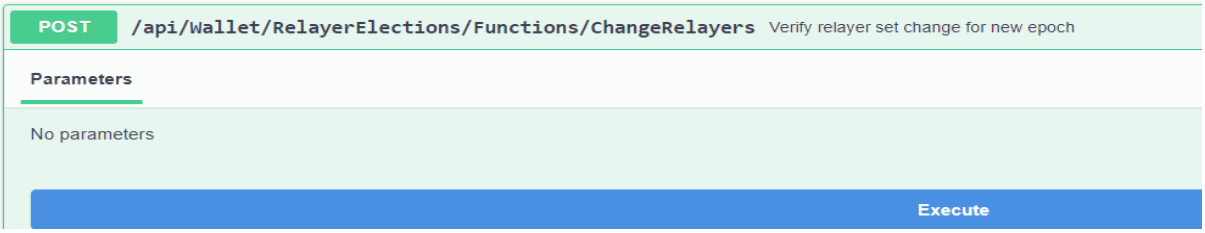

Figure 61: Scenario 4 - Verifying the relayer set change at start of epoch

The new relayers are calculated in the following way: The RSC first calculates the new transaction capacity for the chain. If no delayed payments were recorded in the previous epoch, the capacity increases by 10%. If delayed payments were recorded, the new capacity is equal to the old capacity minus the total amount of delayed payments. After the new transaction capacity is calculated, the RSC scans the shortlist, starting from the candidates with the top stakes and going to the candidates with the lower stakes. If a candidate's maximum transaction capacity is less than the capacity of the chain, he is granted a relay license, and the blockchain transaction capacity for this function is reduced by the candidate's transaction throughput. If a candidate's maximum transaction throughput passes the "current" capacity, he is not granted a license. In the end, all licensed relayers must have a combined maximum transactional throughput equal to or less than the total transaction capacity of the chain. The new relayers are the following:

Response body

```
==========================================================================================
 Price per Mb (in RE-CENT coins): 0.1
 Minimum validator stake (in RE-CENT coins): 400
 Minimum v-witness stake (in RE-CENT coins): 10
 Benign validator penalty percentage % (of initial validator stake):  1.6
 Malicious validator penalty percentage % (of initial validator stake):  50

== CURRENT INFO ==
==========================================================================================
 Current block:  7569
 Current epoch:  22
 Current epoch end (block):  7920
 Current epoch transaction capacity TCr (in number of Txs):  88
 Current epoch delayed payments counter (in number of Txs):  0
 Current validators election end (block):  7890
 Current relayers election end (block):  7860
 Current block reward (in RE-CENT coins):  0

 -> Current validators:
============================================================
 0x1e34c3a6a54b5a1fbebfb84be88c1418e3db4639
 0x6d0d56de06ef88755f8d0dae81c521698e49b3f3
 0x9a551dc0e052a81c831aa2c7b70564ad687d53cd

 -> Current relayers:
============================================================
 0x8bbdb937f216210f6c9dff6d62a6d8a3625e55b9
 0x425f533d0c968ad7db86d613e400d45aae2e8271
```

Figure 62: Scenario 4 - Relayers for new epoch

We notice that Client1 did not make the final shortlist. This is because his maximum transactional throughput was 30, but Relayer1 and Relayer2 already amounted for a combined maximum transactional throughput of 80. Therefore, Client1 could not receive a valid license, because the chain's transaction capacity limit would be violated. We also notice that the total transaction capacity increased for the new epoch, because there were no delayed payments in the previous epoch.

A random network participant like Client2 can now compare the newly licensed relayers of this epoch using the **"RelayerInfo"** API function, and choose the one that fits him most.

Response body

```
-> Sorting relayers based on parameter -- None
-> Sorting order -- None


Relayers in system: 2
************************************************************************
Details:
========================================================================
Relayer address: 0x8bbdb937f216210f6c9dff6d62a6d8a3625e55b9
------------------------------------------------------------------------
Name: Relay1
Domain: http://192.168.1.10:8080
Relay fee: 0.002 RE-CENT coins per Tx
Relay delay: 100 blocks
Delayed payments: 0
Coins inbound: 0 RE-CENT coins (0% coverage)
Coins outbound: 0 RE-CENT coins (0% coverage)
Current users: 0 (0% coverage)
Current gas throughput: 0 gas (0% of maximum allowed in this regulation period)

Balance on relayer: 0 RE-CENT coins
Balance expiration block: 0


Relayer address: 0x425f533d0c968ad7db86d613e400d45aae2e8271
------------------------------------------------------------------------
Name: Relay2
Domain: http://192.168.1.10:8081
Relay fee: 0.01 RE-CENT coins per Tx
```

Figure 63: Scenario 4 - Relay comparison function

A big difference in the relay election compared to the validator election is that witnesses and free-of-charge servers can withdraw their stakes after the election epoch, not two epochs after. They can do this with the API functions **“RelayerElections/WithdrawWitnessStake” and “RelayerElections/WithdrawFoCServerStake”**:

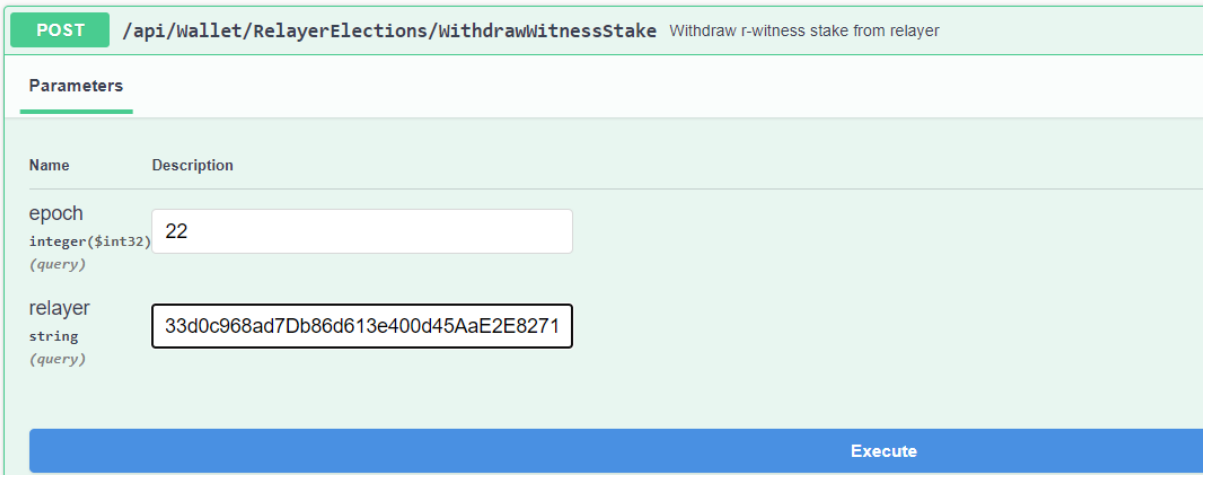

Figure 64: Scenario 4 - Withdrawing witness stake from relay

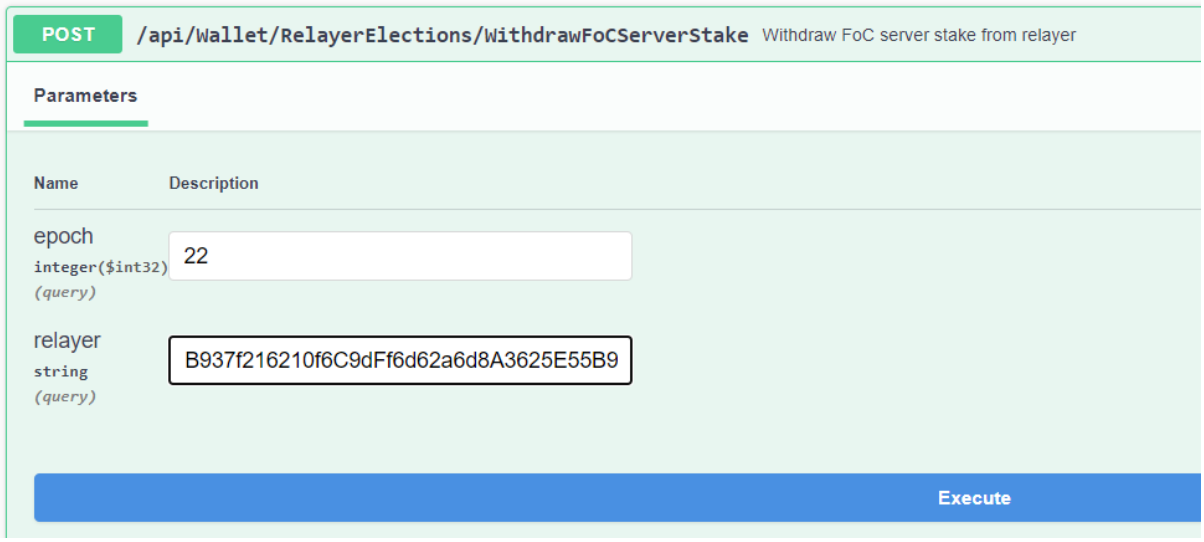

Figure 65: Scenario 4 - Withdrawing FoC server stake from relay

Similar to the validator election, witnesses who supported winning candidates can claim their rewards with the **"RelayerElections/GetWitnessReward"** API function. IMPORTANT: Witnesses must first claim their witness reward and then withdraw their stake. If witnesses withdraw their stake first, they won't be able to claim their reward.

After the end of epoch 22, we can use the **"Info/Summary"** API function to notice that the aforementioned election winners don't have valid relay licenses any more. This is expected, since relay licenses only last for one epoch.

Response body

```
== RE-CENT CHAIN ADJUSTABLE PARAMETERS ==
================================================================================================================
 Price per Mb (in RE-CENT coins): 0.1
 Minimum validator stake (in RE-CENT coins): 400
 Minimum v-witness stake (in RE-CENT coins): 10
 Benign validator penalty percentage % (of initial validator stake):  1.6
 Malicious validator penalty percentage % (of initial validator stake):  50

== CURRENT INFO ==
================================================================================================================
 Current block:  7921
 Current epoch:  23
 Current epoch end (block):  8280
 Current epoch transaction capacity TCr (in number of Txs):  88
 Current epoch delayed payments counter (in number of Txs):  0
 Current validators election end (block):  8250
 Current relayers election end (block):  8220
 Current block reward (in RE-CENT coins):  0

 -> Current validators:
============================================================
 0x1e34c3a6a54b5a1fbebfb84be88c1418e3db4639
 0x6d0d56de06ef88755f8d0dae81c521698e49b3f3
 0x9a551dc0e052a81c831aa2c7b70564ad687d53cd

 -> Current relayers:
============================================================
```

Figure 66: Scenario 4 - Start of next epoch / relay licenses have expired

Relayer1, Relayer2 and Client1 can now withdraw their relay stakes, using the **"RelayerElections/WithdrawRelayerStake"** API function (**"relayerWithdrawRequest"** RSC function).

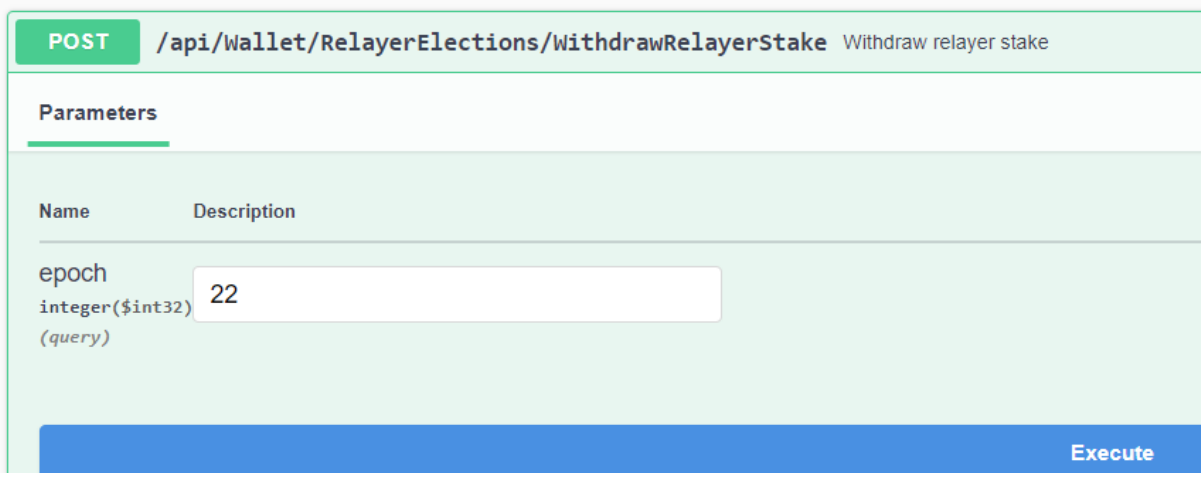

Figure 67: Scenario 4 - Withdrawing relay stake

The election mechanism works as designed.

5.3.5 Scenario 5 – Relay transaction aggregation:

This scenario examines the transaction aggregation functionality of the RSC. We assume that Relayer1 and Relayer2 have won the relay election, and now are whitelisted relayers by the RSC. Client1 and Client2 want to buy some video content from ServiceProvider1. They propose using Relay1, to which ServiceProvider1 agrees to. Before video delivery content starts though, a few things must happen:

5.3.5.1 Scenario summary

For this scenario, the following steps will be followed:

1.) Client1 compares the available relayers.

2.) Client1 deposits funds to Relayer1 (inbound channel).

3.) Client1 requests a new session with ServiceProvider1.

4.) Client2 deposits funds to Relayer1 (inbound channel).

5.) Client2 requests a new session with ServiceProvider1.

6.) Relayer1 deposits funds to ServiceProvider1 (outbound channel).

7.) ServiceProvider1 approves the timeplans from Client1 and Client2 and initiates the sessions.

8.) Session payments are forwarded through the relay.

9.) Before the transactions expire, Relayer1 updates ServiceProvider1's tab on-chain.

10.) Review of Relayer1's running parameters.

11.) ServiceProvider1 withdraws his tab to his account.

12.) Relayer1 withdraws the funds from his outbound channel with ServiceProvider1.

13.) Shortly before the epoch ends, Relayer1 updates the inbound channels of Client1 and Client2.

14.) After the epoch has finished, Client1 and Client2 withdraw their funds from their inbound channels with Relayer1.

5.3.5.2 Runtime

In order to track the chain parameters and the VSC and RSC variables, we use the WebAPI. For the balance tracking of our test nodes, we can use Metamask.

We assume that Relayer1 and Relayer2 have already been granted licenses for this epoch, with the specs defined in the above scenario.

These are the initial balances of all nodes involved in this scenario:

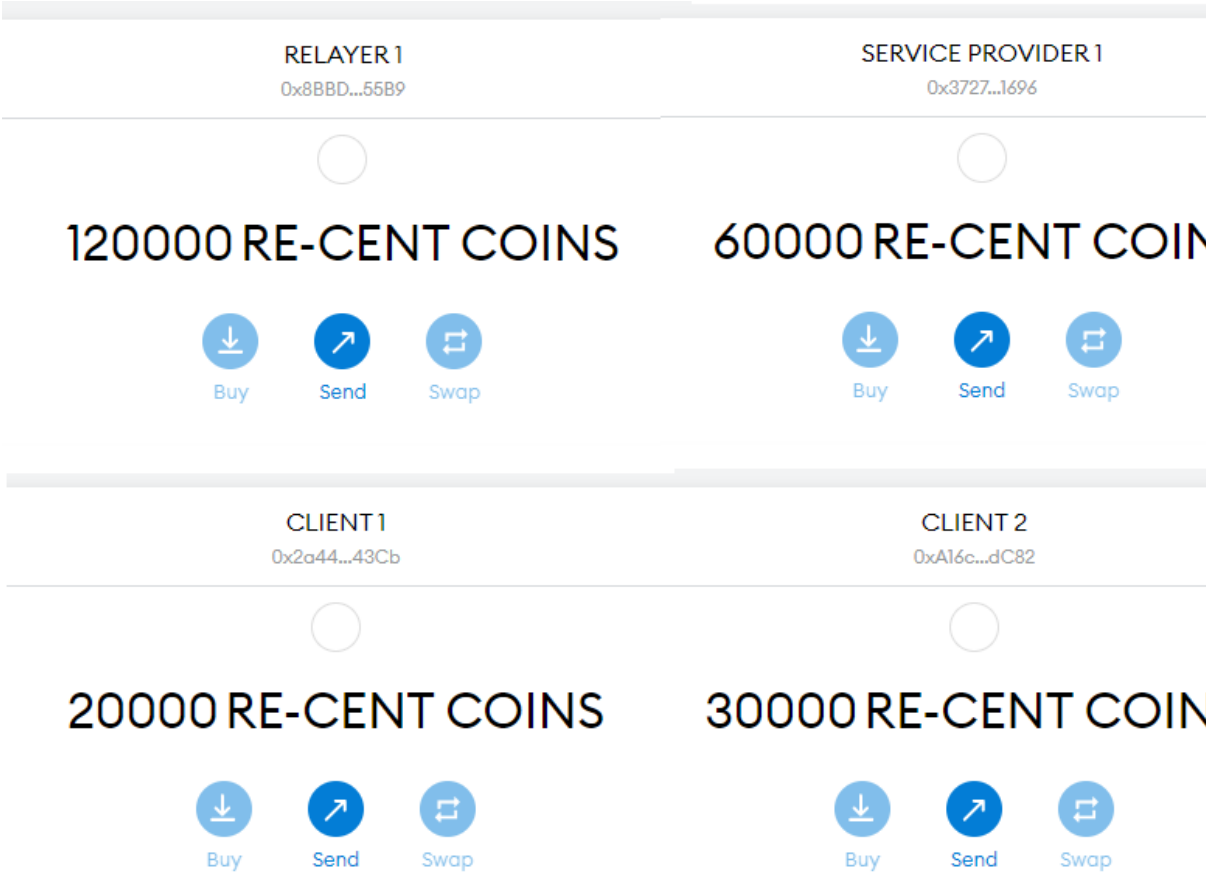

Figure 68: Scenario 5 - Initial balances

We assume that Client1 wants to watch a video. This video is stored by ServiceProvider1, therefore Client1 must communicate with him off-chain and come to an agreement with him about a timeplan. A timeplan is an arrangement between a service provider and a client that defines the time period between micropayments, the duration of the video chunks sent to the client by the service provider (or their size in Mbs) and the value of each micropayment. The

timeplan also defines the relay that will be used for the process.

A random client can use the “Wallet/RelayerInfo” API function to compare relays based on multiple parameters, such as delayed payments, fees, etc.

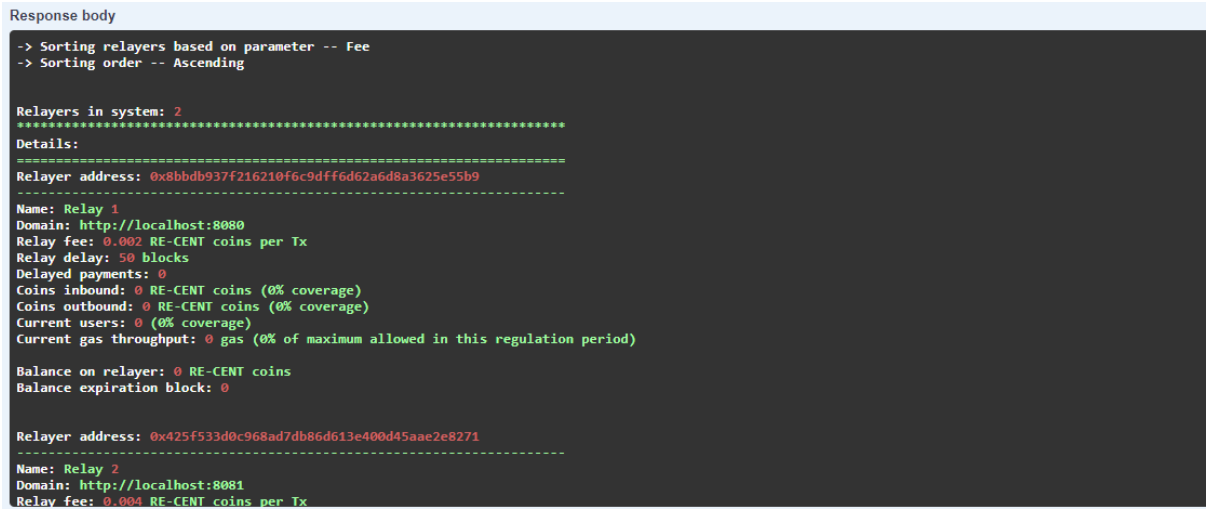

Figure 69: Scenario 5 - Relay comparison

Client1 wants to pick a relay with low fees, therefore he will use Relayer1.

In order to use Relayer1 for his sessions, he must first deposit some coins there. All deposits to relayers last until the end of the epoch. This means the epoch must pass before the client can withdraw his money. The API function the client uses to deposit money to a relayer is **“/Relayer/{address}/Deposit”** (calls the **“depositToRelayer”** RSC function). Client1 deposits 20 coins to Relayer1.

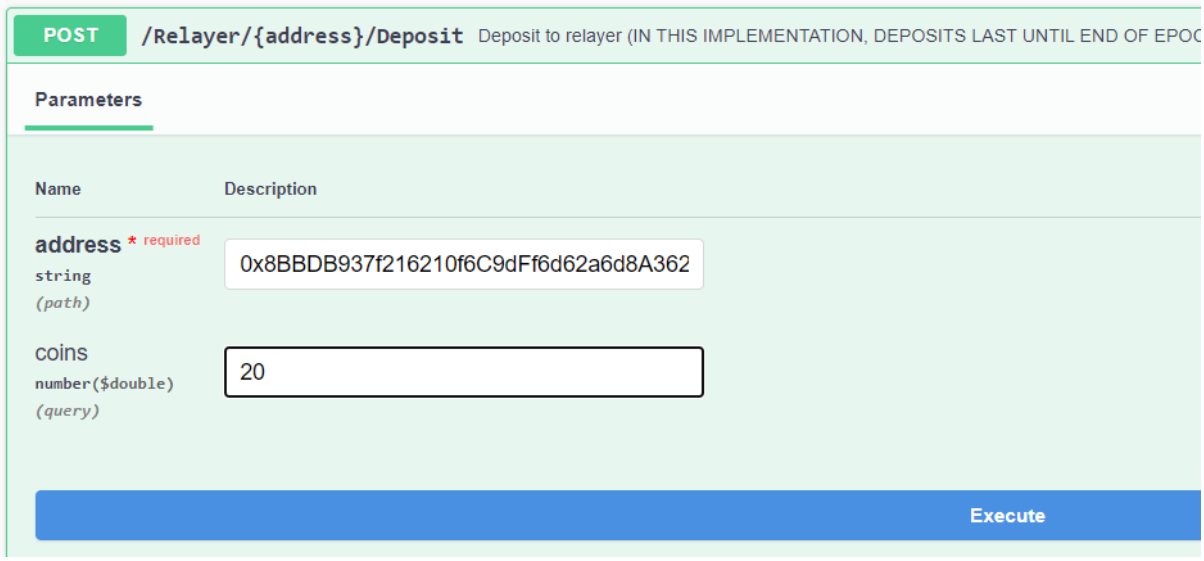

Figure 70: Scenario 5 - Depositing funds to relay

Client1 can now send the details of his video session to ServiceProvider1. In his off-chain message he must include the public address of the service provider, the public address of the relay he'll use, the duration of the video he'll watch, the duration of each micropayment, the cost of each micropayment, and the IP addresses of himself and the service provider. This is done through the **"Wallet/StartSession"** API function. The service provider receives this timeplan in his database, and he can choose to reject or accept it.

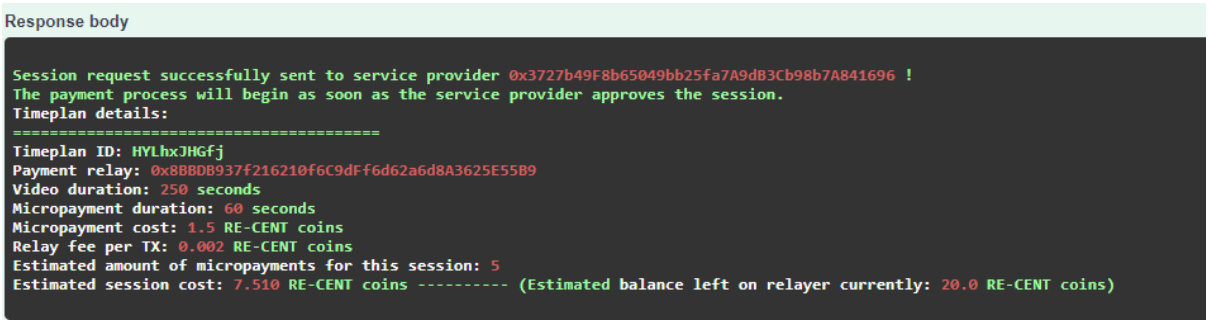

Figure 71: Scenario 5 - Client1/ServiceProvider1 session details

Client2 also wants to watch a video from ServiceProvider1. He follows a similar procedure to the one above. He deposits 10 coins to Relayer1. The details of his timeplan with ServiceProvider1 are the following:

Response body

```
Session request successfully sent to service provider 0x3727b49F8b65049bb25fa7A9dB3Cb98b7A841696 !
The payment process will begin as soon as the service provider approves the session.
Timeplan details:
=====================================
Timeplan ID: hdftzauagn
Payment relay: 0x8BBDB937f216210f6C9dFf6d62a6d8A3625E55B9
Video duration: 195 seconds
Micropayment duration: 20 seconds
Micropayment cost: 0.1 RE-CENT coins
Relay fee per TX: 0.002 RE-CENT coins
Estimated amount of micropayments for this session: 10
Estimated session cost: 1.020 RE-CENT coins ---------- (Estimated balance left on relayer currently: 10.0 RE-CENT coins
```

Figure 72: Scenario 5 - Client2/ServiceProvider1 session details

In order for the sessions to start taking place, apart from the service provider's approval, Relayer1 must also deposit funds to ServiceProvider1. In more advanced implementations of the relay service this could be done intelligently using data or other mechanisms, but for the sake of this scenario we'll do it manually, using the **"/relayer/Server/{address}/Deposit"** API function (calls the **"depositToServer"** RSC function). The relay's total outbound channel balance must be less than his inbound channel balance at all times. This means that Relayer1 cannot deposit more than 20+10=30 coins to ServiceProvider1. For this scenario, he'll deposit 25 coins, which are more than enough.

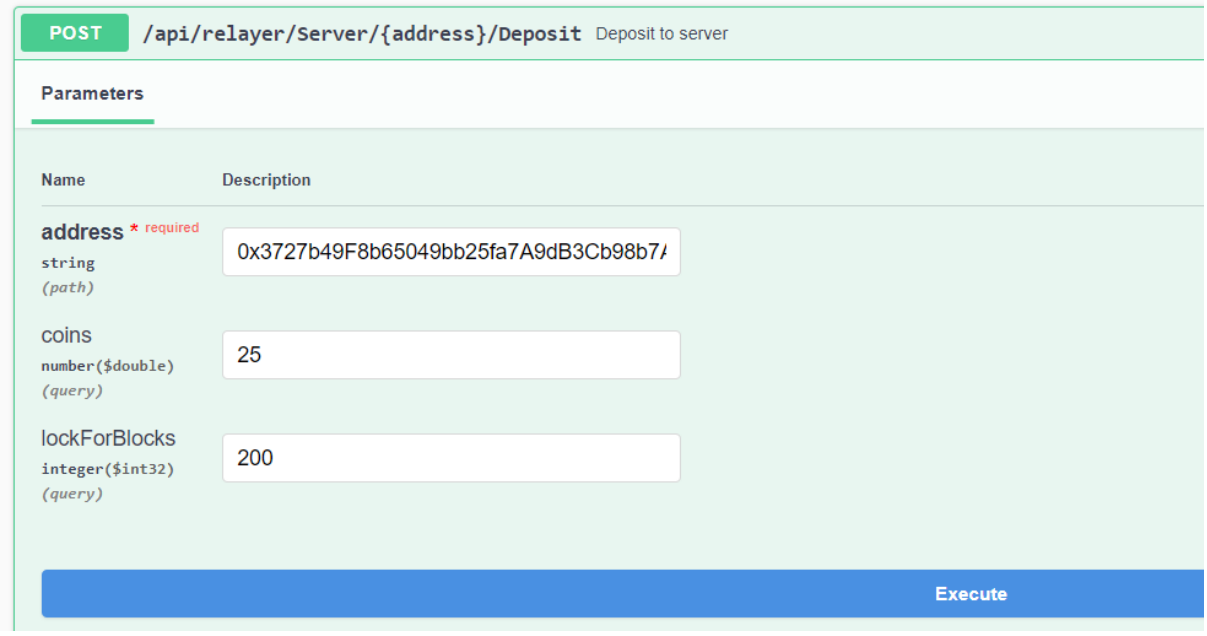

Figure 73: Scenario 5 - Relay deposits funds to server

Since both the inbound channels from the clients to the relayer and the outbound channel from

the relayer to the service provider is adequately funded, the session can begin. The session begins when the service provider approves timeplans sent to him by the clients. Any service provider can inspect the timeplans sent to him with the **"serviceprovider/TimeplanSummary"** API function, then filter them by cost or client, etc.

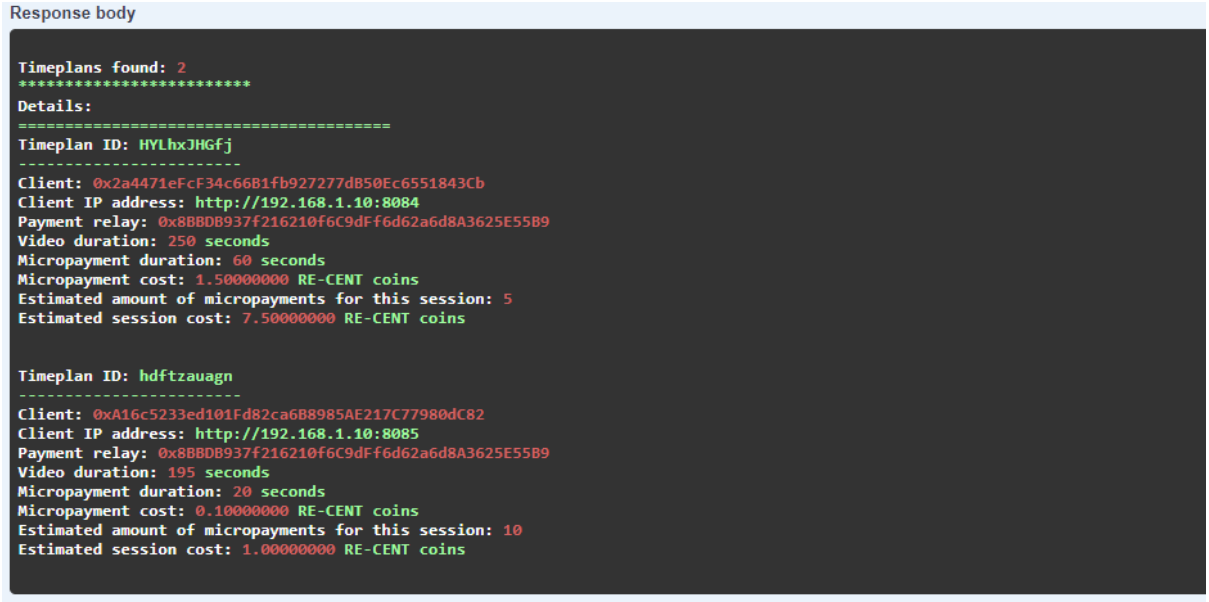

Figure 74: Scenario 5 - Session requests for ServiceProvider1

But to approve a timeplan, a service provider must use the **"serviceprovider/ApproveTimeplan"** API function, and provide the client and the timeplan ID as inputs. Once a timeplan is approved, the service provider sends an acknowledgement to the client, and the session can begin. ServiceProvider1 in our scenario initiates sessions for Client1 and Client2.

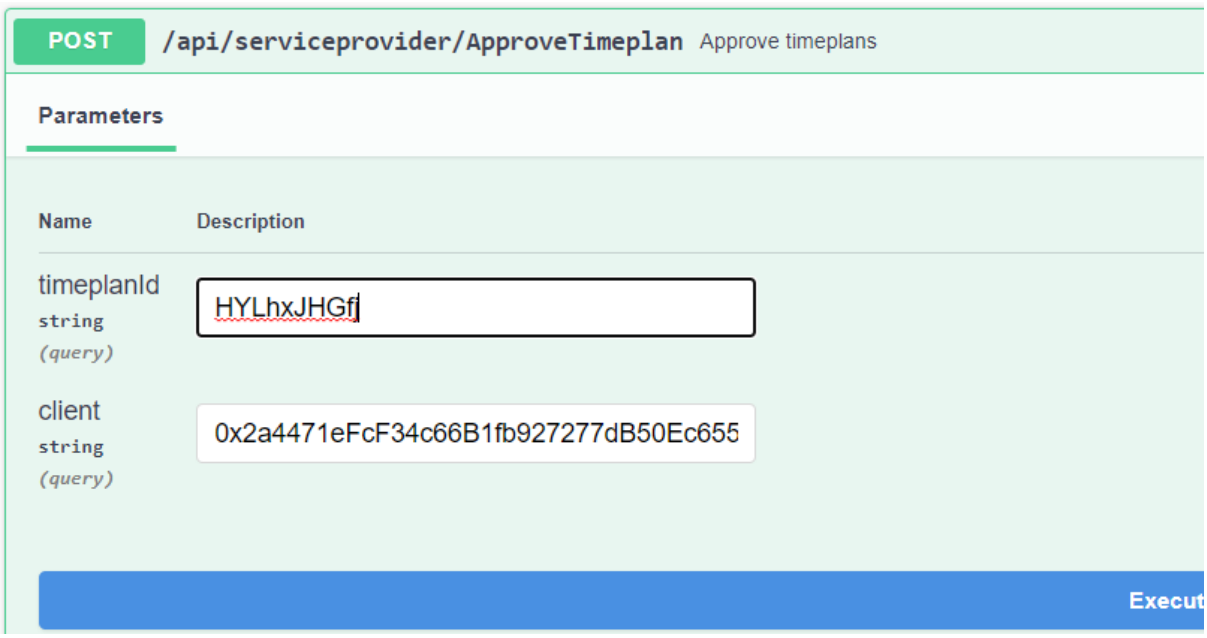

Figure 75: Scenario 5 – *Service provider approves session requests*

A session goes as following: The service provider sends a video chunk to a client (unimplemented in this demo). Once he finishes sending it, he then creates a payment request for this video chunk using the timeplan details, then signs it and sends it to the relayer. The relayer then forwards the request to the client, who decides whether to pay it or ignore it. If he ignores it, the session is aborted and the service provider stops sending video chunks to the client. If the client chooses to respond to the payment request, he must formulate a payment by using the request data, the timeplan data and his signature. He then forwards the payment to the relay who in turn forwards it to the service provider. If the service provider is happy with the payment he received, he can send more video chunks to the client, and repeat the process until all video chunks are sent to the client and the session is complete.

This process is done automatically in this demo. Each node involved in the RE-CENT ecosystem has a specific service running, based on their role in the network (client, service provider, relay).

The **client service** does the following:

1. Scans incoming payment requests, only considers those tied to active sessions.
2. Forwards payments to relays based on incoming requests.
3. Scans blockchain for relay misbehaviour against him (mainly fund theft).
4. Initiates dispute procedure against dishonest relayers.

The **service provider** service does the following:

1. Inspects active sessions.

2. Forwards payment request to relays for clients fulfilling their payment obligations.

3. Scans the blockchain for relay misbehaviour against him (mainly delayed or incorrect payments).

4. Initiates dispute procedures against dishonest relayers.

5. Scans the blockchain for outbound channels opened with him by relayers.

The **relay service** does the following:

1. Forwards payment requests to clients.

2. Forwards payments to service providers.

3. Aggregates payments directed towards the same server based on his relay delay.

4. Updates the on-chain balance of service providers.

5. Updates inbound and outbound channels.

6. Handles disputes against him based on a specific logic.

7. Scans the blockchain for new inbound payment channels and disputes against him.

In the picture below, we see how these services facilitate the payment flow as described above. No on-chain transactions have taken place yet, everything is off-chain.

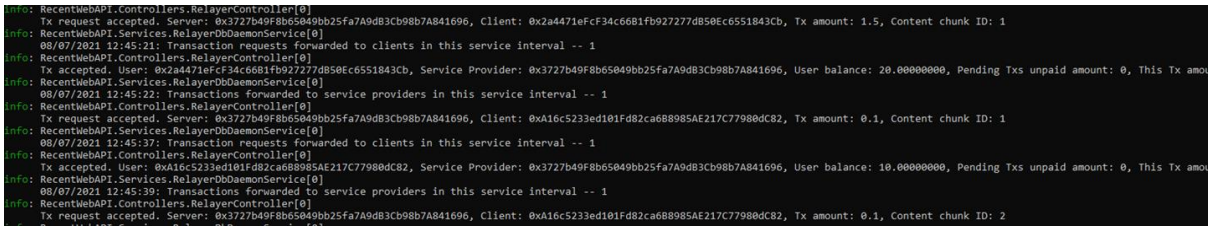

Figure 76: Scenario 5 - The relay service forwards transactions and requests

At some point, the sessions are concluded. A small peek in the relay's database proves that all micropayments were forwarded correctly, since there are 15 micropayments in the relayer's transaction database. This is expected because Client1's timeplan included 5 micropayments (incomplete micropayments are considered full) and Client2's timeplan included 10 micropayments.

	Id	TransactionId	TxReceivedBlock	TxStatusChang...	TxLastException	TxUntilBlock	TxHash	TxHashByte	TxAmount	User	Relay
▶	547	1625748321-503...	116	157	NULL	166	eTzu/Hi5t9HVN...	0x793CEEFC78B...	1.50000000	0x2a4471eFcF3...	0x8BBDB937f21...
	548	1625748337-507...	119	157	NULL	169	n9WNmPq202R...	0x9FD58D98FA...	0.10000000	0xA16c5233ed1...	0x8BBDB937f21...
	549	1625748359-505...	124	157	NULL	173	kvXUnn+t0s2w...	0x92F5D49E7FA...	0.10000000	0xA16c5233ed1...	0x8BBDB937f21...
	550	1625748381-501...	128	157	NULL	178	Fcv3msqB4KO8...	0x15CBF79ACA...	0.10000000	0xA16c5233ed1...	0x8BBDB937f21...
	551	1625748383-505...	128	157	NULL	178	lXeklSzD/mfkYz...	0x2177A4212C...	1.50000000	0x2a4471eFcF3...	0x8BBDB937f21...
	552	1625748403-505...	133	157	NULL	182	mMxrDuveaDY...	0x98CC6B0EEB...	0.10000000	0xA16c5233ed1...	0x8BBDB937f21...
	553	1625748425-505...	137	157	NULL	187	67BbMxIlYxpw...	0xEBB05B33122...	0.10000000	0xA16c5233ed1...	0x8BBDB937f21...
	554	1625748445-502...	141	157	NULL	190	mn80IjyZfMDv...	0x9A7F34963C9...	1.50000000	0x2a4471eFcF3...	0x8BBDB937f21...
	555	1625748447-508...	141	157	NULL	191	AEv4SLMTQM...	0x004BF848B31...	0.10000000	0xA16c5233ed1...	0x8BBDB937f21...
	556	1625748469-503...	146	157	NULL	195	rJ3sXAJ4ATsagx...	0xAC9DEC5C02...	0.10000000	0xA16c5233ed1...	0x8BBDB937f21...
	557	1625748491-509...	150	157	NULL	200	rgHtVWYGqBB...	0xAE01ED55660...	0.10000000	0xA16c5233ed1...	0x8BBDB937f21...
	558	1625748507-505...	153	157	NULL	203	zEvw1VOEIDa6ij...	0xCC4BF0D553...	1.50000000	0x2a4471eFcF3...	0x8BBDB937f21...
	559	1625748513-509...	155	157	NULL	204	C/dPlQVB0Bm3...	0x0BF74F21054...	0.10000000	0xA16c5233ed1...	0x8BBDB937f21...
	560	1625748535-506...	159	159	NULL	209	u3ZNcxwRkBcR...	0xBB764D731C...	0.10000000	0xA16c5233ed1...	0x8BBDB937f21...
	561	1625748569-509...	166	166	NULL	215	huetpbBkm5vw...	0x86E7ADA5B0...	1.50000000	0x2a4471eFcF3...	0x8BBDB937f21...
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 77: Scenario 5 - Transaction database for relay (15 entries expected)

Now the relay service must aggregate payments, before posting the on-chain update. Each transaction must be posted on-chain before the “txUntilBlock” interval, which is basically the moment the relay received the payment request plus the relay delay. The relay service is designed around aggregating similar payments (meaning, directed at the same service provider), despite the client who forwarded it and their deadline. Therefore, the relay can aggregate many payments into one payment and post an on-chain balance update for ServiceProvider1. Since in this simulation the relay’s delay is only 50 blocks, we needed 2 on-chain updates to include all transactions to the service provider. The on-chain transactions and relevant console outputs are seen below:

```
2021-07-08 12:48:45  Imported #157 0x6207…0968 (0 txs, 0.00 Mgas, 0 ms, 0.55 KiB)
2021-07-08 12:48:50  Imported #158 0xdb04…fd15 (1 txs, 0.17 Mgas, 3 ms, 0.94 KiB)
2021-07-08 12:48:55  Imported #159 0xc798…c8ac (0 txs, 0.00 Mgas, 0 ms, 0.55 KiB)
2021-07-08 12:49:00  Imported #160 0x5740…df65 (0 txs, 0.00 Mgas, 0 ms, 0.55 KiB)
```

●
●
●

```
2021-07-08 12:52:20  Imported #200 0x84a5…15af (1 txs, 0.12 Mgas, 10 ms, 0.94 KiB)
2021-07-08 12:52:25  Imported #201 0x2a0f…8793 (0 txs, 0.00 Mgas, 0 ms, 0.55 KiB)
2021-07-08 12:52:30  Imported #202 0x0d84…dedd (0 txs, 0.00 Mgas, 0 ms, 0.55 KiB)
```

Figure 78: Scenario 5 - Service provider balance updates confirmed on-chain

Figure 79: Scenario 5 - Relay service (service provider on-chain updates)

Since the relayer posted an on-chain transaction, we can see that his gas throughput has increased for this regulation period. Additionally, his outbound channel balance was decreased, since it was expended in an on-chain update.

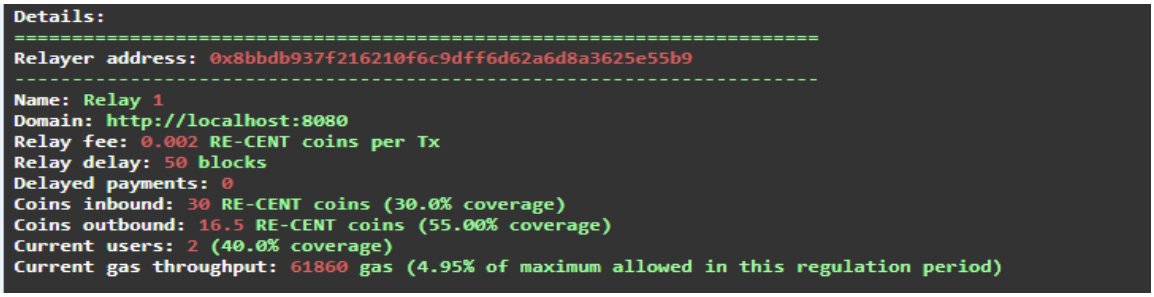

Figure 80: Scenario 5 - New relay data after service provider balance updates

To verify that he received all his due payments, ServiceProvider1 can inspect his tab using the **"ServerTab/Info"** API function.

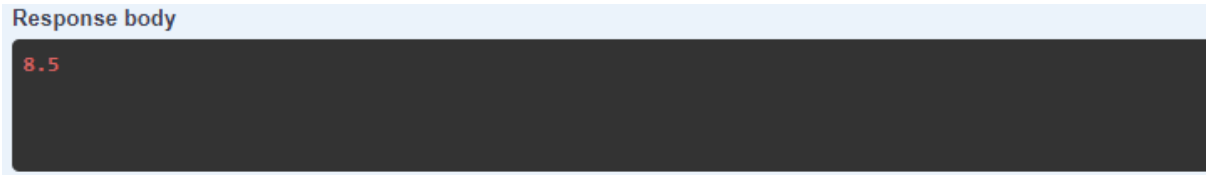

Figure 81: Scenario 5 - Service provider tab after updates

The service provider received the money he expected (5 micropayments * 1.5 coins + 10 micropayments * 0.1 coins = 8.5 coins), therefore we can deduce that all parties behaved in an honest way. The service provider can then either ignore his tab and withdraw it at a later date, or withdraw his funds at once using the **"ServerTab/Withdraw"** API function.

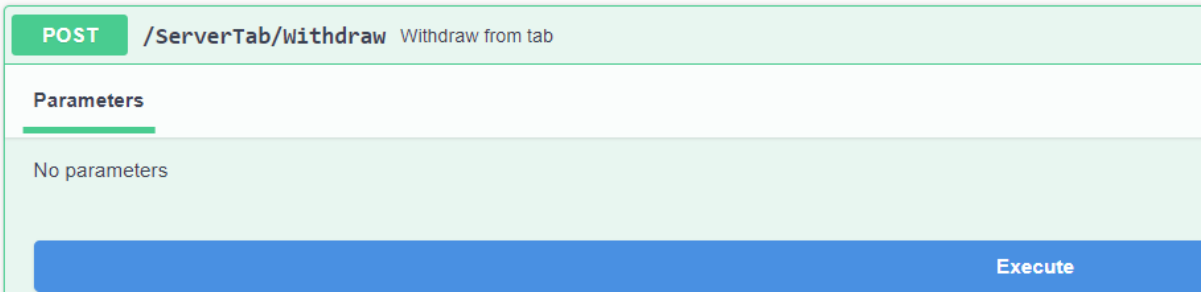

Figure 82: Scenario 5 - Service provider withdrawing tab funds

Since the epoch is drawing to an end and the outbound channel to ServiceProvider1 has expired, Relayer1 can withdraw his funds (partly or the entirety) using the **"Server/{address}/Withdraw"** API function (calls the **"withdrawFromServer"** RSC function).

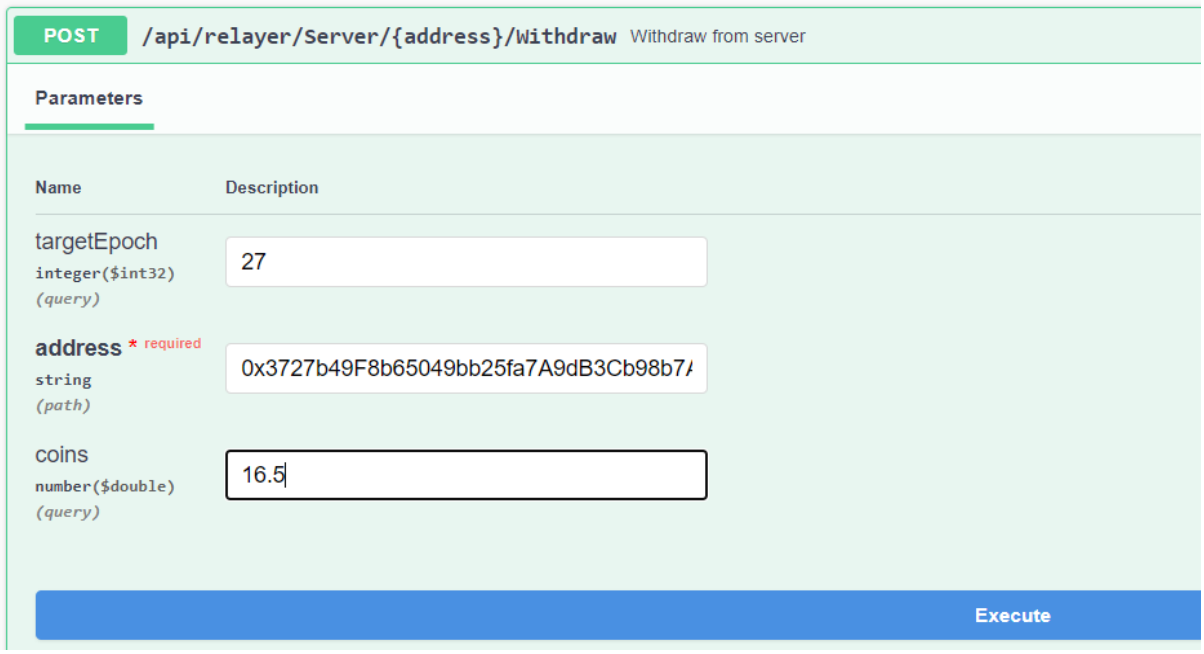

Figure 83: Scenario 5 - Relay withdrawing remaining funds from outbound channel

Before the epoch ends, the relay service also updates the inbound channels. In case this doesn't happen, the clients can withdraw their funds intact, meaning that they never paid for their transactions. Since clients can only withdraw their funds after the epoch is over, it makes sense that the relay waits before updating inbound channels until the very end of an epoch (but NOT after).

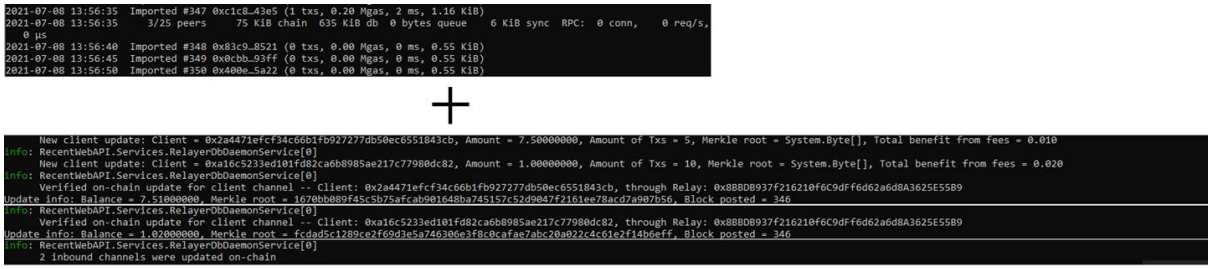

Figure 84: Scenario 5 - Relay updates inbound channels

Since the relay forwarded 15 micropayments, his total relay fee gain will be relayFee * 15 (in our case, 15*0.002 = 0.03).

After the epoch is over, both Client1 and Client2 can withdraw their unspent funds from their inbound channel with Relayer1, using the **“Relayer/{address}/Withdraw”** API function (calls the **“withdrawFundsFromRelayer”** RSC function).

The final balance of all nodes involved is the following. We notice that ServiceProvider1 is richer by 8.5 RE-CENT coins (the combined cost of the sessions he offered), the relay is richer by 0.03 RE-CENT coins which is equal to the fees he claimed, and each of the clients has spent an amount of money equal to the sessions he initiated, plus the relay fees.

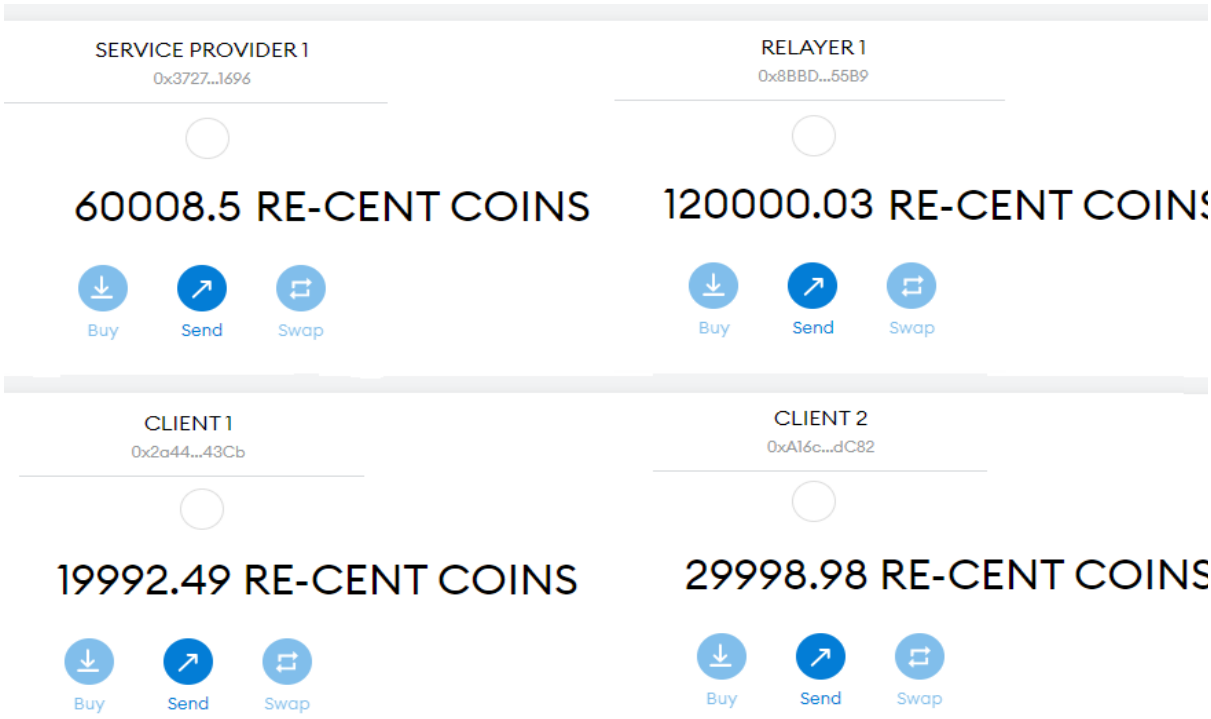

Figure 85: Scenario 5 - Final balances

This is a typical example of payment aggregation in the RE-CENT chain.

5.3.6 Scenario 6 – Onchain dispute mechanism – Delayed payments:

Scenarios 6 and 7 examine the dispute mechanisms of the RSC. Scenario 6 handles service provider – relay disputes and scenario 7 handles client – relay disputes.

In scenario 6, we investigate the possibility where a service provider and a relay often engage in disputes about the timely payment of the service provider. In multiple occasions, the service provider realizes that his added balance on-chain is less than the sum of signed transactions he has received from the relay. Therefore, he deduces that the relay has omitted posting on-chain some of the transactions he agreed to post. Thus, the relay owes money to the service provider.

In another situation, a service provider decides to report illicit behaviour from a relay, even though none has taken place. The relay picks up on the service provider's unfounded reports, and verifies the validity of his activities using specific RSC methods.

After a number of lost disputes, the relay's penalty fund is depleted. In situations like these, the RSC decides to revoke the relay's license, meaning that RE-CENT nodes can no longer use this relay for their sessions.

In conclusion, the RSC gives the ability to service providers who haven't received the payments they should to report dishonest relay behaviour, and get their money back, in addition to the gas costs they paid to report the dishonest behaviour on-chain. Additionally, relays who have done no wrongdoing are given the ability to defend themselves from disputes by dishonest service providers. If a relay's penalty fund has been depleted, then his license is automatically revoked by the RSC, and a relevant event is emitted.

All disputes must be resolved in the dispute resolution window (D_R).

5.3.6.1 Scenario summary

For this scenario, the following steps will be followed:

1.) Client1 initiates his first session with ServiceProvider1, using Relayer1 as the relay.

2.) Relayer1 forwards the payments, but goes offline before updating ServiceProvider1's on-chain balance.

3.) ServiceProvider1 initiates an on-chain dispute with Relayer1 to claim his rightful funds.

4.) Relayer1 never posted the update as he should, therefore loses the dispute automatically.

5.) ServiceProvider1 claims his refund, alongside the gas costs he paid for the dispute.

6.) Review of Relayer1's running parameters.

7.) Client1 initiates his second session with ServiceProvider1, using Relayer1 as the relay.

8.) Relayer1 forwards the payments, and updates ServiceProvider1's on-chain balance on time.

9.) ServiceProvider1 initiates an on-chain dispute with Relayer1 for the latest session, even though his on-chain balance was properly updated.

10.) Relayer1 responds to the dispute and proves that he posted the disputed transactions on-chain.

11.) Relayer1 is the winner of the dispute, and ServiceProvider1 gets no refunds.

12.) Client1 initiates his third session with ServiceProvider1, using Relayer1 as the relay.

13.) Relayer1 forwards the payments, but goes offline before updating ServiceProvider1's on-chain balance.

14.) ServiceProvider1 initiates an on-chain dispute with Relayer1 to claim his rightful funds. Since Relayer1 is offline, ServiceProvider1 wins the dispute automatically and claims his refund.

15.) Client1 initiates his fourth session with ServiceProvider1, using Relayer1 as the relay.

16.) Relayer1 forwards the payments, but goes offline before updating ServiceProvider1's on-chain balance.

17.) ServiceProvider1 initiates an on-chain dispute with Relayer1 to claim his rightful funds. Since Relayer1 is offline, ServiceProvider1 wins the dispute automatically and claims his refund.

18.) Review of the new relayer set.

19.) Review of Relayer1's current parameters.

5.3.6.2 Runtime

In order to track the chain parameters and the VSC and RSC variables, we use the WebAPI. For the balance tracking of our test nodes, we can use Metamask.

For this scenario we assume that B_R = 36000 blocks. We increase the epoch duration to make sure that no disputes will span multiple epochs, for simplicity purposes.

We assume that Relayer1 has already acquired a valid relay license for this epoch. We also assume that Client1's inbound channel to Relayer1 and Relayer1's outbound channel to ServiceProvider1 are both adequately funded for a session to take place between them.

Using the **"Info/Summary"** API function, we notice that Relayer1 indeed has a valid license at the start of the scenario:

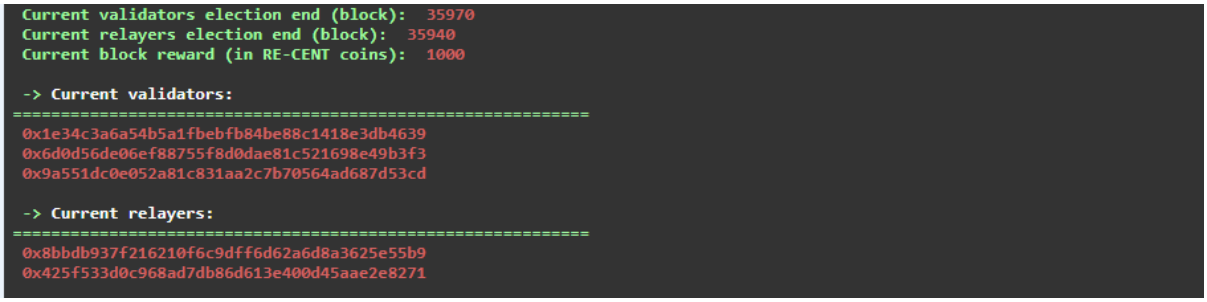

Figure 86: Scenario 6 - Valid relayers at the start of the scenario

This is ServiceProvider1's tab at the start of the scenario:

Response body

```
0
```

Figure 87: Scenario 6 - Initial ServiceProvider1 tab

The relay's complete specs are seen below. The **"Relayers/{epoch}"** API function is used. For this scenario we set a pretty low penalty fund, so it can be depleted after 2-3 lost disputes and the relay license revoking mechanism can be showcased.


```
{
  "name": "Relay 1",
  "owner": "0x8bbdb937f216210f6c9dff6d62a6d8a3625e55b9",
  "domain": "http://localhost:8080",
  "maxUsers": 5,
  "maxCoins": 100,
  "maxGasThroughput": 1500000,
  "offchainTxDelay": 50,
  "fee": 0.002,
  "relayerStake": 212,
  "currentUsers": 0,
  "currentCoinsInbound": 0,
  "currentCoinsOutbound": 0,
  "currentGasThroughput": 0,
  "lastBlockInSubepoch": 0,
  "delayedPayments": 0,
  "clientMirrorFund": 100,
  "serverMirrorFund": 100,
  "remainingPenaltyFunds": 12
},
```


Figure 88: Scenario 6 - Initial Relayer1 specs

We assume that at some point during the epoch, ServiceProvider1 approves a timeplan from Client1, and a session between them commences. The details of the timeplan are the following:

Response body

```
Session request successfully sent to service provider 0x3727b49F8b65049bb25fa7A9dB3Cb98b7A841696 !
The payment process will begin as soon as the service provider approves the session.
Timeplan details:
=======================================
Timeplan ID: yHMXG0Y23h
Payment relay: 0x8BBDB937f216210f6C9dFf6d62a6d8A3625E55B9
Video duration: 60 seconds
Micropayment duration: 100 seconds
Micropayment cost: 1 RE-CENT coins
Relay fee per TX: 0.002 RE-CENT coins
Estimated amount of micropayments for this session: 1
Estimated session cost: 1.002 RE-CENT coins ---------- (Estimated balance left on relayer currently: 50 RE-CENT coins)
```

Figure 89: Scenario 6 - First session details

The session goes on like normal, and the payment is forwarded to the service provider. This is a prerequisite for a successful dispute effort, since payments that haven't been signed by the relay or haven't been forwarded to the service provider cannot be disputed.

```
info: RecentWebAPI.Services.ServiceProviderObDaemonService[0]
      Tx request accepted by relay and stored to server database: Client = 0x2a4471eFcF34c66B1fb927277dB50Ec6551843Cb, Relay = 0x8BBDB937f216210f6C9dFf6d62a6d8A3625E55B9, Beneficiary = 0x3727b49F8b65049bb25fa7A9dB3Cb98b7A84
tionBlock = 457, Amount = 1, Content Chunk ID = 1
info: RecentWebAPI.Services.ServiceProviderObDaemonService[0]
      Timeplan "yHMXG0Y23h" updated -- 1 micropayments sent to client "0x2a4471eFcF34c66B1fb927277dB50Ec6551843Cb", (0 micropayments left)
 Previous payment timestamp = 01/01/0001 00:00:00, Current payment timestamp = 15/07/2021 11:17:09
info: RecentWebAPI.Services.ServiceProviderObDaemonService[0]
      Timeplan "yHMXG0Y23h" is complete -- Session over.
info: RecentWebAPI.Services.ServiceProviderObDaemonService[0]
      15/07/2021 11:17:09: Transaction requests forwarded to relays in this service interval -- 1
info: RecentWebAPI.Controllers.ServiceProviderController[0]
      Tx accepted by server: Client = 0x2a4471eFcF34c66B1fb927277dB50Ec6551843Cb, Relay = 0x8BBDB937f216210f6C9dFf6d62a6d8A3625E55B9, Beneficiary = 0x3727b49F8b65049bb25fa7A9dB3Cb98b7A841696, TxFinalizationBlock = 457, Amou
```

Figure 90: Scenario 6 - First session transactions forwarded (ServiceProvider1 service)

However, at some point before the on-chain update of ServiceProvider1 (aka before the relay delay interval has passed), Relayer1 goes offline, for some reason. Since relays need to be online to commit on-chain updates to the blockchain, ServiceProvider1's balance isn't updated on-chain as it should.

```
2021-07-15 11:20:55  Imported #453 0x8264…ddae (0 txs, 0.00 Mgas, 0 ms, 0.55 KiB)
2021-07-15 11:21:00  Imported #454 0xcc2b…0825 (0 txs, 0.00 Mgas, 0 ms, 0.55 KiB)
2021-07-15 11:21:05  Imported #455 0xe7f0…1140 (0 txs, 0.00 Mgas, 0 ms, 0.55 KiB)
2021-07-15 11:21:10  Imported #456 0xee2f…a2f3 (0 txs, 0.00 Mgas, 0 ms, 0.55 KiB)
2021-07-15 11:21:15  Imported #457 0x8fb1…0145 (0 txs, 0.00 Mgas, 0 ms, 0.55 KiB)
2021-07-15 11:21:16     3/25 peers     94 KiB chain  697 KiB db  0 bytes queue    3 KiB sync  RPC:  0 conn,    0 req/s,    0 µs
2021-07-15 11:21:20  Imported #458 0x14a1…6a8c (0 txs, 0.00 Mgas, 0 ms, 0.55 KiB)
```

Figure 91: Scenario 6 - Onchain update not posted on time

The service provider service is designed to scan the blockchain for new on-chain updates and cross-validate them with the transactions in the service provider's database. When the service provider detects that his total sum of on-chain updates is less than the total sum of expired

transactions directed at him, he realizes that the relay delayed to verify some transactions on-chain. In order to get his owed funds, the RSC gives him the ability to report a delayed payment from a specific relay. In order for the report to be valid, the service provider must also post the signed payload of a transaction, alongside its properties. This is all done automatically, no human intervention is required. The RSC method that allows the service provider to initiate disputes is **"reportDelayedPayment"**.

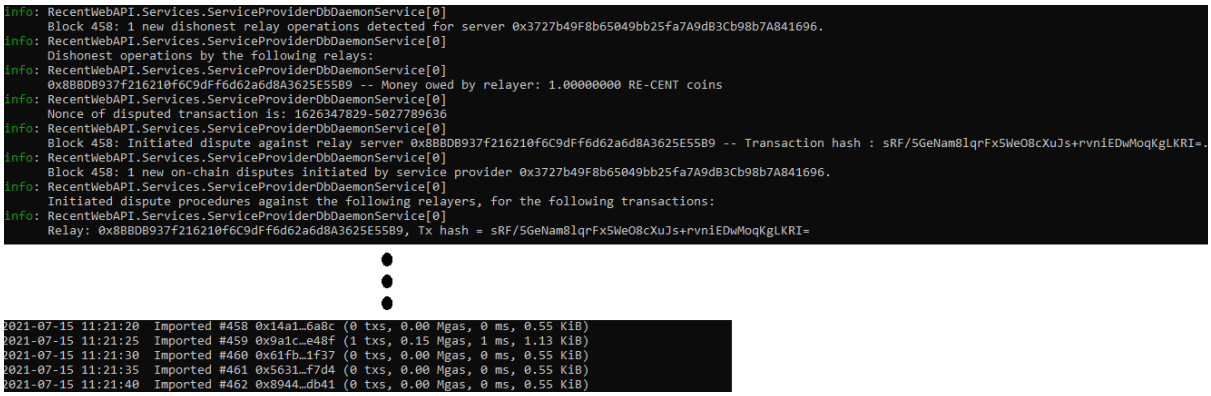

Figure 92: Scenario 6 - Service provider initiates dispute for first session

The moment the dispute is initiated, the relay has a small time window (D_R) to respond to this accusation. However, in this scenario the relay can't justify himself, since he was offline during the time he should post the on-chain balance update. Therefore, he acted negligently, and the service provider will be the winner of this dispute. Once the dispute resolution period is over, the service provider service calls the **"serverRefund"** RSC method, which refunds the service provider in case of a victorious dispute with the value of the transaction plus the gas costs he paid to report the relay's illicit behaviour.

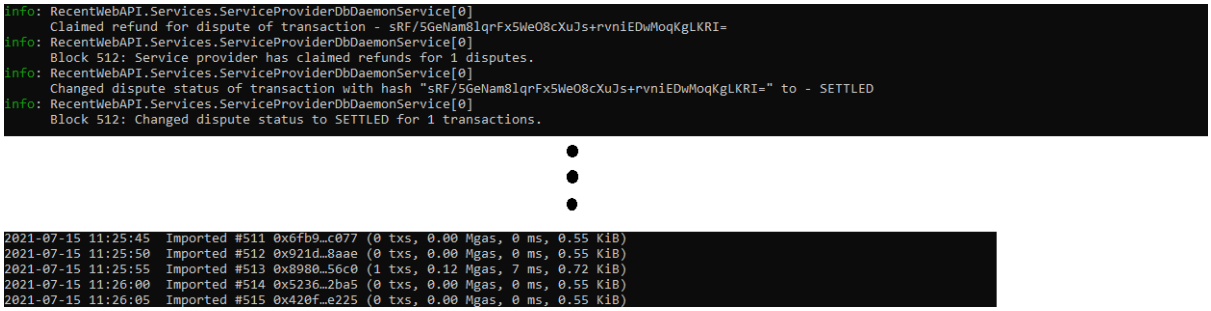

Figure 93: Scenario 6 - Service provider claims refund for first session

The funds go to the server's tab, who then withdraws them:

Response body

```
1.0000000000001823
```

Figure 94: Scenario 6 - Server's tab after first dispute

Notice that the gas costs he paid to initiate the dispute are also included in his tab. This mechanism is in place to encourage service providers to initiate disputes, if they're positively sure that they're in the right.

Additionally, the relay is punished for delaying payments, by having part of his stake slashed (penalty equal to the transaction value for his server mirror fund, plus an additional penalty for delaying a payment to his penalty funds. Additionally, his delayed payments counter is increased (so does the epoch's delayed payments counter). If a relay's penalty fund is reduced to zero due to penalties, his license is automatically revoked.


```
{
  "name": "Relay 1",
  "owner": "0x8bbdb937f216210f6c9dff6d62a6d8a3625e55b9",
  "domain": "http://localhost:8080",
  "maxUsers": 5,
  "maxCoins": 100,
  "maxGasThroughput": 1500000,
  "offchainTxDelay": 50,
  "fee": 0.002,
  "relayerStake": 209.74999999999983,
  "currentUsers": 1,
  "currentCoinsInbound": 50,
  "currentCoinsOutbound": 40,
  "currentGasThroughput": 0,
  "lastBlockInSubepoch": 0,
  "delayedPayments": 1,
  "clientMirrorFund": 100,
  "serverMirrorFund": 99,
  "remainingPenaltyFunds": 10.749999999999817
},
```


Figure 95: Scenario 6 - Relay's specs after the first dispute

At some point later, the ServiceProvider1 and Client1 agree to a new session, using Relayer1 as the relay. The sessions details are the following:

Response body

```
Session request successfully sent to service provider 0x3727b49F8b65049bb25fa7A9dB3Cb98b7A841696 !
The payment process will begin as soon as the service provider approves the session.
Timeplan details:
=====================================
Timeplan ID: Yyhe9ErX5i
Payment relay: 0x8BBDB937f216210f6C9dFf6d62a6d8A3625E55B9
Video duration: 90 seconds
Micropayment duration: 100 seconds
Micropayment cost: 2 RE-CENT coins
Relay fee per TX: 0.002 RE-CENT coins
Estimated amount of micropayments for this session: 1
Estimated session cost: 2.002 RE-CENT coins ---------- (Estimated balance left on relayer currently: 49 RE-CENT coins)
```

Figure 96: Scenario 6 - Second session details

The session commences, but this time the relay posts the update on-time:

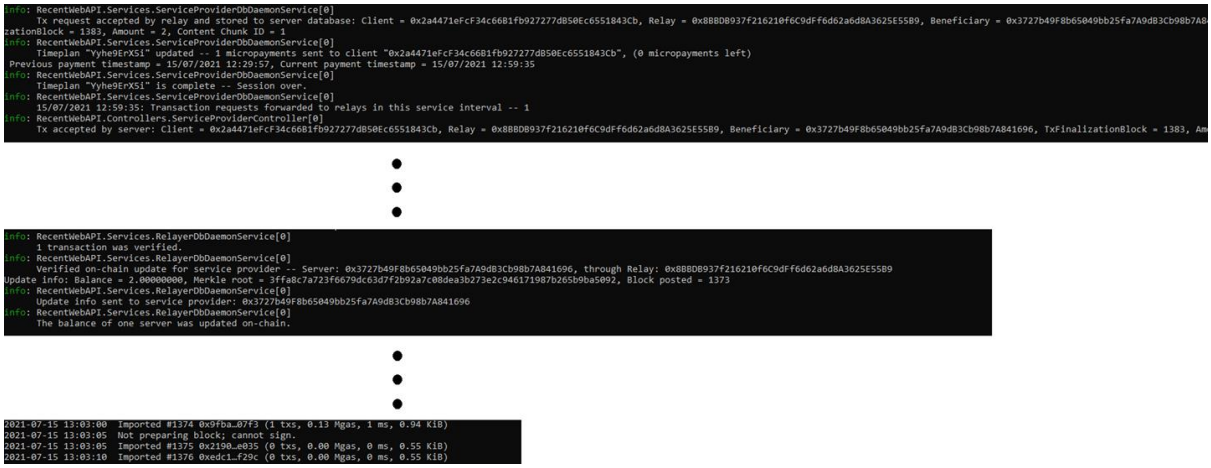

Figure 97: Scenario 6 - Second session commences, relay posts onchain update

Therefore, the service provider can simply withdraw his funds and continue his activities like normal. However, we assume that either due to mistake, either because the service provider believes he can trick the RSC and get more funds from the relayer than he is owed, after withdrawing he initiates a dispute for the same payment the relay just posted on-chain.

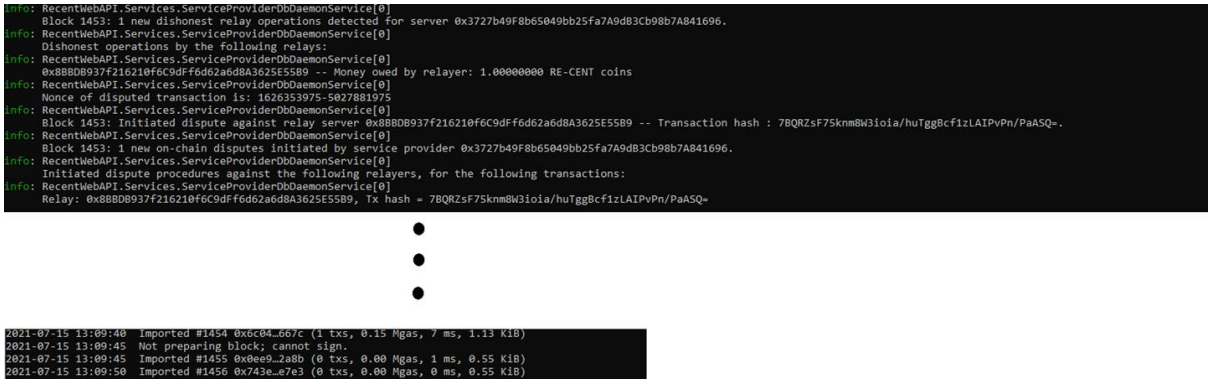

Figure 98: Scenario 6 - ServiceProvider1 incorrectly disputes latest onchain update

The relay service is designed to constantly scan the blockchain for any possible disputes opened by service providers against the relay. Each of these disputes is examined further. If the disputed transaction has been posted on-chain by the relay, the relay can defend himself by using the **“respondDelayedPayment_verifyUserPayloads”**, **“respondDelayedPayment_verifyRelayPayloads”** and **“respondDelayedPayment_settleDispute”** RSC functions. These functions are called automatically by the relay service in this order, allow the relay to prove that he has in fact updated the in question service provider’s balance in an honest manner by providing the complete payloads of specific updates (which should include the payload of the disputed transaction), and proving that the sum tied to these updates is equal to the expected. In this case, the RSC protects the relay’s funds and the service provider doesn’t get any refunds for his false accusation. In other cases, where the relay has committed a mistake, he can still call these functions to verify his activities and willingly give what he owes back to the service provider. This way, penalties towards him are milder and more manageable.

Following the service provider’s dishonest dispute, the relay has provides proof to the RSC that he did in fact update ServiceProvider1’s balance in a timely manner.

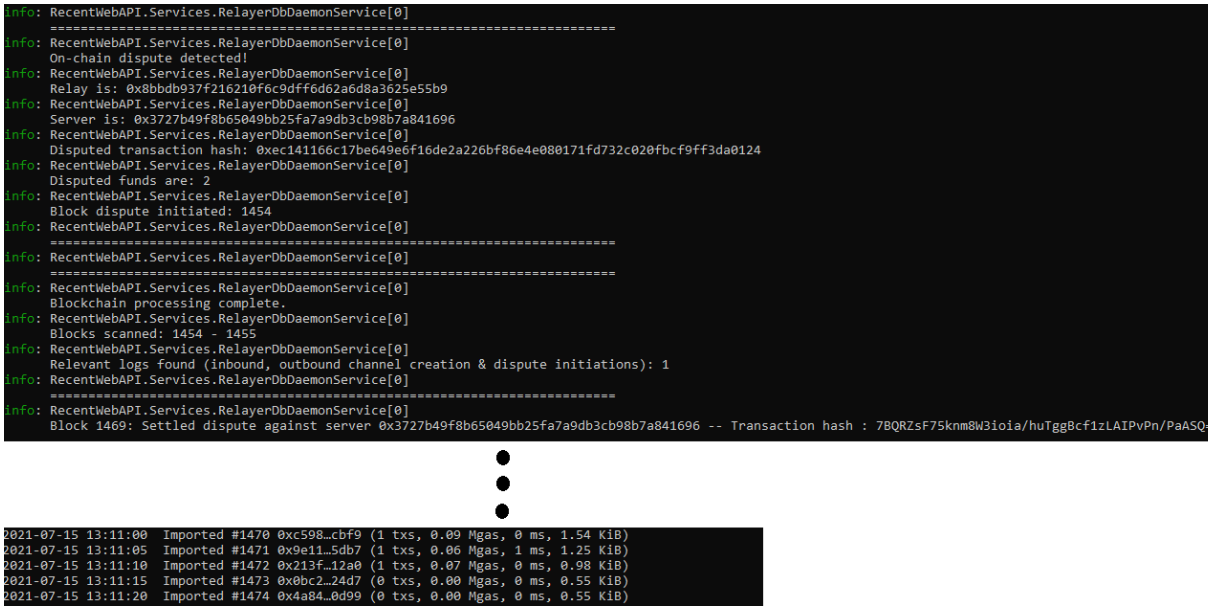

Figure 99: Scenario 6 - Relayer1 responds to ServiceProvider1's dispute and wins

Since the dispute is now settled by the relay, the service provider cannot get any refunds. This is the RSC's way to dissuade service providers from initiating disputes in case of honest relay operation. However, there is no way for the relay to get a gas refund for responding to the dispute.

Following the successful response to the service provider's dispute, the relay's penalty funds are untouched:

```
{
  "name": "Relay 1",
  "owner": "0x8bbdb937f216210f6c9dff6d62a6d8a3625e55b9",
  "domain": "http://localhost:8080",
  "maxUsers": 5,
  "maxCoins": 100,
  "maxGasThroughput": 1500000,
  "offchainTxDelay": 50,
  "fee": 0.002,
  "relayerStake": 209.74999999999983,
  "currentUsers": 1,
  "currentCoinsInbound": 50,
  "currentCoinsOutbound": 37,
  "currentGasThroughput": 97890,
  "lastBlockInSubepoch": 1374,
  "delayedPayments": 1,
  "clientMirrorFund": 100,
  "serverMirrorFund": 99,
  "remainingPenaltyFunds": 10.749999999999817
},
```


Figure 100: Scenario 6 - Relay's specs after second dispute, penalty funds untouched

Some time later, ServiceProvider1 and Client1 engage in yet another session. The session's details are seen below:

Response body

```
Session request successfully sent to service provider 0x3727b49F8b65049bb25fa7A9dB3Cb98b7A841696 !
The payment process will begin as soon as the service provider approves the session.
Timeplan details:
=====================================
Timeplan ID: q6vHNrwu1W
Payment relay: 0x8BBDB937f216210f6C9dFf6d62a6d8A3625E55B9
Video duration: 150 seconds
Micropayment duration: 150 seconds
Micropayment cost: 3 RE-CENT coins
Relay fee per TX: 0.002 RE-CENT coins
Estimated amount of micropayments for this session: 1
Estimated session cost: 3.002 RE-CENT coins --------- (Estimated balance left on relayer currently: 50 RE-CENT coins)
```

Figure 101: Scenario 6 - Third session details

Like in the first session, the relay forwards the transactions but goes offline before he can verify the update on-chain. The service provider picks up on this, and initiates a dispute against the relay:

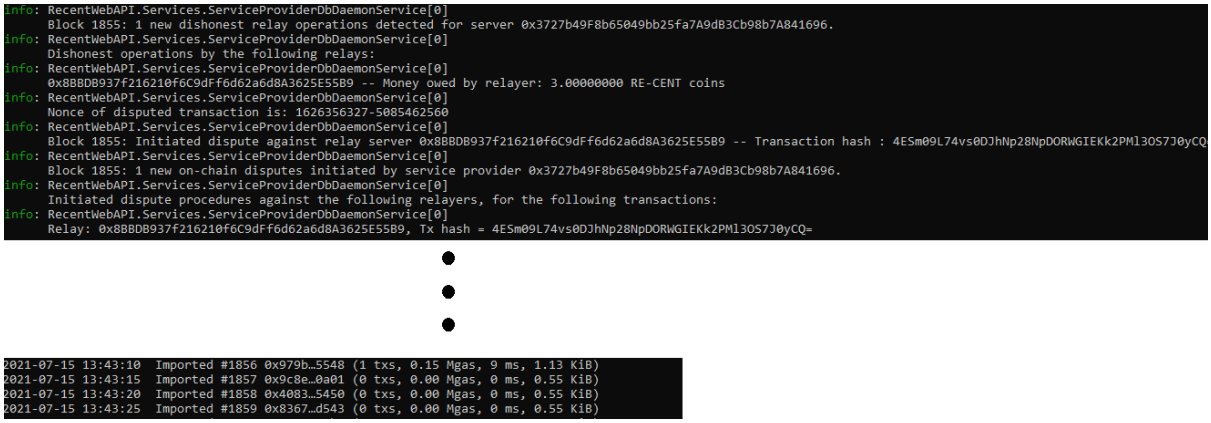

Figure 102: Scenario 6 - Third dispute against Relayer1 from ServiceProvider1

The relay is offline and can’t respond, therefore ServiceProvider1 is the victor in the dispute. He claims his refund after the dispute resolution passes, and his tab is now the following:

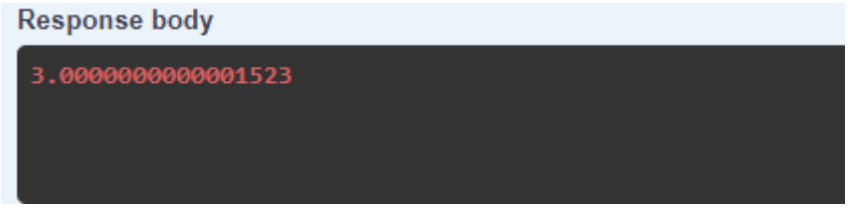

Figure 103: Scenario 6 - Server's tab after third dispute

Relayer1’s penalty fund has been decreased even further:

```
{
  "name": "Relay 1",
  "owner": "0x8bbdb937f216210f6c9dff6d62a6d8a3625e55b9",
  "domain": "http://localhost:8080",
  "maxUsers": 5,
  "maxCoins": 100,
  "maxGasThroughput": 1500000,
  "offchainTxDelay": 50,
  "fee": 0.002,
  "relayerStake": 202.06249999999966,
  "currentUsers": 1,
  "currentCoinsInbound": 50,
  "currentCoinsOutbound": 37,
  "currentGasThroughput": 97890,
  "lastBlockInSubepoch": 1374,
  "delayedPayments": 2,
  "clientMirrorFund": 100,
  "serverMirrorFund": 96,
  "remainingPenaltyFunds": 6.062499999999665
},
```


Figure 104: Scenario 6 - Relay's specs after 3rd dispute

We'll repeat this procedure for yet another session. The relay will forward the transactions, but then will go offline and won't post the on-chain update. The session details will be the following:

Response body

```
Session request successfully sent to service provider 0x3727b49F8b65049bb25fa7A9dB3Cb98b7A841696 !
The payment process will begin as soon as the service provider approves the session.
Timeplan details:
=====================================
Timeplan ID: M6BBAG7cdi
Payment relay: 0x8BBDB937f216210f6C9dFf6d62a6d8A3625E55B9
Video duration: 90 seconds
Micropayment duration: 90 seconds
Micropayment cost: 5 RE-CENT coins
Relay fee per TX: 0.002 RE-CENT coins
Estimated amount of micropayments for this session: 1
Estimated session cost: 5.002 RE-CENT coins ---------- (Estimated balance left on relayer currently: 42 RE-CENT coins)
```

Figure 105: Scenario 6 - Fourth session details

Since Relayer1's penalty fund was already very low, we can assume that due to this last dispute it has now been depleted. Using the **"Info/Summary"** API function we notice that Relayer1 is no longer included in the list of active relayers. This is also confirmed when we attempt to run Relayer1's relay service.

Figure 106: Scenario 6 - Relayer1's relay license no longer valid

We now look up Relayer1's specs again. We notice that his delayed payments counter has increased. We also notice that his penalty fund has been depleted. His stake has also been decreased to zero, meaning he cannot withdraw the remainder of his initial stake that wasn't expended in penalties. We confirm that his relay license is no longer valid.


```
{
  "name": "Relay 1",
  "owner": "0x8bbdb937f216210f6c9dff6d62a6d8a3625e55b9",
  "domain": "http://localhost:8080",
  "maxUsers": 5,
  "maxCoins": 100,
  "maxGasThroughput": 1500000,
  "offchainTxDelay": 50,
  "fee": 0.002,
  "relayerStake": 0,
  "currentUsers": 1,
  "currentCoinsInbound": 50,
  "currentCoinsOutbound": 37,
  "currentGasThroughput": 97890,
  "lastBlockInSubepoch": 1374,
  "delayedPayments": 3,
  "clientMirrorFund": 100,
  "serverMirrorFund": 91,
  "remainingPenaltyFunds": 0
}
```


Figure 107: Scenario 6 - Relay's final specs (license revoked)

The service provider dispute mechanism works as designed.

5.3.7 Scenario 7 – Onchain dispute mechanism – Unauthorized payments:

This scenario covers disputes between RE-CENT clients and relays. We identify two sub-scenarios:

In the first subscenario (scenario 7.1), a client who's had a number of transactions with a relay discovers at some point that his on-chain inbound channel balance is less that the total sum of transactions he has signed. This means that the relay used up more channel funds than he should, either by mistake or maliciously. The client is given the ability to report the relay on-chain and claim the money that was illegally taken from him, in addition to the gas costs he paid to report the relayer's dishonest behaviour.

In the second subscenario (scenario 7.2), a client accuses a relay of illegal behaviour, when there is none. Just like in the service provider disputes, the relay is given the ability to respond to disputes from clients, and prove his innocence.

All disputes must be resolved in the dispute resolution window (D_R).

5.3.7.1 Scenario summary

For subscenario 1 (client victory), the following steps will be followed:

1.) Review of Client1's inbound channel with Relayer1.

2.) Relayer1 withdraws funds from Client1's inbound channel, even though no sessions took place.

3.) Review of Client1's inbound channel with Relayer1.

4.) Client1 disputes the inbound channel update on-chain.

5.) Relayer1 cannot justify the update since client1 didn't sign any transactions, therefore he automatically loses the dispute.

6.) Client1 claims his refund, after the dispute resolution period is over.

7.) Review of Relayer1's running parameters.

For subscenario 2 (relay victory), the following steps will be followed:

1.) Review of Client1's inbound channel with Relayer1.

2.) Client1 engages in a session with ServiceProvider1, using Relayer1 as a relay.

3.) Relayer1 withdraws funds from Client1's inbound channel, in accordance with the transactions he signed.

4.) Client1 wrongfully disputes the inbound channel update on-chain.

5.) Relayer1 verifies the validity of the inbound channel update using Client1's signed transactions.

6.) Client1 loses the dispute and gets no refunds.

7.) Review of Relayer1's running parameters.

5.3.7.2 Runtime

In order to track the chain parameters and the VSC and RSC variables, we use the WebAPI. For the balance tracking of our test nodes, we can use Metamask.

For both sub-scenarios of scenario 7 we assume that B_R = 36000 blocks. We increase the epoch duration to make sure that no disputes will span multiple epochs, for simplicity purposes.

We assume that Relayer1 has already acquired a valid relay license for this epoch. We also assume that Client1's inbound channel to Relayer1 and Relayer1's outbound channel to ServiceProvider1 are both adequately funded for a session to take place between them.

== Scenario 7.1 – Client victory ==

For this subscenario, we assume that even though Client1 had deposited a number of funds to Relayer1, no transaction or session ever took place between them. In other words, Client1 didn't send out a single payment. Assuming honest operation, the relay would leave the inbound channel intact and after the epoch has passed, the client would withdraw an amount of funds equal to his deposit.

In our case however, the relay due to either mistake or on purpose, he reduces Client1's inbound channel balance by 3 coins.

The client's initial deposit to to Relay1 was 10 coins. We can track the on-chain balance of every inbound channel with the **"User/{clientAddress}/Info"** API function.

Response body

```
Client deposit data:
====================
Client address: 0x2a4471eFcF34c66B1fb927277dB50Ec6551843Cb
Relay address: 0x8BBDB937f216210f6C9dFf6d62a6d8A3625E55B9
Amount: 10
Expiration block: 36000
```

Figure 108: Scenario 7.1 - Initial client deposit data

This is Client1's account balance before the dispute procedure:

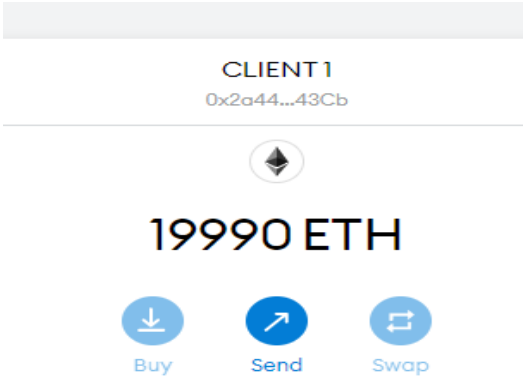

Figure 109: Scenario *7.1 – Client1 initial* account balance

And Relayer1's data:

```
{
  "name": "Relay 1",
  "owner": "0x8bbdb937f216210f6c9dff6d62a6d8a3625e55b9",
  "domain": "http://localhost:8080",
  "maxUsers": 5,
  "maxCoins": 100,
  "maxGasThroughput": 1500000,
  "offchainTxDelay": 50,
  "fee": 0.002,
  "relayerStake": 1000,
  "currentUsers": 1,
  "currentCoinsInbound": 10,
  "currentCoinsOutbound": 0,
  "currentGasThroughput": 0,
  "lastBlockInSubepoch": 0,
  "delayedPayments": 0,
  "clientMirrorFund": 100,
  "serverMirrorFund": 100,
  "remainingPenaltyFunds": 800
},
```


Figure 110: Scenario 7.1 - Initial relay data

Relayer1 decides to reduce three coins from Client1's inbound channel, his motive being unknown. The relay service calls the RSC and conducts the illegal balance update:

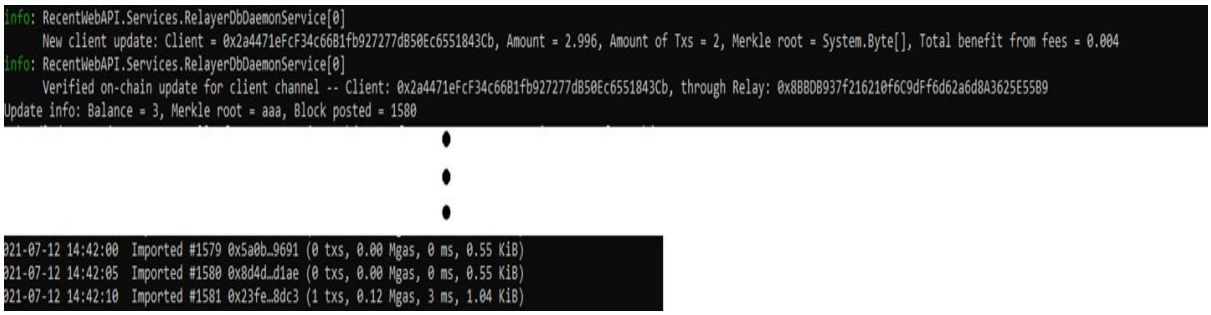

Figure 111: Scenario 7.1 - Illegal inbound channel update by relay

After the illegal inbound channel reduction by the relay, Client1's inbound channel balance is now reduced:

Response body

```
Client deposit data:
====================
Client address: 0x2a4471eFcF34c66B1fb927277dB50Ec6551843Cb
Relay address: 0x8BBDB937f216210f6C9dFf6d62a6d8A3625E55B9
Amount: 7
Expiration block: 36000
```

Figure 112: Scenario 7.1 - New channel deposit data

The client service is designed to pick up on illicit relay activity. If the sum of his inbound channel reductions by a relay is greater than the sum of payments he has forwarded to service providers through this relay, he initiates a dispute procedure against the relay, for the update he thinks he was mistreated in. In order to initiate the dispute, the client just provides the merkle root hash of an update to the RSC via the **"reportOverwithdraw"** method.

The client service initiates a dispute for the relay's update to the client's inbound channel:

```
info: RecentWebAPI.Services.ClientDbDaemonService[0]
      Block 1618: 1 new on-chain balance updates for user 0x2a4471eFcF34c66B1fb927277dB50Ec6551843Cb.
info: RecentWebAPI.Services.ClientDbDaemonService[0]
      Block 1618: 1 new dishonest relay operations detected for user 0x2a4471eFcF34c66B1fb927277dB50Ec6551843Cb.
info: RecentWebAPI.Services.ClientDbDaemonService[0]
      Dishonest operations by the following relays:
info: RecentWebAPI.Services.ClientDbDaemonService[0]
      0x8BBDB937f216210f6C9dFf6d62a6d8A3625E55B9 -- Additional money relayer withdrew: 3 RE-CENT coins
info: RecentWebAPI.Services.ClientDbDaemonService[0]
      Block 1618: 1 new on-chain disputes initiated by user 0x2a4471eFcF34c66B1fb927277dB50Ec6551843Cb.
info: RecentWebAPI.Services.ClientDbDaemonService[0]
      Initiated dispute procedures against the following relayers and merkle root update hashes:
info: RecentWebAPI.Services.ClientDbDaemonService[0]
      Relay: 0x8BBDB937f216210f6C9dFf6d62a6d8A3625E55B9, Update root hash = 6161610000000000000000000000000000000000000000000000000000000000
```

```
021-07-12 14:45:20  Imported #1619 0x599b…4aec (1 txs, 0.12 Mgas, 7 ms, 0.79 KiB)
021-07-12 14:45:25  Imported #1620 0xf2b6…b52d (0 txs, 0.00 Mgas, 0 ms, 0.55 KiB)
021-07-12 14:45:30  Imported #1621 0x64d6…2fec (0 txs, 0.00 Mgas, 0 ms, 0.55 KiB)
021-07-12 14:45:35  Imported #1622 0x92eb…d980 (0 txs, 0.00 Mgas, 0 ms, 0.55 KiB)
021-07-12 14:45:40  Imported #1623 0x1721…1e12 (0 txs, 0.00 Mgas, 0 ms, 0.55 KiB)
```

Figure 113: Scenario 7.1 - Client disputes illegal update

Since the relay doesn't have any signed payloads from the client, he cannot defend himself in this dispute and therefore loses. The client then waits until the dispute resolution period is over. After, the client service automatically calls the **"clientRefund"** RSC function, which refunds the client and punishes the relay.

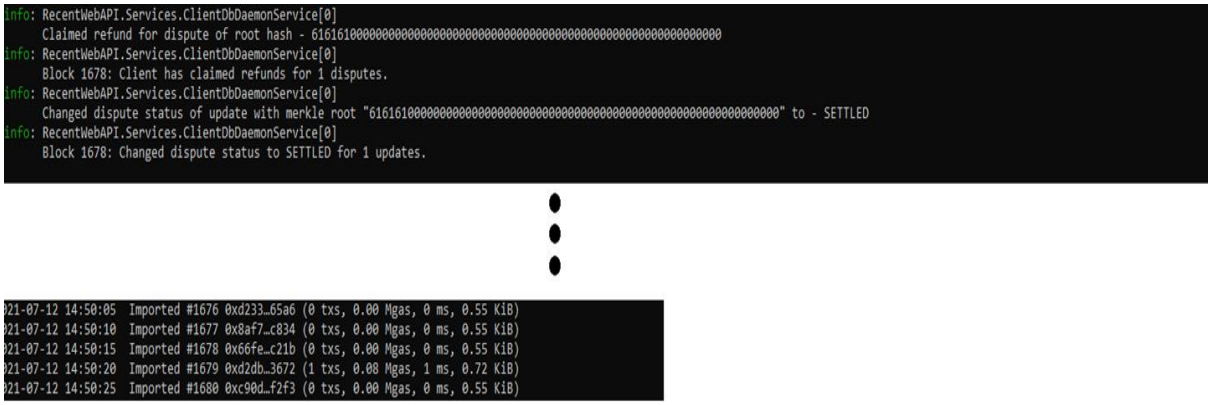

Figure 114: Scenario 7.1 - Client claims refund for illegal update

The client's refund (including the gas costs he spent to initiate the dispute) are sent straight to his account balance:

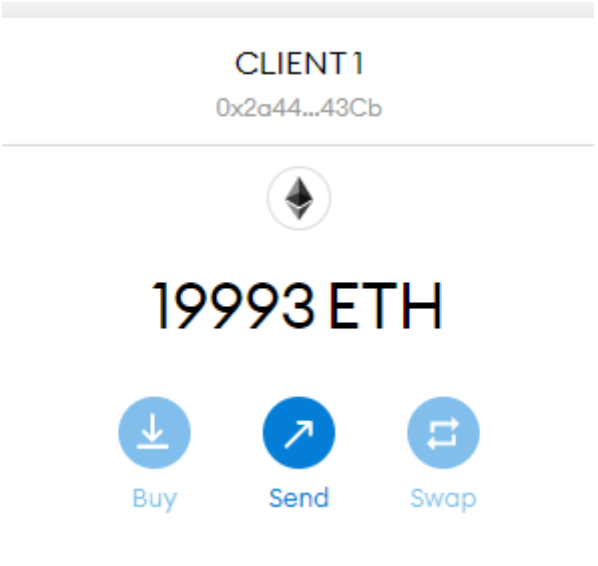

Figure 115: Scenario 7.1 - Client1 account balance after dispute

His inbound channel however isn't refunded:

Response body

```
Client deposit data:
====================
Client address: 0x2a4471eFcF34c66B1fb927277dB50Ec6551843Cb
Relay address: 0x8BBDB937f216210f6C9dFf6d62a6d8A3625E55B9
Amount: 7
Expiration block: 36000
```

Figure 116: Scenario 7.1 - Client deposit data after dispute

And the relay is punished by having his stake slashed (client mirror fund decreased by the value of the update, penalty funds charged extra) and his delayed payments counter increased by one.


```
{
  "name": "Relay 1",
  "owner": "0x8bbdb937f216210f6c9dff6d62a6d8a3625e55b9",
  "domain": "http://localhost:8080",
  "maxUsers": 5,
  "maxCoins": 100,
  "maxGasThroughput": 1500000,
  "offchainTxDelay": 50,
  "fee": 0.002,
  "relayerStake": 993.2499999999999,
  "currentUsers": 1,
  "currentCoinsInbound": 7,
  "currentCoinsOutbound": 0,
  "currentGasThroughput": 0,
  "lastBlockInSubepoch": 0,
  "delayedPayments": 1,
  "clientMirrorFund": 97,
  "serverMirrorFund": 100,
  "remainingPenaltyFunds": 796.2499999999999
},
```


Figure 117: Scenario 7.1 - Relay data after dispute

== Scenario 7.2 – Relay victory ==

This scenario is a variation of the previous scenario. We assume that Client1 requests a session with ServiceProvider1 via Relayer1. We also assume that Client1 has deposited 10 coins to Relayer1, and that Relayer1 deposited 5 coins to ServiceProvider1. The session's details were the following:

Response body

```
Session request successfully sent to service provider 0x3727b49F8b65049bb25fa7A9dB3Cb98b7A841696 !
The payment process will begin as soon as the service provider approves the session.
Timeplan details:
=====================================
Timeplan ID: O2y0F8Du4L
Payment relay: 0x8BBDB937f216210f6C9dFf6d62a6d8A3625E55B9
Video duration: 60 seconds
Micropayment duration: 100 seconds
Micropayment cost: 2 RE-CENT coins
Relay fee per TX: 0.002 RE-CENT coins
Estimated amount of micropayments for this session: 1
Estimated session cost: 2.002 RE-CENT coins ---------- (Estimated balance left on relayer currently: 10 RE-CENT coins)
```

Figure 118: Scenario 7.2 - Session details

The initial server tab for ServiceProvider1 and Client1's deposit data are the following:

Response body

```
0
```

Response body

```
Client deposit data:
====================
Client address: 0x2a4471eFcF34c66B1fb927277dB50Ec6551843Cb
Relay address: 0x8BBDB937f216210f6C9dFf6d62a6d8A3625E55B9
Amount: 10
Expiration block: 36000
```

Figure 119: Scenario 7.2 - Initial ServiceProvider1 tab and Client1 channel details

The session begins and is concluded shortly after. The Service provider's balance is updated properly, and shortly after Client1's inbound channel balance is also updated:

Figure 120: Scenario 7.2 - Final ServiceProvider1 tab and Client1 channel details

For his own personal reasons though, Client1 decides to dispute the update, even though there was no wrongdoing from the relay in this particular update.

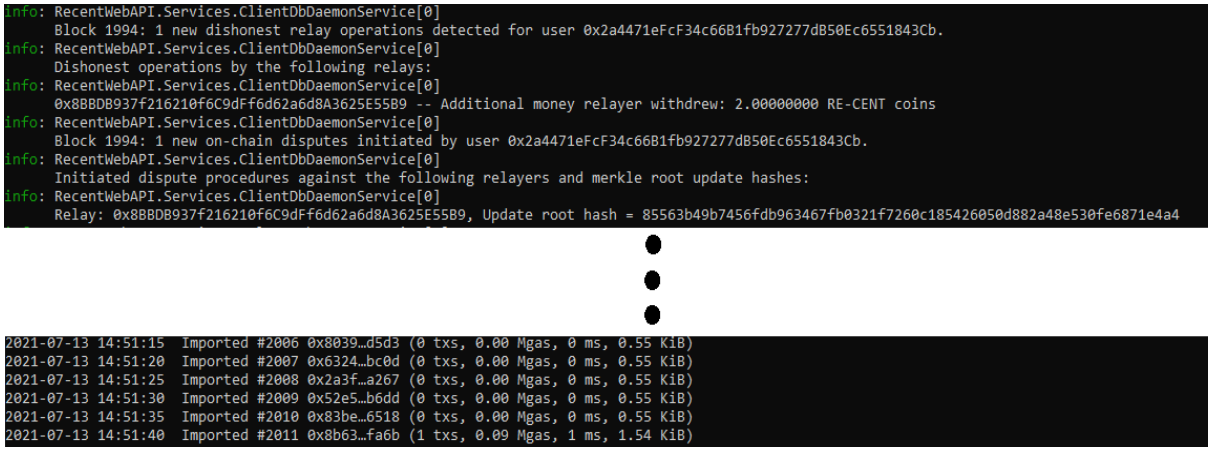

Figure 121: Scenario 7.2 - Client disputes the relay's last update

The relay service is designed to constantly scan the blockchain for any possible disputes opened by clients against the relay. Each of these disputes is examined further. If the disputed update is perfectly legit, the relay can defend himself by using the **“respondDelayedPayment_verifyUserPayloads”** and **“respondOverdispute_settleDispute”** RSC functions. These two functions are called automatically by the relay service in this order, allow the relay to prove that a particular update is indeed honest, by providing all the payloads of all transactions tied to this update. If the sum of the transactions in the update is equal to the value of the update, the relay is considered

the victor in the dispute. In this case, the RSC protects the relay's funds and the client doesn't get any refunds for his false accusation.

In this subscenario, the relay first notices the client's dispute by scanning the chain, and then provides proof to the RSC that he did in fact update Client1's inbound channel balance properly.

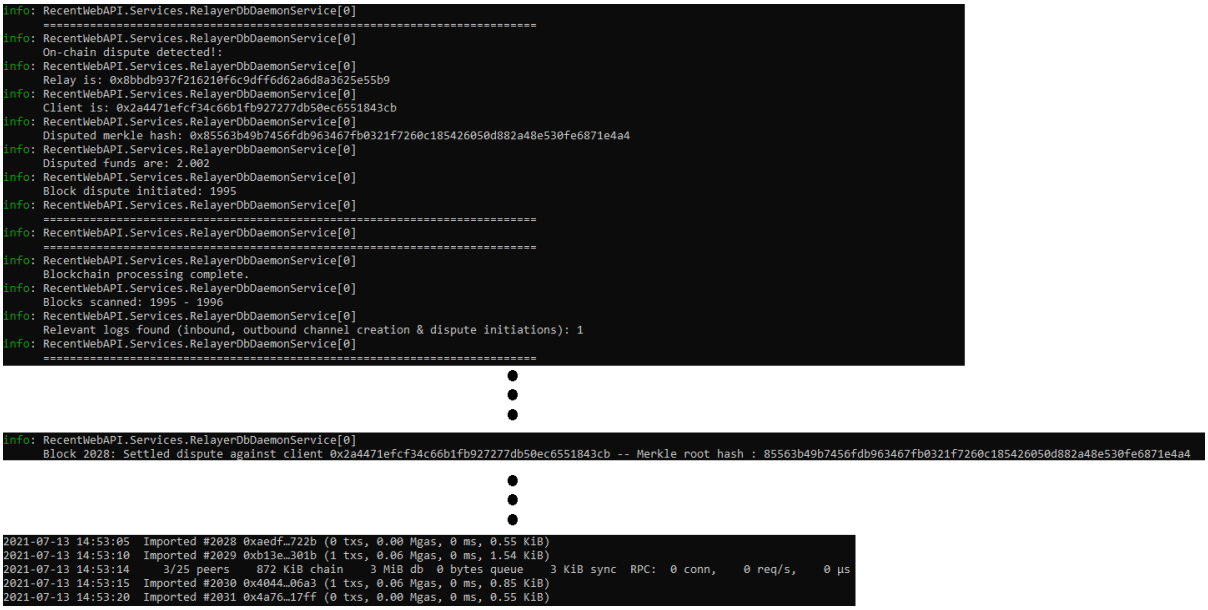

Figure 122: Scenario 7.2 - Relay detects, then settles false dispute

Since the dispute is now settled by the relay, the client cannot get any refunds. This is the RSC's way to dissuade clients from initiating disputes in case of honest relay operation. However, there is no way for the relay to get a gas refund for responding to the dispute.

5.3.8 Scenario 8 – Relay throughput regulation mechanism:

Scenarios 8 examines the relay throughput regulation mechanism, which assures that the maximum gas throughput defined in a relay's license won't be surpassed in a throughput regulation period. This mechanism is put in place to motivate relays to aggregate transactions better (otherwise they'll be left wide open to disputes) and to help the system preserve its scalability:

The mechanism works as follows:

According to the RSC, each epoch is decomposed into B_R / k_R subepoch slots, in which the relay should never exceed the maximum amount of gas M[r] allowed in his license. At this

point, we need to mention that even though relayers specify a maximum transaction limit per epoch during their license request, in the end the RSC also keeps track of their maximum gas consumption limit, since in the Ethereum platform gas is what fills the blocks and not transactions (even though there is a connection between the two, there are cases in which network participant could initiate a single transaction costing more gas than 50 simple transactions, for example). Therefore, it's more important that we keep track of the current gas throughput for any relay, instead of their current transaction throughput. In our model, each transaction in the relay's maximum transaction limit is equal to 150000 gas. This number is justified because of the multiple storage changes associated with the balance update of a single server, alongside the constant cost of any on-chain transaction (21000 gas).

The relay throughput monitoring mechanism is activated every time a relayer calls the "releaseServerFunds" RSC function (aka, when server balances are updated on-chain). The RSC first calculates on which subepoch slot the function was called. Assuming the epoch started on block b_0 and the current block is b, the RSC checks if the last recorded counter updated in lb[r] belongs to the current subepoch slot. This holds true if:

$$Ceil((b - b_0)/ k_R) = Ceil((lb[r] - b_0)/ k_R)$$

The left side of the equation is the current subepoch slot, and the right side of the equation is the last subepoch slot a "releaseServerFunds" operation was recorded in. If the equation is true, the currentGasThroughput (G_C) metric of the relay is increased by the amount of gas used by the "releaseServerFunds" method call. Assuming that:

$$G_C < M[r]$$

The transaction can go on like normal. If G_C surpasses M[r] though, the RSC considers that the relay has surpassed his maximum gas throughput allowed by his license in this subepoch slot. Therefore, the transaction is reverted (meaning that the on-chain balance updates can't go through) and an event is emitted to the blockchain, asserting that the relay has reached his gas throughput limit for this epoch. This event acts as a warning to nodes wanting to use the relay, a warning that basically means that there is a good chance that they won't receive their money on time.

If the above equation is false, then G_C is set equal to the gas costs used up by the function call (assuming that's lower than M[r], and lb[r] is set to b, indicating that the subepoch slot has changed.

The relay throughput regulation mechanism doesn't deal any penalties to the relays directly, but it leaves them defenceless against possible delayed payment disputes. Thus, it helps discourage dishonest or inefficient relay behaviour, and assures that the system's scalability is preserved at all times.

In this scenario, we'll examine this regulatory mechanism and how surpassing it can negatively affect a relay.

5.3.8.1 Scenario summary:

For this scenario, the following steps will be followed:

1.) Client1 initiates a session with ServiceProvider1, using Relayer1 as the relay.

2.) Relayer1 forwards the payments, and then updates the on-chain balance of ServiceProvider1 on time.

3.) Review of Relayer1's running parameters.

4.) Client1 initiates another with ServiceProvider1, using Relayer1 as the relay.

5.) Relayer1 forwards the payments, but cannot update the on-chain balance of ServiceProvider1 on time due to gas limitations for this regulation period.

6.) ServiceProvider1 initiates a dispute with Relayer1, concerning the payments of the second session.

7.) Relayer1 never posted the transactions on-chain therefore he automatically loses the dispute. Relayer1 receives the appropriate penalties, and ServiceProvider1 gets his refund.

8.) Review of Relayer1's running parameters.

5.3.8.2 Runtime:

In order to track the chain parameters and the VSC and RSC variables, we use the WebAPI. For the balance tracking of our test nodes, we can use Metamask.

The addresses that will participate in this scenario are Relay1, ServiceProvider1 and Client1. For simplicity, we assume that Relay1 has already won a valid license through the relay election mechanism, and that Relay1's inbound channel with Client1 and outbound channel with ServiceProvider1 are adequately funded. We also assume that the relay has a very low transaction throughput, equal to 1 transactions per subepoch slot (which is equal to about 100000 gas).

Like in the dispute scenarios, we assume that B_R = 36000 blocks, and we also consider k_R = 1000 blocks, in order to better understand the throughput regulation mechanism. We assume that the appropriate payment channels have already been set up. This is Relay1's specs and current data at the start of this scenario, alongside Client1's deposit details

(**"Wallet/RelayerInfo"** API function):

```
Relayer address: 0x8bbdb937f216210f6c9dff6d62a6d8a3625e55b9
------------------------------------------------------------------------
Name: Relay 1
Domain: http://localhost:8080
Relay fee: 0.002 RE-CENT coins per Tx
Relay delay: 50 blocks
Delayed payments: 0
Coins inbound: 50 RE-CENT coins (50.0% coverage)
Coins outbound: 40 RE-CENT coins (80.0% coverage)
Current users: 1 (20.0% coverage)
Current gas throughput: 0 gas (0% of maximum allowed in this regulation period)
Last regulation period an on-chain update was posted: No on-chain updates posted for this relay -- (Current regulation period = 1)
Balance on relayer: 50 RE-CENT coins
Balance expiration block: 36000
```

Figure 123: Scenario 8 - Initial relay data

Client1 decides to initiate a session with ServiceProvider1, using Relay1 as the relay. The timeplan is seen below:

Response body

```
Session request successfully sent to service provider 0x3727b49F8b65049bb25fa7A9dB3Cb98b7A841696 !
The payment process will begin as soon as the service provider approves the session.
Timeplan details:
=====================================
Timeplan ID: AXSNkIxZvG
Payment relay: 0x8BBDB937f216210f6C9dFf6d62a6d8A3625E55B9
Video duration: 60 seconds
Micropayment duration: 100 seconds
Micropayment cost: 2 RE-CENT coins
Relay fee per TX: 0.002 RE-CENT coins
Estimated amount of micropayments for this session: 1
Estimated session cost: 2.002 RE-CENT coins ---------- (Estimated balance left on relayer currently: 50 RE-CENT coins)
```

Figure 124: Scenario *8* - First session specs

The service provider agrees to the timeplan, and the session can begin. The transactions are forwarded normally, and the service provider's balance is updated on-chain soon after.

Figure 125: Scenario 8 - First on-chain update

The service provider's tab is now equal to the session cost:

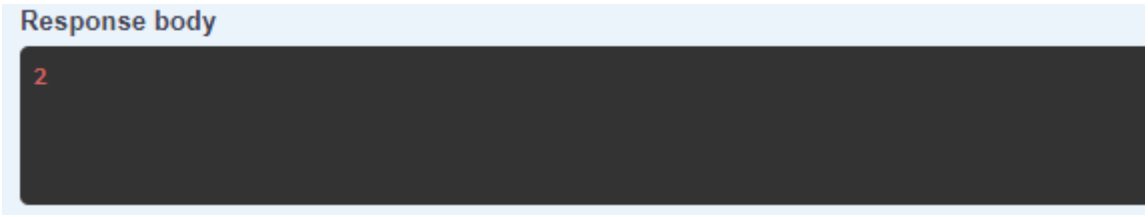

Figure 126: Scenario 8 - Server's tab after first update

A quick peek into the relay's current info reveals that aside from his outbound funds being depleted, his current gas throughput metric has increased, and is now much closer to the limit.

Figure 127: Scenario 8 - Relay data after first server update

Generally, this would act as a warning to relays, to not accept payments indiscriminately until the subepoch slot has changed. However, to showcase the regulation mechanism, we assume that Relay1 will accept any transactions directed through him.

Client1 decides to initiate another session with ServiceProvider1. The details are the following:

Response body

```
Session request successfully sent to service provider 0x3727b49F8b65049bb25fa7A9dB3Cb98b7A841696 !
The payment process will begin as soon as the service provider approves the session.
Timeplan details:
=====================================
Timeplan ID: npVOYap4Qy
Payment relay: 0x8BBDB937f216210f6C9dFf6d62a6d8A3625E55B9
Video duration: 95 seconds
Micropayment duration: 100 seconds
Micropayment cost: 3 RE-CENT coins
Relay fee per TX: 0.002 RE-CENT coins
Estimated amount of micropayments for this session: 1
Estimated session cost: 3.002 RE-CENT coins ---------- (Estimated balance left on relayer currently: 48 RE-CENT coins)
```

Figure 128: Scenario 8 - Second session specs

The service provider accepts the timeplan and the session begins. The transactions are forwarded like normal, but when it's time for the relay to update the server's balance on-chain, we notice that the relay service returns an error. This is because the relay's current gas throughput will surpass the maximum gas throughput defined by his license if the update goes through. Therefore, the on-chain update can't go through.

```
nfo: RecentWebAPI.Controllers.RelayerController[0]
     Tx request accepted. Server: 0x3727b49F8b65049bb25fa7A9dB3Cb98b7A841696, Client: 0x2a4471eFcF34c6681fb927277dB50Ec6551843Cb, Tx amount: 3, Content chunk ID: 1
nfo: RecentWebAPI.Services.RelayerObDaemonService[0]
     12/07/2021 11:06:54: Transaction requests forwarded to clients in this service interval -- 1
nfo: RecentWebAPI.Controllers.RelayerController[0]
     Tx accepted. User: 0x2a4471eFcF34c6681fb927277dB50Ec6551843Cb, Service Provider: 0x3727b49F8b65049bb25fa7A9dB3Cb98b7A841696, User balance: 50.00000000, Pending Txs unpaid amount: 0, This Tx amount: 3
nfo: RecentWebAPI.Services.RelayerObDaemonService[0]
     12/07/2021 11:06:56: Transactions forwarded to service providers in this service interval -- 1

     .
     .
     .

fail: RecentWebAPI.Services.RelayerObDaemonService[0]
     The on-chain call for the "releaseServerFunds" RSC function has been reverted. Either the outbound channel balances aren't enough or the relay has surpassed the maximum gas throughput for this regulation period! Curre
put = 112860, Maximum gas throughput = 150000
```

Figure 129: Scenario 8 - Relay can't forward update due to maximum gas throughput being surpassed

The service provider realizes that his balance hasn't been updated, therefore initiates a dispute procedure against Relay1. Since the relay hasn't uploaded the latest transactions on-chain, he cannot defend himself, and loses the dispute. ServiceProvider1 claims his dispute rewards shortly after:

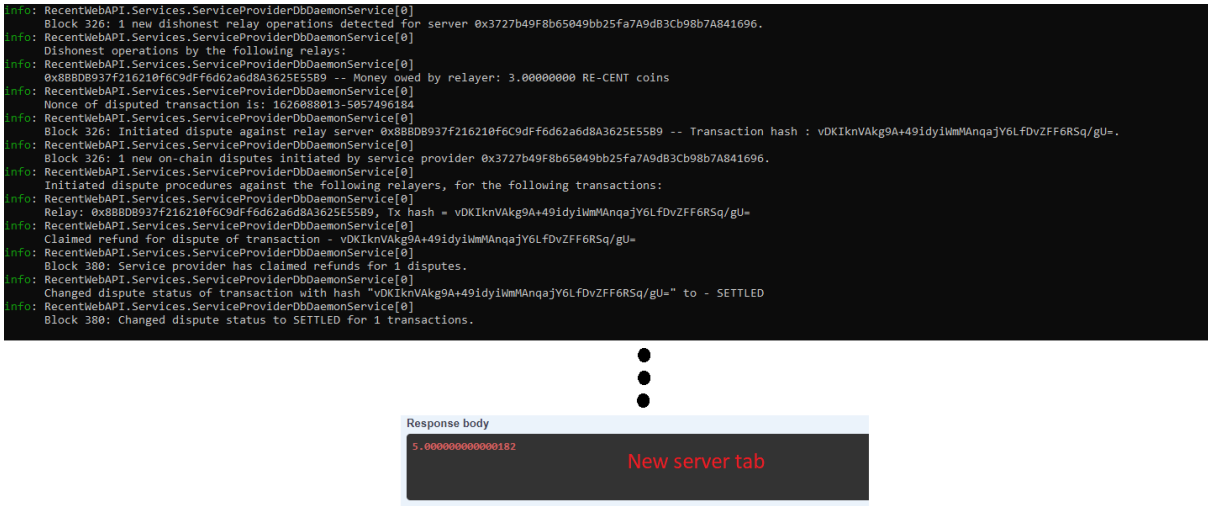

Figure 130: Scenario 8 - ServiceProvider1 claims delayed transaction funds

A quick peek into Relay1's current info reveals that his stake has been reduced, and his delayed payments counter has increased. Even though he never went online, he was still punished because he surpassed his allowed gas throughput limit. Relays must protect themselves from this mechanism by aggregating transactions better, or by selectively accepting only the transactions they know they'll be able to post on-chain (the picture below is from the **"Relayers/{epoch}"** function - we prefer this to Wallet/RelayerInfo in this case because it also includes details about the relay's stake).

```
{
  "name": "Relay 1",
  "owner": "0x8bbdb937f216210f6c9dff6d62a6d8a3625e55b9",
  "domain": "http://localhost:8080",
  "maxUsers": 5,
  "maxCoins": 100,
  "maxGasThroughput": 150000,
  "offchainTxDelay": 50,
  "fee": 0.002,
  "relayerStake": 993.2499999999998,
  "currentUsers": 1,
  "currentCoinsInbound": 50,
  "currentCoinsOutbound": 38,
  "currentGasThroughput": 112860,
  "lastBlockInSubepoch": 171,
  "delayedPayments": 1,
  "clientMirrorFund": 100,
  "serverMirrorFund": 97,
  "remainingPenaltyFunds": 796.2499999999998
},
```


Figure 131: Scenario 8 - Relay data after 2nd session (stake slashed, delayed payments increased)

After the subepoch passes, the relay's current gas throughput metric can be considered zero again, therefore he's free to post new updates on-chain.

```
Relayer address: 0x8bbdb937f216210f6c9dff6d62a6d8a3625e55b9
--------------------------------------------------------------------------------
Name: Relay 1
Domain: http://localhost:8080
Relay fee: 0.002 RE-CENT coins per Tx
Relay delay: 50 blocks
Delayed payments: 1
Coins inbound: 50 RE-CENT coins (50.0% coverage)
Coins outbound: 38 RE-CENT coins (76.00% coverage)
Current users: 1 (20.0% coverage)
Current gas throughput: 0 gas (0% of maximum allowed in this regulation period)
Last regulation period an on-chain update was posted: Regulation period = 1 -- (Current regulation period = 2)
Balance on relayer: 50 RE-CENT coins
Balance expiration block: 36000
```

Figure 132: Scenario 8 - Relay data, next regulation period

This is how the transaction throughput regulation mechanism works for relays, in a nutshell.

6. CONCLUSIONS

In this work, we have provided a comprehensive overview of blockchain technologies and protocols that can be used to implement blockchain-backed network asset trading over 5G and Beyond networks. The emphasis was given on the development of solutions enabling i) a high degree of decentralization upon blockchain consensus, ii) multi-million transactions per second to be performed and iii) anonymous instant off-chain payments enabling fair-exchange of network and blockchain level resources. We have further presented the RE-CENT mobile data access model and elaborated on the performance benefits that it can deliver in view of mobile video content delivery. An innovative mechanism for fully decentralized DPoS consensus has been presented to enable different roles and levels of engagement in view of the myriads components populating the 5G and Beyond network.

Founded on payment channels and off-chain payments, we have also proposed a forward-thinking payment relay service that enables multi-million transactions per second to be aggregated within the transactions capacity of the today's SC-enabled platforms whereas, building on-top of this service, we have also developed an innovative coin mixing service that enables payment relays to support anonymous yet fair exchange of blockchain and network-level resources. Using sophisticated incentive engineering mechanism, we have showed that the proposed protocols are secure against inadvertent or malicious behavior of the RE-CENT nodes providing also detailed runtime examples and discussing necessary implementation details.

Extensive simulation results have demonstrated the unique performance trade-offs of blockchain-backed service delivery and have enabled us to draw valuable design guidelines for blockchain-driven mobile data access in 5G and Beyond networks. Future work includes full-scale implementation of the proposed protocols (expected by the end of 2021), mathematical assessment and formalization of the RE-CENT protocols for blockchain decentralization, scalability and anonymity as well as extension of the proposed RE-CENT blockchain-backed service model in IoT networks

Extensive testing and validation of the proposed RE-CENT logic has also took place in WP4, providing necessary guidelines and steps towards using the RE-CENT testbed.

7. REFERENCES

[1] Ericsson Mobility Report, Ericsson, Sweden, Nov. 2020.
[2] NR; NR and NG-RAN Overall Description; Stage-2, document TS 38.300 V15.3.1, 3GPP, Jul. 2018.
[3] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, ‘‘Network slicing and softwarization: A survey on principles, enabling technologies, and solutions,’’ IEEE Commun. Surveys Tuts., vol. 20, no. 3, pp. 2429–2453, 3rd Quart., 2018.
[4] S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin, K.-W. Wen, K. Kim, R. Arora, A. Odgers, L. M. Contreras, and S. Scarpina, ‘‘MEC in 5G networks,’’ 1st ed., Eur. Telecommun. Standards Inst., Sophia-Antopolis, France, ETSI White Paper 28, Jun. 2018.
[5] T. Ouyang, Z. Zhou, and X. Chen, ‘‘Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing,’’ IEEE J. Sel. Areas Commun., vol. 36, no. 10, pp. 2333–2345, Oct. 2018.
[6] Service Requirements for the 5G System, document TS22.261, Version 17.3.0, 3GPP, Jul. 2020. [Online]. Available: http://www.3gpp.org/ftp// Specs/archive/22_series/22.261/22261-h30.zip
[7] Multi-Access Edge Computing (MEC); Framework and Reference Architecture, document ETSI GS MEC 003, Version 2.1.1, Jan. 2019.
[8] Multi-Access Edge Computing (MEC); Radio Network Information API, document ETSI GS MEC 012, Version 2.1.1, Dec. 2019.
[9] Q.-V. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le, W.-J. Hwang, and Z. Ding, ‘‘A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art,’’ IEEE Access, vol. 8, pp. 116974–117017, 2020.
[10] M. Jiangy, D. Xenakis, S. Costanzo, N. Passas, and T. Mahmoodi, ‘‘Radio resource sharing as a service in 5G: A software-defined networking approach,’’ Comput. Commun., vol. 107, pp. 13–29, Jul. 2017.
[11] A. Tsiota, D. Xenakis, N. Passas, and L. Merakos, ‘‘On jamming and black hole attacks in heterogeneous wireless networks,’’ IEEE Trans. Veh. Technol., vol. 68, no. 11, pp. 10761–10774, Nov. 2019.
[12] D. Tsolkas, E. Liotou, N. Passas, and L. Merakos, ‘‘A survey on parametric QoE estimation for popular services,’’ J. Netw. Comput. Appl., vol. 77, pp. 1–17, Jan. 2017.
[13] S. Nakamoto. (May 2008). Bitcoin: A Peer-to-Peer Electronic Cash System. [Online]. Available: https://bitcoin.org/bitcoin.pdf
[14] V. Buterin, ‘‘A next-generation smart contract and decentralized application platform,’’ Etheureum.org, Switzerland, White Paper, 2014. [Online]. Available: https://cryptorating.eu/whitepapers/Ethereum/ Ethereum_white_paper.pdf
[15] N. Laoutaris, ‘‘Why online services should pay you for your data? The arguments for a human-centric data economy,’’ IEEE Internet Comput., vol. 23, no. 5, pp. 29–35, Sep. 2019.
[16] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, ‘‘A survey of distributed consensus protocols for blockchain networks,’’ IEEE Commun. Surveys Tuts., vol. 22, no. 2, pp. 1432–1465, 2nd Quart., 2020.
[17] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim, ‘‘A survey on consensus mechanisms and mining strategy management in blockchain networks,’’ IEEE Access, vol. 7, pp. 22328–22370, 2019.
[18] YouTube. Official YouTube Stats Report. Accessed: May 14, 2021. [Online]. Available: https://www.youtube.com/intl/en-GB/about/press/

[19] Omnicore Agency. (Jan. 6, 2021). Facebook by the Numbers: Stats, Demographics & Fun Facts. [Online]. Available: https://www.omnicoreagency. com/facebook-statistics/

[20] Blockchain. StatisticsonBitcoin. Accessed: Sep. 21, 2020. [Online]. Available: https://www.blockchain.com/charts/transactions-per-second

[21] Blockchair. Statistics on Ethereum. Accessed: Sep. 21, 2020. [Online]. Available: https://blockchair.com/ethereum/charts/transactions-persecond|

[22] B. Pirus. (Jun. 2020). ETH Scalability Isn't Going to Be an Issue Soon, Suggests Vitalik Buterin. [Online]. Available: https://cointelegraph.com/ news/eth-scalability-isn-t-going-to-be-an-issue-soon-vitalik-buterinposits

[23] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, G. M. Voelker, S. Savage, and D. McCoy, ''A fistful of bitcoins: Characterizing payments among men with no names,'' in Proc. Conf. Internet Meas. Conf., Barcelona, Spain, Oct. 2013, pp. 127–140.

[24] F. Tschorsch and B. Scheuermann, ''Bitcoin and beyond: A technical survey on decentralized digital currencies,'' IEEE Commun. Surveys Tuts., vol. 18, no. 3, pp. 2084–2123, 3rd Quart., 2016.

[25] W. Chan and A. Olmsted, ''Ethereum transaction graph analysis,'' in Proc. 12th Int. Conf. Internet Technol. Secured Trans. (ICITST), Cambridge, U.K., Dec. 2017, pp. 498–500.

[26] F. Béres, I. A. Seres, A. A. Benczúr, and M. Quintyne-Collins, ''Blockchain is watching you: Profiling and deanonymizing ethereum users,'' May 2020, arXiv:2005.14051. [Online]. Available: http://arxiv.org/ abs/2005.14051

[27] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten, ''Mixcoin: Anonymity for bitcoin with accountable mixes,'' in Proc. Int. Conf. Financial Cryptogr. Data Secur., in Lecture Notes in Computer Science, vol. 8437. Berlin, Germany: Springer, Nov. 2014, pp. 486–504.

[28] L. Valenta and B. Rowan, ''Blindcoin: Blinded, accountable mixes for bitcoin,'' in Proc. Int. Conf. Financial Cryptogr. Data Secur., in Lecture Notes in Computer Science, vol. 8976. Berlin, Germany: Springer, Sep. 2015, pp. 112–126.

[29] E. Heilman, L. AlShenibr, F. Baldimtsi, A. Scafuro, and S. Goldberg, ''TumbleBit: An untrusted bitcoin-compatible anonymous payment hub,'' in Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS), San Diego, CA, USA, 2017, pp. 1–36.

[30] G. Maxwell. (2013). Coinjoin: Bitcoin Privacy for the Real World. [Online]. Available: https://bitcointalk.org/?topic=279249

[31] T. Ruffing, P. Moreno-Sanchez, and A. Kate, ''CoinShuffle: Practical decentralized coin mixing for bitcoin,'' in Proc. Eur. Symp. Res. Comput. Secur., in Lecture Notes in Computer Science, vol. 8713. Cham, Switzerland: Springer, Sep. 2014, pp. 345–364.

[32] G. Bissias, A. P. Ozisik, B. N. Levine, and M. Liberatore, ''Sybil-resistant mixing for bitcoin,'' in Proc. 13th Workshop Privacy Electron. Soc., Nov. 2014, pp. 149–158.

[33] S. Meiklejohn and R. Mercer, ''Mobius: Trustless tumbling for transaction privacy,'' in Proc. Privacy Enhancing Technol. (PETS), Barcelona, Spain, Jun. 2018, pp. 105–121.

[34] Barrywhitehat. (2018). Miximus. [Online]. Available: https://github.com/ barryWhiteHat/miximus

[35] I. A. Seres, D. A. Nagy, P. Burcsi, and C. Buckland, ''MixEth: Efficient, trustless coin mixing service for Ethereum,'' in Proc. Int. Conf. Blockchain Econ., Sec. Protoc. (Tokenomics), Paris, France, May 2019, pp. 13:1–13:20.

[36] R. Dingledine, N. Mathewson, and P. Syverson, ''TOR: The secondgeneration onion router,'' in Proc. 13th USENIX Sec. Symp., 2004, p. 21.

[37] E. Letsoalo and S. Ojo, ''Survey of media access control address spoofing attacks detection and prevention techniques in wireless networks,'' in Proc. IST-Africa Week Conf., Durban, South Africa, May 2016, pp. 1–10.

[38] N. E. Hastings and P. A. McLean, ‘‘TCP/IP spoofing fundamentals,’’ in Proc. IEEE 15th Annu. Int. Phoenix Conf. Comput. Commun., Scottsdale, AZ, USA, Mar. 1996, pp. 22–218.

[39] O. A. Osanaiye, ‘‘IP spoofing detection for preventing DDoS attack in cloud computing,’’ in Proc. 18th Int. Conf. Intell. Next Gener. Netw., Paris, France, 2015, pp. 139–141.

[40] Y.-H. Chang, K.-B. Yoon, and D.-W. Park, ‘‘A study on the IP spoofing attack through proxy server and defense thereof,’’ in Proc. Int. Conf. Inf. Sci. Appl. (ICISA), Suwon, South Korea, Jun. 2013, pp. 1–3.

[41] Re-Cent: Integrated Network Resource Sharing Service for User-CENTric Digital Content Delivery in 5G Mobile Data Networks| GitHub OpenSource Repository. Accessed: May 14, 2021. [Online]. Available: https://github.com/Fogus-Gr/RE-CENT2021

[42] Digiconomist. Bitcoin Energy Consumption Index. [Online]. Available: https://digiconomist.net/bitcoin-energy-consumption

[43] VISA. Visa Fact Sheet. Accessed: May 14, 2021. [Online]. Available: https://usa.visa.com/dam/VCOM/download/corporate/media/visanettechnology/aboutvisa factsheet.pdf

[44] S. King. (Jul. 2013). Primecoin: Cryptocurrency With Prime Number Proof-of-Work. Accessed: Nov. 3, 2018. [Online]. Available: http://primecoin.io/bin/primecoin-paper.pdf

[45] S. Park, K. Pietrzak, J. Alwen, G. Fuchsbauer, and P. Gazi, ‘‘Spacecoin: A cryptocurrency based on proofs of space,’’ Int. Assoc. Cryptol. Res., Lyon, France, Tech. Rep. 2015/528, 2015. [Online]. Available: https://eprint.iacr.org/2015/528

[46] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz, ‘‘Permacoin: Repurposing bitcoin work for data preservation,’’ in Proc. IEEE Symp. Secur. Privacy, May 2014, pp. 475–490.

[47] Filecoin: A Decentralized Storage Network, Protocol Labs, San Francisco, CA, USA, Aug. 2017.

[48] The Linux Foundation. (2018). Hyperledger Sawtooth Project. [Online]. Available: https://www.hyperledger.org/projects/sawtooth

[49] A. Kiayias, A. Russell, B. David, and R. Oliynykov, ‘‘Ouroboros: A provably secure proof-of-stake blockchain protocol,’’ in Proc. 37th Annu. Int. Cryptol. Conf. (CRYPTO), Santa Barbara, CA, USA, Aug. 2017, pp. 357–388.

[50] J. Kwon. (2014). Tendermint: Consensus Without Mining (Draft). [Online]. Available: https://tendermint.com/static/docs/tendermint.pdf

[51] V. Buterin and V. Griffith, ‘‘Casper the friendly finality gadget,’’ Jan. 2019, arXiv:1710.09437. [Online]. Available: http://arxiv.org/abs/1710.09437

[52] Parity Documentation—Aura—Authority Round. Accessed: May 14, 2021. [Online]. Available: https://openethereum.github.io/Aura

[53] I. Barinov, V. Arasev, A. Fackler, V. Komendantskiy, A. Gross, A. Kolotov, and D. Isakova, ‘‘POSDAO: Proof of stake decentralized autonomous organization, version v1.7.1,’’ Tech. Rep., May 2020, doi: 10.2139/ssrn.3368483.

[54] P. Ekparinya, V. Gramoli, and G. Jourjon, ‘‘The attack of the clones against proof-of-authority,’’ in Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS), San Diego, CA, USA, Feb. 2020, pp. 1–14. [Online]. Available: https://www.ndss-symposium.org/wp-content/uploads/2020/02/24082paper.pdf

[55] The Etheureum Hub. Ethereum 1.x Roadmap. Accessed: May 14, 2021. [Online]. Available: https://docs.ethhub.io/ethereum-roadmap/ethereum1.x/

[56] J. Poon and T. Dryja. (Jan. 2016). The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. [Online]. Available: https://lightning.network/lightning-network-paper.pdf

[57] Raiden Network. Accessed: May 14, 2021. [Online]. Available: https:// raiden.network/

[58] J. Poon and V. Buterin. (Aug. 2017). Plasma: Scalable Autonomous Smart Contracts. [Online]. Available: https://plasma.io/plasma.pdf

[59] J. Coleman, L. Horne, and L. Xuanji. (Jun. 2018). Counterfactual: Generalized State Channels. [Online]. Available: https://l4.ventures/ papers/statechannels.pdf

[60] D. Chaum, ‘‘Blind signature system,’’ in Advances in Cryptology. Boston, MA, USA: Springer, 1984.

[61] X. Ling, J. Wang, T. Bouchoucha, B. C. Levy, and Z. Ding, ‘‘Blockchain radio access network (B-RAN): Towards decentralized secure radio access paradigm,’’ IEEE Access, vol. 7, pp. 9714–9723, 2019.

[62] IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Standard 802.11-2012 (Revision of IEEE Std 802.11-2007), March, 2012.

[63] D. Xenakis, N. Passas, L. Merakos, and C. Verikoukis, ‘‘ANDSF-assisted vertical handover decisions in the IEEE 802.11/LTE-advanced network,’’ Comput. Netw., vol. 106, pp. 91–108, Sep. 2016.

[64] E-UTRA and E-UTRAN Overall Description, document TS 36.300, V10.7.0, 3GPP, Mar. 2012.

[65] D. Xenakis, N. Passas, L. Merakos, and C. Verikoukis, ‘‘Mobility management for femtocells in LTE-advanced: Key aspects and survey of handover decision algorithms,’’ IEEE Commun. Surveys Tuts., vol. 16, no. 1, pp. 64–91, 1st Quart., 2014.

[66] D. Rusakov, A. Tseitlin, D. Rubtsov, and N. Kuznetsov, ‘‘First Internet marketplace power by P2P VPN network on blockchain, release 5.0,’’ Privatix.io, Israel, White Paper, May 2018.

[67] A. Kosta, N. Pappas, and V. Angelakis, ‘‘Age of information: A new concept, metric, and tool,’’ Found. Trends Netw., vol. 12, no. 3, pp. 162–259, 2017.

[68] D. Tsolkas, E. Liotou, N. Passas, and L. Merakos, ‘‘A survey on parametric QoE estimation for popular services,’’ J. Netw. Comput. Appl., vol. 77, pp. 1–17, Jan. 2017.

[69] H. Kim and K. Chung, ‘‘Multipath-based HTTP adaptive streaming scheme for the 5G network,’’ IEEE Access, vol. 8, pp. 208809–208825, 2020.

[70] C. Baena, S. Fortes, E. Baena, and R. Barco, ‘‘Estimation of video streaming KQIs for radio access negotiation in network slicing scenarios,’’ IEEE Commun. Lett., vol. 24, no. 6, pp. 1304–1307, Jun. 2020.

[71] A. Zhu, S. Guo, B. Liu, M. Ma, J. Yao, and X. Su, ‘‘Adaptive multiservice heterogeneous network selection scheme in mobile edge computing,’’ IEEE Internet Things J., vol. 6, no. 4, pp. 6862–6875, Aug. 2019.

[72] M. Asad, A. Basit, S. Qaisar, and M. Ali, ‘‘Beyond 5G: Hybrid end-to-end quality of service provisioning in heterogeneous IoT networks,’’ IEEE Access, vol. 8, pp. 192320–192338, 2020, doi: 10.1109/ACCESS.2020.3032704.

[73] Z. Xiong, S. Feng, W. Wang, D. Niyato, P. Wang, and Z. Han, ‘‘Cloud/fog computing resource management and pricing for blockchain networks,’’ IEEE Internet Things J., vol. 6, no. 3, pp. 4585–4600, Jun. 2019.

[74] C. Fang, H. Yao, Z. Wang, W. Wu, X. Jin, and F. R. Yu, ‘‘A survey of mobile information-centric networking: Research issues and challenges,’’ IEEE Commun. Surveys Tuts., vol. 20, no. 3, pp. 2353–2371, 3rd Quart., 2018.

[75] E. Liotou, K. Samdanis, E. Pateromichelakis, N. Passas, and L. Merakos, ‘‘QoE-SDN APP: A rate-guided QoE-aware SDN-APP for HTTP adaptive video streaming,’’ IEEE J. Sel. Areas Commun., vol. 36, no. 3, pp. 598–615, Mar. 2018.

[76] System Architecture for the 5G System (5GS), document TS 23.501, Version 16.8.0, Release 16, 3GPP, Mar. 2021.

[77] J. Waring. Blog: How Many Global Base Stations are there Anyway? Accessed: May 14, 2021. [Online]. Available: https://www. mobileworldlive.com/blog/blog-global-base-station-count-7m-or-4times-higher

[78] The State of Mobile Internet Connectivity, GSMA, London, U.K., 2020.

[79] Cisco Annual Internet Report (2018–2023) White Paper, Cisco, San Jose, CA, USA, Mar. 2020.

[80] Omnicore. (Jan. 6, 2021). YouTube by the Numbers: Stats, Demographics & Fun Facts. [Online]. Available: https://www.omnicoreagency.com/ youtube-statistics/

[81] MarketingCharts. (Oct. 2019). Online Video Consumption Continues to Rise Globally. [Online]. Available: https://www. marketingcharts.com/digital/video-110520

[82] D. Xenakis, I. Zarifis, P. Petrogiannakis, A. Tsiota, N. Passas, ""Blockchain-driven mobile data access towards fully decentralized mobile video trading in 5G networks", 2020 IEEE International Conference on Communications (ICC), 7-11 June 2020, Dublin, Ireland.

[83] D. Xenakis, A. Tsiota, C. Koulis, C. Xenakis and N. Passas, "Contract-less Mobile Data Access Beyond 5G: Fully-decentralized, high-throughput and anonymous asset trading over the Blockchain", IEEE Access, vol. 9, pp. 73963-74016, 2021. doi: 10.1109/ACCESS.2021.3079625.