Enabling Collaborative Multi-Domain Applications: A Blockchain-Based Solution with Petri Net Workflow Modeling and Incentivization

Reginald Cushing
Netherlands eScience Center
Amsterdam, Netherlands
R.Cushing@esciencecenter.nl

Xin Zhou
University of Amsterdam
Amsterdam, Netherlands
X.Zhou@uva.nl

Adam Belloum
University of Amsterdam
Amsterdam, Netherlands
A.S.Z.Belloum@uva.nl

Paola Grosso
University of Amsterdam
Amsterdam, Netherlands
P.Grosso@uva.nl

Tom van Engers
University of Amsterdam
Amsterdam, Netherlands
T.M.vanEngers@uva.nl

Cees de Laat
University of Amsterdam
Amsterdam, Netherlands
C.T.A.M.deLaat@uva.nl

***Abstract*—The increasing value of data and the emergence of programmable infrastructures have paved the way for collaborative multi-domain applications across industries such as healthcare and airlines. However, such collaborations come with significant challenges, including application coordination, incentivization, and validation of execution. In this paper, we propose a novel solution that leverages blockchain technology and utilizes Petri nets for workflow modeling. Our approach involves implementing a smart contract-based workflow coordinator on the blockchain and employing a three-layered architecture to coordinate off-chain tasks. Additionally, we demonstrate the use of Petri nets for modeling economy tokens, which serve as incentives to foster collaboration among workflow parties. To validate our solution, we present a proof of concept through a simulated use case involving a multi-domain workflow for mitigating a DDoS attack. In this use case, domains collaborate by blocking offending IPs, incentivized by acquired tokens required to invoke workflows.**

***Index Terms*—Petri net, Hyperledger, blockchain, workflow, smart contract, incentives**

I. Introduction

In the past decade, blockchain-based smart contracts have rapidly developed and have been widely applied in industries, varying from finance [3], internet of things(IoT) [4], [5], supply chain [6], [7] to electronic health [8]. To leverage blockchain-based smart contracts to enforce multi-domain workflows, protocols [9] or policies [10], [11], we first need to model and verify smart contracts at the *contract-level* [12] to ensure that the translated functions describing the interactions among domains can precisely express the norms. Secondly, we need to execute the smart contract safely, which requires a collaborative infrastructure that can execute the applications in the workflows (also called applications) at the *program-level*.

However, even if these two requirements are satisfied, non-compliant behaviors can still occur during the execution of the workflow, particularly when the workflow involves tasks that must be executed off-chain, since these off-chain tasks are not visible to all parties involved in the workflow execution. To address this challenge, one approach is introducing triggers to receive and send messages between on-chain smart contract and off-chain interfaces, by this way, triggers connect blockchain to the internal process of domains and further enforce the regulated workflow [13]. In reality, however, domains may still choose to deviate from regulations even after receiving messages from the blockchain, motivated by maximizing their own utilities. For example, they might terminate certain off-chain tasks that have been triggered while claiming that the tasks have been well executed.

In such circumstances, incentives, which enable the punishment and reward of domains based on their past performance, have great potential to be deployed within the smart contract. By incorporating incentives at the contract-level, the adherence of parties to executing on-chain and off-chain tasks can be enhanced, fostering cooperation and trust among the involved parties. Therefore, the requirements for blockchain-based smart contracts are as following:

- **Contract-level**: map, encode, verify, and coordinate the tasks within the workflow, while ensuring parties are incentivized to fullfill their tasks, especially the off-chain ones
- **Program-level**: build collaborative infrastructure that is capable for identity management, data-flow management, and control-flow management

In this work, we propose a solution that leverages Petri net and blockchain technologies to coordinate multi-domain work-

The work as presented in this paper has been done as part of the Dutch Research project ¨Data Logistics for Logistics Data" (DL4LD), supported by the Dutch Organisation for Scientific Research (NWO), the Dutch Institute for Advanced Logistics TKI Dinalog (http://www.dinalog.nl/) and the Dutch Commit-to-Data initiative (http://www.dutchdigitaldelta.nl/big-data/over-commit2data). Special thanks goes to CIENA for their continued support of this research through funding.

flows with incentives integrated, meeting the requirements of contract-level and program-level. The remainder of this paper is organized as follows: Section II introduces the fundamental concepts involved in the one-box solution; followed by the concrete deployment of the blockchain-based Petri nets in Section III; in Section IV, we illustrate the functionality of the proposed solution through a distributed denial-of-service (DDoS) use case. Finally, we conclude the findings and compare our solution with related works in section V.

II. Preliminaries

A. Blockchain and Smart Contract

Blockchain is a distributed ledger of blocks that preserves the integrity and immutability of data. All blocks are linked by cryptographic hashes that contain information from the previous block, the timestamp, and transaction data, allowing to create immutable chains of blocks. The key feature of blockchain technology is the ability for each party to maintain a consistent copy of the blockchain. This mechanism enables multiple domains to transact without the need for a trusted central server.

The functionality of blockchain is extended and expanded with the emergence and development of smart contracts. As a computer protocol designed for executing applications, smart contract can maintain the workflow among multi-domain when coupled with blockchain [15]. Consequently, the combination of smart contracts with blockchain technologies has made it possible to enforce more complex rules, contracts, and policies among multiple domains meanwhile tracking the progress of execution.

Blockchain-based smart contract has been widely and rapidly developed [2]. The foundation of blockchain-based smart contracts is built on four key components as illustrated in Figure 1: *smart contracts*, which enable the execution of complex agreements by automating the processes and enforcing the tasks in the workflow; *ledger*, which provides complete and immutable records shared by all the peers; *wallets*, which utilize a Public Key Infrastructure(PKI) system to allow users access to the ledger, and assign ownership of ledger records using key signatures; and *consensus*, which ensures the agreement of peers regarding the ledger. These four components work together to create a secure and decentralized system for executing agreements across multiple domains.

It is worth noting that since all parties update the same ledger, there is a potential risk of breaking consistency. Consensus itself can be an attack vector, for example in Sybil attack, where a peer controlling the ordering can control what gets written. To mitigate the risk, most public blockchain setups apply proof-of-work consensus, which requires an attacker to control more than 50% of the compute power to control the network [16]. Proof-of-work consensus provides higher security at the cost of high computational power. However, in less malicious environments, traditional consensus such as Paxos or Raft can be used [17] to decrease costs. In this study, we consider the scenario that all parties intend to collaborate and have a semi-trustful relationship with other peers. Hence, we leverage Hyperledger default consensus algorithm which is Raft.

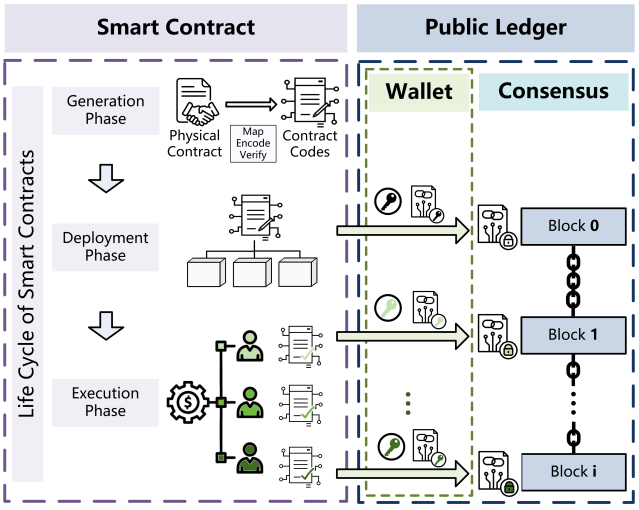

Fig. 1. Four pillars of the blockchain-based smart contract: smart contract, ledger, wallets, and consensus. The life cycle of smart contracts consists of three phases: generation, deployment, and execution. Starting from the deployment phase, the completion of every activity or operation is recorded as a new block on the public ledger. The successful recording relies on wallet and consensus. The wallet uses a public key infrastructure system, enabling the identification of block creator. Meanwhile, the consensus algorithm ensures that all parties agree on the records on the public ledger.

B. Petri Nets

As Figure 1 illustrates, smart contract generation is the starting point of smart contract life cycle. At this phase, the physical contract needs to be mapped and encoded into executable codes, where the properties of interactions and the external environment can to be expressed and verified. Approaches such as process algebras [18], set-based methods [19], and state-transition systems [20] are commonly used. Considering the aim of this work, and the fact that state-transition systems can naturally model the business artifacts in process-oriented contracts, we choose the state-transition systems to map and verify the workflow among the domains.

Petri net is a representative state-transition language proposed by Carl Adam Petri [21]. They consist of three fundamental elements: *places*, *transitions*, *tokens*, and *arcs* which enable Petri nets to model the process in workflows, policies, or protocols. Figure 2 gives an example of a classical Petri net. A Petri net is a graph of places and transitions. Places are token holders and transitions move token from input places to output places. A transition can be considered as an action and is connected to input/output places by arcs. When all input places have sufficient tokens, the following transition will be fired, during which the actions that represented by the transition is executed. Subsequently, new tokens are generated by the fired transition, and placed in all the output places of the fired transition. The distribution of tokens among places is called a *marking*, which denotes the current state of the net.

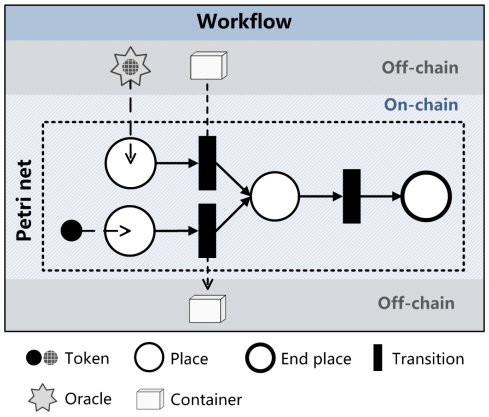

Fig. 2. The Petri net is a state-transition language used to concurrent processes, consisting of three fundamental elements: tokens, places, and transitions. Places are connected by transitions. A transition is fired when all input places have the necessary amount of tokens (classically it is one token). On firing a transition, tokens are produced in all output places of the transition. Markings, which record the current distribution of tokens, reflect the state of the Petri net.

Petri nets not only have the advantages in intuitively represent the entire workflow as well as the real-time process state, but also have well-defined mathematical properties such as reachability, liveness, deadlock and boundedness which can be used for verifying the correctness of the application. In [22] the mathematical properties are used to enhance the security of smart contracts. Petri nets have been applied in coordinating multi-domain workflows, for example [23] applied an extended Petri nets that equipped with Oracle interfaces to cope with workflows that require external data. As Figure 2 shows, when the external requirement is satisfied, Oracle interface can put the token into places and trigger the associated tasks[1]. However, the off-chain tasks are not limited to data access control, some complicated tasks might cannot be completely enforced. For instance, in a supply chain workflow, a manufacturer may fail to deliver the product to the wholesaler, but still declare and add the record of "task accomplished" on the blockchain. Such deviation to the workflow might lead to the collapse of cooperation. Therefore, this work aims at implementing incentives into the workflow that modelled by Petri nets for further enhancing the enforcement of off-chain tasks in multi-domain workflows, and promoting cooperation and trust among parties.

C. Token Economy

Token economy is prevalent in behavior modification programs in social science [24], [25]. These programs typically consist of three essential components: the *target behaviors* that wish to reinforce; the *tokens* earned for engaging in those behaviors; and the *back-up reinforcers* that can be obtained by exchanging tokens as rewards. For example, in a supply chain workflow, a wholesaler may incentivize manufacturers to deliver products on time by awarding badges to who consistently perform the desired behavior. Manufacturers with the highest number of badges are then rewarded the privilege of extending the cooperation period. In this example, the badges are tokens, and the privilege of extending cooperation period is the back-up reinforcer.

It is important to note that the term "token" in the context of "Token Economy" has a different definition and functions than in Petri net. In token economy, a token is an abstract concept that can take the form of any objects or symbols that work as secondary enforcers. Tokens themselves are worthless, but they can be exchanged for other valuable things. Hence, participants are motivated to engage in desired behaviors to earn tokens. The primary function of tokens in token economy is serving as a intermediary in enforcing incentives. In contrast, in Petri net, tokens are a fundamental element that triggers the firing of transitions. They do not serve as incentives but are essential for executing tasks. Without tokens, transitions cannot be fired, and tasks cannot be performed. To distinguish these two types of token, we denote the secondary enforcers in Token Economy as *E-tokens*, and the ones in Petri nets as tokens.

In this work, we realize token economy by employing a peer-audit based E-tokens assignment process in the classical Petri nets. More concretely, parties involved in the blockchain are audited by their peers. Those who successfully complete their on-chain and off-chain tasks have a higher chance of being assigned E-tokens. For the first cooperation, each party is assigned one E-token by default. E-tokens are required to authorize and activate the Petri nets, hence only parties with E-tokens can participate in the workflow. As being involved in the workflow is valuable for participants, the opportunity for future involvement serves as a backup reinforcer to incentivize the cooperation of parties. We refer to these token economy implemented workflow as "incentive-integrated workflows".

Incentive-integrated workflows not only encourage parties to fulfill their tasks in the workflow but also encourage cooperation by incentivizing the validation of the peers and rewarding correct execution of workflow tasks. In the following section, we present detailed steps of implementing our incentive-integrated workflows using Petri nets for modeling and coordinating multi-domain applications.

III. Incentive-integrated workflows on Hyperledger

In this section, we first illustrate how we integrate "E-token assignment" in Petri nets, and then introduce the three-layer architecture that enables Petri nets to coordinate the on-chain and off-chain tasks involved in incentive-integrated workflows.

A. Incentive-integrated Workflows

A classical workflow describes the schedule of tasks needed to complete the application. In our incentive-integrated workflows, we include an additional stage called the "incentive stage" before the workflow is completed, as shown in Figure 3. The incentive stage is for generating and assigning E-tokens,

[1]There are many variations of Petri nets, in our work we limit the usage of classic Petri nets to model workflow-like applications.

it consists two types of transitions: "peer audit" transitions and "E-token assignment" transitions. As shown in Figure 3, when generated tokens are placed in "P1" and "P2", "peer audit" transitions are triggered

In each peer audit transition, the parties decide whether to assign an E-token to the party being evaluated based on their own observation of that party's task execution. For example, in a supply chain workflow, if the wholesaler observes that the manufacturer did not delivery the products as regulated, despite the manufacturer's claim of doing so, the wholesaler has the authority to withhold the assignment of an E-token to the manufacturer.

The Petri net records the successful peer audit transitions through its token mechanism. When the E-token assignment transition has tokens in all its inputs it will be fired and generate an E-token for the party under evaluation. The E-token is automatically translated to an authorization token which is needed to start the whole workflow in the first place.

Various aggregation algorithms for the final E-token assignment decision can be implemented, such as "veto power" [26], "majority rule" [27], etc. The concrete aggregation mechanism can be implemented by designing the E-token assignment process of Petri nets. For example, in both Figure 3 and Figure 6, the veto power is implemented: only when all other parties have vote for the party under evaluation, it will be rewarded with one E-token. After completing the peer audit and E-token assignment transitions, the marking of the Petri net finally reaches the end places.

For the first round of the workflow, E-tokens are automatically assigned to all parties by default. However, for the subsequent rounds of the same workflow, E-tokens are assigned based on the aforementioned E-token assignment process. As a result, parties who gain E-tokens can participate in the next round of cooperation, while parties who receive no E-tokens are excluded from future rounds of cooperation. This peer monitoring mechanism ensures that tokens serve as timely feedback on the behaviors of the involved parties, which encourage them to complete their off-chain tasks fully and contribute to the workflow in an honest manner. Incentives are hence integrated in the workflow using Petri nets to model it. The following section elaborates on how to deploy and realize this incentive-integrated workflow at the program-level, especially involving multiple domains on the blockchain.

B. Implementation on Hyperledger

Blockchain technology provides a shared and transparent ledger, which enables all parties involved in a workflow to track the progress of the workflow in real-time. Additionally, blockchain provides a platform for executing on-chain tasks, such as calculation, verification, authentication, and others. However, for complicated workflows, a central broker is often necessary to coordinate the multi-domain composition of tasks. One popular approach for achieving this coordination is choreography, where a smart contract acts as a broker to invoke applications and monitor tasks [28]. In this work, we use Petri-nets as the broker of the choreography, allowing for

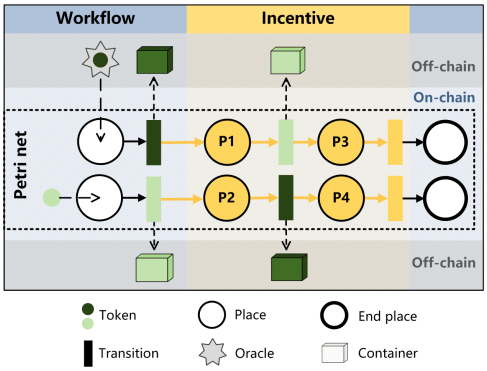

Fig. 3. An example of incentive-integrated workflows. There are two stages in the workflow, the "work stage" and the "incentive stage". Within the work stage, the first two transitions correspond to two off-chain tasks executed respectively by the involved two parties, distinguished by the colors deep green and light green. After these two workflow transitions are fired, tokens are generated and placed in "P1" and "P2". Peer audit transitions in the incentive stage are then triggered, where parties decide whether assign E-tokens to each other. Since only two parties are involved, each of them is then evaluated by the other. If parties decides to assign an E-token to the being evaluated one, new tokens will be generated and placed at places "P3" or "P4", which trigger the subsequent on-chain E-token assignment transitions. Consequently, the Petri net reaches the end place. Only parties who have received tokens are able to authorize and activate the next round of the workflow.

a decentralized coordination of both on-chain and off-chain tasks. In the following, we introduce how we deploy incentive-integrated workflow on blockchain and how we use Petri net to enable the decentralized coordination of the workflow tasks.

1) *Map Petri nets to smart contracts*

In this work, we use Hyperledger[2] as the basis of our blockchain technology. Hyperledger is an open source blockchain technology with comparable functionalities to other smart-contract able public blockchain such as Etherium. A main difference with public blockchain is that Hyperledger is a permission-based blockchain meaning that anyone can setup a private ledger using the concept of channels. This gives us the ability to setup private ledgers specific to consortia collaborations and allows the setup of semi-trusted environments. In Hyperledger, smart contract functionality is supported through the concept of chaincode, which are JavaScript or Go scripts running as peers in the different domains. Chaincode allows us to map Petri nets to smart contracts. This is achieved by implementing a Petri net executor as a JavaScript chaincode.

The Petri net executor exposes functions that modify the Petri net such as the function *PutToken()* which puts a token in a place. This function will (after putting a token) analyse the Petri net to find which transition(s) need to be fired and generates firing event(s). Event(s) will trigger the off-chain functionality which in-turn triggers more movement of tokens.

The actual Petri net graph is modelled as a set of assets on Hyperledger. These assets are: *tokens*, *places*, *transitions* and *arcs* as introduced in Section II-B. To make our Petri net function like a workflow we have two types of tokens: data-

[2]www.Hyperledger.org

tokens which carry data and pass data between transitions/-tasks; auth-tokens which act as authorization tokens and are the product of the E-token incentive phase. One of the initial places of the Petri net is designed to accept only auth-tokens, effectively limiting the parties who can execute the Petri net. Since any element of a Petri net is modelled as an asset this also means that tokens, places, transitions and arcs are owned by organizations. This ownership introduces an extra level of control e.g. an auth-token cannot be generated by the party that it belongs to, but has to be generated by an other party and then transferred. Similarly, transitions can only be modified by the party who owns them.

Places are typed, they can either accept auth-tokens or data-tokens. Tokens can be reusable or disposable (by default). This means when a token is used to fire a transition, its state becomes *USED* and can not be used anymore (need auth-tokens) while data-tokens can be set to reusable so that the same token can be used in multiple runs of the workflow. Transitions have an associated function to execute as part of the transition. This would be the actual workflow-task modelled by the transition. The architecture implements a plug-in architecture to extend the functionality of tasks. E.g. a *Docker* plug-in is able to run off-chain tasks in Docker containers.

To deploy Petri nets on the blockchain (as presented in the Deploy Phase of Figure 1), parties need to define the four fundamental elements of Petri nets: places, transitions, tokens and arcs. This is done using a JSON file where each element is described and its parameters defined (e.g. owner). These elements are considered as private asset of the parties. For instance, an organization can define the required number of tokens in the input places of a transition; and the number of generated tokens and output places of a transition. In this way, complicated workflows can be mapped to Petri nets. Once Petri nets are well defined, they can be deployed on the blockchain. However, since workflows can involve on-chain and off-chain tasks, to trigger the off-chain applications, the blockchain layer and the infrastructure layer needs to communicate. To achieve this, a three-layer architecture, as shown in Figure 4, is used to coordinate multi-domain workflows involving both on-chain and off-chain tasks.

2) *Three-layer architecture*

Figure 4 presents the three-layer architecture composed of *blockchain layer*, *network layer*, and *infrastructure layer*. On the blockchain layer, a Petri net is deployed as a smart contract on the Hyperledger. Parties with wallets can invoke functions on the smart contract, updating the ledger with markings that represent the progress of workflow execution through the distribution of tokens; and can also listen for specific markings, such as firing events and then trigger the corresponding off-chain tasks at the proper time.

Since parties are in possession of a wallet (private/public key), they can sign any off-chain action with their key, thus any off-chain activity can validate the command is issued by the correct wallet/organization. The party is also responsible to call back to the ledger once the off-chain task has been completed by calling a *CompleteTransition()* function which will put tokens in the output places of the transition. The transactions are recorded to the ledger and the execution of the Petri net continues.

The parties' blockchain interface clients (programs with wallets talking to the blockchain) collectively form a bridge between the on-chain and off-chain, and we denote this bridge as the *network layer*. The network layer serves as a bridge between the blockchain layer and the infrastructure layer where off-chain tasks are executed. To enable the interaction between these two layers: system architecture including hardware and software of parties at the network layer keeps listening, and executes the off-chain tasks once monitors the corresponding input places are placed tokens; after the off-chain task is completed, the latest marking is recorded as a new block and linked to the Hyperledger on the blockchain layer. Wallets used by the different parties in the network layer help keep track of who did what as recorded in the ledger. This tamper-proof synchronization of the markings enables all parties to easily track the progress of the workflow. The interaction among these three layers ensures the coordination and tracing of on-chain and off-chain tasks in the workflow(Figure 4).

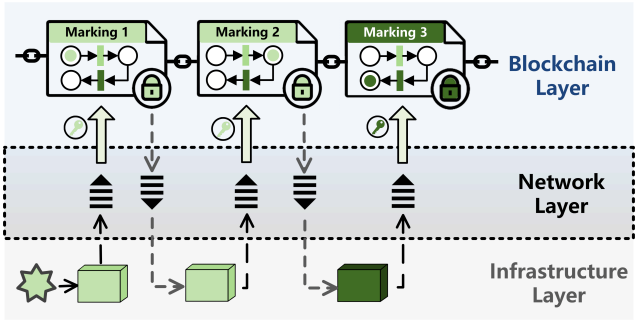

Fig. 4. The three-layer architecture for coordinating multi-domain workflow using Petri Nets. In this architecture, Petri nets depicting the abstract multi-domain workflows are deployed on the blockchain layer. When executing Petri nets, the completion of each task accompanies with new tokens being generated and corresponding markings being created and linked to the existing blocks. When the tasks are executed off-chain, the Hyperledger is updated through the network layer. Meanwhile, the network layer allows off-chain architecture components to listen to the latest markings on the blockchain layer, enabling domains to execute tasks at the appropriate time. In this manner, this three-layer architecture realizes a decentralized and transparent choreography of multi-domain workflows.

Noteworthy, in order to enhance the security and ensure the confidence among parties, our solution includes an *activation step* after deploying the Petri net. This step requires all parties to authorize the deployed Petri nets by signing the Petri net (the arcs asset on Hyperledger). Because unlike a singular domain where places, transitions, and tokens are owned by one domain, in a multi-domain Petri net, these three elements are owned by different domains. It necessitates the agreement of all parties on the Petri net. Therefore, the activation step ensures a global agreement.

So far, we discussed the deployment and execution of incentive-integrated workflow. In the next section, we illustrate the application of our in-box solution through a concrete use

case, where an alignment of semi-trust parties cooperate to enforce a designed workflow for mitigating DDoS attacks.

IV. A DDoS USE CASE

A distributed denial-of-service (DDoS) attack is a type of services attack that overloads a system by flooding it with requests, preventing legitimate requests from being executed. One of the challenges in containing DDoS attacks is that blocking the detected IP address of the malicious host is always not enough for the attacked domains. The illegitimate requests can easily break the block from other unblock legitimate domains since domains are connected [29]. Therefore, one possible solution is building an alliance where members can share information and block the illegitimate IP addresses at the same time [30]. In our use case, we assume there is such an semi-trust alliance to prevent DDoS attacks through the following protocol:

> When a certain sensor in domain 'A' is attacked by an illegitimate IP address, 'A' needs to block the IP address and notify other members in the alliance, who have the obligation to block the illegitimate IP address after receiving the notification.

This protocol outlines the workflow for a semi-trust alliance to mitigate DDoS attacks. We first map this workflow into an Petri net, and then simulate its enforcement. In our use case, the alliance consists of three domains, represented by different colors in Figure 5. When a malicious host attacks a domain, the domain triggers the activated Petri net by placing its auth-token at the start place. The subsequent transition, notifying alliance members with the illegitimate IP address, is then fired, followed by the generation of new tokens placed at the output places of the transition. The next three parallel transitions represent the blocking the IP address for each domain. Although these transitions are fired in one sequence in Figure 5, they can be executed in different orders in practice. Each transition results in generated tokens and markings, followed by the incentive stage.

To ensure the successful mitigation of DDoS attacks, it is crucial that all members of the alliance adhere to the predefined workflow. To motivate semi-trust members to fulfill their obligations, we implement incentives by integrating an incentive stage, which composes of the auditing transition and the E-token assignment transition, into the workflow. In the auditing transition, each domain is evaluated by the other domains in the alliance. Only if all other domains vote to assign, can the subsequent on-chain E-token assignment transition be triggered, and the evaluated domain receive an E-token. Similar to the blocking transitions, the peer audit transitions of all domains are parallel and can be accomplished in any order in practice. For simplicity, Figure 6 only presents the incentive stage of the attacked domain, the incentive stage of the other two domains are similar. Ideally, after the incentive stage, the dishonest domains will receive no E-tokens (namely no auth-tokens). Consequently, by such in-time feedback on performance, domains are motivated to be honest and responsible.

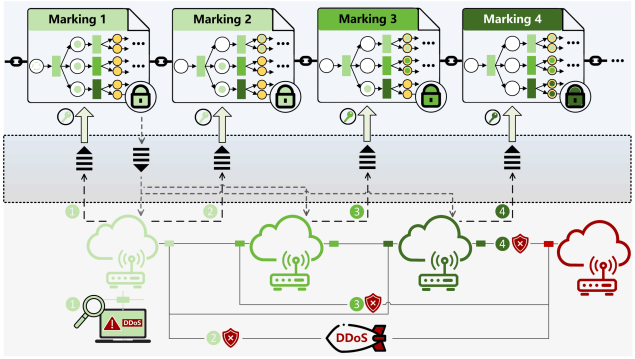

Fig. 5. The choreography of workflow in a semi-trust alliance leveraging the three-layer architecture. First, the workflow of the DDoS-resistant protocol is mapped to a Petri net and deployed on Hyperledger. When a malicious host attacks a domain, the domain triggers the activated Petri net by placing its token at the starting point. This initiates the subsequent transition to notify alliance members of the illegitimate IP address, followed by the creation of a new token at the output place of the transition. When the corresponding new marking published on the Hyperledger, alliance members can listen to it through the network layer and execute the following parallel transitions, referring to blocking the IP address. Each completed transition generates tokens which trigger the subsequent peer audit transitions in the incentive stage.

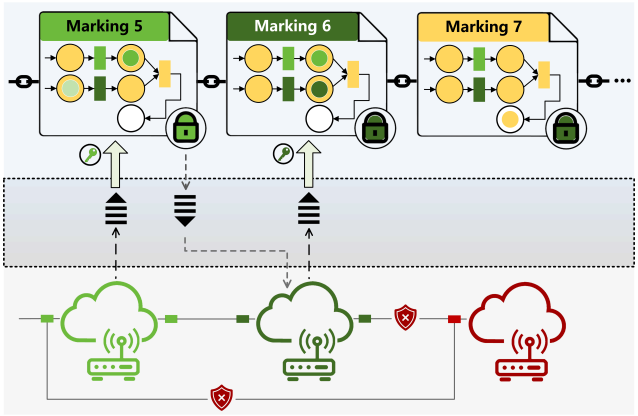

Fig. 6. The incentive stage of Petri net in the DDoS use case. Two types of transitions are involved in this incentive stage, 1) peer audit transitions, where parties are audited by other cooperators who decide whether to vote; 2) E-token assignment transitions, in which the evaluated parties are assigned auth-tokens. But whether the E-token assignment transitions are triggered depends on both the voting decisions of parties and the design of the aggregation algorithm. For example, in this case we implement the veto power to aggregate the result of peer audit by requiring two tokens to trigger the E-token assignment transitions. Only when both the other two parties vote will the evaluated party be assigned an auth-token.

To simulate the DDoS use case we are using Kathara network emulator[3] to create a hypothetical Internet scenario. Kathara emulates a network as a set of Docker containers, where each device (e.g. router, web-server) is a container and collision domains are Docker networks. This enables a functioning network where we can interact with the actual devices at the infrastructure layer. To connect the infrastructure

[3]www.kathara.org/

layer to the blockchain layer, we use an MQTT[4] message queue at the network layer. Parties' blockchain interface clients at the network layer sign and send commands to the MQTT server/s which are subsequently picked up by the router devices in the Kathara emulator. The devices have a white list of public keys to verify the signature of the command. Upon validation of the signature the device proceeds to execute the command (update routes to block IP).

Our use-case employs a Petri net workflow, where each party corresponds to an Internet Autonomous System (AS). Each AS manages its own network router, forming an interconnected network for Internet traffic. The workflow directs ASs to block offending IPs on their routers when an attack is detected by any party. Collaboration comes into play because tokens owned by a number of parties are signalling other parties to trigger transitions. Following router transition actions, a two-step cross-validation process ensures successful off-chain actions. First, an application-specific off-chain routine is executed, followed by ledger updates. It's worth noting that in our demo, we haven't yet implemented the off-chain routine, which would involve one AS verifying another's IP blocking, requiring additional infrastructure monitoring services. The validation steps yield authorization tokens that enable parties to participate in the subsequent invocations of the workflow. The code for the demo is available online[5]

V. Conclusion and Discussion

To enforce and coordinate multi-domain workflows, mapping predefined tasks to smart contracts on the contract level and building collaborative infrastructures in the program-level are two unavoidable requirements. To satisfy the first requirement, this work generates smart contracts based on Petri nets, due to their advantages in workflow descriptiveness and process logic verification. For the second requirement, we construct a three-layer architecture where the blockchain layer serves as a trusted storage of the smart contract and the execution state of the workflow; at the infrastructure layer, the private system architecture of domains can be triggered at the proper time to execute the corresponding activities by listening to the state of the workflow through the network layer. Once domains accomplished activities, corresponding new blocks that record the latest state of the workflow will be added to the blockchain. Accordingly, our proposed solution enables a in-time tractable and auditable multi-domain workflow coordination.

Lots of explorations have been made in utilizing blockchain in facilitating secure inter-organizational workflow [32]. Since 2016, Weber et al., has proposed an approach that uses Solidity smart contract to execute multi-domain workflow [13], [33], where the smart contract is deployed on-chain and works as a centralized mediator or choreography monitor. Instead of focusing on monitoring and mediating, [34] proposed the framework "ChorChain" for better enforcing and auditing the activities in the workflow through event-based gates to allow only conforming operations being executed, and providing records retrieval interfaces for users. Some other works focus on multi-domain confidential data sharing within workflow, for example, [31] proposed an "Encrypter" framework ensuring data integrity and the confidentiality of data exchanged on the blockchain, where data exchange is encrypted and can only be decrypted by authorized organizations. [28], in contrast, addressed the challenge of enforcing access control policies by managing the data access authorizations in the coordination layer implemented on the blockchain framework.

These approaches can handle workflows that involve on-chain tasks, such as data storage and computation, as well as tasks that require off-chain data access. However, some off-chain tasks are difficult to monitor or enforce using these approaches. For example, in the use case of DDoS attack, it is challenging to guarantee that domains adhere to the protocol that blocking the illegitimate IP address, even if domains declare their accomplishment of the task by creating new markings on the Hyperledger. This is an inherited limit of smart contracts. To address this issue and facilitate cooperation among such semi-trust domains, we integrate an incentive stage into traditional Petri nets, where each party is audited by their peers. Non-compliant parties that fail to fulfill their obligations are less likely to receive E-tokens, and further lose their chance for future collaboration since E-tokens are required to activate Petri nets. Notably, when a Petri net is deployed for the first time, E-tokens are by default assigned to every party. This mechanism eliminates non-compliant domains and encourages domains to be honest and compliant, thus improving the mutual trust among parties.

To conclude, implementing the Petri net on the three-layer architecture can choreograph multi-domain workflow and enforce the predefined tasks to a greater extent, where on-chain tasks are enforced by smart contract, while off-chain tasks are incentivized through E-tokens. Additionally, our solution records the state of the workflow on the Hyperledger in real-time, providing the benefits for tracking and post-auditing. To the best of our knowledge, this is the first work that executes Petri nets with incentives into smart contracts in multi-domain workflow coordination. Our solution can be applied to various workflows involving hard-to-enforce tasks, such as those in the internet of things, supply chain management, or federated learning.

It is important to note that although we use peer-auditing to monitor semi-trust parties, this approach relies on the assumption that failures to adhere to the workflow can be observed by the parties. For example, in our use case, a domain that fails to block the illegitimate IP address in an DDoS attack may impact other domains, enabling other domains to recognize and evaluate their peers' performance. However, there are instances where non-compliant behaviors may not be immediately recognized due to their high concealment or delayed impact. In such cases, our integrated incentives cannot give effective feedback. To address this limitation, external auditing system implementing incentives may be nec-

[4]mqtt.org

[5]doi.org/10.5281/zenodo.8341111

essary. Nevertheless, the Hyperledger which records domains' declared successful activities still benefits the post-auditing process by recording the asserted accomplished activities. Therefore, our solution can be considered as a complementary approach to existing solutions in further enforcing off-chain workflow activities.

Another limitation of the solution is the simplification of the aggregation algorithm in the E-token assignment, the veto power that we implemented may result in the honest and reliable domains being squeezed out if any of the parties maliciously give poor evaluation. To avoid this risk, future work should explore more comprehensive aggregation algorithms. Additionally, inspired by previous work [35], where a reputation system is built, and only domains have high reputation is able to build the Hyperledger, we suggest that another way to address the E-token assignment matter is to build an independent reputation chain and allow domains to select collaborators based on the reputation chain. This would encourage domains to behavior well for maintaining a good reputation and increasing their chances of future cooperation, while also prevent domains from being completely deprived of the opportunity to collaborate.

REFERENCES

[1] D. Yaga, P. Mell, N. Roby, K. Scarfone, Blockchain technology overview, 2018. doi:https://doi.org/10.6028/NIST.IR.8202.

[2] S.-Y. Lin, L. Zhang, J. Li, L.-l. Ji, Y. Sun, A survey of application research based on blockchain smart contract, Wireless Networks 28 (2022) 635–690.

[3] F. Schär, Decentralized finance: On blockchain-and smart contract-based financial markets, FRB of St. Louis Review (2021).

[4] A. Dorri, S. S. Kanhere, R. Jurdak, Blockchain in internet of things: challenges and solutions, arXiv preprint arXiv:1608.05187 (2016).

[5] T. H. Pranto, A. A. Noman, A. Mahmud, A. B. Haque, Blockchain and smart contract for iot enabled smart agriculture, PeerJ Computer Science 7 (2021) e407.

[6] R. C. Koirala, K. Dahal, S. Matalonga, Supply chain using smart contract: a blockchain enabled model with traceability and ownership management, in: 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), IEEE, 2019, pp. 538–544.

[7] S. E. Chang, Y. Chen, When blockchain meets supply chain: A systematic literature review on current development and potential applications, IEEE Access 8 (2020) 62478–62494.

[8] R. Casado-Vara, J. Corchado, Distributed e-health wide-world accounting ledger via blockchain, Journal of Intelligent & Fuzzy Systems 36 (2019) 2381–2386.

[9] R. Li, T. Song, B. Mei, H. Li, X. Cheng, L. Sun, Blockchain for large-scale internet of things data storage and protection, IEEE Transactions on Services Computing 12 (2018) 762–771.

[10] X. Zhou, R. Cushing, R. Koning, A. Belloum, P. Grosso, S. Klous, T. van Engers, C. de Laat, Policy enforcement for secure and trustworthy data sharing in multi-domain infrastructures, in: 2020 IEEE 14th International Conference on Big Data Science and Engineering (BigDataSE), IEEE, 2020, pp. 104–113.

[11] S. Salonikias, M. Khair, T. Mastoras, I. Mavridis, Blockchain-based access control in a globalized healthcare provisioning ecosystem, Electronics 11 (2022) 2652.

[12] P. Tolmach, Y. Li, S.-W. Lin, Y. Liu, Z. Li, A survey of smart contract formal specification and verification, ACM Computing Surveys (CSUR) 54 (2021) 1–38.

[13] I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, J. Mendling, Untrusted business process monitoring and execution using blockchain, in: Business Process Management: 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings 14, Springer, 2016, pp. 329–347.

[14] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, Decentralized Business Review (2008) 21260.

[15] N. Szabo, Smart contracts: building blocks for digital markets, EXTROPY: The Journal of Transhumanist Thought,(16) 18 (1996) 28.

[16] J. Garay, A. Kiayias, N. Leonardos, The bitcoin backbone protocol: Analysis and applications, in: Advances in Cryptology-EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II, Springer, 2015, pp. 281–310.

[17] A. Singh, G. Kumar, R. Saha, M. Conti, M. Alazab, R. Thomas, A survey and taxonomy of consensus protocols for blockchains, Journal of Systems Architecture 127 (2022) 102503. URL: doi:https://doi.org/10.1016/j.sysarc.2022.102503.

[18] M. Qu, X. Huang, X. Chen, Y. Wang, X. Ma, D. Liu, Formal verification of smart contracts from the perspective of concurrency, in: Smart Blockchain: First International Conference, SmartBlock 2018, Tokyo, Japan, December 10–12, 2018, Proceedings 1, Springer, 2018, pp. 32–43.

[19] K. Hu, J. Zhu, Y. Ding, X. Bai, J. Huang, Smart contract engineering, Electronics 9 (2020) 2042.

[20] R. Hull, V. S. Batra, Y.-M. Chen, A. Deutsch, F. F. T. Heath III, V. Vianu, Towards a shared ledger business collaboration language based on data-aware processes, in: Service-Oriented Computing: 14th International Conference, ICSOC 2016, Banff, AB, Canada, October 10-13, 2016, Proceedings 14, Springer, 2016, pp. 18–36.

[21] C. A. Petri, Kommunikation mit automaten (1962).

[22] N. Zupan, P. Kasinathan, J. Cuellar, M. Sauer, Secure smart contract generation based on petri nets, in: Blockchain Technology for Industry 4.0: Secure, Decentralized, Distributed and Trusted Industry Environment, Springer, 2020, pp. 73–98.

[23] P. Kasinathan, J. Cuellar, Workflow-aware security of integrated mobility services, in: Computer Security: 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part II 23, Springer, 2018, pp. 3–19.

[24] A. Kazdin, The token economy: A review and evaluation (2012).

[25] J. W. Ivy, J. N. Meindl, E. Overley, K. M. Robson, Token economy: A systematic review of procedural descriptions, Behavior Modification 41 (2017) 708–737.

[26] D. J. Brown, Aggregation of preferences, The Quarterly Journal of Economics 89 (1975) 456–469.

[27] L. Bouton, A. Llorente-Saguer, F. Malherbe, Get rid of unanimity rule: The superiority of majority rules with veto power, Journal of Political Economy 126 (2018) 107–149.

[28] C. Rondanini, B. Carminati, F. Daidone, E. Ferrari, Blockchain-based controlled information sharing in inter-organizational workflows, in: 2020 IEEE International Conference on Services Computing (SCC), IEEE, 2020, pp. 378–385.

[29] H. A. Khattak, M. A. Shah, S. Khan, I. Ali, M. Imran, Perception layer security in internet of things, Future Generation Computer Systems 100 (2019) 144–164.

[30] M. Hajizadeh, N. Afraz, M. Ruffini, T. Bauschert, Collaborative cyber attack defense in sdn networks using blockchain technology, in: 2020 6th IEEE Conference on Network Softwarization (NetSoft), IEEE, 2020, pp. 487–492.

[31] B. Carminati, C. Rondanini, E. Ferrari, Confidential business process execution on blockchain, in: 2018 ieee international conference on web services (icws), IEEE, 2018, pp. 58–65.

[32] B. Carminati, E. Ferrari, C. Rondanini, Blockchain as a platform for secure inter-organizational business processes, in: 2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC), IEEE, 2018, pp. 122–129.

[33] O. López-Pintado, L. García-Bañuelos, M. Dumas, I. Weber, Caterpillar: A blockchain-based business process management system., BPM (Demos) 172 (2017).

[34] F. Corradini, A. Marcelletti, A. Morichetta, A. Polini, B. Re, F. Tiezzi, Engineering trustable and auditable choreography-based systems using blockchain, ACM Transactions on Management Information Systems (TMIS) 13 (2022) 1–53.

[35] H. Watanabe, S. Fujimura, A. Nakadaira, Y. Miyazaki, A. Akutsu, J. Kishigami, Blockchain contract: Securing a blockchain applied to smart contracts, in: 2016 IEEE international conference on consumer electronics (ICCE), IEEE, 2016, pp. 467–468.