TechRxiv
Powered by IEEE

Quantitative Cooperation Analysis among Cross-chain Smart Contracts

This paper was downloaded from TechRxiv (https://www.techrxiv.org).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

13-05-2021 / 14-05-2021

CITATION

Su, Hong (2021): Quantitative Cooperation Analysis among Cross-chain Smart Contracts. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.14583399.v1

DOI

10.36227/techrxiv.14583399.v1

Quantitative Cooperation Analysis among Cross-chain Smart Contracts

Hong Su

***Abstract*—In cross-chain scenarios, there are different blockchains, which need to cooperate. Cooperation among different blockchains is done by smart contracts that work together to complete cross-chain tasks. When numerous cooperative smart contracts are involved, smart contracts form a complex interaction network, which makes it difficult to evaluate the cooperation. It needs a common model to quantitatively analyze the cross-chain cooperation of associated smart contracts. In this paper, we model the cooperation among smart contracts as conditions and their corresponding actions, the condition-trigger model. Then we propose the method to calculate the cooperation probabilities by the graph weight. As the edge weight lacks the information of interaction probabilities, we introduce the dimension of the edge weight to calculate the probabilities. Finally, we verify the proposed condition-trigger model and its different types. It demonstrates that our proposed methods can effectively analyze the cross-chain cooperation among smart contracts.**

***Index Terms*—cross-chain cooperation, condition-trigger model, all-trigger, dynamic-trigger.**

I. Introduction

IN cross-chain scenarios, a system may consist of different blockchains due to various purposes. To cooperate with different blockchains, cross-chain interoperability is introduced [1] [2], which ensures that the operations or state changes in one blockchain cause the operations or state changes in other blockchain(s). The operation and the state change can be abstracted to an action. Then the cross-chain interoperability is that the source action(s) trigger(s) the target action(s). There are different relationships between the source actions and the target actions. One (source) action triggers one (target) action (one-to-one); one action triggers several actions (one-to-many); several actions trigger one action (many-to-one); or several actions trigger several other actions (many-to-many).

Take Figure 1 for example, which is a system to process the data of foods (fruits). This system consists of four blockchains, the sensor blockchain, the delivery blockchain, the monitoring blockchain, and the customer blockchain. Different blockchains are used mainly due to two reasons (1) different blockchain have different functions, a monitoring blockchain is often permissioned blockchain composed by nodes from government, and it is not suitable to be nodes from the sensor blockchain; multiple blockchains increase the process capabilities, access control. (2) the process capabilities, as there may be many blockchains in a city, if one blockchain is used for all, the process capabilities is limited. Now, we analyze the trigger relationship. The sensor blockchain processes the planting data that it gets from crop sensors. Then the fruits go to the delivery system, and the corresponding information goes to both the delivery system and the food monitoring system. This is a one-to-many relationship. Information of those two blockchains goes to the customer blockchain, which provides information for customers to judge and select fruits. This is a many-to-one format.

In the above format, the triggering relationship is predefined. There are also the cases that the source action(s) only trigger some candidate actions in other blockchains dynamically (this is called the dynamic trigger and the predefined is called the all-trigger; they are further described in Section III-B1 and Section III-B2). [3] [4] [5] propose to use several blockchains (or sidechains) to increase the processing capability. Then, an issue is how to balance the load to those processing blockchains, the load balance issue of blockchains. There are two requirements to choose a target blockchain. (a) The target blockchain should have the capacity for the further process (called the available blockchains). The load is decided by its dynamic processing state that cannot be predicted. (b) The available blockchains should have an equal chance to run the target action. Thus, it is dynamical to choose a blockchain to process, and we abstract this kind of trigger as a dynamic-trigger.

A real system may contain a combination of the above triggers. Predefined triggers are combined with dynamic triggers, and even triggers are cascaded. In Figure 1, action a1 triggers action a2 and a3 as predefined; and one of actions a41 and a42 is dynamically chosen to analyze the data from sensors. As the trigger relationship is complex, it is important to analyze the relationship among associated smart contracts. It brings benefits. (1) It helps us to know whether associated blockchains are cooperative or not. For example, it helps us to know whether a1 triggers a5 or not. (2) It provides the ability to analyze the system in which some actions are added dynamically. In this case, actions are added dynamically to the running systems, and it helps to know how the new action cooperates with others. (3) It helps to know where the bottleneck of a system is. With the quantitative analysis of the system, we can find the bottleneck.

In this paper, we focus on cooperation among different blockchains, including different trigger models and the quantitative analysis of the cooperation among different blockchains. The main contributions are as follows.

(1) We propose the method to analyze the probabilities of cooperation among blockchains, which is done by graph weight calculation with dimensions. As the trigger relationship

H. Su is with the College of Computer Science, Sichuan University, Chengdu, China, 610041.
E-mail: suguest@126.com

may form a complex network, if we use the traditional method to calculate the cooperation probabilities among each paired action, it is complex and even difficult for participants to analyze the whole system. Thus, we adopt the graph theory to analyze cooperation. An action can be seen as a graph node and its interaction with other actions as edges. Those nodes and edges form a graph. However, the network flows in the graph theory cannot reflect the probabilities of blockchain cooperation (referring to III-B2 for more details). Thus, we introduce the weight dimension and the derived dimension to calculate the cooperation probabilities.

(2) We propose the condition-trigger model as the basic interaction model among cross-chain actions. It helps us to analyze the relationship among cross-chain actions without the details of each action (such as which blockchain an action belongs to). If an action triggers another action, the former can be seen as the condition and the latter can be seen as the action.

(3) We propose to abstract the cross-chain cooperation as the all-trigger and the dynamic-trigger models. If the action to be triggered in another blockchain is predefined, it is the all-trigger. If the actions to be triggered are dynamically selected, it is the dynamic-trigger. Those triggers can cascade to form a complex cooperative blockchain system.

The rest of the paper is organized as follows. Section II describes the technical background and related works. Section III introduces the condition-trigger model, and how to calculate their cooperation probabilities among those basic models. Section IV describes how to evaluate the trigger probabilities among actions. Section V describes the evaluation results, and Section VI draws the conclusion.

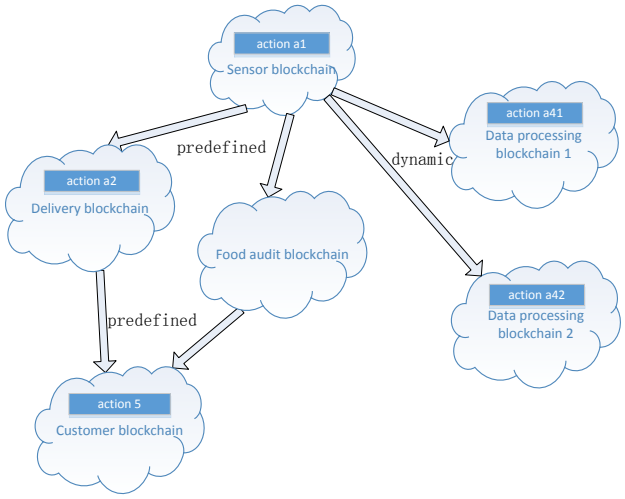

Fig. 1. Cooperation among different blockchains. This is a system composed of several blockchains, aiming to process the data of foods from the planting to the end-user. The sensor blockchain processes the planting data from crops directly. Then the fruit goes to the delivery system, the corresponding information goes to both the delivery system and the food monitoring system. At last, users access the customer blockchain to know the delivery information and the audit information. Meanwhile, there are two data processing blockchain to increase the data analysis capacity.

II. Theoretical Background and Related Works

A. Theoretical Background

1) Cross-chain interoperability and cross-chain cooperation: Cross-chain interoperability is to interact among blockchains [6] [7]. It has two aspects: (1) to ensure that a specific state change in one blockchain causes a specific state change in another blockchain, such as the atomic asset swap [8]; and (2) to ensure that a specific operation in one blockchain causes a specific operation in another blockchain [9]. Previous works focus more on the state synchronization among blockchains. However, the relationship among actions of blockchains is also impotant, and we introduce cross-chain cooperation that focuses on the analysis of actions. Cross-chain cooperation means whether one action in a blockchain will probably trigger another action in a different blockchain or not.

2) Smart contract and action in the cross-chain interoperability: Smart contracts are programs on the blockchain [10]. They automate the operations on the blockchain, which can be used to interact with other blockchains [11] or non-blockchain systems [12]. A smart contract contains several sub-steps that cause operation or state change on the other blockchain. To give a more precise description of the interaction, we introduce a fine-grained component to describe it, the actions of a smart contract. An action is a function or submodule of a smart contract to run when it receives a request from another smart contract or an external application (the first trigger).

B. Related Work

Cross-chain interoperability ensures the correct execution of actions on different blockchains. Hash-locking [13] [14] aims to ensure the atomic exchange among blockchains, which requires a specific sequence of steps. At the initial step corresponding assets are locked on different blockchains; at the second step the first user tries to get its exchanged assets by exposing its own secret; later the second user can retrieve its exchanged asset by that secret. To exchange data among blockchains [15] [16], it requires to ensure the correct execution sequence of actions of different smart contracts. To ensure the correctness of cross-chain payment protocols, a new specification formalism formalizes its protocols and ensures those protocols work correctly even in the presence of a clock skew [17]. Those works focus more on the correct synchronization of actions among different blockchains to achieve a specific goal. They lack of a common model to analyze the relationship of actions used in the cross-action interoperability.

In the early stage of cross-chain interoperability, the relationship among cross-chain actions is certain, in which actions have no competition. It can be seen in the bid process of an e-commerce exchange [19] [20] or in the energy trading process [21] [22]. The participants and their actions are certain, and it can be abstracted as the all-trigger model. Later, to solve the scalability issue, the multiple-blockchain solution has been used. To meet industrial standards, work [3] proposes a novel blockchain architecture, which allows to parallelly run multiple blockchains that have different consensus algorithms.

[4] proposes sidechains to increase the transaction processing capability by offloading the processing from the mainchain to its sidechains. [5] proposes to use the sharding method to increase the transaction throughput by loading tasks to one of several candidate blockchains. They are the detailed format of the dynamic-trigger; while there is no corresponding quantitative analysis model.

There are also works for information propagation among blockchains. Different blockchains exchange information by a blockchain router or a special network. A blockchain router behaves similarly to a router of the Internet [1] [2], in which a central 'hub' blockchain connects other blockchains. Blockchains communicate with the help of the 'hub' blockchain. Blockchains can also exchange information by a special network. Cross-chain exchange methods [2] [23] are by special roles of validator, connector, surveillant, and nominator, with the aim to keep the transactions atomic or ensure data validity. Those methods are the basis for cross-chain communication and can help to transfer cross-chain information.

III. Cooperation Model

In this section, we describe the condition-action model and the different types of triggers. The triggers can be direct or indirect.

(1) Direct-trigger smart contracts are smart contracts that interact without the help of other smart contracts. The co-operation among directly-triggered smart contracts refers to whether one smart contract triggers another one or not. In the dynamic-trigger, a smart contract only has the probabilities to trigger other smart contracts, and therefore we need to evaluate the trigger probabilities.

(2) Indirect-trigger smart contracts are smart contracts that are not invoked directly, and the interaction among them is through other smart contracts. For this kind of triggers, we need to know whether an action in a smart contract leads to the expected action in an indirectly-triggered smart contract or not. Direct-triggers can be cascaded to form the indirect-trigger.

A. Condition-trigger Model

As the action of a smart contract is done under some conditions (request from other smart contracts), we propose the model of condition and trigger, the condition-trigger model. In this model, actions in a blockchain are triggered by their conditions. If another blockchain wants to trigger an action, it should do something to match conditions of the target action. If the actions always happen, the condition can be set as true.

In Figure 2, action b has the condition cb. Action a tries to trigger action b with condition cb. After the condition is passed to blockchain n, action b is triggered. As there may be complicated condition-trigger relationships among blockchains, it is important to analyze their relationships.

Now we describe the condition and trigger formally. A condition (suppose c) is generated by its action (suppose a), and the condition (c) triggers the another action (suppose a'). This procedure is shown in (1).

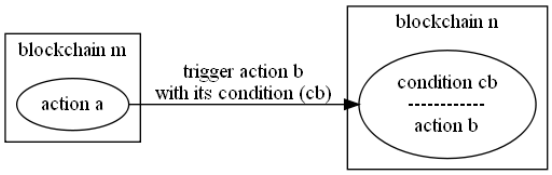

Fig. 2. Condition-trigger model

$$\begin{aligned} a &\xrightarrow{generate} c \\ c &\xrightarrow{trigger} a' \end{aligned} \tag{1}$$

where c means a condition, and a, a' mean two actions.

In (1), there are keywords of "generate" and "trigger". For simplification, we omit those two keywords. An arrow with the default behavior to indicate that a condition triggers an action or an action generates a condition.

To describe relationship among all conditions and actions, we introduce the set of conditions and actions. The notation of C is as the set of conditions in (2), and A is as the set of actions in (3).

$$\mathbf{C} = \{c_1, \ldots, c_i, \ldots, c_n\} \tag{2}$$

where c_i is a condition in a cross-chain cooperation.

$$\mathbf{A} = \{a_1, \ldots, a_i, \ldots, a_n\} \tag{3}$$

where a_i is an action in a cross-chain cooperation.

Then in a cross-chain interaction, the cooperation is among C and A.

$$A \leftrightarrow C \tag{4}$$

Now, we define the condition set C_a_i of an action a_i as (5), and the action set A_c_i of condition c_i as (6).

$$C\_a_i = \{c_i | c_i \in C, \text{ and } c_i \text{ is condition generated by } a_i\} \tag{5}$$

$$A\_c_i = \{a_i | a_i \in A, \text{ and } a_i \text{ is action triggered by } c_i\} \tag{6}$$

If we put the above two expressions together, we get (7), which specifies that two actions can be associated with a condition. In (7), a' is the source action which triggers other action, and a is the target action which is triggered by other actions.

$$a' \rightarrow c \rightarrow a \tag{7}$$

As the condition is triggered by an action, we can omit the condition and directly describe the relation among different actions. Still, we define the (trigger) target set T_a_i of an action a_i as (8), and the (trigger) source set S_a_i of action a_i as (9).

$$T\_a_i = \{a_j | a_j \text{ is triggered by condition generated by } a_i\} \tag{8}$$

$$S\_a_i = \{a_j | a_j \text{ generates the condition that triggers } a_i\} \tag{9}$$

To make our analysis not too complex, there are two preconditions (or limitations) of the model. (1) First one is that each condition c_j of C_a_i can trigger the target action a_i. There is no requirement for the combination of two or more conditions to trigger a target action. This means we only discuss conditions that have a relationship of or. If an action is triggered by the combination of several conditions, they have a relationship of and; the analysis is similar to the or relationship, and we do not discuss it to save space. (2)The second one is that there is no loop among two actions. If one action (a_i) triggers another action (a_j), a_j will not trigger back to a_i. If there is a finite loop, we can replace the loop format to several repeated condition-trigger pairs.

The relationship among trigger pairs can be depicted graphically. Its nodes are actions, and its edges are direct triggers from a source action to its target action(s). All conditions and actions form a directed acyclic graph, called the trigger graph. In the trigger graph, one action may trigger several actions, or one action is triggered by several conditions (each condition can trigger this action). A trigger graph is shown in Figure 3.

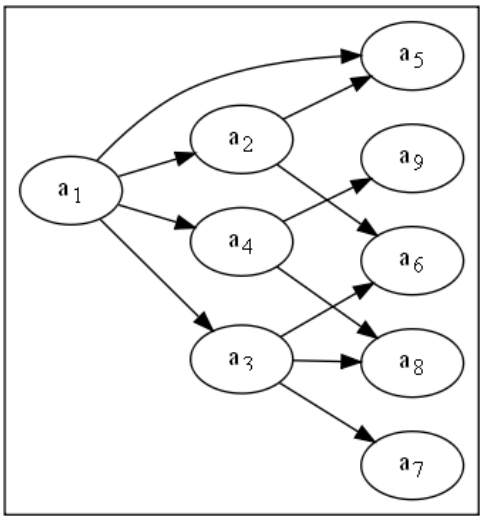

Fig. 3. Interactions among actions form a directed acyclic graph, called the trigger graph.

Cooperation relationships are more visible in a trigger graph. Such as whether two actions are directly-triggered smart contracts (a direct trigger pair) or indirectly-triggered smart contracts (an indirect trigger pair). In Figure 3, a_1 and a_7 are an indirect trigger pair, as a_3 is between them. a_3 and a_7 are a direct trigger pair. If there exist both direct and indirect triggers among two actions, we call them the mixed trigger pair, as a_1 and a_5.

The above types of actions are cooperative as one action can trigger another. Actions are cooperative when there is a directed path from one action to another, and this path is called a trigger path. With the trigger path, two cooperative actions can be defined as in formula (10), in which a_i triggers a_j.

$$\exists tp, a_i \in tp,\ a_j \in tp \tag{10}$$

where tp is a trigger path.

Similarly, we define non-cooperative actions. If there is no trigger path among action a_i and action a_j, those two actions are non-cooperative.

The above describes the cooperation qualitatively, which specifies that one action will trigger another one. But in some cases, one action only has the probability to trigger another one, and it cannot be ensured one action must trigger another action. We need a quantitative assessment of how an action is possible to trigger another action, which is called the quantitative cooperation analysis.

To analyze the cross-chain actions quantitatively, we further adapt the trigger graph. The edge weight stands for the probabilities that how one action triggers another. We first analyze the basic models among direct trigger pairs, and then extend to analyze indirect trigger pairs.

The joint probability method [24] is difficult to analyze the cooperation among actions. The reason is that there may be several paths from the source action (a_s) to the target action (a_t); in each path, a_s can trigger a_t, which may result that the sum of those trigger probabilities is bigger than 1 (referring to part III-B2 for more details). It is not a single joint probability. It is the combination of several joint probabilities. As those actions may form a complex interaction network, it is difficult to iterate and analyze all combinations. Then, in this paper, we utilize the trigger graph for the analysis.

B. Trigger Types

In this section, we describe the basic trigger model between direct actions, the dynamic-trigger and the all-trigger.

1) dynamic-Trigger: In some cases, several actions are candidates to be triggered by a condition, while the number of actually triggered actions is limited. This can be used to do load balancing among several blockchains as described in Section I. In other words, not all direct trigger actions are allowed to be triggered.

This kind of trigger is called the dynamic-trigger model, in which an action sends a trigger condition to other actions, while only one of those actions is allowed to run. When there are n candidate actions, each action has $1/n$ probability to run, as in (11). In this paper, we only limit one action that can be triggered for simplification. If several actions (suppose k actions) can be triggered, we should assign each action with the probability k/n.

$$\forall a_j \in T\_a_i, w_e = 1/n \tag{11}$$

where e is the edge from a_i to a_j, and w_e is its weight.

Figure 4 is a dynamic-trigger, in which action s triggers several actions from t_1 to t_n. For each edge, we give a weight $1/n$ to indicate that this action triggers one of its target actions.

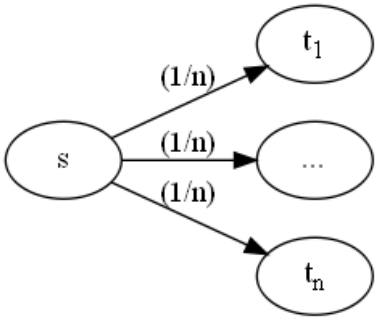

Fig. 4. dynamic-trigger type. Source action s triggers only one of its target actions. Numbers on the edge are the edge weights.

2) All-trigger: In this model, an action triggers all its target actions. All target actions run when the trigger condition matches. Figure 5 shows an example, action s trigger actions from t_1 to t_n.

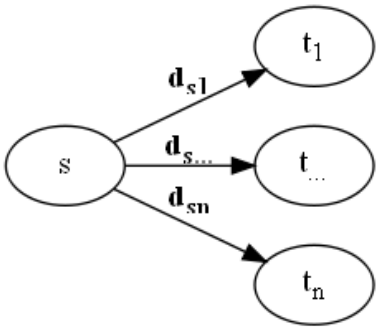

Fig. 5. All-trigger type. Source action s triggers all its target actions. Notations on the edge are weights with dimensions ($\mathbf{d_{s1}}$, $\mathbf{d_{sn}}$, etc.).

We still describe this case by the edge weight. Suppose state s is the source action. Intuitively, weight 1 should be assigned to each edge. However, each of them can be triggered separately, we give each edge weight 1 with a different dimension. This kind of dimension is called the (edge) weight dimension. The name format of a weight dimensions is in (12). For example, $\mathbf{d_{s2}}$ indicates that it is the dimension of the second trigger of action s, and we just use $\mathbf{d_s}$ if the sequence is not necessary to mention.

$$\mathbf{d}_{<\textbf{action name}><\textbf{sequence}>} \text{ or } \mathbf{d}_{<\textbf{action name}>} \tag{12}$$

The reason why different weight dimensions are required is that weights from an all-trigger type cannot be directly added. Figure 6 is one example. Action s (all-trigger) triggers m_1, m_i, and so on. Action m_1 (dynamic-trigger) trigger l_i with 1/2 probability as it only selects one of two actions (l_i and l_j) to run. Similarly, action l_k has 1/2 probability to run. If we directly add them, action l_o is triggered with probability 1(1/2+1/2), which is incorrect as there are possibilities to run action l_j or l_l. If more branches are involved, the sum of the trigger probability can even be bigger than 1.

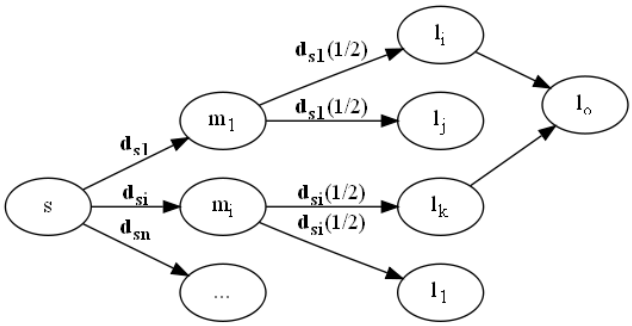

Fig. 6. The trigger action of different weight dimensions. Action s (all-trigger) trigger actions from m_1 to m_i. Action m_1 (dynamic-trigger) triggers l_i with ½ probability as it only selects one of two actions (l_i and l_j) to run. Action l_k also has ½ probability to run. If we directly add them, l_o is triggered with 1(1/2+1/2), which is incorrect as there are probabilities to run action l_j or l_l

Notice, in all-trigger, successive actions are triggered, and the sum of all edges' weights is bigger than 1. This makes it difficult to use traditional methods (such as Markov process [25] or the joint probability method to analyze [24]) as they require that the sum of its outgoing edge (its triggered actions) weight is 1.

C. Trigger Combination

dynamic-trigger and all-trigger types can be combined. Here we give four possible combinations, all-all, all-dynamic, dynamic-all, and dynamic-dynamic combinations.

1) All-all Combination: In this form, there are two cascaded all-trigger types as shown in Figure 7. Two related weight dimensions are generated, and we introduce binary operation '#' that means a new weight dimension ($\mathbf{d_{nd}}$) is generated from the old weight dimension ($\mathbf{d_{od}}$), which forms ($\mathbf{d_{od}}\#\mathbf{d_{nd}}$). $\mathbf{d_{nd}}$ is the child weight dimension of $\mathbf{d_{od}}$, and $\mathbf{d_{od}}$ is the parent weight dimension of $\mathbf{d_{nd}}$. In Figure 7, action r and v are target actions of the second all-trigger of action s. Weight dimension $\mathbf{d_{s_1}}$ is generated at the first trigger. Weight dimensions $\mathbf{d_{m_1}}$ and $\mathbf{d_{m_2}}$ are generated at the second trigger m, which results weights of $\mathbf{d_{s_1}}\#\mathbf{d_{m_1}}$ and $\mathbf{d_{s_1}}\#\mathbf{d_{m_2}}$. Assuming the weight of the first action s is d (following cases are also based on this assumption), the weight of action r (a target action of m) is shown in (13). Notice the weight of an action is the weight of its incoming edge in the trigger graph, which stands for the probabilities of that action to be triggered.

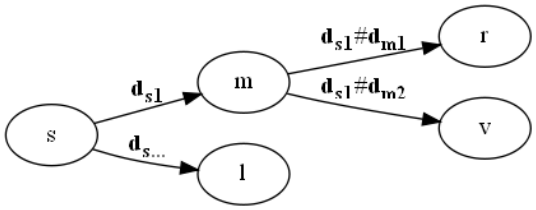

Fig. 7. The combination of all-trigger type. A new dimension is generated after an all-trigger. For example, $\mathbf{d_{s_1}}$ is generated at the first trigger of action s, and $\mathbf{d_{m_1}}$ and $\mathbf{d_{m_2}}$ are generated at the second trigger. Then we get $\mathbf{d_{s_1}}\#\mathbf{d_{m_1}}$ and $\mathbf{d_{s_1}}\#\mathbf{d_{m_2}}$.

$$d \xrightarrow{\text{all-trigger}} d\#\mathbf{d_{s_1}} \xrightarrow{\text{all-trigger}} d\#\mathbf{d_{s_1}}\#\mathbf{d_{m_1}} \tag{13}$$

where d is the weight of action s, and the arrow means that the former action triggers the latter action.

2) All-dynamic Combination: In this type, an all-trigger is followed by a dynamic-trigger. Different weight dimensions are formed in the all-trigger, and weights are divided in the dynamic-trigger. An example is shown in Figure 8.

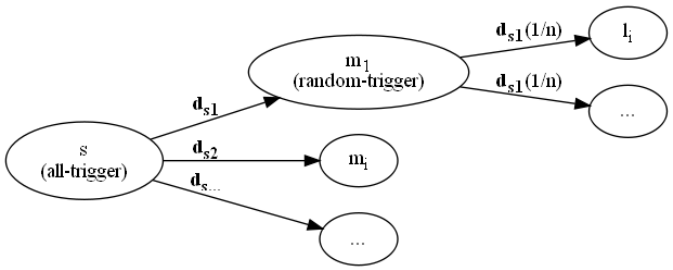

Fig. 8. The combination of all-trigger and dynamic-trigger type. Source action s first generates new weight dimensions $\mathbf{d_{s1}}$, $\mathbf{d_{s2}}$ and $\mathbf{d_{s...}}$ in an all-trigger. And one of its target actions m_1 gives a dynamic-trigger, and successive weights are divided.

In Figure 8, it shows that the source action is s and its target actions of the all-trigger are m_1, m_i, and so on. m_1 has a dynamic-trigger. As the number of m_1's target actions in the dynamic-trigger is n, the weight of each action is $1/n$ of $\mathbf{d_{s_1}}$, and we notate it as $\mathbf{d_{s_1}}(1/n)$. Here we put $1/n$ after the dimension symbol instead of as its coefficient. The reason is that this format keep the division information of this dynamic trigger, which will be used in later calculation. The weight of action l_i (a target action of m_1) is shown in (14).

$$d \xrightarrow{\text{all-trigger}} d\#\mathbf{d_{s_1}} \xrightarrow{\text{dynamic-trigger}} d\#\mathbf{d_{s_1}}(1/n) \tag{14}$$

where d is the weight of action s.

3) dynamic-all Combination: In this type, the first trigger is a dynamic-trigger and the second trigger is an all-trigger. The dynamic trigger divides the trigger probabilities of the former. The weight change process is shown in (15).

$$d \xrightarrow{\text{dynamic-trigger}} d(1/n) \xrightarrow{\text{all-trigger}} d(1/n)\#\mathbf{d_{s_1}} \tag{15}$$

where d is the weight of action s.

In expression (15), former weight is divided into 1/n in the dynamic-trigger, and a new dimension is generated in the following all-trigger. The weight changes from d to $d(1/n)\#\mathbf{d_{s_1}}$.

4) dynamic-dynamic Combination: In this case, there are two successive dynamic-triggers, each of which divides the original value. Suppose the first division is among n paired actions and the second is among m paired action, the weight change process is expression as in (16).

$$d \xrightarrow{\text{dynamic-trigger}} d(1/n) \xrightarrow{\text{dynamic-trigger}} d(1/mn) \tag{16}$$

where d is the weight of action s.

IV. Trigger Probabilities

In section III, we describe the cooperation probabilities among basic trigger pairs and their combinations, in which there is only one path from the source action to the target action. While in a complex network, there may be more than one path from a source action to its target action, which results in different combination of the weight dimensions. In this section, we describe how to calculate cooperation probabilities among two actions when multiple trigger paths exist.

A. Probabilities Calculation

1) Relationship among Different Combination of Weight Dimension : After traversing the trigger graph, we may get several weights from different paths. We put them into a parenthesis and call it the weight array of the result. An example is as follows.

$$(D1(x), D2(y), ...)$$

where $D1$, $D2$ are dimensions, and x, y are probabilities.

The trigger probability can be calculated from the weight array of the result. However, their weight dimensions may be different. Before the calculation, we need to discuss the relationship among different weight dimensions. We start with the definition of the dimension tuple, as in definition 2.

Definition 1: Let w be a weight of an action. Its *dimension tuple* (D) is a tuple composed by all dimensions of w. Its *sub dimension tuple* (S) is a subset of D, where S is either one or several adjacent dimensions of D. Dimensions $D1$ and $D2$ are adjacent dimensions of w, which means that there is no other dimension(s) between $D1$ and $D2$ in w.

For example, a weight is $\mathbf{d_a}(1/2)\#\mathbf{d_b}(1/4)\#\mathbf{d_c}$. Its dimension tuple is$\{\mathbf{d_a}, \mathbf{d_b}, \mathbf{d_c}\}$, and its sub dimension tuples are $\emptyset$, $\{\mathbf{d_a}\}$, $\{\mathbf{d_b}\}$, $\{\mathbf{d_c}\}$, $\{\mathbf{d_a}, \mathbf{d_b}\}$, $\{\mathbf{d_a}, \mathbf{d_c}\}$, $\{\mathbf{d_b}, \mathbf{d_c}\}$, and $\{\mathbf{d_a}, \mathbf{d_b}, \mathbf{d_c}\}$. Or we can also notate those sub dimension tuple as $\emptyset$, $\mathbf{d_a}$, $\mathbf{d_b}$, $\mathbf{d_c}$, $\mathbf{d_a}\#\mathbf{d_b}$, $\mathbf{d_a}\#\mathbf{d_c}$, $\mathbf{d_b}\#\mathbf{d_c}$, and $\mathbf{d_a}\#\mathbf{d_b}\#\mathbf{d_c}$. The latter format is more convenient in this paper.

Based on the sub dimension tuple, we introduce the concept of the common starting sub dimension tuple or the common sub dimension tuple.

Definition 2: Let $D1$, $D2$, Di and so on are several dimension tuples. Their *common sub dimension tuple* is a sub dimension tuple D, which matches the conditions as follows.

$$D1 = D\#D'$$
$$D2 = D\#D''$$
$$Di = D\#D'''$$
$$...$$

where D, D', D'', and D''' are different sub dimension tuples.

For example, given weights of $\mathbf{d_a}\#\mathbf{d_b}\#\mathbf{d_d}$ and $\mathbf{d_a}\#\mathbf{d_b}\#\mathbf{d_f}$, we can choose a common sub dimension tuple D as $\mathbf{d_a}\#\mathbf{d_b}$. Those two expressions can be expressed as $D\#\mathbf{d_d}$ and $D\#\mathbf{d_f}$.

Meanwhile, for the convenience of analysis, we introduce the weight dimension of weight w at position i (the position

is 1-based), notated as d(w, i). For example, suppose weight w is $\mathbf{d_a}(1/2)\#\mathbf{d_b}(1/4)\#\mathbf{d_c}$; then d($w$, 1) is $\mathbf{d_a}$, and d(w, 3) is $\mathbf{d_c}$.

2) Calculation Among the Same Dimension: In this case, all weights are of the same dimension tuple. Suppose the dimension tuple is D, and weights are $D(x)$, $D(y)$, and $D(z)$. They are results of one or several dynamic triggers after D. Otherwise, if they encounter any all-trigger after D, there is at least a dimension difference. The calculation is to directly add them, as shown in (17).

$$(D(x), D(y), D(z)) \rightarrow D\,(x + y + z) \tag{17}$$

This kind of format can be extended. Suppose there are two dimension D', and D'' as in (18).

$$\begin{aligned} w' &= D1(x_1)\#D2(x_2)\#Di(x_i)\#...\#Dn(x_n) \\ w'' &= D1(y_1)\#D2(y_2)\#Di(y_i)\#...\#Dn(y_n) \end{aligned} \tag{18}$$

As they have a common sub dimension tuple as in (19), then they be calculated as in (20).

$$S = D1\#D2\#Di\#...\#Dn \tag{19}$$

$$\begin{aligned} w &= w' + w'' \\ &= D1(x_1 + y_1)\#D2(x_2 + y_2)\#Di(x_i + y_i)\#... \end{aligned} \tag{20}$$

We briefly explain this process with Figure 9. The original trigger relationship is in the upper part. In the first step, we merge the trigger probabilities (x_1 and y_1) of D1, as they must be from different branches of the same dynamic trigger(s). The lower part is step 2, in which x_2 and y_2 are calculated similarly. Then all independent possibilities are merged in this way.

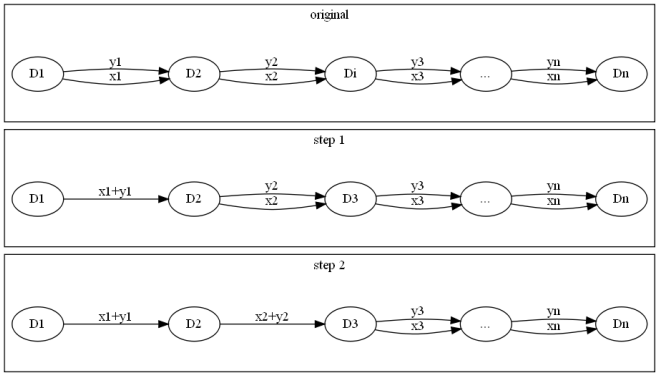

Fig. 9. Calculation among the same dimension.

With this calculation, the same weight arrays has been calculated.

3) Calculation Among Dimension and its Derived Dimension: In this case, the weight calculation is among a dimension and its derived dimension. Suppose two inputs are $D(x1)$ and its derived dimension $D(x2)\#\mathbf{d_k}(y)$. The corresponding dimension tuple are D and its derivation $D\#\mathbf{d_k}$.

In this case, (1) those two weights are from two branches of a dynamic-trigger, and (2) one of its branches passes an all-trigger, which results in $D(x2)\#\mathbf{d_k}$. The second is obvious. We try to explain the first point that those two inputs are from different branches of a dynamic-trigger. As those two inputs are separate, they must come from two branches of either a dynamic-trigger or an all-trigger. If they come from an all-trigger, there is a dimension difference, such as $D\#\mathbf{d_{k1}}$ and $D\#\mathbf{d_{k2}}$ instead of D and $D\#\mathbf{d_k}$.

As those two actions come from two dynamic branches of an action, the target can be triggered by any of them. The probability of the target action to be triggered is the sum of the probabilities of the two inputs, shown in (21).

$$(D(x1), D(x2)\#\mathbf{d_k}(y)\ldots) \rightarrow D\,(x1 + x2 * y) \tag{21}$$

4) Calculation Among Different Dimensions Of One Action: Input weights are branches of an all-trigger, such as $D\#\mathbf{d_{k1}}(x)$, $D\#\mathbf{d_{k2}}(y)$, $D\#\mathbf{d_{k3}}(z)$. Characteristics of this case, (1) inputs are branches of an all-trigger to generate different dimensions ($\mathbf{d_{k1}}$, $\mathbf{d_{k2}}$, $\mathbf{d_{k3}}$); (2) inputs may later encounter a dynamic-trigger to change the coefficient (x , y and z may not be equal). Thus, the probability is the complement probability that no action is triggered.

$$\begin{aligned} &(D\#\mathbf{d_{k1}}(x), D\#\mathbf{d_{k2}}(y), D\#\mathbf{d_{k3}}(z)) \\ &\rightarrow D\,(1 - (1 - x) * (1 - y)(1 - z)) \end{aligned} \tag{22}$$

5) Calculation Among Different Dimension: In this case, inputs are from different paths. Characteristics of this case: for any inputs $i1$ and $i2$, d($i1$,1) and d($i2$,1) are (1) not from an all-trigger but (2) from a dynamic-trigger. An example is shown in Figure 10. Different branches of a dynamic-trigger can be triggered independently. Then the trigger probabilities are the sum of all probabilities.

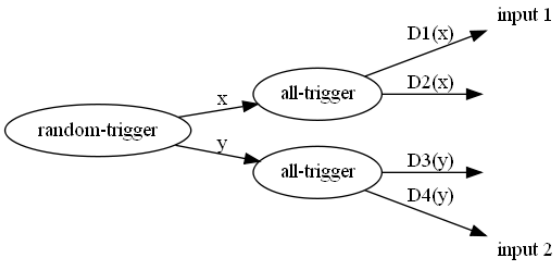

Fig. 10. Path with different dimension. Input 1 and input 2 are from two different paths, which means they are not from an all-trigger.

$$(D1(x), D2(y), D3(z)) \rightarrow (x + y + z) \tag{23}$$

B. Calculations Precedence of Probabilities

Trigger probabilities can be calculated by the above methods. However, different calculation orders may result differently. For example in (24), the calculation $\mathbf{d_j}(x)\#\mathbf{d_{k1}}(y1)$ and $\mathbf{d_j}(x)\#\mathbf{d_{k2}}(y2)$ are between different dimensions of one action; the calculation of $\mathbf{d_j}(x)\#\mathbf{d_{k1}}(y1)$ and $\mathbf{d_j}(x)$ are between dimension and its derived dimension. These two different calculations have different results.

$$(\mathbf{d_i}(s), \mathbf{d_j}(x), \mathbf{d_j}(x)\#\mathbf{d_{k1}}(y1), \mathbf{d_j}(x)\#\mathbf{d_{k2}}(y2), \mathbf{d_n}(t)\#\mathbf{d_o}(r)) \tag{24}$$

Thus, we need a calculation precedence. The calculation precedence concerns the time order of actions and whether one calculation will impact others or not.

(1) The same dimension is calculated at first. It reduces the number of weights to be calculated, and it does change the calculation order of other dimensions.

(2) The second is either the calculation of different dimensions among one action (2a) or the calculation of dimension and its derived dimensions (2b).

As 2a and2b may contain each other mutually, then we calculate this in the time sequence (to choose the pairs first which happen at the latest time). There is a simple way to get the time order by comparing which combination is the longest.

(3) Different dimensions are calculated at last. The reason is that those paths do not affect each other.

C. Probabilities and Cooperations

After the calculation, we get the final probability that an action is triggered by other actions, and we call it the final trigger probability or the trigger probability. In this subsection, we use the trigger probability to analyze the several aspects of the cooperation.

Cooperative Judgment. Suppose the trigger probabilities among actions a_a and a_b is p_t. If p_t is bigger than 0, then a_a and a_b are cooperative. And the trigger probabilities reflect how frequently an action triggers another action. The bigger p_t is, more frequently a_b will be triggered after action a_a. If p_t is 1, a_b will be triggered by action a_a.

Bottleneck analysis. In some case, there are bottlenecks which block the further processing of an action (suppose action as). We can use the trigger probabilities to find the corresponding blockchains. A bottleneck blockchain has the feature that its actions have **high** trigger probabilities to be triggered by action as. Suppose there are two blockchains BCa and BCb, and they have similar processing capabilities. If the trigger probabilities of BCa is 0.95 and BCb is 0.1, we should choose to add new blockchains to reduce the load of BCa. Meanwhile, when new blockchains of BCa has been increased, the successive blockchains may be the bottleneck blockchains, and the trigger probabilities can be used in further analysis of the successive blockchains. Thus, we can use the trigger probabilities to find where the bottleneck is and increase its corresponding blockchains.

V. Evaluation

In this section, we try to evaluate the results by experiments. We first introduce the test environment and then evaluate the all-trigger and dynamic-trigger separately.

A. Environment

This subsection includes two major parts, the blockchains and the communication protocol used for the cross-chain trigger. The communication protocol should be (1) secure and (2) confirmed in the blockchain to keep a consist state. Then we put the condition of a trigger as data in a transaction to be confirmed in its blockchain. Different blockchains synchronize those data among them.

1) Blockchains: There are either 5 or 7 blockchains used in the evaluation depending on different test cases. The consensus algorithms of verified blockchains are PoW. Those blockchains are publically available blockchains, which means (1) the nodes of one blockchain can connect to the nodes of the other blockchain by the P2P method, and (2) the nodes of one blockchain can get chain(s) of blocks from other blockchains and fetches corresponding transactions for the condition-trigger model.

2) Connection Topology: The topology among blockchains is a ring topology. There are two subtypes of topology, one with 5 blockchains, and another one with 7 blockchains, shown in Figure 11.

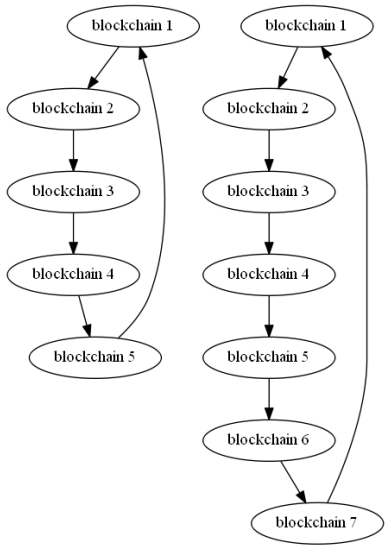

Fig. 11. Topology for the verification. The left one is the ring topology with 5 blockchains and the right one with 7 blockchains.

3) Data Format: In the proposed condition-trigger model, conditions are generated by corresponding actions, which further triggers other actions. The condition is identified by a special starting string (”Cross-chain:trigger:condition”), and it is put to the data structure of a transaction. This transaction has a special type to mark it as cross-chain transactions. Blockchains synchronize cross-chain transactions instead of all transactions, aiming to reduce the amount of the data to be synchronized.

4) Data Propagation: There are different ways to propagate the message among different blockchains. It includes the non-blockchain intermediate method and the blockchain-based propagation method. The non-blockchain intermediate method is by a third-party (or the user) to take the data from the source blockchain to the target blockchain. This method relies on some methods to ensure the validation of the data, such as the way in the hash-locking method [13].

The blockchain-based propagation method can be based on a centered blockchain, such as the blockchain router method

[2], which helps to propagate transactions among blockchains. Different blockchains connect to the blockchain router, posting the data to it and getting data from the blockchain router.

The cross-chain data can also be propagated directly among blockchains instead of the introduction of a centered blockchain. The number of connection increases with the square of the number of blockchains if each one connects to all other blockchains. We adopt a ring topology of blockchain connection in which each blockchain only connects to another blockchain. Directly connected blockchains transform and seal the corresponding transaction of its neighbored blockchain. This process iterates on to process among indirectly connected blockchains.

In the method used in the evaluation, the blockchain seals related information into its own blockchain after the chain of blocks is got from another blockchain. And to encourage the miner to mine information from other blockchains, the reward mechanism has an adaption for this in the verification blockchain. That is an extra reward is given to the block miner. The reward for miners contains three parts. (1) A relatively big reward (1 internal coin in the verification blockchain. It is not fixed and different blockchains can have its own reward value) is given to the miner who mines a block, as it uses heavy CPU resources to mine a blockchain. (2) A relatively low reward (1/8 internal coin in the verification blockchain) are given if a new block contains transaction(s) for the condition-trigger model. It uses the network and corresponding CPU resources to process external transactions, and thus it should be rewarded. However, this process does not take as high resources as the process to mine a block. (3) A little reward (0.001 coins in the verification blockchain) is given for each condition-trigger transaction, as after the synchronization and corresponding process, rewarded in (2), there is not much work to do. The reward for mining extra data ($rewardExtra$) is shown in (25).

$$rewardExtra = 5/4coins + 0.001coins * n \tag{25}$$

where n means the number of transactions which contain data from other blockchains of condition-trigger model.

5) Limit the Number Actions to Be Triggered: There may be a time issue when actions try to run after they receive a dynamic-trigger. Assume a dynamic-trigger is among several actions a_1, a_2, to a_3. If one of them (suppose a_1) finds that there are no other actions to run, it tries to run; while at the same time, another one (suppose a_3) may also want to run and finds no other actions are running. Then a_1 and a_3 begin to run at the same time. This results that more than one action runs.

To avoid this, one kind of transaction (a special data begin with "Cross-chain:trigger:data", trigger token, TT) is required to indicate that one action is ready to run. If there is only one TT within a certain time, the corresponding action gets the token to run. If there is more than one TT is sent out, we use the function of selecttAction (described in the previous "dynamic-trigger" part) to select an action to run.

When there is only one token, its sending action should wait a certain time (the waiting time) to avoid time issue. Then we should analyze the time. It is based on the propagation method and topology proposed above.

The time taken in the propagation includes procedures to transfer a TT over the network and mine it. Then the propagation time t_{p_i} in blockchain i contains the time used in the network t_{n_i} and the confirmation time (the mining time) t_{c_i}. As the ring topology is used, the waiting time t_w should be the sum of all propagation time among them, as in (27). As those time may change, we also add an additional time t_a to ensure the propagation and confirmation are done.

$$t_{p\_i} = t_{n\_i} + t_{c\_i} \tag{26}$$

$$t_w = \sum_{i=1}^{n} t_{p\_i} + t_a \tag{27}$$

t_a is relatively long to ensure that corresponding information has been propagated to other blockchains. In fact, t_a can be optimized. If a smart contract does not respond to a dynamic-trigger, it can send a token denying token (TDT) to notify others. Other smart contracts run without having to wait an extra time t_a. Then (27) is reformed to (28), in which t_w is the max time of t_{p_i}.

$$t_w = \max(\textstyle\sum_{i=1}^{n} t_{p\_i}) \tag{28}$$

where i stands for blockchain i.

B. All-trigger

In this subsection, we verify the all-trigger model. The topology is the 5-blockchain topology. Three kinds of verifications are carried out, A1, A2 and A3, whose logic connection is shown in Figure 12. Notice, a logic connection shows how actions are triggered, which is different from the topology diagram. For example, blockchain 3 is logically triggered by blockchain 1 although blockchain 3 does not directly connect to blockchain 1 (referring to Figure 11).

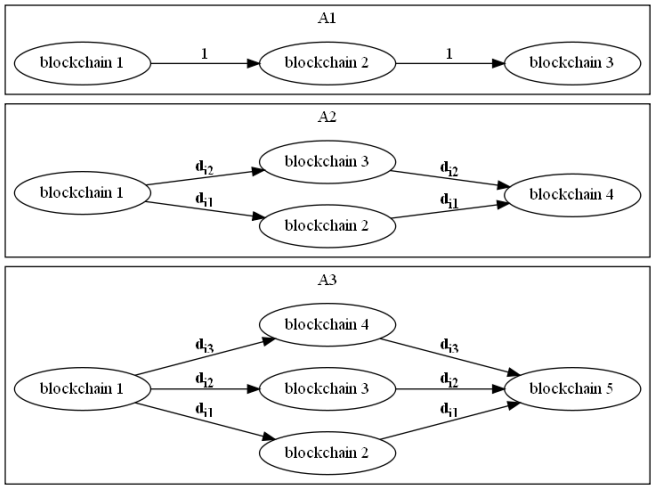

Fig. 12. The logic connection of case A1, A2, and A3

In the A1 case, a user triggers the initial action with a transaction in blockchain 1, which can be seen as probability

1 (the user is not shown in Figure 12). Then an associated smart contract on blockchain 2 gets the condition and does the corresponding action in blockchain 2. Later, blockchain 3 gets the corresponding state from blockchain 2, and the associated action is triggered.

A2 case is similar to A1. The difference is that the condition of an all-trigger from blockchain 1 has been sent to two blockchains, blockchain 2 and 3. The target blockchain 4 gets two all-triggers from blockchain 2 and 3. When the first one comes, its action is triggered. A3 case has three blockchains in the middle to receive and send out all-trigger from blockchain 1.

The weight calculation is shown in follows, which indicates that the target actions are triggered with the trigger probability of one.

Formula (29) is for A1.

$$1 * 1 = 1 \tag{29}$$

Formula (30) is for A2, in which there are two inputs ($\mathbf{d_{i_1}}$, $\mathbf{d_{i_2}}$) for the target smart contract on blockchain 4.

$$1 - (1 - 1) * (1 - 1) = 1 \tag{30}$$

Formula (31) is for A3, in which there are three inputs ($\mathbf{d_{i_1}}$, $\mathbf{d_{i_2}}$, $\mathbf{d_{i_3}}$) for the target smart contract on blockchain 5.

$$1 - (1 - 1) * (1 - 1)(1 - 1) = 1 \tag{31}$$

We carried out 27 round verifications. We show and compare their completion time. The result is shown in Figure 13.

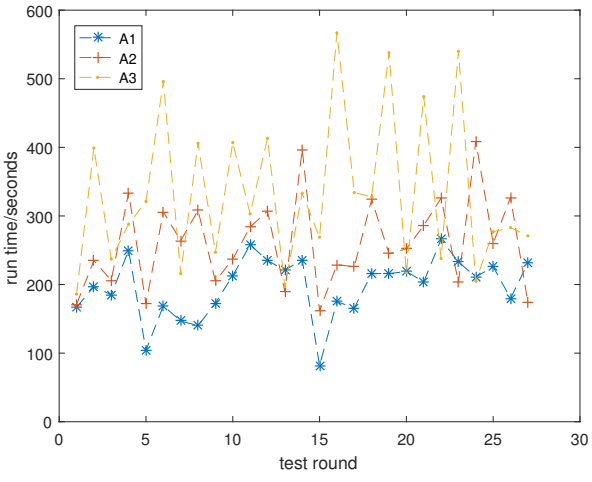

Fig. 13. The run time of A1, A2, and A3. All target actions are triggered and then finish before timeout.

From Figure 13, we can see that A3 takes the longest time, an average of 43.72 seconds. The reason is that the target blockchain (blockchain 5) requires information that needs to propagate from another three blockchains (blockchain 2, 3 and 4). Although any one of the all-trigger can trigger the action on blockchain 5; while transactions need to propagate over three blockchains to blockchain 5 as it is a ring topology.

The run time is different among A1, A2, and A3, while the target blockchain is triggered in all those 81 test rounds. It demonstrates that a target action can be triggered among an all-trigger path. We also dynamically shut down one or two blockchain nodes. The action is still triggered, which indicates that partial node failures do not affect the whole blockchain. The reason is that the blockchain consensus algorithm can reach an agreement and keep the final status unified.

C. dynamic-trigger

There are also three kinds of verifications for the dynamic-trigger type, named D1, D2, and D3, and their logic connections are shown in Figure 14. In the D1 case, a user triggers the initial action in blockchain 1, which sends a dynamic-trigger to blockchain 2 and blockchain 3. Then either blockchain 2 or 3 wins the trigger, which sends an all-trigger to blockchain 4.

D2 case is similar to D1 in the first steps that blockchain 1 sends a dynamic-trigger to blockchain 2 and blockchain 3. The difference is that blockchain 2 sends an all-trigger to blockchain 5 and blockchain 3 sends an all-trigger to blockchain 4. As only blockchain 2 or 3 gets the dynamic-trigger, then the target blockchain (blockchain 5) is triggered dynamically with the probability of 1/2.

In the D3 case, the first step is still similar. While blockchain 1 sends a dynamic-trigger to blockchain 2, 3, and 4. Only blockchain 2 sends the all-trigger to blockchain 5 if blockchain 2 wins the dynamic-trigger. As there are three blockchains to race for one dynamic-trigger, blockchain 5 triggered dynamically with the probability of 1/3.

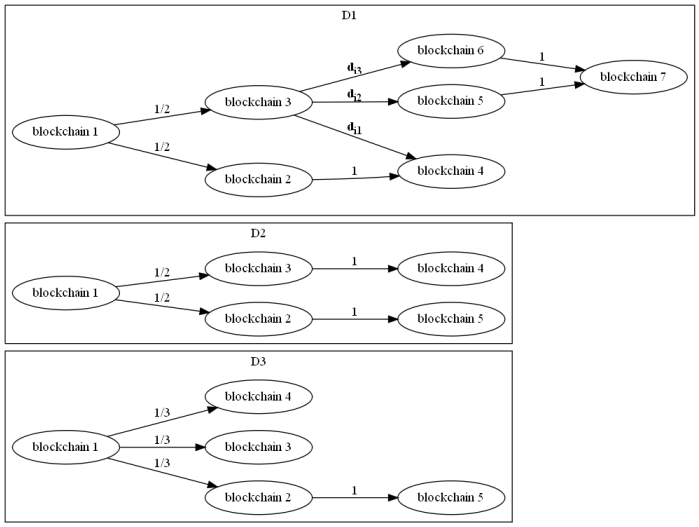

Fig. 14. Logic connection of dynamic-trigger D1, D2, and D3. Different numbers of blockchains to race for one possible trigger.

The trigger results are shown in Figure 15, 16 and 17. The y-axis is the test round, and if the action in a blockchain is triggered in a test round, we draw a small circle for the blockchain at that test round.

Figure 15 shows the results of D1, from which we see that the probability to win the dynamic trigger from blockchain

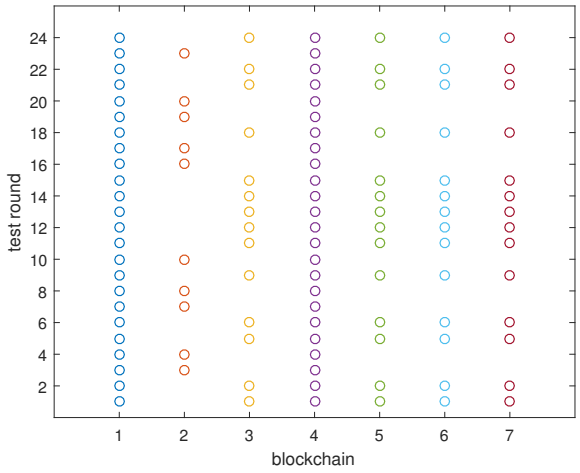

Fig. 15. The trigger distribution of D1. If the target action is triggered, a small circle is drawn at the corresponding test round (it is same for the following two figures).

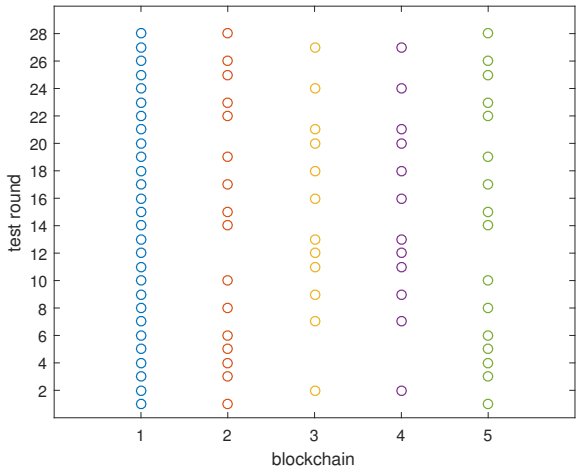

Fig. 16. The trigger distribution of D2.

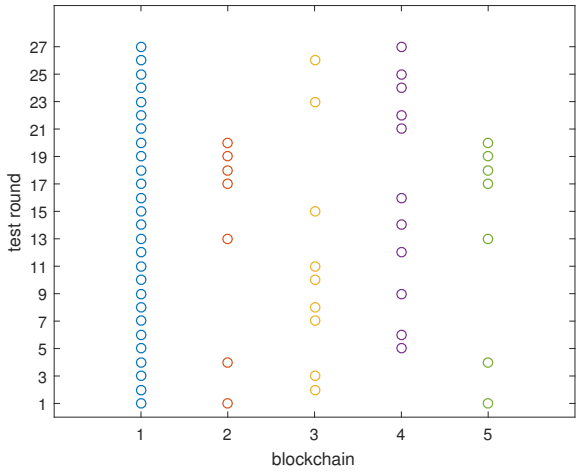

Fig. 17. The trigger distribution of D3.

1 is dynamic for blockchain 2 and 3. This also verifies the dynamic-trigger function described in Section III-B1.

For blockchain 4, it is triggered with the probability of 1. Its weight array is $(1/2,\ 1/2\mathbf{d_{i_1}})$. It can be regarded as the calculation among dimension (1) and its derived dimension (i), and then the final weight is $1/2 + 1/2 = 1$.

For blockchain 7, it only triggers with the probability of 1/2. Its weight array is $(1/2 * \mathbf{d_{i_2}},\ 1/2 * \mathbf{d_{i_3}})$, and the final weight is $1/2*(1-(1-1)*(1-1)) = 1/2$. This demonstrates that dimensions from the same action can not be added. It also indicates that the dimension information should be kept during the calculation process.

From Figure 16 and 17, we see with the increase of the number of racing blockchains, the probability for a blockchain to run is smaller. The probability is the reciprocal of blockchain numbers, as each number has equal chance to run.

1) Optimization of Waiting Time: In this part, we try to verify the optimization of the waiting time by TDT (time denying transaction). The logic connection is shown in Figure 18. There is a dynamic-trigger from blockchain 1 to blockchain 2 and 3. We measure the effect of the optimization by the completion time of a smart contract (named SOp), which runs on blockchain 4 and can be triggered by an action from either blockchain 2 or blockchain 3. To make the process reflect the propagation and confirmation time, the logic of SOp is simple; it only logs its start time and completion time after it receives its trigger.

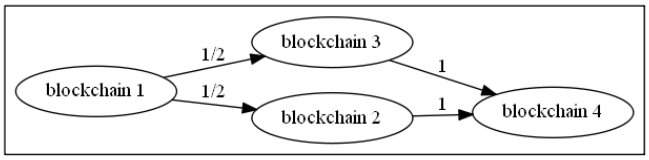

Fig. 18. Logic connection used in optimization of dynamic-trigger

Two kinds of comparable experiments are carried out, O1 and O2. In both experiments, we dynamically choose blockchain 2 or 3 not to respond (the dynamic is generated also by the hash of that dynamic-trigger), called non-responsible blockchain (NRB). While in O1, NRB does not send TDT to notify other blockchains, and then other blockchains wait until timeout (the timeout value is set to 1200 seconds). In O2, NRB sends a TDT. We have carried out 35 test rounds for O1 and O2 separately. The results are shown in Figure 19.

From Figure 19, we see that the run time in O2 changes from the minimum value (212 seconds) to the maximum value (821 seconds). While in O1, dynamic-triggers are done until it times out. No matter what the performance of blockchain networks is, the run time of O1 is still 1200 seconds. Another point is that the timeout value should be big enough, with the aim to ensure that dynamic-triggers can be propagated and confirmed in other blockchains. In this verification, its value (1200 seconds) is much larger than the biggest values (821 seconds) in the O1 case.

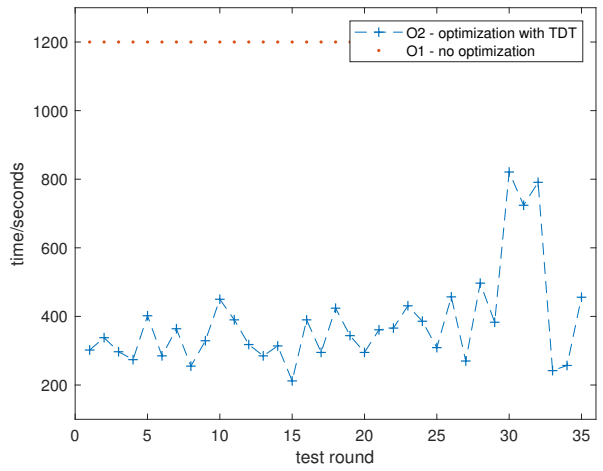

Fig. 19. Time comparison between dynamic-trigger with (O2) and without (O1) optimization.

VI. CONCLUSION

In this paper, we address the issues of the triggering relationship among cross-chain smart contracts, which are deployed on different blockchains. We first propose the condition-trigger model as the basic model for the analysis, its different trigger types, including the all-trigger, and the dynamic-trigger, concerning whether it limits the number of successive smart contracts to be triggered or not. Second, we propose to use the weights of the graph (formed by different actions) to analyze the probability of cooperation. As the edge weight cannot be used directly as the cooperation probabilities, we introduce the dimension in the edge weight. At last, we evaluate the condition-trigger model and its different trigger types, the all-trigger and dynamic-trigger. The evaluation results show that our proposed methods can analyze the (action) cooperation relationship among smart contracts in different blockchains.

ACKNOWLEDGMENT

The authors thank the anonymous reviewers for their constructive comments, which help us to improve the quality of this paper. This work was supported in part by the National Natural Science Foundation of China under Grant No. 61772352; the Science and Technology Planning Project of Sichuan Province under Grant No. 2019YFG0400, 2018GZDZX0031, 2018GZDZX0004, 2017GZDZX0003, 2018JY0182, 19ZDYF1286.

REFERENCES

[1] Kan L , Wei Y , Hafiz Muhammad A , et al. [IEEE 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C) - Lisbon, Portugal (2018.7.16-2018.7.20)] 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C) - A Multiple Blockchains Architecture on Inter-Blockchain Communication[C]// 2018:139-145.

[2] Wang H, Cen Y, Li X. Blockchain Router: A Cross-Chain Communication Protocol[AC]//Proceedings of the 6th International Conference on Informatics, Environment, Energy and Applications. ACM, 2017: 94-97.

[3] Li, W., Sforzin, A., Fedorov, S., & Karame, G. (2017). Towards Scalable and Private Industrial Blockchains. Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts, 9-14.

[4] Gaži P, Kiayias A, Zindros D. Proof-of-stake sidechains[C]//2019 IEEE Symposium on Security and Privacy (SP). IEEE, 2019: 139-156.

[5] Dang H, Dinh T T A, Loghin D, et al. Towards scaling blockchain systems via sharding[C]//Proceedings of the 2019 international conference on management of data. 2019: 123-140.

[6] Kannengießer N, Pfister M, Greulich M, et al. Bridges between islands: Cross-chain technology for distributed ledger technology[C]//Proceedings of the 53rd Hawaii International Conference on System Sciences. 2020.

[7] Borkowski M. Cross Blockchain Technologies: Review, State of the Art, and Outlook. 2019[J]. URL: http://dsg. tuwien. ac. at/projects/tast/pub/tast-white-paper-4. pdf. White Paper, Technische Universität Wien. Version, 1.

[8] Zie J Y, Deneuville J C, Briffaut J, et al. Extending atomic cross-chain swaps[M]//Data Privacy Management, Cryptocurrencies and Blockchain Technology. Springer, Cham, 2019: 219-229.

[9] Liu Z, Xiang Y, Shi J, et al. Hyperservice: Interoperability and programmability across heterogeneous blockchains[C]//Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. 2019: 549-566.A

[10] Dolgui A, Ivanov D, Potryasaev S, et al. Blockchain-oriented dynamic modelling of smart contract design and execution in the supply chain[J]. International Journal of Production Research, 2020, 58(7): 2184-2199.

[11] Qiu H, Wu X, Zhang S, et al. ChainIDE: A cloud-based integrated development environment for cross-blockchain smart contracts[C]//2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom). IEEE, 2019: 317-319.

[12] Su H, Guo B, Shen Y, et al. A Solution for State Conflicts of Smart Contract in Interaction with Non-blockchain[C]//IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, 2020: 382-387.

[13] Dai B, Jiang S, Zhu M, et al. Research and Implementation of Cross-Chain Transaction Model Based on Improved Hash-Locking[C]//International Conference on Blockchain and Trustworthy Systems. Springer, Singapore, 2020: 218-230.

[14] Herlihy M. Atomic cross-chain swaps[C]//Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing. ACM, 2018: 245-254.

[15] Wang Q , Wang S , Zhang P , et al. An Achieving Data Exchange Cross-Chain Alliance Protocol[J]. Journal of Physics: Conference Series, 2019, 1213:042037-.

[16] Lin W, Yin X, Wang S, et al. A Blockchain-enabled decentralized settlement model for IoT data exchange services[J]. Wireless Networks, 2020: 1-15.

[17] Van Glabbeek R , Gramoli V , Tholoniat P . Cross-Chain Payment Protocols with Success Guarantees[J]. 2019.

[18] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. bitcoin.org, 2008.

[19] Chen Y H, Chen S H, Lin I C. Blockchain based smart contract for bidding system[C]//2018 IEEE International Conference on Applied System Invention (ICASI). IEEE, 2018: 208-211.

[20] Mengelkamp E, Notheisen B, Beer C, et al. A blockchain-based smart grid: towards sustainable local energy markets[J]. Computer Science-Research and Development, 2018, 33(1-2): 207-214.

[21] S. Karumba, S. S. Kanhere, R. Jurdak and S. Sethuvenkatraman, "HARB: A Hypergraph-based Adaptive Consortium Blockchain for Decentralised Energy Trading," in IEEE Internet of Things Journal, doi: 10.1109/JIOT.2020.3022045.

[22] Zhang B, Jiang C, Yu J L, et al. A contract game for direct energy trading in smart grid[J]. IEEE Transactions on Smart Grid, 2016, 9(4): 2873-2884.

[23] Ilham A. Qasse, Manar Abu Talib, and Qassim Nasir. 2019. Inter Blockchain Communication: A Survey. In Proceedings of the ArabWIC 6th Annual International Conference Research Track (ArabWIC 2019). Association for Computing Machinery, New York, NY, USA, Article 2, 1–6.

[24] Kakimoto M, Endoh Y, Shin H, et al. Probabilistic solar irradiance forecasting by conditioning joint probability method and its application to electric power trading[J]. IEEE Transactions on Sustainable Energy, 2018, 10(2): 983-993.

[25] Azaïs R, Bardet J B, Génadot A, et al. Piecewise deterministic Markov process—recent results[C]//ESAIM: Proceedings. EDP Sciences, 2014, 44: 276-290.