Bridge Oracle
First Ever Public Oracle System on TRON Network

Bridge.link

Bridge Oracle System

First Ever Public Oracle System on TRON Network

by Bridge Oracle

Spring 2020

Last Update: July 29th 2020

White Paper Version: v 0.1.1

Abstract

Smart contracts are going to change the world's businesses by providing an infrastructure for creating Dapps (decentralized applications) and DAOs (Decentralized Autonomous Organizations) in order to automate jobs based on blockchain technology. However, one should note that due to underlying consensus protocols, the blockchain on which smart contract is deployed and triggered, cannot access external data.

In order to solve this problem, a kind of special technology was proposed which is called *Oracle*. Oracles inject information of the real world into the blockchains whenever needed. Without oracles, blockchains can be only used for tokenization purposes; however, by setting oracle technology on a blockchain, smart contracts can be programmed as required and perform as a decentralized autonomous organization.

In this paper, Bridge project is presented as the first ever dedicated public oracle system on TRON network that is introduced to serve smart contracts which are deployed on TRON blockchain. At the end of the paper, roadmap of the project is declared which is consisted of adding various data sources, responses with various kind of proofs, developing decentralized oracle service which is an incentive based system for reporting aggregating response to blockchain and providing possibilities for launching enterprise oracles on behalf of companies to sell dedicated data.

Keywords

smart contract, dapp, decentralized application, DAO, decentralized autonomous organization, blockchain, consensus protocol, deploy, trigger, oracle, tokenization, proof, incentive, aggregating response

Contents

1. Introduction

Smart contracts are a kind of application with tamperproof context and clauses which are deployed and executed on blockchain networks. It means that no party (and even creators) are able to manipulate the codes and functions after deployment. Thus, unlike traditional paper contracts or digital ones which are programmed on centralized platforms that are exposed to alteration, termination and deletion by a trusted party or a third person, smart contracts bring parties into agreement and generate a novel and powerful class of trust without relying on trust in any party or intermediaries. This feature turns smart contracts into a superior tool for realizing and implementing digital agreements.

It should be noted that smart contracts are trying to digitalize real-world agreements. Consequently, in order to carry out such a task, they need to access real-world data. However, due to special underlying consensus protocols, blockchains are avoided to connect external data sources, therefore they cannot access outside. Thus, smart contract developers encounter a *connectivity* issue according to which majority of smart contracts are not able to function practically. Without connectivity, smart contracts and blockchains are only tools and platforms, respectively by which tokenization of shares and assets of organizations are possible and they are not flexible enough to be used as a programming ecosystem. For this cause the main usage of smart contracts which are deployed in TRON network are low price and weak tokens which is an ordinary and common use case in any other blockchain networks.

Current concentration on tokens is due to lack of a public and user-friendly oracle service in TRON network. Thus, beside blockchain technology, oracle system is a vital requirement with which blockchain platforms acquire the flexibility of handling all kind of applications and digital autonomous organizations by accessing external data. Oracles are a kind of technology through which people are able to inject real-world data into their smart contracts. The most useful data source that is utilized as data feed in order to inject information, is HTTP/HTTPS API endpoint. As it was declared before, due to consensus mechanisms used by blockchains, they are unable to directly fetch such critical data.

In this paper, connectivity issue of TRON network is addressed and subsequently a solution for injecting external data into smart contracts is provided. Bridge is the first ever dedicated public oracle technology on TRON network which is developed to provide ability of accessing external data for ordinary users' smart contracts. Because making smart contracts to be externally aware and capable of accessing off-chain resources is a vital aspect, if they are expected to be substituted with manual and digital agreements in use today.

The features due which Bridge oracle system shows off, are consisted of following clauses:

- The ability to access external data using various form of APIs and parsing helpers such as JSON, XML, HTML.
- The ability to add extra various data sources such as BTFS, WolframAlpha, Random, etc.
- The ability to add various kinds of proof to prove the authenticity of the injected data.
- The ability to develop a decentralized oracle service upon current infrastructure.
- The ability to add dedicated oracle data carriers for special enterprises in order to sell dedicated raw data.
- The ability to accept two various payment methods including native coin of TRON blockchain network (TRX) and project related token (BRG) with a special discount.

Before proceeding to the main project, examples of potential next-generation smart contracts and their needs to external data are presented which are expected to be substituted by their traditional equivalent:

- *Normal Contracts*: For example, contract of promoting a video to reach a certain number of views. When the views of the video reach to a certain number, promoter should receive money.
  Requirements: Accessing to number of views using scraping by xpath language.
- *Securities*: For example, bonds, interest rate derivatives, etc.
  Requirements: Accessing to APIs which report market prices and market reference data such as interest rate.
-

- *Insurance*: For example, if fire extinguisher system of a building gets the ability to record fire data, estimate the destruction and inject it into insurance smart contract, insurance contracts can pay the compensation. Requirements: Accessing to IoT (Internet of Things) data feeds.
- *Trade finance*: For example, when transit of goods is completed to a special region, freight is paid using smart contract.
  Requirements: Accessing to GPS data about shipment and alike information in order to fulfill contractual obligations.

  .

  .

  .
- ***Dapp*: Generally, any kind of decentralized application which is operated by smart contract as backend codes. One can implement any kind of application using smart contracts in which any kind of information may be required.
- ***DAO*:** Generally, any kind of decentralized autonomous organization in which usage of any kind of information is possible.

The underlying reasons of using smart contracts for developing Dapps and DAOs are as follows:

- No need to backend codes on centralized servers.
- Handling security of contract by blockchain technology.
- Providing possibility of tokenizing your platform and create dedicated assets.
- Using peer to peer payment methods through native coin and tokens of the same blockchain network.
- Registering transactions publicly and transparently which can be used as receipt by customers.
- Etc.

Lack of a public oracle system in TRON network is strongly felt. Creating smart contracts without oracle technology is almost impossible and without oracle technology, blockchains use case is limited to be a tokenization platform. Although powerful enterprises are able to have dedicated oracle systems, people with limited budget, ability, time or any other reason are unable to prepare one.

However, as you well know, in all over the world, small businesses are more important than strong organizations and great economic enterprises. Thus, providing an infrastructure for attracting small businesses and implementing their businesses on TRON blockchain is critical and vital for progression and expansion of the network. For instance as emerging data centers and server services providers together with ready-to-use content manager systems including WordPress, Joomla, etc. led to spread of implementing small businesses on internet easily, creating a public oracle system on TRON network will cause people to be able to implement their local or small business on blockchain, as well and benefit from peer to peer payment methods of TRON network using native coin and tokens.

In the following, Bridge oracle is presented.

2. Bridge Oracle

2.1. Oracle Structure

As it was said previously, oracle is a kind of technology with which blockchains are able to access real-world data. Simple schematic structure of Bridge oracle system can be seen in Fig. 1:

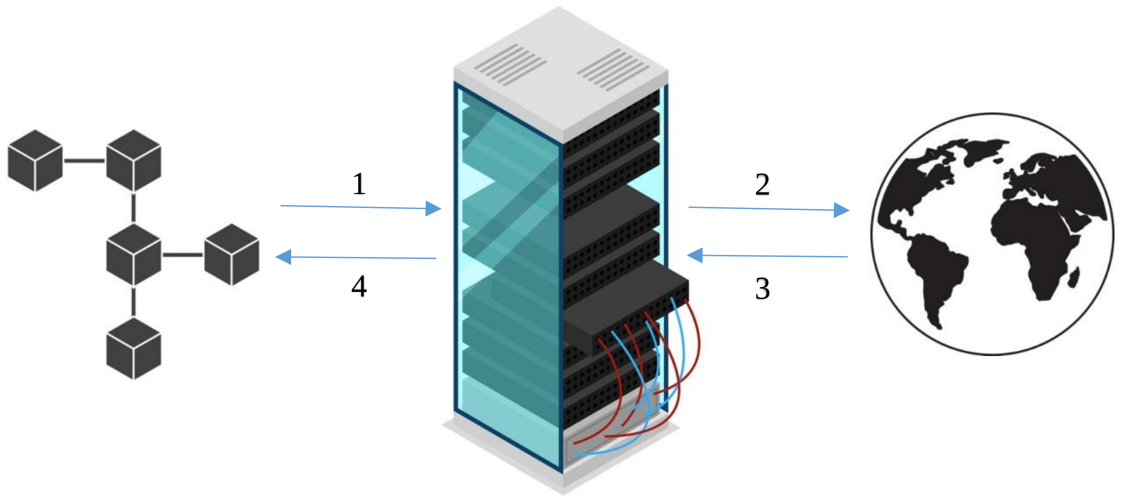

Figure 1. Simple Schematic Structure of Bridge Oracle System

Bridge oracle system is composed of 2 distinct parts including smart contracts which perform in blockchain network and oracle data carriers that bridges between blockchain and real-world data.

2.1.1. Smart Contracts

Bridge oracle is consisted of 3 various smart contracts that interact with each other. One of the contracts is the Bridge API contract that client's smart contract should inherit from this contract through which client's smart contract can connect to Bridge oracle and benefit from provided services. The other contract is Bridge oracle address resolver which redirects client's requests to the correct service including public oracle system, decentralized oracle system, a specific enterprise oracle system, etc. The last contract is Bridge oracle connector that receives client's request and process it in format of a query and emit specific data which can be read by watching oracle data carriers.

Notice: Note that interactions between Bridge's smart contracts are established in such a way that upgrading Bridge oracle system to a higher version with more features and without occurrence of any disruption in previous users' contract is possible.

2.1.2. Data Carriers

After emitting information in blockchain by Bridge oracle connector contract, Bridge oracle data carriers that are watching connector smart contract, receive a group of information from which they recognize blockchain requirements of real-world data.

2.2. Oracle Performance

Let's investigate how an oracle system works generally. By considering Fig. 1, in the first step, blockchain emits special data using logging events that inspires its requirements of real world-data. In second step, oracle data carriers that are watching such events, understand blockchain's needs of data and try to fetch them from real world by sending requests to various data sources using different kind of methods such as APIs, downloading files, asking questions, etc. In the third step, intended resource respond to the oracle data carriers' request and send requested data to the oracle data carriers. In the last step, oracle data carriers inject final result to the blockchain and eventually blockchain's need for external information is met.

Fig. 2 shows interaction of smart contracts of Bridge oracle system with together and with oracle data carriers in details. As it can be seen in Fig. 2, in the first step, proper API contract should be imported into user's contract based on owner's requirements. There are various API contracts, each has a certain parameter written in it from which user's contract connects to the appropriate connector contract.

After importing proper API contract to user's contract, in second step and before sending query to the proper contract, user's contract needs to know address of the appropriate connector contract in order to send the query. So, asking the address of the proper contract is carried out using the certain parameter which is sent to oracle address resolver contract from utilized API contract.

In third step oracle address resolver contract finds the appropriate connector contract address using a hash table that is provided in it. This hash table receives the certain parameter which mentions the proper connector contract, and returns the address of the connector to the user's contract.

In fourth step, user's contract sends the query to the determined connector address on which a special kind of oracle data carriers including public, decentralized or enterprise are watching.

In fifth step the connector contract sorts input data in a standard pattern and logs corresponding event based on user's needs and requirements.

In sixth step valid oracle data carriers that are watching the connector contract, receives user's request in format of events.

In seventh step valid oracle data carriers send request to the data source which is requested by user's contract and ask for required data.

In eighth step requested data source responds to data carriers' requests and delivers appropriate data.

In ninth step which is the last step of the procedure, oracle data carriers inject users' requested data into their smart contracts.

Note that final results are injected to __callback function of users' smart contracts which are only allowed to be triggered by Bridge's valid oracle data carriers in order to avoid injecting fake data by evil and fraud sources.

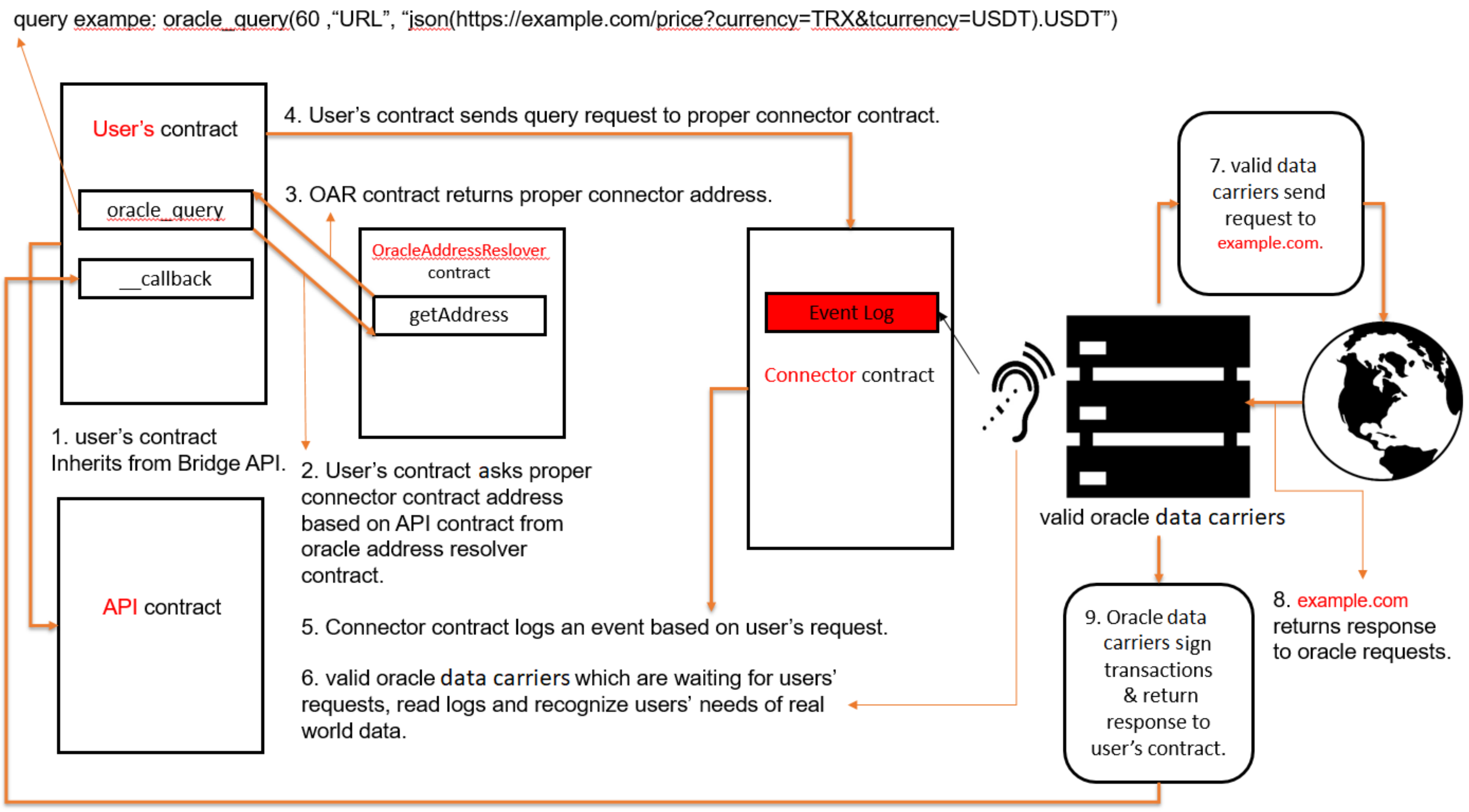
query exampe: oracle_query(60 ,“URL”, “json(https://example.com/price?currency=TRX&tcurrency=USDT).USDT”)
User’s contract
oracle_query
__callback
4. User’s contract sends query request to proper connector contract.
3. OAR contract returns proper connector address.
OracleAddressReslover contract
getAddress
Event Log
Connector contract
7. valid data carriers send request to example.com.
valid oracle data carriers
1. user’s contract Inherits from Bridge API.
2. User’s contract asks proper connector contract address based on API contract from oracle address resolver contract.
API contract
5. Connector contract logs an event based on user’s request.
6. valid oracle data carriers which are waiting for users’ requests, read logs and recognize users’ needs of real world data.
9. Oracle data carriers sign transactions & return response to user’s contract.
8. example.com returns response to oracle requests.

Figure 2. Interaction of Smart Contracts of Bridge Oracle System with Together and with Oracle Data Carriers

2.3. Oracle API Contracts

As it was said in previous section, proper API contract should be imported into user's contract based on owner's requirements. So, there are various API contracts in Bridge ecosystem, each has special functionality. Various API contracts include:

- Public API contract
- Decentralized API contract
- Enterprise API contracts that are dedicated to each company or organization which is volunteer to sell data for BRG token

It should be noted that using each of above contracts depends on user's requirements.

2.3.1. Public API Contract

Public API contract has been designed to simply obviate requirements from external data that the validity of data or proving the validity of data is not that important for user. Of course, in this method various kinds of proof will be provided to resolve proof of validity of data in a near future.

2.3.2. Decentralized API Contract

Decentralized API contract has been prepared for those users who need to implement a fully decentralized platform. Thus, they need to fetch external data using a decentralized oracle system as well. As developing such infrastructure is in our future plans, details of designing such oracle system will be reported in future.

2.3.3. Enterprise API Contract

Enterprise API contract has been provided for enterprise companies and organizations that own dedicated and special data which are not published and shared publicly. These companies and organizations can sell such data using Bridge system and users who need such data can import that company's dedicated API contract. This API contract redirects users' requests to that company's connector on which company's dedicated oracle data carriers watch and respond users' contracts directly and inject dedicated data to their contract privately. Companies and organizations can exchange their dedicated information for BRG token which is paid by users' contracts. In fact, enterprise organizations can make money by selling data to Bridge users. This is one of Bridge system's business plans. Business plans will be discussed in next sections.

Notice: Note that only public API contract is prepared yet and other oracle systems are going to be created in a near future based on roadmap plans.

2.4. Public Oracle Data Sources

Data sources are various kind of trusted references such as a website or a web API from which required data is requested by oracle data carriers based on users' choice. One should note that each data source has a special use case and functionality and choosing a suitable data source is of great importance. There are several critical data sources that are expected to exist in Bridge oracle system. Various data sources that are supported by Bridge's public oracle system include:

- URL
- Complex URL
- WolframAlpha
- Random
- BTFS
- Nested
- Etc.

2.4.1. URL

This data source is a general data source that enables users to fetch every kind of data using http/https request APIs that related responses are in format of one of the following cases:

1. JSON
2. XML
3. HTML

2.4.2. Complex URL

URL data source is able to fetch a parameter from http/https request APIs that is consisted of a solo set of JSON/XML/HTML parameters. However, there are special cases in which one encounters an array of multiple set of JSON/XML/HTML parameters. For instance, an array consisted of multiple set of parameters that relates to matches of an event. Suppose that we need to know the winner of a match with a special matchID that exists in the array. In order to extract a data like matchWinner using a deterministic parameter like matchID, such data source is provided.

2.4.3. WolframAlpha

This data source redirects users' requests and questions to computational knowledge engine of WolframAlpha Company which is able to answer what one wants to calculate or know about. WolframAlpha computational intelligence answers the inquiries and corresponding responses are returned to the users' smart contracts.

2.4.4. Random

This data source generates random numbers and injects them into users' contracts. Random number has many critical use cases including statistical sampling, computer simulation, cryptography, completely randomized design, scientific calculations, etc.

2.4.5. BTFS

BTFS is decentralized storage system of TRON network. Using this data source enables users' contracts to be able to interact with the BitTorrent File System.

2.4.6. Nested

This case enables users to utilize the combination of different types of data source or multiple requests of the same data source that returns a unique result into users' smart contracts.

Notice 1: It should be noted that only URL data source is ready to use for now and other data sources are being developed to be available in a near future.

Notice 2: Off-chain architecture has been developed based on polymorphism provision according to which additional data sources can be implemented easily. Thus, in order to have more practical data sources in Bridge oracle, users of all around the world can suggest new ones and Bridge team will provide them in turn based on priority.

Notice 3: Note that above data sources are those references that are accessible publicly and data which are returned in such data sources are open to all. In these cases, Bridge oracle is only a gateway to access to such public real-world data. However, in order to access dedicated data by special companies and organizations, people need to import company's dedicated API contract in order to connect to their dedicated connector contract and oracle data carriers and receive their dedicated rare data directly from their data carriers in exchange for BRG tokens.

2.5. Responding Time of Requests

It is obvious that users need to receive real-world data into their smart contract on a certain time. Various kind of data react differently over time. A group of information is time-dependent data which alters continuously, for example TRX/USD live price. On the other hand, there are some information that is almost time-independent, for instance context of a post in a social media application that was sent previously and has not been modified or deleted yet. Fetching such data is independent of time and any time a user fetches the data, a similar result will be received. Also, there are some cases that user doesn't know when an event will be occurred. For example, no one can guess the exact ending time of a snooker match. Thus, such an event has an open ending time.

Based on above descriptions, one can guess that any event may have one of the three above conditions. According to this fact, Bridge oracle provides three various time-dependent options for requesting external data. These three options include:

1. On time queries
2. Scheduled queries
3. Open ending time queries

2.5.1. On Time Queries

By using this option in users' contract, responses will be returned by Bridge oracle as soon as possible. When transactions of users' requests are confirmed and corresponding information is emitted in connector contract and they are received by oracle data carriers, the requests are processed right away and the results are returned immediately.

2.5.2. Scheduled Queries

Using this option enables users' contracts to submit queries in connector contract which should be responded in the future and after a certain period of time or on a certain time stamp. For example, users are able to ask Bridge oracle to return price of TRX/USD after an hour or even tomorrow at 10'oclock through setting the exact time stamp.

2.4.7. / 2.5.3. Open Ending Time Queries

This part has been classified as both sections 2.4.7 and 2.5.3. This is due to the fact that this section is a distinct data source which has special use cases. Also requests that are sent using this feature, has a specific time-dependent functionality. For instance, consider a snooker game match. No one is able to predict ending time of the match exactly. Now suppose that a person wants to inject winner of the match and carry out some operations based on the winner. However, match is not finished to have a static result and no one can even guess the ending time of the match yet. In such cases there are three approaches based on user's opinion and intended server's capability:

1. Web socket protocol
2. Long polling method
3. Recursive HTTP/HTTPS request method

Users should utilize first method in cases that using web socket protocol to make connection to API link is possible. This method is an optimized approach and receiving response through this approach into users' contract is the most affordable method in comparison to the others. In this method, Bridge oracle data carriers make a connection to the target link using web socket protocol and waits for a deterministic parameter to be responded. In above example, Bridge oracle will wait for winner parameter of the snooker match to be fulfilled. As soon as the match is finished and Bridge oracle receives intended response, it terminates the connection and injects received data to user's contract.

Long polling method is a similar (inefficient) approach which is provided in old servers in instead of web socket protocol.

Now consider that a user needs to access a data from a link with similar condition; however, making a web socket or long polling connection to the link is impossible. In this cases recursive HTTP/HTTPS requests method should be utilized. Before using the method, user should estimate the ending time of the event. Then the user should determine the time step after each, Bridge oracle ask the latest result from intended link for intended parameter. By setting these two elements in user's contract, process is started and Bridge oracle tries to get the intended result in several efforts. After a certain time, which was predicted by user, oracle tries for the first time to get the result. If oracle achieves the result, it will terminate the connection and returns the result. Otherwise it will hold the request and try after the time step again. This process will be continued until the oracle get intended result. As soon as Bridge oracle gets the result in its latest query, it will terminate the job and injects received data to the user's smart contract.

2.5.4. Recursive Queries

This section is a special case that is not anything new in comparison to above time dependent functionalities; however, this part has a special use case that should be considered. This option has been provided for injecting any kind of data using various methods repeatedly and continuously based on an intended time step. For instance, one can consider a DEX which wants to have up to date price of TRX in its smart contract, continuously. In such smart contract TRX price is injected to blockchain every *n* seconds which is addressed as time step. The approach to resolve such cases is recursive queries. In order to investigate implementation of recursive queries in smart contract using Bridge oracle, one can refer to Bridge document in bridge.link website.

2.6. Off-Chain System Architecture

In this section, off-chain system architecture is presented. Fig. 3 shows off-chain system architecture, schematically.

This architecture is presented to declare that what happens after users' requests are received by Bridge oracle's data carriers. As it was discussed before, in very first step, requests which are logged in blockchain, are received by off-chain system. This task is done by the messaging queue system that is responsible for arranging query execution queue and balancing loads of oracle data carriers. After receiving user's request, messaging queue system logs the query into a temporary database. This temporary database only logs those requests which has not been processed yet and corresponding result has not been injected to user's smart contract. After completion of the process of a certain request, it is removed from this database. This database is located on random access memory (RAM) in order to benefit from maximum speed of logging and fetching data. This pattern is implemented in order to avoid losing any of users' queries in high load condition, especially in TRON network in which block time is 3 seconds and confirming a transaction is carried out rapidly.

After that, received tasks are separated and assigned to each oracle data carrier to be processed based on complexity of a task and each oracle data carrier’s processing load in order to preserve optimality of performance of the system. After assigning a task to a data carrier, data carrier id is added to log of that request for further actions. This id shows that which computer is responsible for a certain query.

It should be noted that messaging queue system is always in interaction with main oracle data carriers and monitors their situation, including on/off status, executing load, etc. This part is provided to not only preserve optimality of the system but also maintain stability of it. In order to know, how this part maintain stability of the system, suppose that for example oracle data carrier 2 crashes or it is shut down due to maintenance or any other reason. According to ceaseless interaction of messaging queue system and oracle data carriers, messaging queue system recognizes that oracle data carrier 2 is out of service; however, there should be several tasks which were assigned to be carried out by oracle data carrier 2 that is out of service right now. In such a condition, messaging queue system retrieve those queries which were assigned to data carrier 2 instantly using logged id for each request and based on other data carriers’ processing load, separates and assigns retrieved tasks between other them to be executed in an optimized manner. Using this approach messaging queue system maintains optimality and stability of Bridge oracle system.

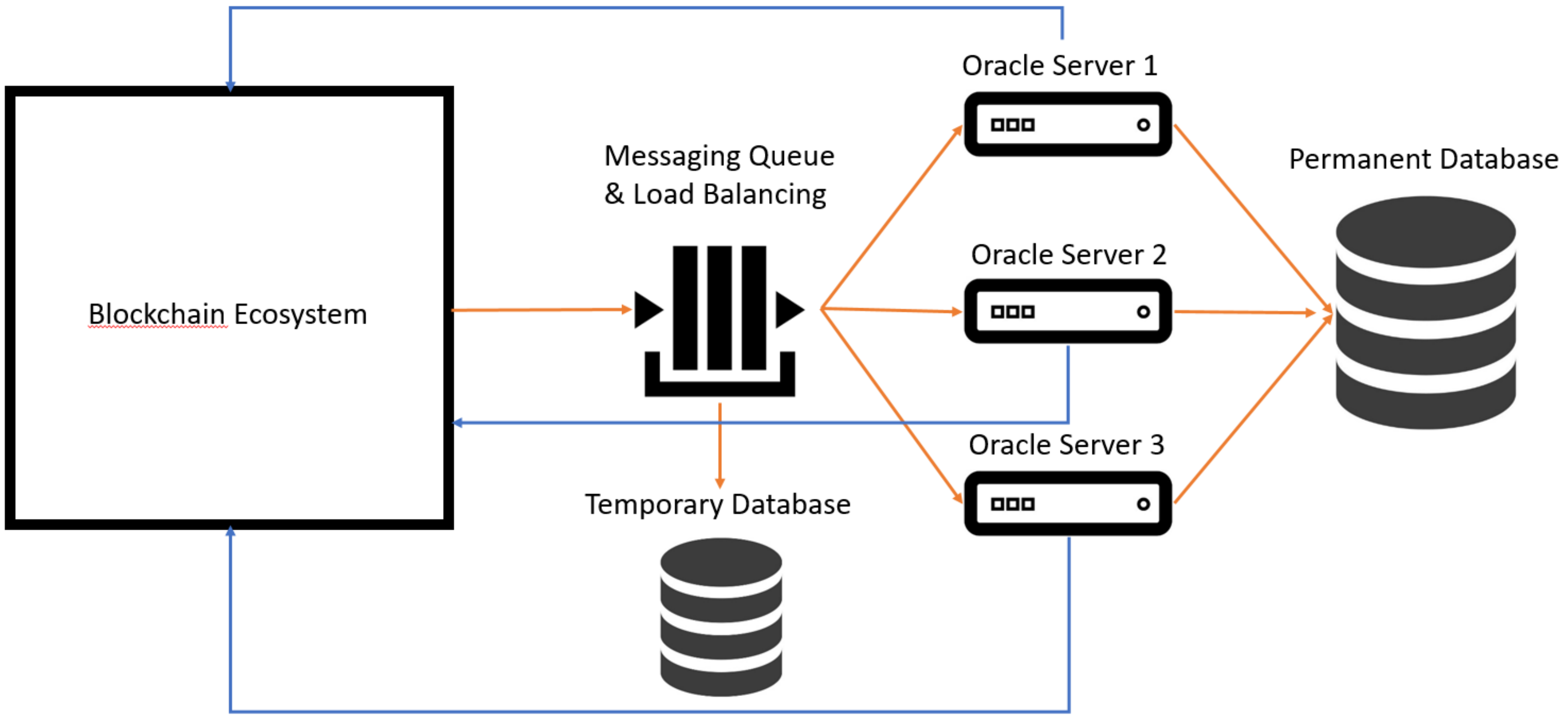

Figure 3. Schematic of Bridge Oracle's Off-Chain System Architecture

Now suppose that messaging queue system crashes. In such a case, after restarting the service it can retrieve all requests from temporary database and continue to process the requests again.

After forwarding a request to a certain oracle data carrier by messaging queue system, request is sent to intended resource in order to receive proper response based on user's request. If an appropriate response is received by data carriers, it will be sent to user's contract. However, in some cases intended resource is not responding due to being down or busy or any other reason due which proper response is not received by oracle data carriers. In such cases oracle data carriers hold the task and try it a couple of times more. If Bridge oracle data carriers cannot receive a proper response after several attempts, a blank string will be injected into user's contract.

Before injecting received response into user's contract by Bridge oracle data carriers, last result which is going to be injected to user's contract is logged in a permanent database together with further corresponding information including query id, etc. that enables users to inquire about the response which is forwarded to their smart contract using query id. Of course, it should be noted that information of queries cannot be stored for ever in permanent database and after several months, this information will be removed in order to have an optimized searching process for fetching queries information.

2.7. Payment Methods

In order to place a request of injecting data in Bridge oracle system, users must pay the cost. Bridge oracle provides two various payment methods for users to pay the cost of their requests including:

1. BRG (Bridge token)
2. TRX

Price of a request is calculated based on a certain algorithm and the amount is paid automatically from user's contract. In fact, in order to use Bridge oracle system, user's contract is charged by BRG or TRX.

Before discussing payment process, note that price of the request is calculated based on TRX. Thus, in order to have the price based on BRG token, up to date exchange rate of BRG/TRX should be available every moment for our pricing system with which BRG-based price can be easily calculated. For solving this issue, a bot has been developed to watch BRG/TRX exchange rate continuously and compare it with the rate which was injected to Bridge oracle's pricing system, previously. If the difference is more than 1%, this bot updates the price and injects the new exchange rate to the blockchain and cost of next requests are calculated based on the new exchange rate.

In first step Bridge oracle checks BRG balance of user's contract. If there is enough balance in user's contract according to the cost of the request, Bridge will charge user's contract by BRG token, automatically. If there is not enough BRG balance in user's contract, in second step Bridge oracle will check TRX balance of user's contract. This time if there is enough TRX balance in user's contract, Bridge will charge user's contract by TRX. If there is not enough BRG and TRX balance in user's contract, user's request will be rejected and no response will be reported into user's smart contract by Bridge oracle system.

Notice: Charging user's contract by BRG token include a percentage of discount in comparison with TRX payment method. In fact, paying BRG token is more affordable than paying by TRX for placing a request in Bridge oracle system.

2.8. Roadmap

Due to complexity of the system and use cases, providing a prioritized roadmap with exact details is impossible; however general purposes are provided as following:

- Providing a feedback section in Bridge oracle's official website, bridge.link to receive users' suggestions.
- Adding various data sources based on priority.
- Adding various kinds of proof in order to prove that users' query is not manipulated before it is delivered into their smart contract
- Providing complete compatibility with various social media apps such as Instagram, Telegram, etc. by adding dedicated data sources. Predicted sections includes:
    - Sending private post requests using user's encrypted access token that should be sent on behalf of user's identity.
    - Sending public post requests that doesn't matter the owner of access token such as number of views for an Instagram post.

- Wide marketing in order to establish enterprise and dedicated oracles for companies and organizations which are able to sell special and dedicated data.
- Developing an incentive-based decentralized oracle system for both people who wants to participate as a node and earn income and people who needs to have a completely decentralized smart contract.

2.9. Business Plans

There are several kinds of plan with which people can participate in Bridge's system and earn income from this project as below:

- Buying and investing on BRG tokens as shares of a high potential project
- Trading BRG tokens in exchanges after listing
- Selling dedicated and special data for BRG tokens as enterprise oracles
- Participating in Bridge's decentralized oracle system as a node and earn incentive rewards

3. Conclusion

Smart contracts are going to change businesses and automate activities of an organization; however, without feeding information of real world into smart contracts, they could only be utilized as a tool for tokenizing shares of a company. Thus, a technical system is needed through which real-world data is injected into smart contracts that are deployed on decentralized blockchain networks. This technology is *Oracle* system.

TRON network claims to have the best platform for developing Dapps and TRON foundation is promoting TRON network, continuously in order to attract more and more developers to increase the number of Dapps on the network and subsequently widen its ecosystem and community. Of course, TRON network stands in second place of blockchains ranking from number of Dapps viewpoint with almost 750 Dapps which is a good overall success; however, in Ethereum blockchain that stands in the first place of blockchains ranking, more than 3000 Dapps have been established and such a difference between number of Dapps of first and second placed blockchains is egregious. Such a big gap originates from an important deficiency which is felt significantly in TRON network. There should be a practical solution to fill the gap, based on which TRON is enabled to outpace Ethereum and stands on first place of blockchains ranking. The solution is providing a public oracle system. Unfortunately, there is no public oracle system in TRON network, thus public users cannot participate in TRON ecosystem as an active user and from ordinary users' point of view, TRON network is only a tokenization ecosystem. It can be easily said that creating Dapp for an ordinary user without a public oracle service is almost impossible.

As declared at the beginning of the context, in all over the world small businesses are more important than strong organizations and great economic enterprises. Thus, providing an infrastructure for attracting small businesses and implementing their businesses in TRON blockchain is critical and vital for progression and expansion of the network. Let's discuss an analogy in web-based projects. As emerging data centers and server service providers together with ready-to-use content manager systems including WordPress, Joomla, etc. that led to spread of implementing small businesses on internet extensively, creating a public oracle system on TRON network will have a similar effect and cause the ordinary users to be able to implement their local and small businesses on blockchain, as well and benefit from peer to peer payment methods of TRON network using native coin and tokens, adaptively.

Bridge project is the first ever public oracle system which has been developed to solve such issue and provide such services.