

The Introduction of about B-WSM

1. Introduction to the system

1.1 Here I open source the code of our system, but only for the reference of researchers. The copyright of the open source project and introduction in this paper belongs to the author, and it is forbidden for others to copy the author's labor achievements.

1.2 The system is mainly designed and developed for web service governance model based on block chain technology, and is also designed and developed based on the paper. The web service governance model we designed (**B-WSM: A decentralized web services governance model based on blockchain**) based on the idea of block chain is composed of several B-nodes, and the nodes are all equal. Each node stores the complete data of the model, and the data is distributed storage. The nodes communicate with each other through P2P, and the private data communicated between nodes adopts SHA256 encryption algorithm. According to the different needs of users, the model realizes the joining and exiting of nodes, the joining and exiting of Web services and the acquisition of Web services. This model realizes the decentralization of Web service governance and solves the problem of lack of trust in Web service governance. In addition, the model is mainly developed based on Java, HTML, javascript and other programming languages, and the data is stored in the form of JSON array.

1.3 The system is implemented in the design way of front and back end separation. The back-end code is implemented in Java, and the front-end code is implemented in HTML, javascript, jquery and other programming languages. The backend development tool of the system is Eclipse, the front-end development tool is Hbuilderx, and the Java JDK environment is 1.8.131.

1.4 The code of the open source system includes a complete system of Web services governance model based on blockchain technology. There are raw project files on the eclipse backend, front-end code files, and JAR files packaged with Eclipse's Maven.

1.5 The data of the three tables (NodeInfo, ServiceUserInfos, and WebServicesInfo) of the model structure stored by each node is stored in json arrays in the corresponding TXT file in the TXT folder of the local C drive (the location of the file can also be used) Set in code).

1.6 The model system needs to deploy relevant environment to run at the initial stage. In the initial stage, a Chuangshi node (the first node in the model) must be set in the model system to run normally. Then, if a new node wants to apply for joining, it can send the application to Chuangshi Node. You can send a request to any node in the model if a new node wants to join. With the addition of nodes, the model becomes a circular structure.

2. Introduces the architecture of the model system

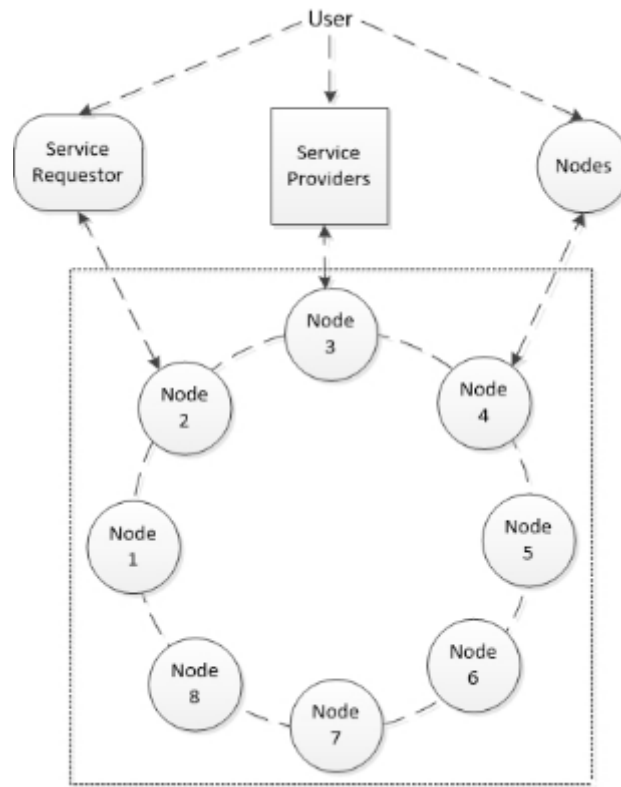


Fig.1 Model structure

This model is a decentralized structure in which each node holds the same information data, including: 1) node information table, 2) Web service information table, and 3) service provider account information table. The nodes send and receive messages through P2P communication. When nodes join or quit, service release or quit, and service provider account registration, the nodes send and receive messages, and the nodes add or delete relevant data from the local according to the message type, so as to ensure the consistency of node information in the model. The privacy data of communication is encrypted with SHA256 asymmetric algorithm to improve data security and prevent data leakage. According to the different needs of users, we designed three identities: node, service provider and service requester. The model is initially composed of a founding B-node, and then users who fit the deployment environment of the model can choose node identity, service provider identity, or service requester identity according to their requirements, so as to obtain the corresponding identity permissions. Here are three identities in the model structure.

3. Introduce the role of the three identities in the model system

3.1 Node identity

Each node has the same structural characteristics and records the same data information, including all node information tables, service provider account information tables and Web service information tables in the model.

Table* *1 Node information table (NodeInfo)

| *Fields* | *Description* |
|-----------------|---|
| nodename | Node Name |
| flag | Online/offline |
| port | Port number for opening websocket connections |
| ip | ip address of the node |
| jointime | The creating time of the node |

Table* *2 Service provider account information table (ServiceUserInfos)

| *Fields* | *Description* |
|-----------------|---|
| account | Accounts registered by service providers |
| password | Password for the service provider account (encrypted with SHA256) |

Table* *3 Web service information table (WebServiceInfo)

| *Fields* | *Description* |
|------------------------|--|
| belongtoNodeip | The ip address of the node to which the service belongs |
| belongtoServiceaccount | Account of the service provider to which the service belongs |
| inputandoutputdesc | Description of the input and output of the service |
| methodname | Method name of the service |
| serviceCode | The authentication code of the service, the credentials when joining or exiting the service, encrypted with SHA256 |
| serviceName | Service Name |
| serviceaddress | Access the address where the service is located |
| serviceexplain | Description of the service method |
| servicetype | Service Type |

Node identity has six function, respectively is: 1) the user as the node join or quit model, 2) node can release or quit the web service, 3) node to release on their service is a key to exit the permissions, node once the exit of the service will also exit model, 4) to get the release on his all web services, 5) Access to all nodes and Web services in the current model, and 6) P2P communication with other nodes in the model.

3.2 Service provider identity and requester identity

Service provider identity has three functions, namely: 1) users register accounts, log in accounts to become service providers, and can freely publish or cancel Web services on nodes in the model; 2) users can obtain all published Web services of their own accounts; 3) users can obtain all nodes and Web services in the current model.

What the service requester identity does: Users can enter retrieval information into the model and find the desired target Web service information.

4. How does the model system work

4.1 Deployment of model systems

The basic environment of the experiment of this model needs to configure the Java JDK environment. Our JDK environment is JDK1.8.131. And the environment is running on 4 machines in the same LAN, 4 host systems are windows10. The IP addresses of the four hosts are 1:192.168.1.3, 2:192.168.112.128, 3:192.168.112.129, and 4:192.168.112.130 respectively.

4.2 Use eclipse development tools to set up genesis nodes and file storage locations in the model system

Into the com. WebServiceModeBasedonBlockchain. MSG packets under Constant. Java Settings file (constants).

NodeInfo_FilePath: specifies the path for saving the node information table (NodeInfo table)

ServiceInfo_filePath: saves the Web service information table (ServiceUserInfos table)

SERVICE_USERINFOS: Directory where the service provider account information table (WebServiceSInfo table) is saved

FISTNODE_IP: IP address of genesis node

4.3 Package your project with Maven using Eclipse

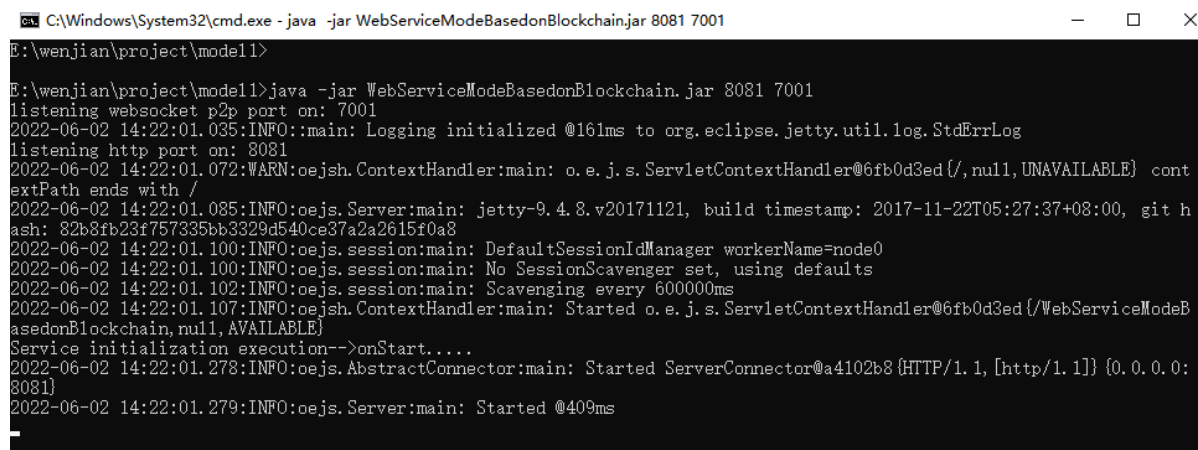
Right click the project -> run as-> Maven install-> Copy in the project target folder generated lib folder and WebServiceModeBasedonBlockchain jar file -> Put it in a folder on your local computer->Front-end ServiceModeBasedonBlockchain also in the same folder. As shown in Fig.2:

| 名称 | 修改日期 | 类型 | 大小 |
|------------------------------------|----------------|---------------------|-------|
| lib | 2022/6/2 14:12 | 文件夹 | |
| ServiceModeBasedonBlockchain | 2022/6/2 14:12 | 文件夹 | |
| WebServiceModeBasedonBlockchain... | 2022/6/2 14:08 | Executable Jar File | 57 KB |

Fig 2

4.4 Project to run the model system

Terminal command CMD into the project directory, then run the command `Java -jar WebServiceModeBasedonBlockchain.jar` (Fig 3), 7001 8081 8081 for HTTP protocol port here, 7001 is the Websocket port. Then enter `ServiceModeBasedonBlockchain` file under the view folder in the browser open `FunctionSelection`. The HTML file (e.g., Fig. 4). Here, I open the file with Google browser and the page is displayed (as shown in Fig 5), where Join as a node is entered as a node, Publish a service is entered as a service publisher. Join as a Service Requester is entered as a Service requester.



```
C:\Windows\System32\cmd.exe - java -jar WebServiceModeBasedonBlockchain.jar 8081 7001
E:\wenjian\project\model1>
E:\wenjian\project\model1>java -jar WebServiceModeBasedonBlockchain.jar 8081 7001
listening websocket p2p port on: 7001
2022-06-02 14:22:01.035:INFO::main: Logging initialized @161ms to org.eclipse.jetty.util.log.StdErrLog
listening http port on: 8081
2022-06-02 14:22:01.072:WARN:oejsh.ContextHandler:main: o.e.j.s.ServletContextHandler@6fb0d3ed{/ ,null,UNAVAILABLE} contextPath ends with /
2022-06-02 14:22:01.085:INFO:oejs.Server:main: jetty-9.4.8.v20171121, build timestamp: 2017-11-22T05:27:37+08:00, git hash: 82b8fb23f757335bb3329d540ce37a2a2615f0a8
2022-06-02 14:22:01.100:INFO:oejs.session:main: DefaultSessionIdManager workerName=node0
2022-06-02 14:22:01.100:INFO:oejs.session:main: No SessionScavenger set, using defaults
2022-06-02 14:22:01.102:INFO:oejs.session:main: Scavenging every 600000ms
2022-06-02 14:22:01.107:INFO:oejsh.ContextHandler:main: Started o.e.j.s.ServletContextHandler@6fb0d3ed{/WebServiceModeBasedonBlockchain,null,AVAILABLE}
Service initialization execution-->onStart....
2022-06-02 14:22:01.278:INFO:oejs.AbstractConnector:main: Started ServerConnector@a4102b8 [HTTP/1.1, [http/1.1]] {0.0.0.0:8081}
2022-06-02 14:22:01.279:INFO:oejs.Server:main: Started @409ms
```

Fig 3

> wenjian > project > model1 > ServiceModeBasedonBlockchain > view



















| 名称 | 修改日期 | 类型 | 大小 |
|---|-----------------|-----------------------|------|
|  AddNode.html | 2022/4/21 16:09 | Microsoft Edge HTM... | 4 KB |
|  AddServices.html | 2022/4/21 16:09 | Microsoft Edge HTM... | 7 KB |
|  AllNodesForNode.html | 2022/4/21 16:09 | Microsoft Edge HTM... | 6 KB |
|  AllNodesForService.html | 2022/4/22 10:30 | Microsoft Edge HTM... | 5 KB |
|  AllNodesForTour.html | 2022/4/21 16:33 | Microsoft Edge HTM... | 3 KB |
|  AllServicesInfosForNode.html | 2022/4/21 16:11 | Microsoft Edge HTM... | 6 KB |
|  AllServicesInfosForService.html | 2022/4/21 16:13 | Microsoft Edge HTM... | 6 KB |
|  AllServicesInfosForTour.html | 2022/4/21 16:16 | Microsoft Edge HTM... | 6 KB |
|  FunctionSelection.html | 2022/4/22 10:24 | Microsoft Edge HTM... | 4 KB |
|  LogoutServicesForService.html | 2022/4/21 16:17 | Microsoft Edge HTM... | 4 KB |
|  MyAllServicesForNode.html | 2022/4/21 16:19 | Microsoft Edge HTM... | 4 KB |
|  MyAllServicesForService.html | 2022/4/22 10:32 | Microsoft Edge HTM... | 7 KB |
|  NodeMeServiceDetailsForNode.html | 2022/4/21 16:21 | Microsoft Edge HTM... | 5 KB |
|  ServiceDetailsForNode.html | 2022/4/21 13:06 | Microsoft Edge HTM... | 5 KB |
|  ServiceDetailsForService_Allserviceinfos.html | 2022/4/21 14:02 | Microsoft Edge HTM... | 5 KB |
|  ServiceDetailsForService_Me.html | 2022/4/21 14:15 | Microsoft Edge HTM... | 5 KB |
|  ServiceDetailsForTour.html | 2022/4/21 11:13 | Microsoft Edge HTM... | 5 KB |
|  ServiceMain.html | 2022/4/21 16:23 | Microsoft Edge HTM... | 5 KB |

Fig 4

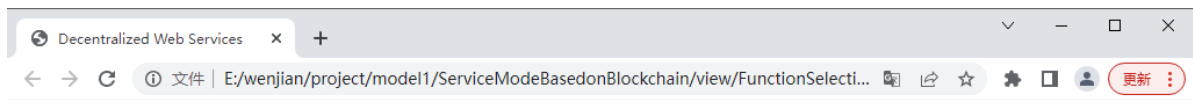


Fig 5

After running, three tables (nodeInfo. TXT, ServiceUserInfos. TXT, webServicesInfo. TXT) will appear in the local storage address for recording model data, among which the information of The Genesis node needs to be set in the nodeInfo. TXT table. Flag is the online/offline state, IP is the IP of the Chuangshi node, nodename is the name of the Chuangshi node, port is the websocket port of the Chuangshi node, and JoinTime is the time when the node joins the model.

```
[
  {
    "flag": "1",
    "ip": "192.168.1.3",
    "nodename": "node1",
    "port": "7001",
    "jointime": "2022-07-08 11:39:29"
  }
]
```

4.5 Node set

Run the model system project for the remaining hosts 2, 3 and 4 respectively. Then host 2,3 were set as nodes in the model, and host 4 simulated the process of adding nodes.

4.6 Nodes to join

1) Host 4 (IP: 192.168.112.130) as the node (click Join as a node in Fig 6) - > Enter the node list information interface of the model system (see Fig.7)

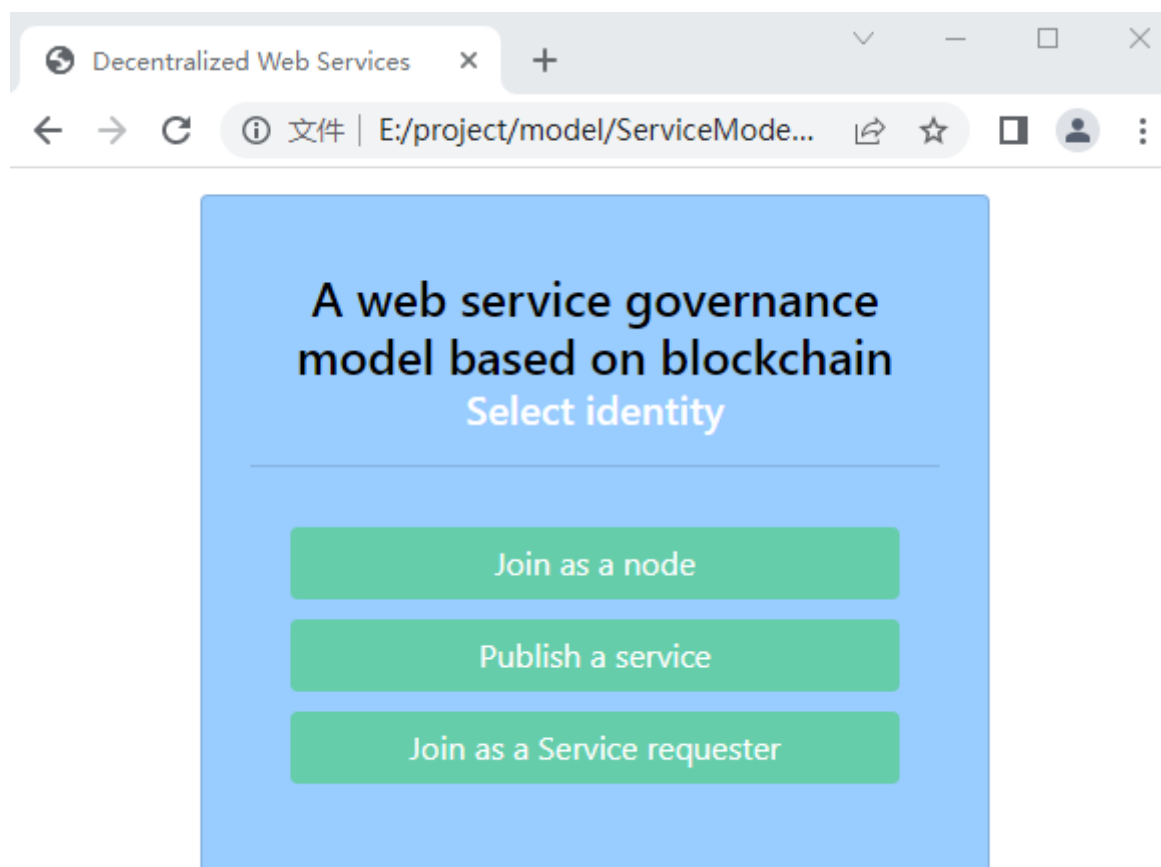


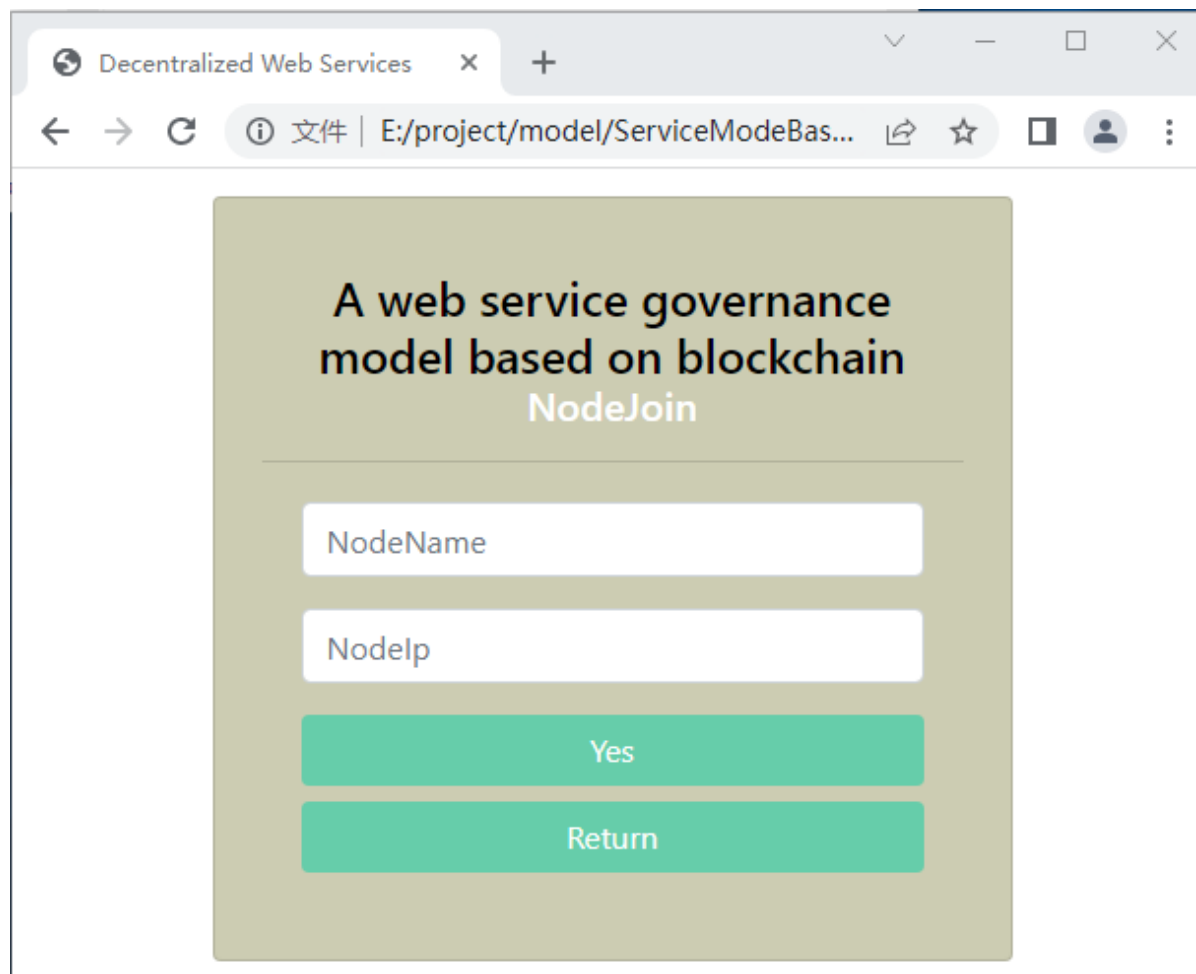
Fig 6

| NodeInfos AllServiceInfos MyServiceInfosForNode | | | | |
|---|------|---------------|----------------|---------------------|
| NodeList | | Identity:node | | Cancel |
| Ip | Port | Nodename | NodeFunction | Create time |
| 192.168.1.3 | 7001 | node1 | Join as a node | 2022-07-08 11:39:29 |
| 192.168.112.128 | 7001 | node2 | Join as a node | 2022-07-08 16:21:26 |
| 192.168.112.129 | 7001 | node3 | Join as a node | 2022-07-08 17:25:58 |

Fig 7

2) In Figure 7, Nodeinfos is the information list interface of all nodes in the node identity model, AllServiceInfos is the information list interface of all Web services in the model, MyServiceInfosForNode is the Web service added under this node, NodeList is a list of nodes, Identity: Node refers to access as a node, Cancel refers to exit as a node, Ip refers to the Ip address of a node, Port refers to the communication Port enabled between nodes, Nodename refers to the node name, NodeFunction refers to the joining or exiting functions that can be performed on a node, Join as a node refers to Join as a node, Create time is the create time of the node.

3) You can see that there are already three nodes in the model, and host 4 is not yet a node in the model. Host 4 can select any node in the model to send a request to join as a node. Here, I select node 2 (IP: 192.168.112.128) to send the Join request, click Join as a node after node 2, and enter the interface of input node information (as shown in Fig 8).



The screenshot shows a web browser window with the title 'Decentralized Web Services'. The address bar displays 'E:/project/model/ServiceModeBas...'. The main content area features a form titled 'A web service governance model based on blockchain NodeJoin'. The form includes two text input fields labeled 'NodeName' and 'NodeIp'. Below these fields are two prominent green buttons labeled 'Yes' and 'Return'.

Fig 8

4) In Fig 8, NodeName is the NodeName to be entered, and NodeIp is the IP address of the node to be entered (also the ipv4 address of the localhost). In this case, NodeName enters node4 and NodeIp enters 192.168.112.130. Then click Yes to send the join request to node 2. After receiving the node joining request, node 2 stores the information about the new node 4 to the local computer and forwards the information to other nodes. Other nodes also store the data of the new node to the local computer. Node 2 sends a message to host 4 agreeing to join the new node.

4.7 Node exit

Here, we select Node4 to simulate node exit, and node4 enters The node information list interface (as shown in Fig 9). Selecting any of The node logout indicates that Node4 sends exit request to all nodes in The model, and other nodes receive exit notification. Update data Deletes node 4 data from, for example, local data.

NodeInfos

AllServiceInfos

MyServiceInfosForNode

NodeList

Identity:node

Cancel

| Ip | Port | Nodename | NodeFunction | Create time |
|-----------------|------|----------|-----------------|---------------------|
| 192.168.1.3 | 7001 | node1 | The node logout | 2022-07-08 11:39:29 |
| 192.168.112.128 | 7001 | node2 | The node logout | 2022-07-08 16:21:26 |
| 192.168.112.129 | 7001 | node3 | The node logout | 2022-07-08 17:25:58 |
| 192.168.112.130 | 7001 | node4 | The node logout | 2022-07-15 01:09:53 |

Fig 9

4.8 Publication of Web services

1) Here we select Publish a Service in Fig 6 to access the model as a service provider. Enter the service provider account registration/login information interface (Fig 10). Enter the account and password for registration. I register an account as Sunny's account, and the password of registration information is encrypted with SHA256. After successful registration, select Login to Login and enter the interface of model node information list of the service provider (as shown in Fig 11).

A web service governance model
based on blockchain

Login Or Register Account For Service

please input account

please input password

Login

Register

Return

Fig 10

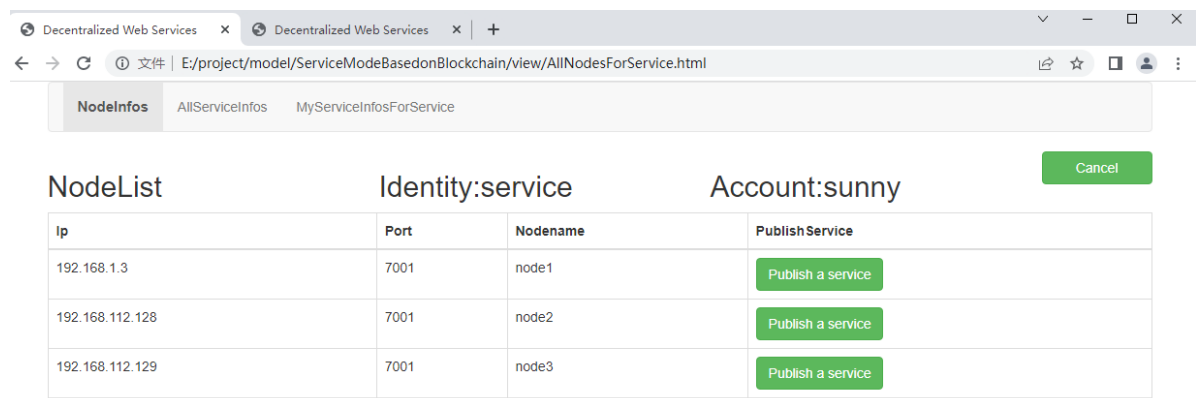


Fig 11

2) In Fig 11, Nodeinfos is the information list interface of all nodes in the service provider model, AllServiceInfos is the information list interface of all Web services in the model, MyServiceInfosForService is the information list interface of web services added under this account, NodeList is a list of nodes, Identity: service refers to access as a service provider, Account: Sunny is the account name of the service provider, Cancel is the identity exit, Ip is the Ip information of the node, Port is the communication Port enabled between nodes, Nodename is the name of the node, ServiceJoin is the account to join the Web service function on the node. Publish a service publishes a Web service.

3) Publish a Service after either node is selected to Publish the Web service. Here I choose Node3 to publish the Web service. Click Publish a Service to enter the input information screen for publishing the Web service (see Figure 12). In figure 12, where ServiceName is the name of the web service, ServiceType is the type of the web service, ServiceAddress is the address of the interface where the web service is called, MethodName is the MethodName of the web service, MethodExplain is a service method that InputAndOutputDescription is the input and output of the service method, ServiceCode refers to the service authentication code, use SHA256 encryption, exit when web services need to input validation, Yes: Publishes the Web service. Return Returns to the previous page. After clicking Yes, the host where the account belongs sends a Web service joining request to node 2. After receiving the Web service joining request, node 2 stores the web service information to the local data and forwards the web service joining information to other nodes in the model, and other nodes store the new service information locally. Node 2 sends a notification to the host that sends the Web service. After receiving the notification, the host updates data to complete the joining of the Web service. The Web services we referenced in this study were from www.webxml.com.cn website.

A web service governance model based on blockchain

PublishService

ServiceName

ServiceType

ServiceAddress

MethodName

MethodExplain

InputAndOutputDescription

ServiceCode

Fig 12

4) Click Yes to publish the Web service and publish it successfully. Enter the web service information interface published by this account (see Fig 13). In Fig 13, MyAllServicesForService is the published Web service interface for the service provider account. AllServiceList is a list of all Web services. NodeInfos is the interface for listing all nodes in the service provider's model. MyServiceList is a list of Web service information published by the account of the service provider. The search box on the left of Search is to enter the name of the service you want to search. Click Search to find the target service name (enter lowercase letters for fuzzy search). ServiceName is the ServiceName of all the web services displayed. BelongToNode refers to the node to which the published Web service belongs. BelongToAccount refers to the account to which the published Web service belongs. ServiceType indicates the type of the Web service. ServiceInfo indicates the

details of the service. The Service Logout under Logout indicates that The Web service is logged out.

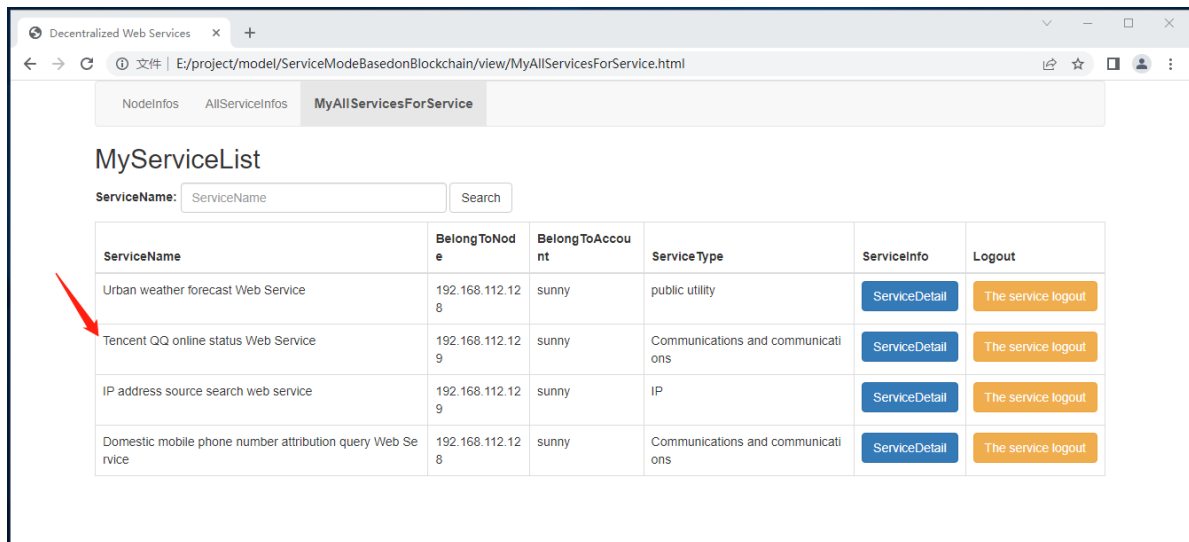


Fig 13

4.9 Web Service Exit

I used sunny account to log in to the interface in Fig 13 and exit the Tencent QQweb service in the list. Click The Service logout after The service. Enter the interface for inputting the service verification code (for example, Fig 14), where the service verification code uses SHA256 encryption. Enter the verification code and click Yes to submit the Web service. After exiting, the Fig 15 page is displayed, and the Web service information is no longer displayed. After the nodes in the model communicate, the data is synchronized and all nodes in the model no longer have the Web service information.

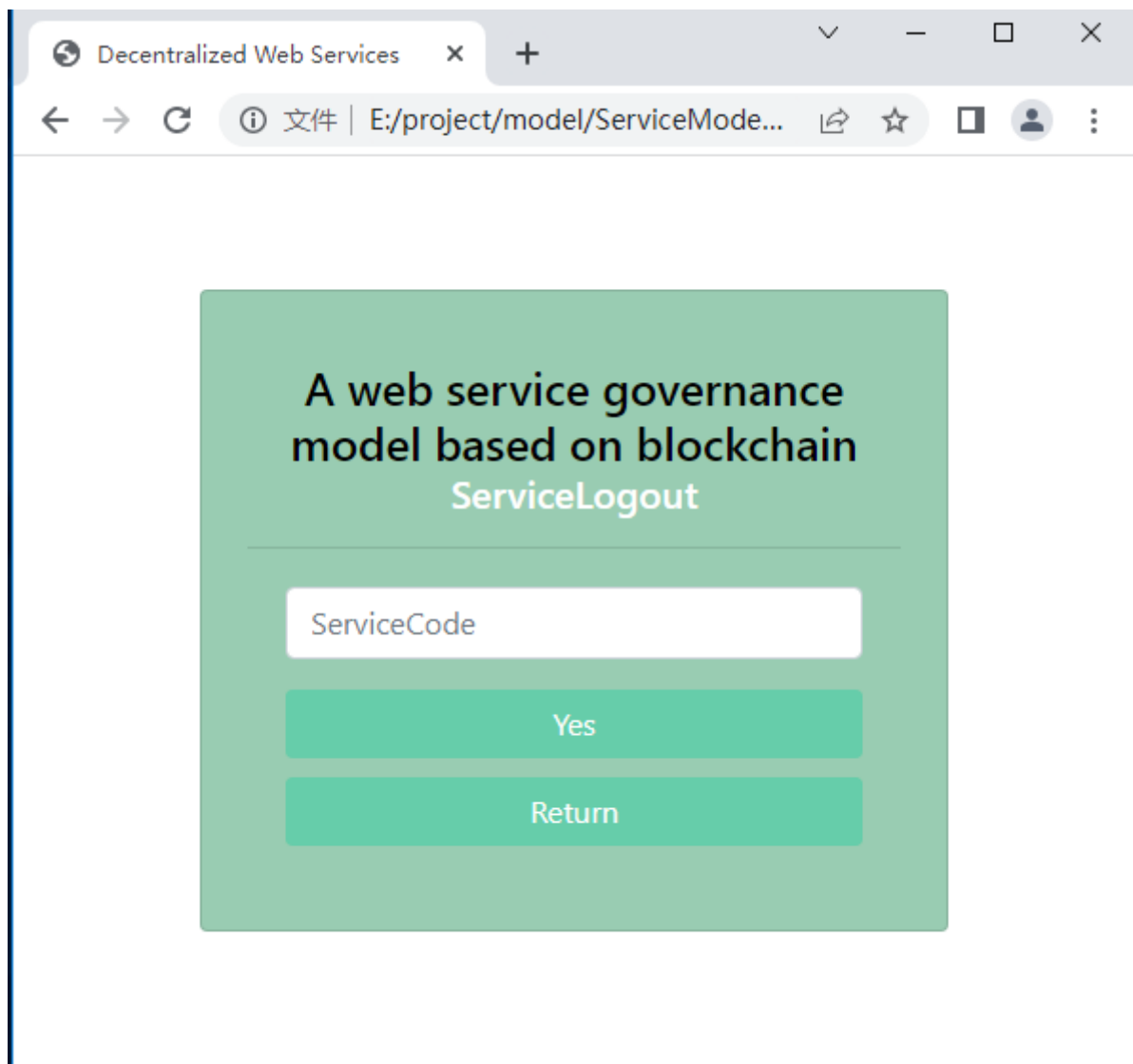


Fig 14

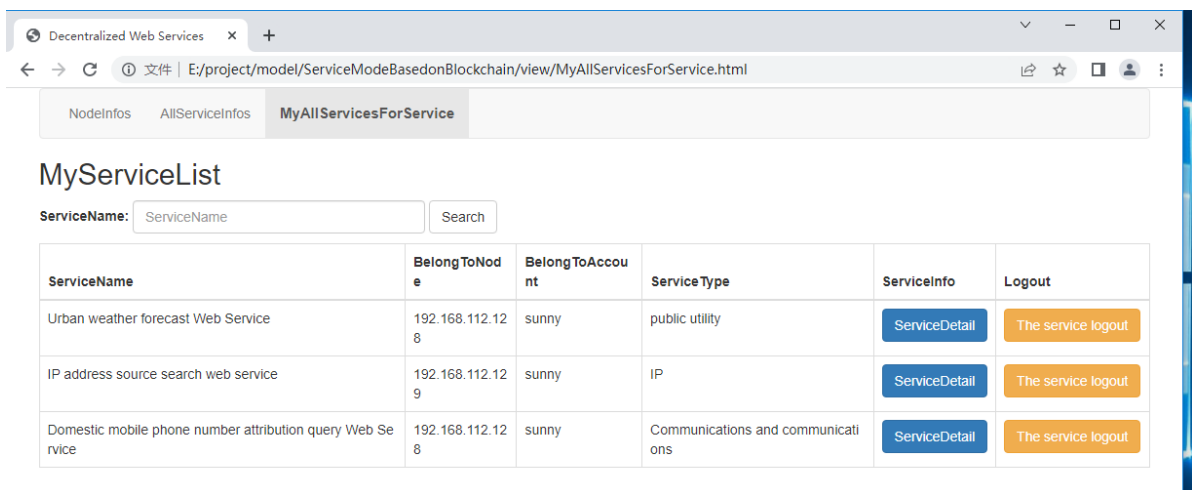


Fig 15

4.10 The service requester gets the Web service

I access the model with the identity of the service acquirer. Click Join as a Service Requester in Fig 16 to enter the model. Enter the list of all Web services information in the service acquirer's model (see Figure 17).

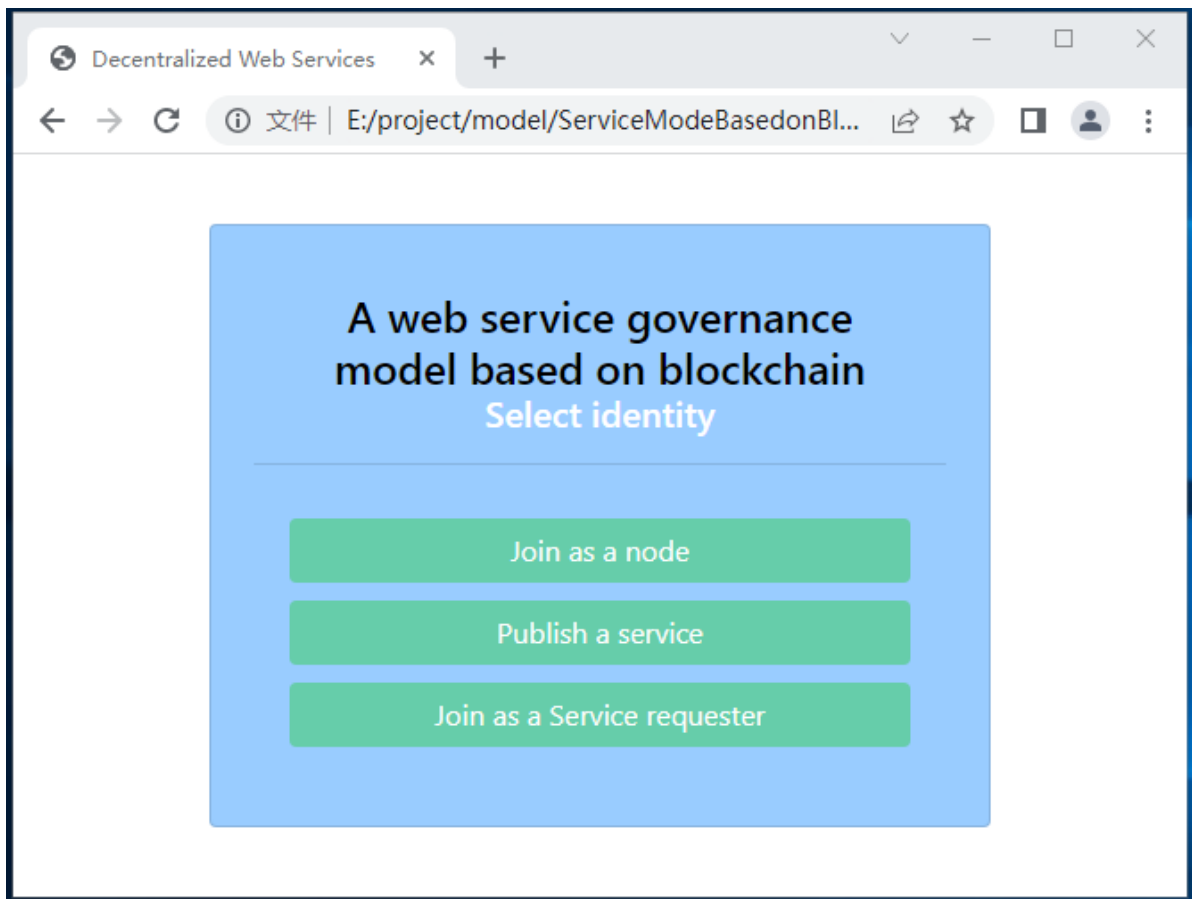


Fig 16

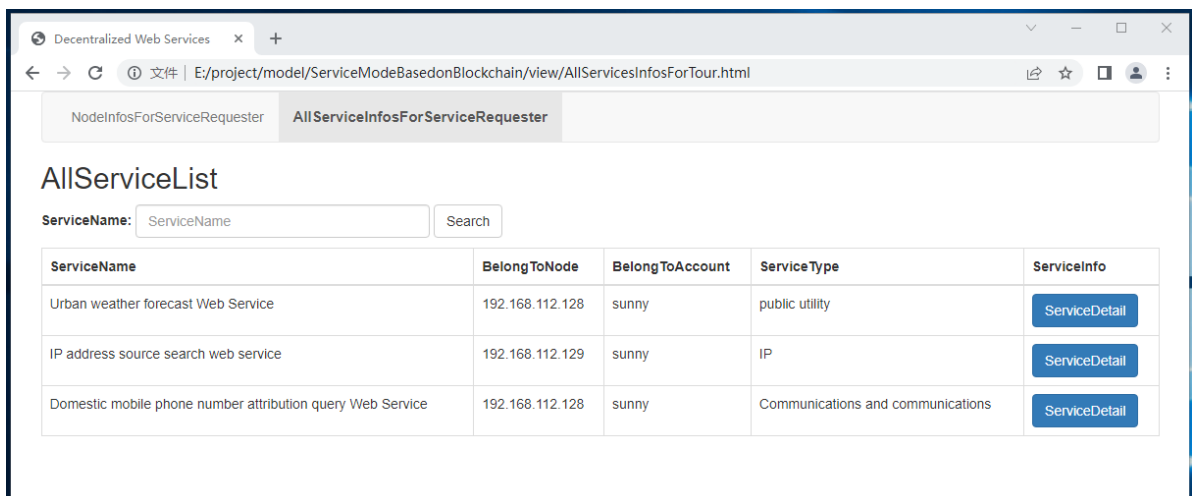


Fig 17

In Fig 17 NodeInfosForServiceRequester of them for the service request list of all the nodes in the model information interface, AllServiceInfosForServiceRequester is all the web services in the model of the service requester information list interface, AllServiceList column is all web service information, search the search box on the left is a type in the name of the service they want to retrieve, click sear Ch search the target ServiceName (enter lowercase letters for fuzzy search),ServiceName is the ServiceName of all web services displayed, BelongToNode means the node where the published web service belongs, and BelongToAccount means the account where the published web service belongs. ServiceType indicates the type of the Web service. ServiceInfo indicates the service details page. Service requesters can obtain the web service they want based on their requirements.

4.11 A decentralized Web services governance system

1) All nodes in the model are peer, recording the same structural data in the model, and there is no central node. Even if a single node fails, the model will work as long as there is one node in the model. The Web services on the non-working nodes are still on all nodes in the model. When a node exits or joins, the working nodes in the model will record the information data. When the non-working nodes work properly, the latest data in the model can still be obtained from the model.

2) There are four nodes in the model: 1,2,3,4. When node4 (IP: 192.168.112.130) does not work properly (as shown in figure 18), the web service on node4 is still available (as shown in figure 19).

| NodeInfos AllServiceInfos MyServiceInfosForNode | | | | |
|---|------|---------------|-----------------|---------------------|
| NodeList | | Identity:node | | Cancel |
| Ip | Port | Nodename | NodeFunction | Create time |
| 192.168.1.3 | 7001 | node1 | The node logout | 2022-07-08 11:39:29 |
| 192.168.112.128 | 7001 | node2 | The node logout | 2022-07-08 16:21:26 |
| 192.168.112.129 | 7001 | node3 | The node logout | 2022-07-08 17:25:58 |

Fig 18

| Decentralized Web Services x + 文件 E:/project/model/ServiceModeBasedonBlockchain/view/AllServicesInfosForNode.html | | | | |
|---|-----------------|-----------------|-----------------------------------|---------------|
| NodeInfos AllServiceInfos MyServiceInfosForNode | | | | |
| AllServiceList | | | | |
| ServiceName: <input type="text"/> ServiceName Search | | | | |
| ServiceName | BelongToNode | BelongToAccount | Service Type | ServiceInfo |
| Urban weather forecast Web Service | 192.168.112.128 | sunny | public utility | ServiceDetail |
| IP address source search web service | 192.168.112.129 | sunny | IP | ServiceDetail |
| Domestic mobile phone number attribution query Web Service | 192.168.112.128 | sunny | Communications and communications | ServiceDetail |
| Verification code image web service supports Chinese, letters and numbers | 192.168.112.130 | sunny | Text conversion picture | ServiceDetail |
| Verification code image web service | 192.168.112.130 | sunny | Text conversion picture | ServiceDetail |

Fig 19

3) Publish a new Web service on Node3 when node4 does not work properly (Fig 20):

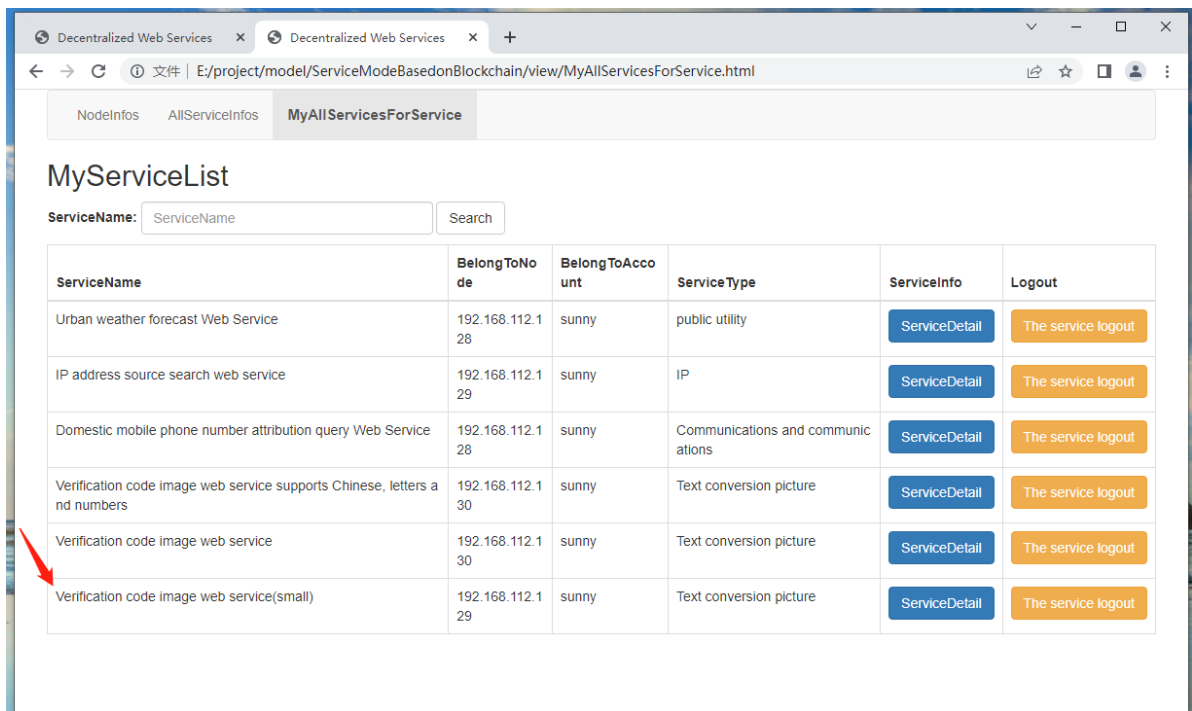


Fig 20

4. When Node4 works properly (as shown in Fig 21), host 4 accesses the model as a node. Node4 gets the latest data from the model, and the new Web service data is displayed in node4's Web service Information list interface (see Fig 22).

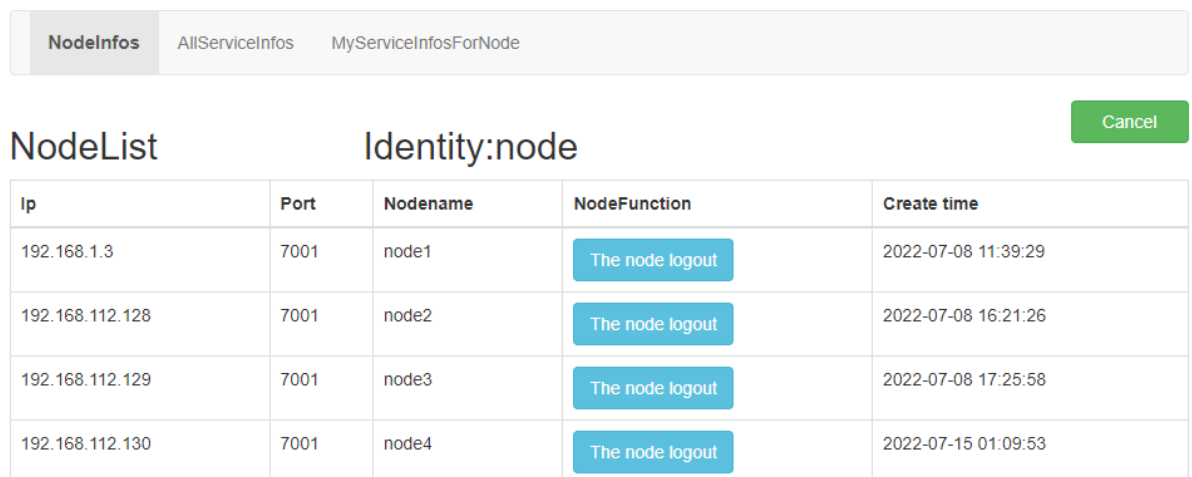


Fig 21

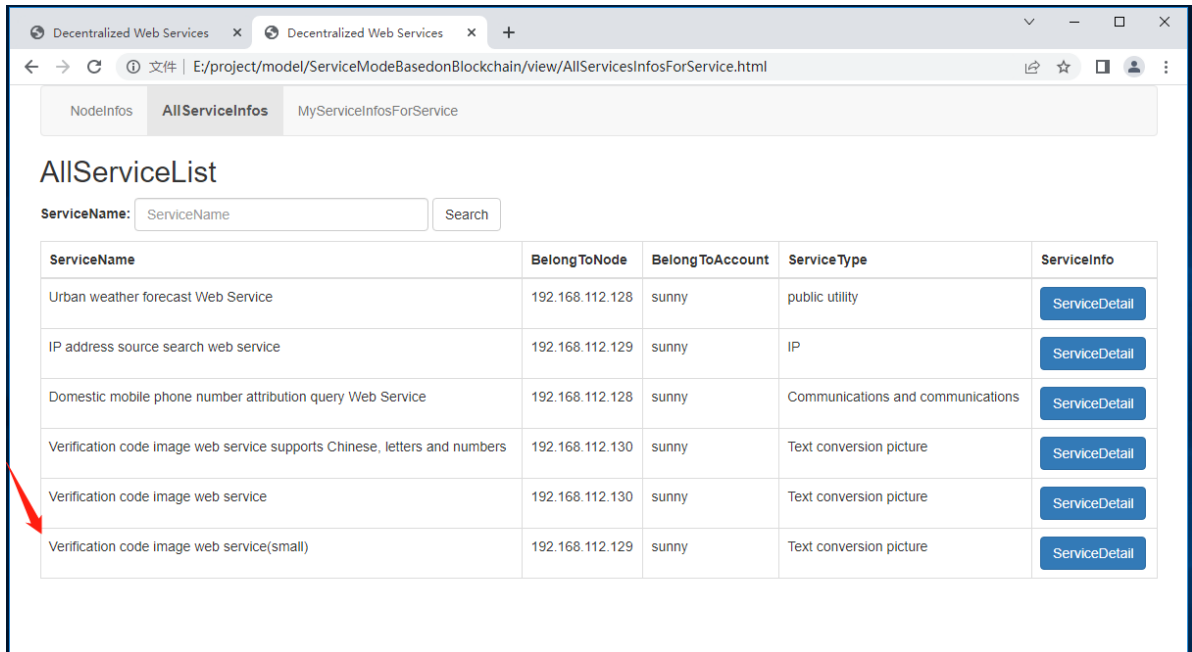


Fig 22

4.12 The node has one-click exit permission for Web services published on it

- 1) When a node exits the model, all Web services published on that node also exit the model.
- 2) There are two Web services on Node4 (IP: 192.168.112.130, as shown in Fig 23), and when node4 exits the model, the web services published on Node4 also exit the model (as shown in Fig 24).

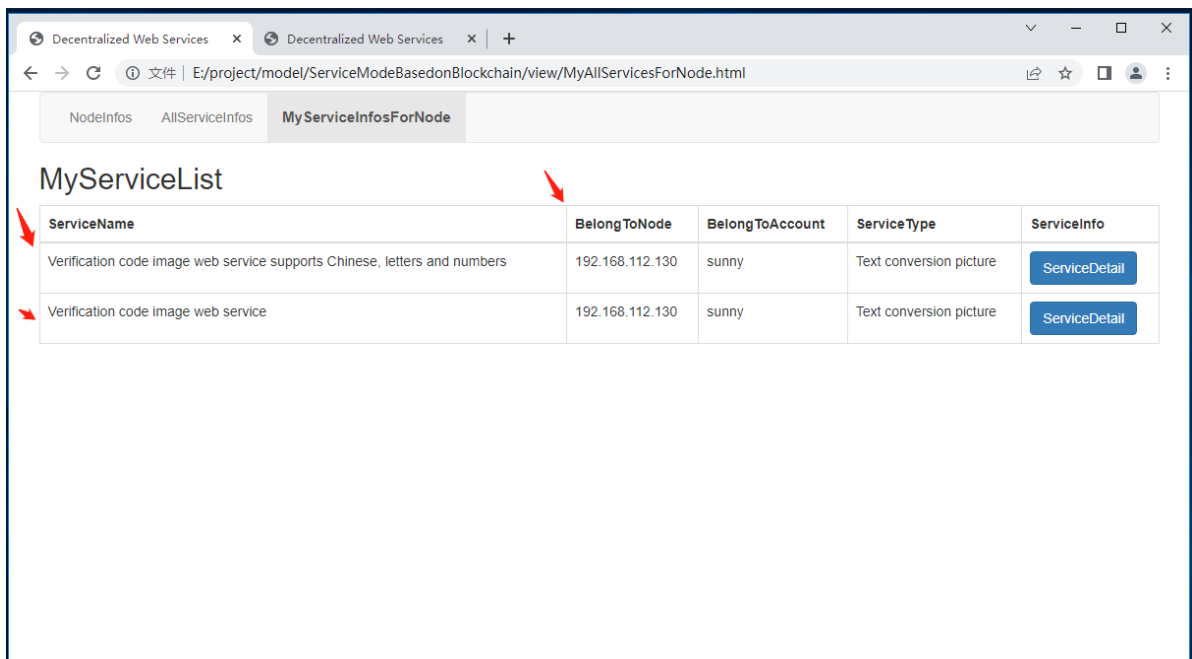


Fig 23

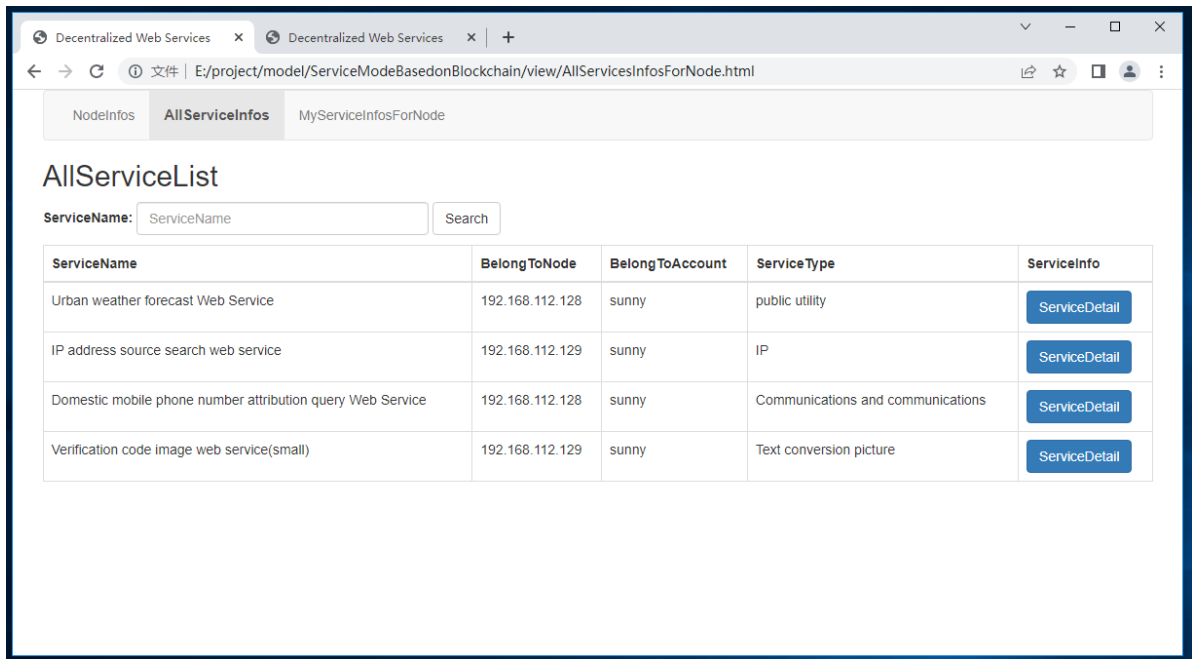


Fig 24

5. conclusion

Through experiments, the web service governance model based on block chain designed in this paper is verified, which solves the problems of decentralized governance of Web services and lack of trust, and the distributed storage of data improves the autonomy of service governance and data security, and proves the feasibility of the implementation of the model.