基于区块链技术的web服务治理模型系统介绍

1. 系统的介绍

1.1 在这里我把我们实现的系统的代码开源，但是只供研究者参考学习，本文所开源的项目和介绍的版权属于作者本人，禁止他人抄袭作者劳动成果。

1.2该系统主要是针对于基于区块链技术的web服务治理模型而设计开发的，也是依托论文而设计开发。而我们设计的基于区块链思想的web服务治理模型B-WSM.是由若干个B-node组成,节点之间都是对等的,每个节点都存储有模型的完整数据,数据采用分布式存储,节点之间通过P2P通信,节点间通信的隐私数据采用SHA256加密算法.模型根据用户不同的需求实现了节点的加入和退出,web服务的加入和退出以及web服务的获取.该模型实现web服务治理的去中心化,解决了web服务治理缺乏信任的问题。另外该模型主要是基于java，html，javascript等编程语言开发，数据存储以json数组的形式存储。

1.3 该系统是前后端分离的设计方式实现，后端代码用Java实现，前端代码用html，javascript以及jquery等编程语言实现。该系统的后端开发工具是eclipse，前端开发工具是Hbuilderx，java的jdk环境是1.8.131。

1.4 开源的系统的代码包括基于区块链技术的web服务治理模型的一整套系统。其中有eclipse后端的原始项目文件，前端的代码文件以及用eclipse的maven打包后的jar文件。

1.5 每个节点存储的模型结构的3个表（NodeInfo、ServiceUserInfos以及WebServiceSInfo3个表下文会进一步介绍）的数据都是以json数组的形式存储到本地的c盘txt文件夹下的对应txt文件中（存储的文件位置也可以在代码中进行设置）。

1.6 模型系统初始需要部署相关环境才能运行，初始中模型系统中必须设置一个创世节点（模型中的第一个节点），模型才能正常运行，随后如果有新节点要申请加入，可以向创世节点发送加入申请。后面有新节点要申请加入可以向模型中的任意一个节点发送加入请求。随着节点的加入模型就成为了一个环状模型结构。

2. 介绍模型系统的体系结构

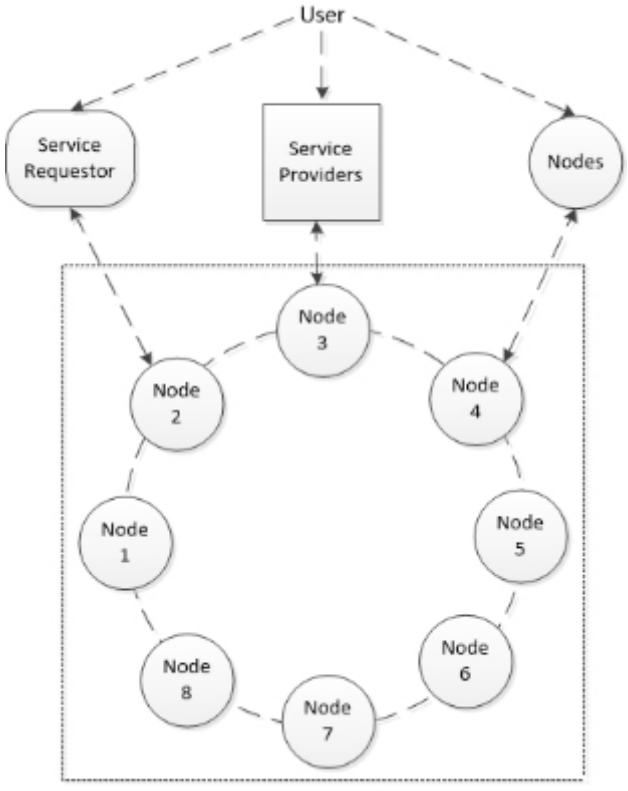

Fig.1 Model structure

这个模型是个去中心化的结构，每个节点保存有同样的信息数据，这些信息包括：1)节点信息表，2)web服务信息表，3)服务提供者账户信息表.节点之间通过P2P通信来收发消息，当有节点加入或退出、服务发布或退出以及服务提供者账号注册时，节点间会收发消息，节点会根据消息类型把相关数据从本地加入或者删除，从而保证模型中节点信息的一致性. 通信的隐私数据采用SHA256非对称算法加密从而提升数据的安全性防止数据泄露。我们根据不同需求的用户，设计了3种身份权限，节点，服务提供者，服务请求者。模型初始由一个创始B-node组成，之后符合模型部署环境的用户可以根据需求选择节点身份，服务提供者身份，或者服务请求者身份，从而获得对应身份的权限。下面要对模型结构中的三种身份进行介绍。

3. 介绍一下模型系统中的3种身份的作用

3.1节点身份

每个DWSGN节点都有一样的结构特征，记录有相同的数据信息，记录有模型中的全部节点信息表，服务提供者账户信息表以及web服务信息表。

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 joining 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 (WebServiceSInfo)

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

节点身份有6个作用，分别是:1)用户作为节点加入或者退出模型，2)节点上可以发布或者退出web服务，3）节点对发布在自己上的服务具有一键退出权限，节点一旦退出其上的服务也将退出模型，4）可以获取发布在自己上的所有的web服务，5）可以获取当前模型中的所有节点和web服务，6）与模型中的其它节点进行P2P通信。

3.2 服务提供者身份和请求者身份

服务提供者身份有3个作用，分别是：1)用户注册账户，登陆账户成为服务提供者，可以自由的在模型中的节点发布或者注销web服务，2)可以获取自己账户的已发布的所有web服务，3)可以获取当前模型中的所有的节点和web服务。

服务请求者身份的作用：用户可以在模型中输入检索信息然后查找自己想要的目标web服务信息。

4. 介绍一下模型系统如何运行工作

4.1模型系统的部署

该模型的实验的基础环境需要配置java的jdk环境，我们的jdk环境是jdk1.8.131，并且环境是在同一个局域网的4台机器上运行，4台主机系统都是windows10，4台主机的ip分别是主机1:192.168.1.3,主机2:192.168.112.128、主机3:192.168.112.129，主机4:192.168.112.130。

4.2用eclipse开发工具对模型系统的创世节点和文件存储位置进行设置

进入到com.WebServiceModeBasedonBlockchain.msg包下的Constant.java(常量设置文件)中。

其中NodeInfo_FilePath：节点信息表(NodeInfo表)保存路径

ServiceInfo_filePath：web服务信息表(ServiceUserInfos表)的保存路径

SERVICE_USERINFOS：服务提供者账号信息表(WebServiceSInfo表)的保存路径

FISTNODE_IP：创世节点的ip

4.3用eclipse对项目进行maven打包

右击项目->run as->Maven install->复制在项目的target文件夹下生成的lib文件夹和WebServiceModeBasedonBlockchain.jar文件->放到本地电脑的一个文件夹下->前端文件ServiceModeBasedonBlockchain也放到同一个文件夹下。如Fig 2所示：

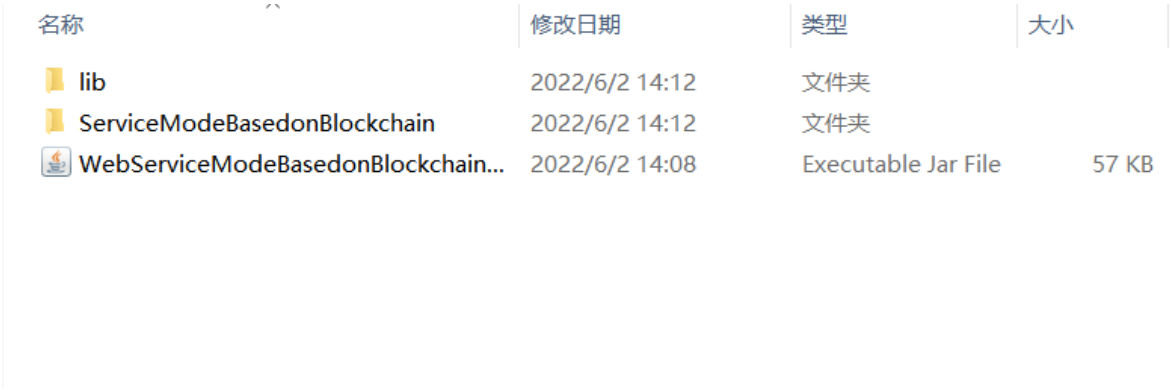

Fig 2

4.4运行模型系统的项目

终端指令cmd进入到项目所在目录，然后运行指令java -jar WebServiceModeBasedonBlockchain.jar 8081 7001（如Fig 3），在这里8081为http协议端口，7001为websocket端口。然后进入到ServiceModeBasedonBlockchain文件下view文件夹下用浏览器打开FunctionSelection.html文件（如Fig 4）。在这里我用google浏览器打开文件后页面展示（如Fig 5）,其中Join as a node是以节点身份进入，Publish a service是以服务发布者身份进入，Join as a Service requester是以服务获取者身份进入。

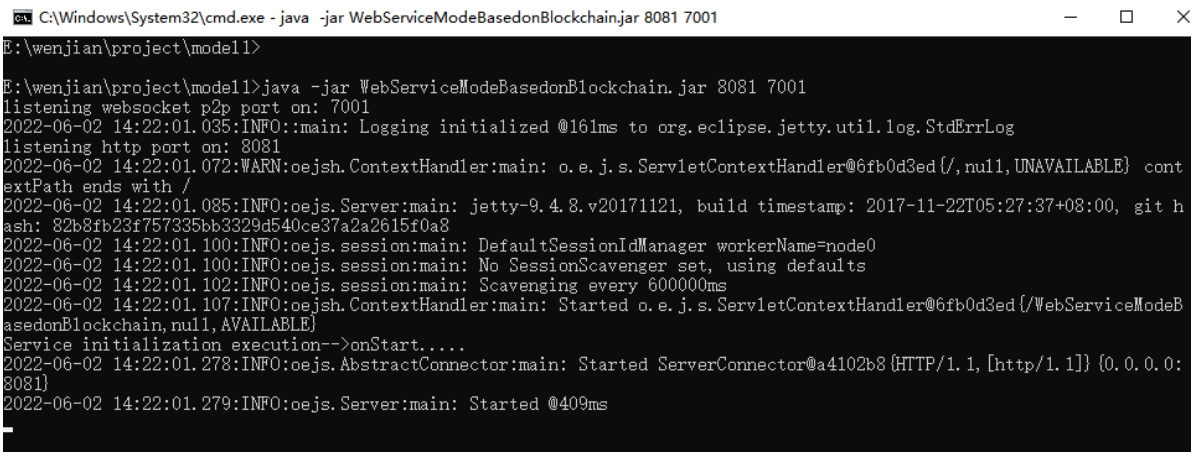

Fig 3

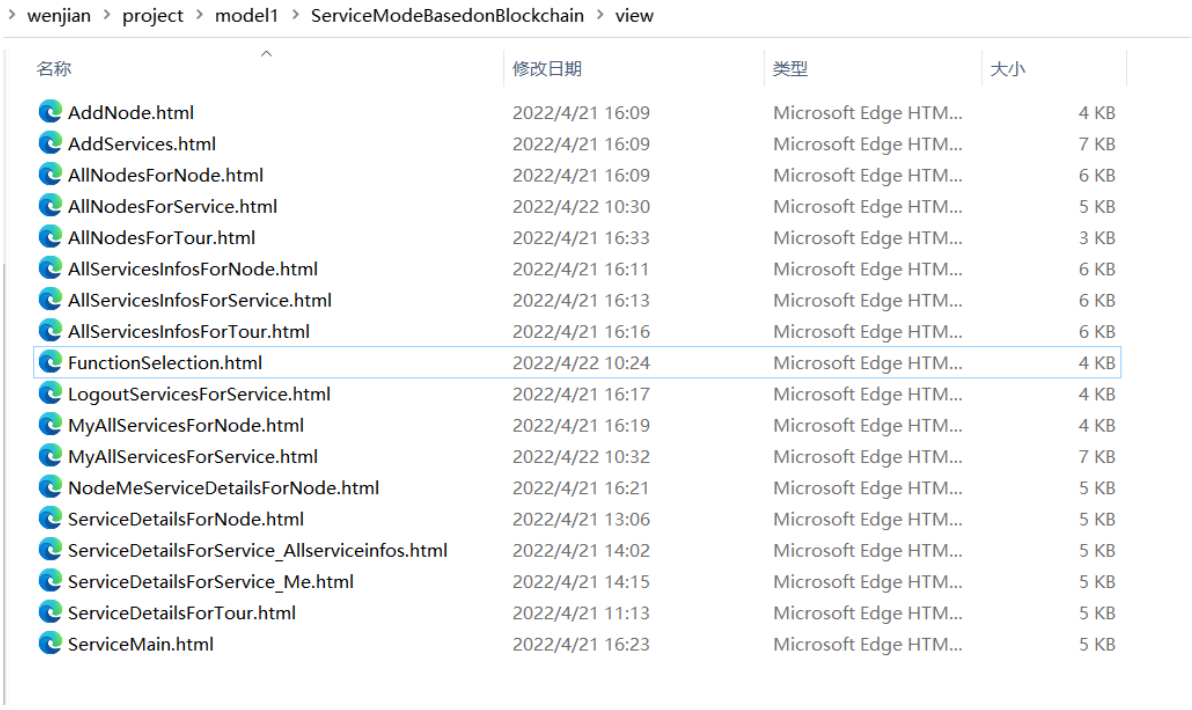

Fig 4

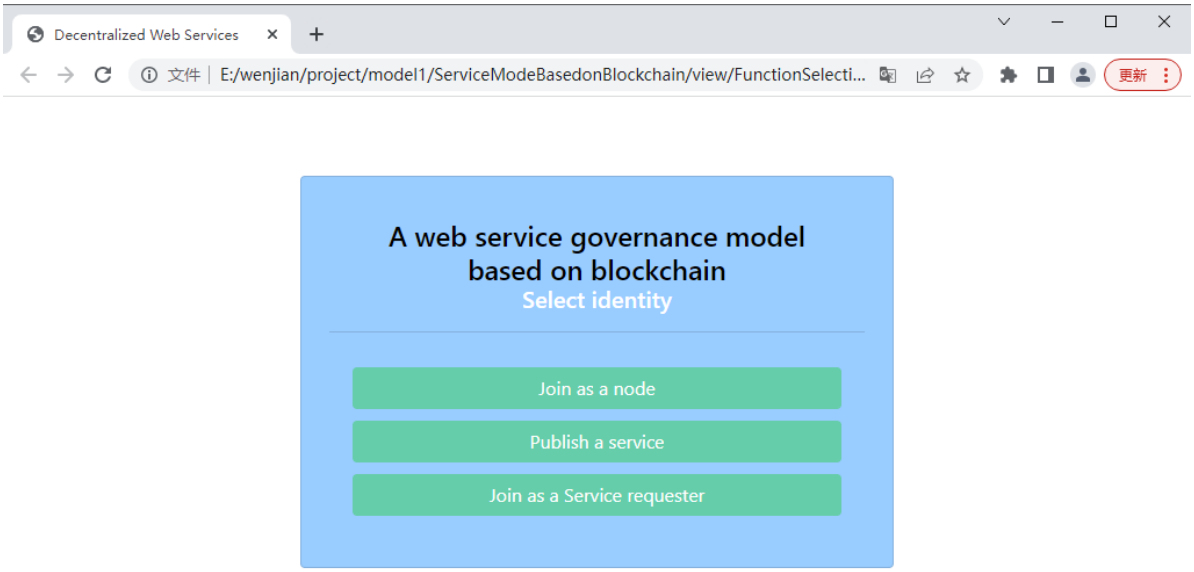

Fig 5

运行完后，对应记录模型数据的本地存储地址中会出现3个表（NodeInfo.txt，ServiceUserInfos.txt，WebServiceSInfo.txt），其中创世节点信息需要在NodeInfo.txt表中设置一下。flag是在线/离线状态，ip是创世节点的ip，nodename是创世节点的名字，port是创世节点的websocket端口,jointime是节点加入模型时的时间。

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

]
```


4.5节点设置

对剩下的主机2，3，4分别运行模型系统项目。然后设置主机2，3为模型中的节点，主机4模拟完成节点加入过程

4.6节点加入

1）主机4（ip：192.168.112.130）以节点身份（点击Fig 6的Join as a node）->进入到模型系统的节点列表信息界面（如Fig 7）

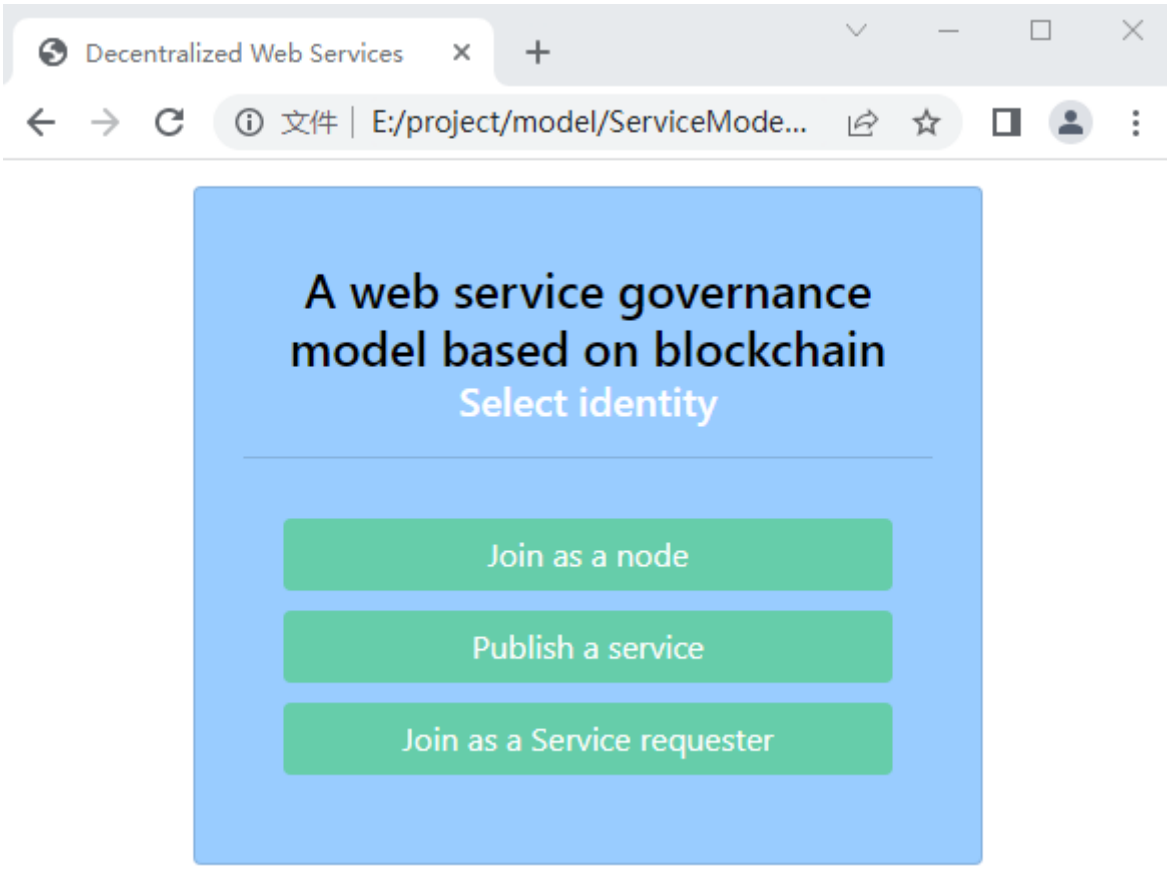

Fig 6

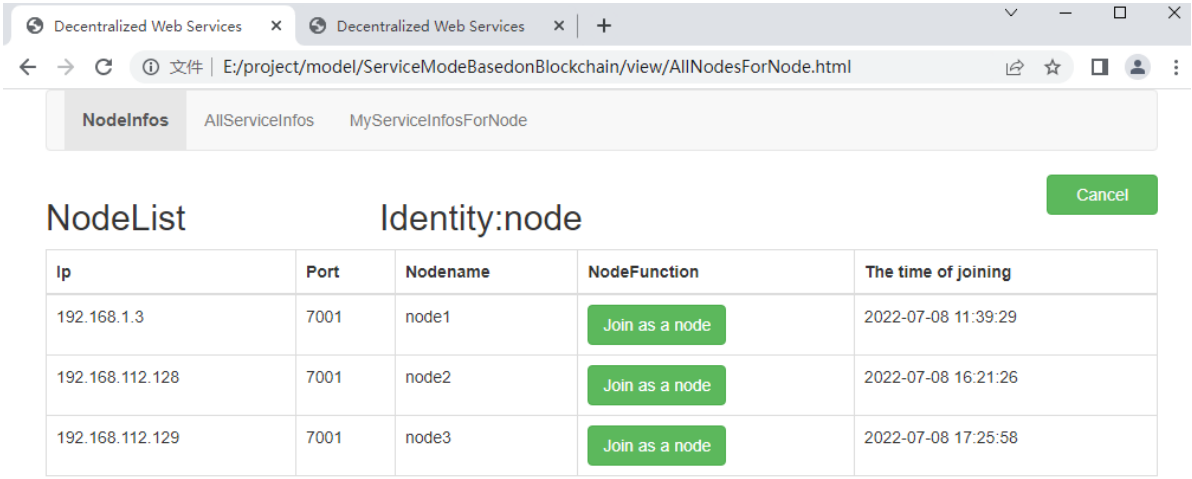

Fig 7

2）在Fig7中，其中Nodeinfos是对于节点身份模型中所有节点信息列表界面，AllServiceInfos是模型中所有web服务信息列表界面，MyServiceInfosForNode是该节点下加入的web服务，NodeList是节点列表，Identity：node是指以节点身份访问，Cancel是身份退出，Ip是节点的ip信息，Port是指节点间开启的通信端口，Nodename是节点名称，NodeFunction是节点当前能执行的加入或者退出功能，Join as a node是加入成为节点，The time of joining代表此节点加入模型的时间。

3）可以看到模型中已有3个节点，而主机4还没成为模型中的一个节点。主机4可以选择模型中的任意一个节点去发送加入成为节点的请求。在这里我选择节点2（ip：192.168.112.128）发送加入请求，点击节点2后的Join as a node，进入到输入节点信息界面（如Fig 8）。

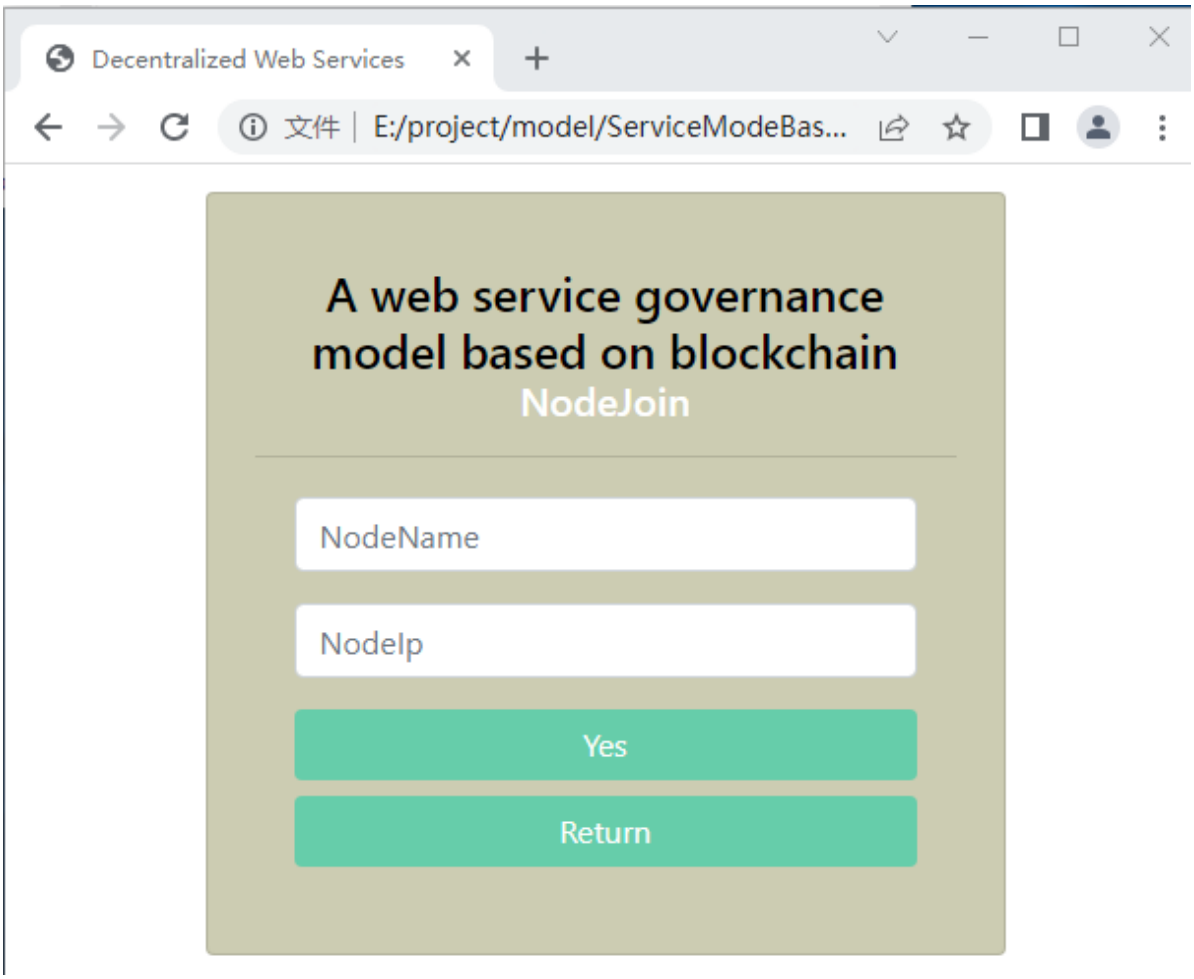

Fig 8

4）在Fig 8中，NodeName是要输入节点名，NodeIp是要输入的节点的ip（也是本地主机的ipv4的地址）。在这里，NodeName输入node4,NodeIp输入192.168.112.130.然后点击Yes，向节点2发送加入请求。节点2收到节点加入请求后把新节点4的信息存储到本地，并把节点加入信息转发给其它节点，其它节点也把新节点数据存储到本地，节点2并返回主机4同意加入消息，主机4更新本地数据，主机4成为新的节点4。

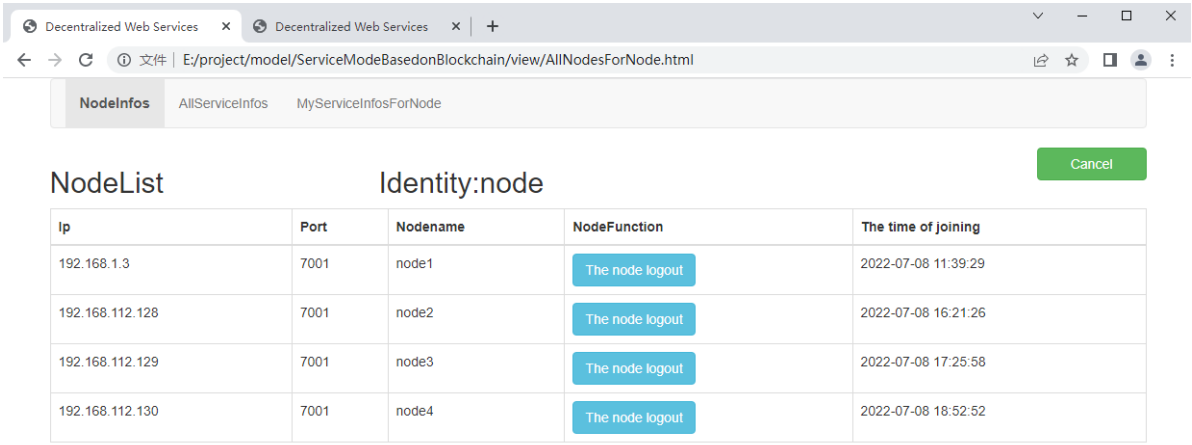

Fig 9

4.7节点退出

在这里我们选择node4模拟节点退出，node4进入节点信息列表界面（如Fig 11），选择任意一个The node logout表示node4向模型中的所有节点发送退出请求，其它节点收到退出通知，更新数据把节点4的数据从比如本地数据中删除。

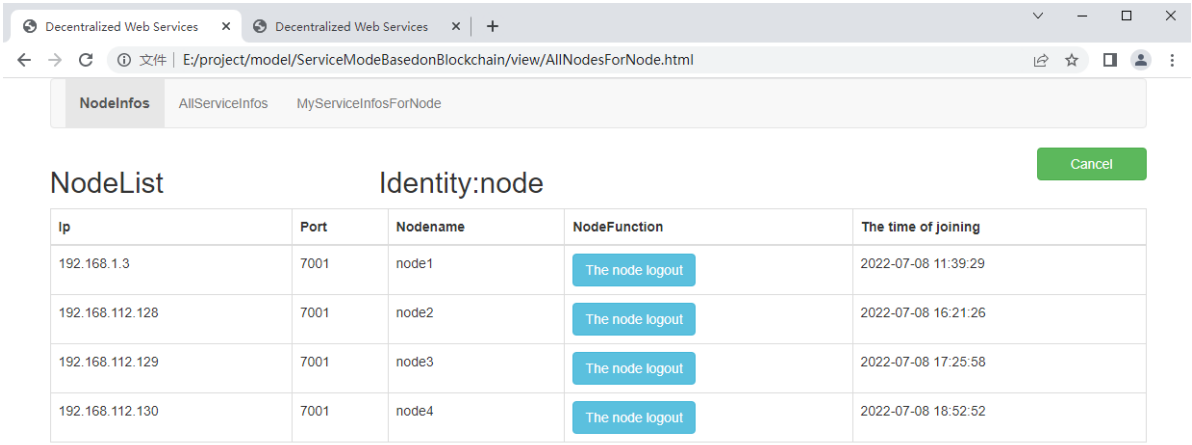

Fig11

4.8 web服务的发布

1）在这里我们选择Fig12中的Publish a service，以服务提供者身份访问模型。进入到服务提供者账号注册/登录信息界面（如Fig 12）。输入账号和密码进行注册，我注册一个账号为sunny的账号，在这里注册信息的密码用SHA256加密。注册成功后选择Login进行登录，登陆进入到服务提供者的模型节点信息列表界面（如Fig 13）。

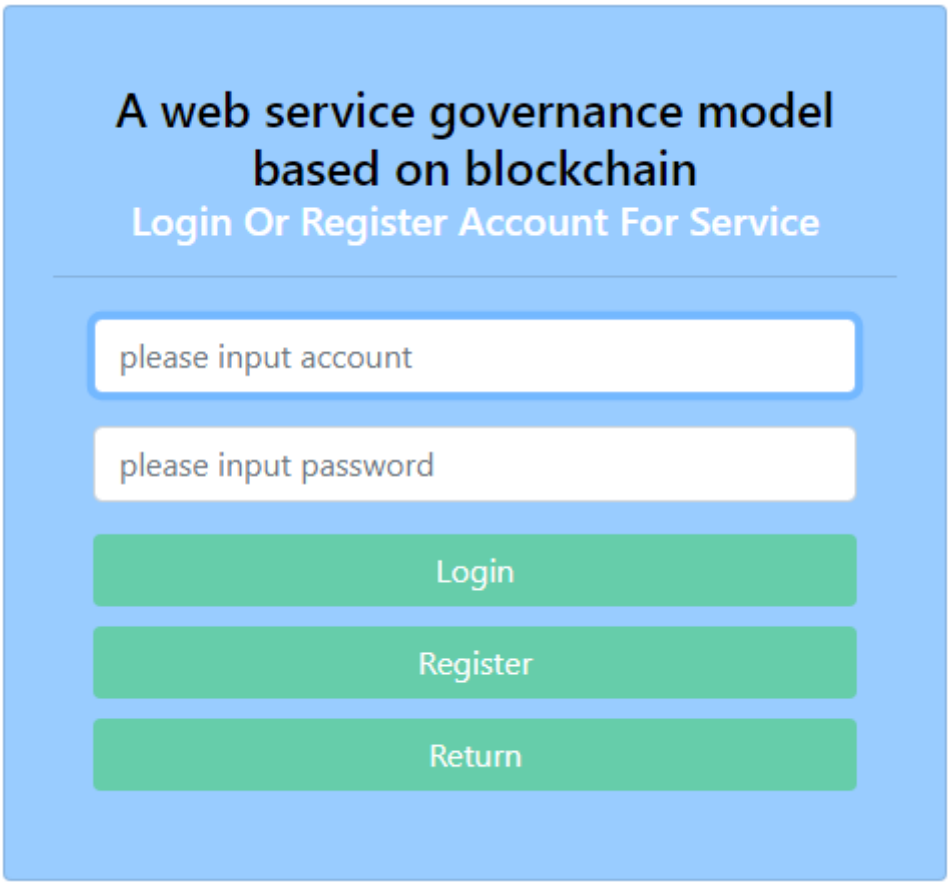

Fig 12

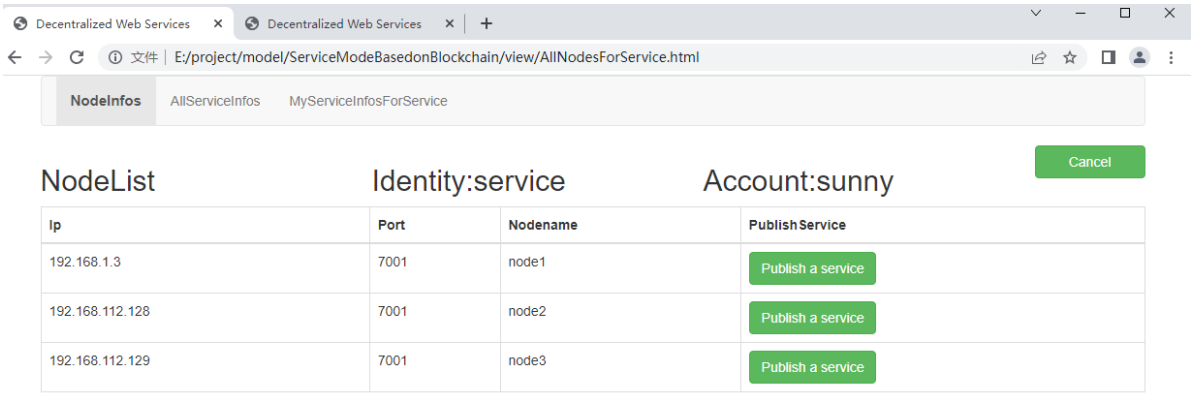

Fig 13

2）在Fig13中，其中Nodeinfos是对于服务提供者模型中所有节点信息列表界面，AllServiceInfos是模型中所有web服务信息列表界面，MyServiceInfosForService是该账号下加入的web服务信息列表界面，NodeList是节点列表，Identity：service是指以服务提供者身份访问，Accout：sunny是服务提供者的账号名，Cancel是身份退出，Ip是节点的ip信息，Port是指节点间开启的通信端口，Nodename是节点名称，ServiceJoin是账号在节点上加入web服务功能，publish a service是发布一个web服务。

3）选择其中任意一个节点后的Publish a service发布web服务。在这里我选择node3发布web服务。点击Publish a service，进入到发布web服务的输入信息界面（如Fig 14）。在Fig14中，其中ServiceName是web服务名称，ServiceType是web服务的类型，ServiceAddress是调用web服务所在的接口地址，MethodName是web服务的方法名，MethodExplain是服务方法的说明，InputAndOutputDescription是服务方法的输入输出说明，ServiceCode是指该服务的验证码，采用SHA256加密，退出web服务的时候需要输入校验，Yes是发布web服务，Return返回上一界面。点Yes确认后该账号所在的主机向节点2发送一个web服务加入请求，节点2收到web服务加入请求后把加入的web服务信息存储到本地数据，并把web服务加入信息转发给模型中其它节点，其它节点把新服务信息

存储到本地，节点2返回给发送web服务的所在主机同意加入通知，主机收到同意通知后更新数据，完成web服务的加入。我们在本研究测试的时候引用的web服务来自于www.webxml.com.cn网站。

Fig 14

4）点击Yes发布web服务，发布成功后。进入到本账号发布的web服务信息界面（如Fig 15）。在Fig15中MyAllServicesForService是服务提供者账号的已发布web服务界面。AllServiceList是所有web服务信息列表界面。NodeInfos是服务提供者的模型中所有节点信息列表界面。MyServiceList是指服务提供者账号发布的web服务信息列表。search左边的搜索框是输入自己想要检索的服务名称,点击search查找目标服务名（输入小写字母小写进行模糊检索）。ServiceName是展示的所有web服务的服务名称。BelongToNode是指发布的web服务所属的节点。BelongToAccount是指发布web服务所属的账号。ServiceType是web服务的类型，ServiceInfo是点击进入服务的详细信息界面。Logout下的The service logout表示对该web服务进行退出。

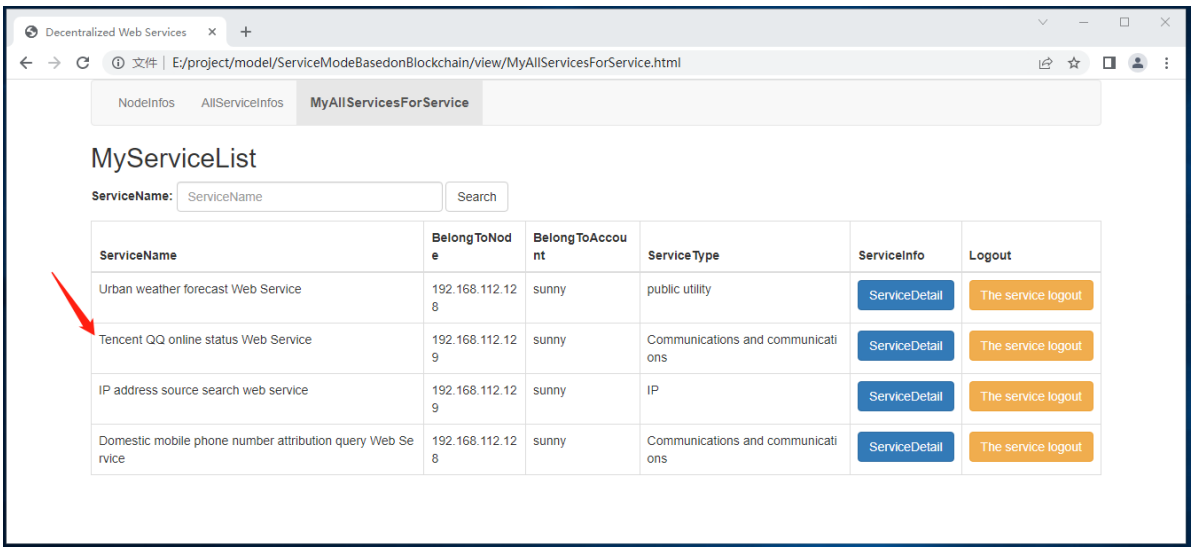

Fig15

4.9 web服务退出

我用sunny账号登录进入到Fig15中的界面，对列表中的Tecent QQweb服务进行退出。点击该服务后的The service logout。进入到输入该服务验证码界面（如Fig16），其中服务验证码采用SHA256加密。输入验证码后点击Yes进行web服务提出。退出后进入Fig17界面，不再显示该web服务信息。模型中的节点进行通信后，同步数据，模型中的所有的节点不再有该web服务信息。

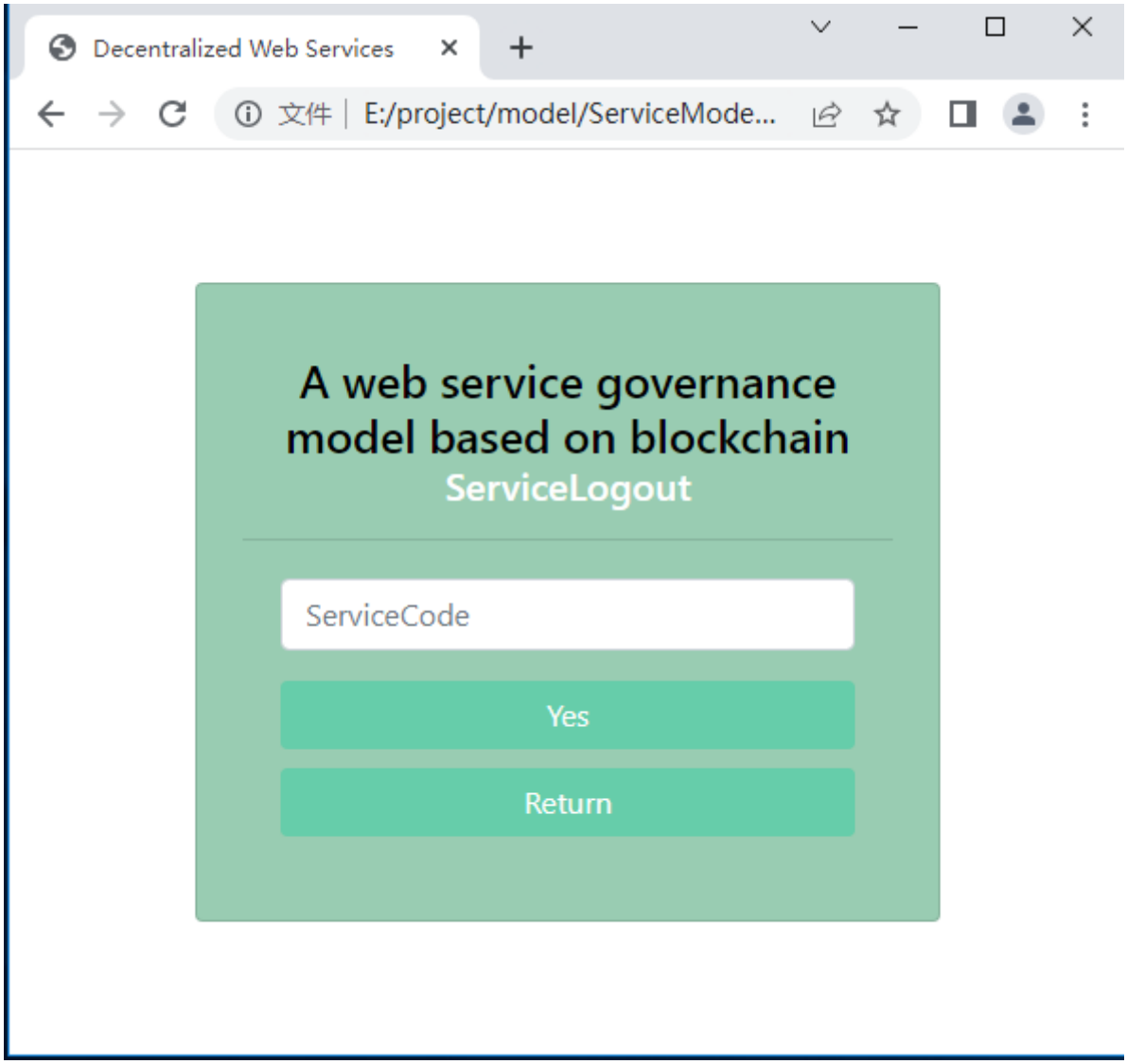

Fig 16

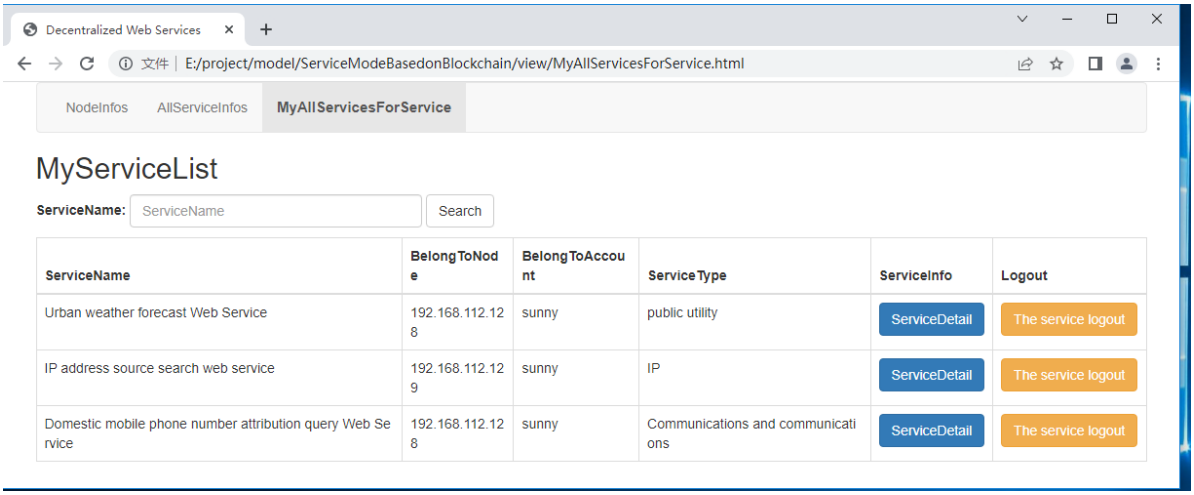

Fig17

4.10 服务请求者获取web服务

我用服务获取者的身份访问模型。点击Fig18中的Join as a Service requester进入模型。进入到服务获取者的模型中所有web服务信息列表界面（如Fig19）。

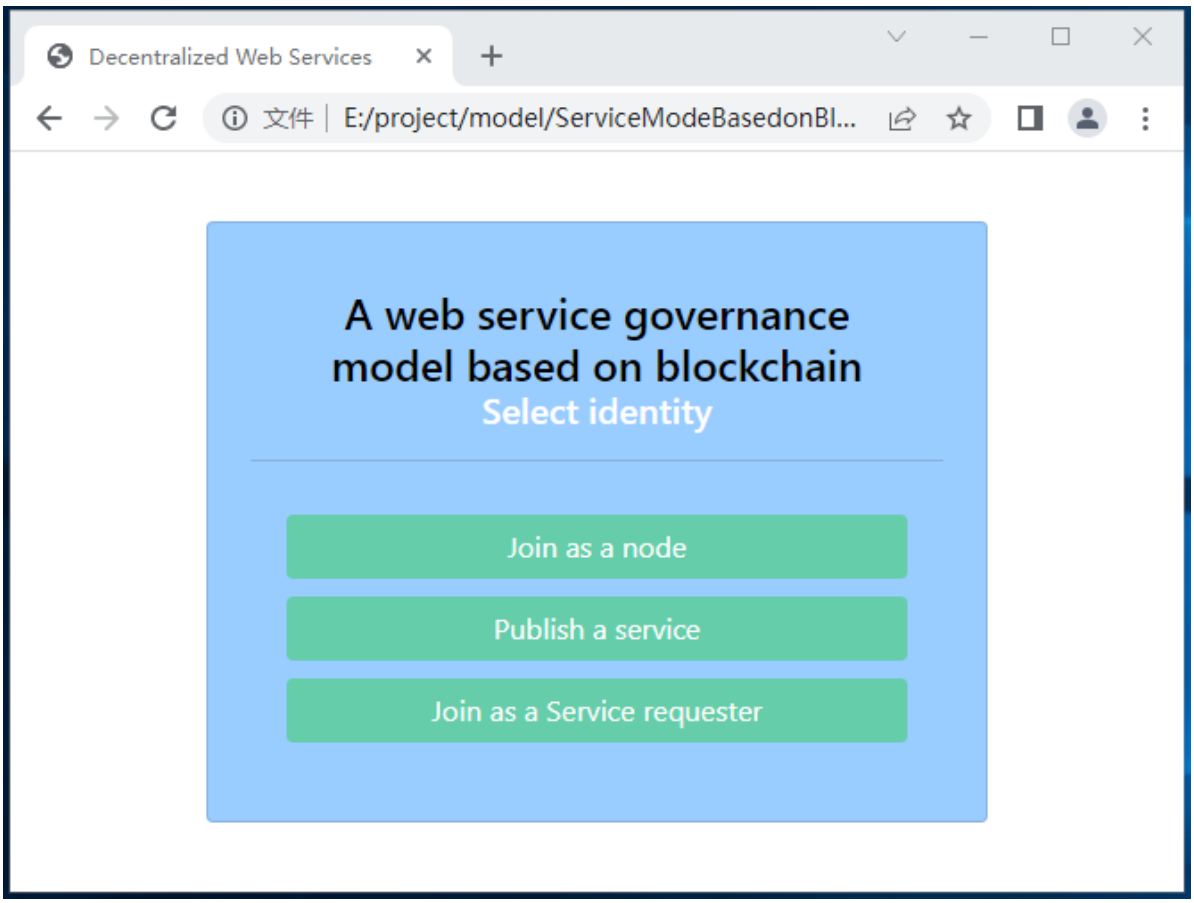

Fig18

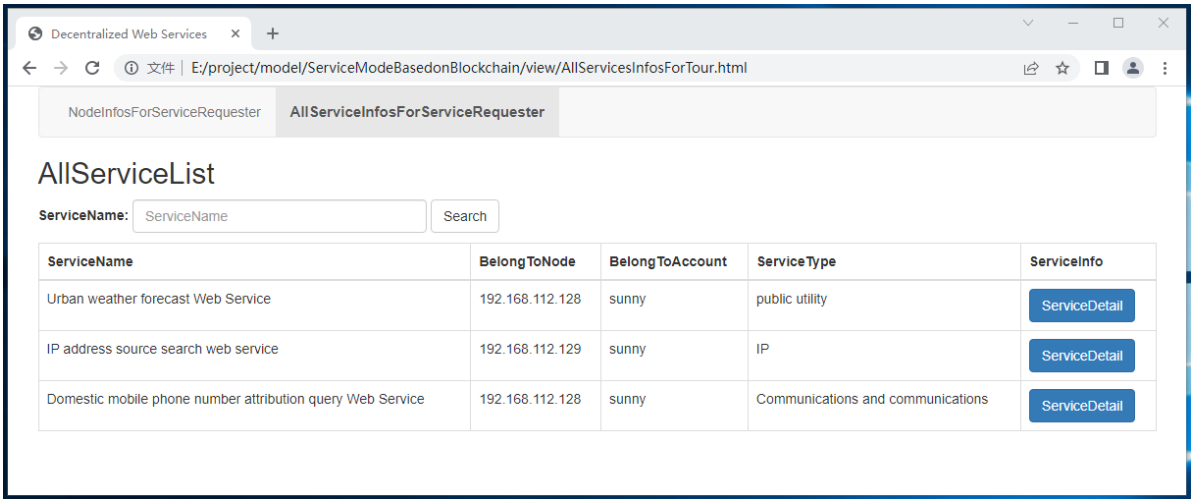

Fig19

在Fig19中，其中NodeInfosForServiceRequester是对于服务请求者的模型中所有节点信息列表界面，AllServiceInfosForServiceRequester是对于服务请求者的模型中的所有web服务信息列表界面,AllServiceList是所有web服务信息列,search左边的搜索框是输入自己想要检索的服务名称,点击search查找目标服务名（输入小写字母小写进行模糊检索）,ServiceName是展示的所有web服务的服务名称，BelongToNode是指发布的web服务所属的节点，BelongToAccount是指发布web服务所属的账号，ServiceType是web服务的类型，ServiceInfo是点击进入服务的详细信息界面，服务请求者可以根据需求获取自己想要的web服务。

4.11 去中心化的web服务治理体系

1）模型中所有节点都是对等的，都记录模型中的同样的结构数据，无中心节点。即使有单个节点不能正常工作，只要模型中还有1个节点，模型依然可以正常运行。不能正常工作的节点上的web服务依然在模型中的所有节点上，当有节点退出或者加入，模型中正常工作的节点会记录这些信息数据，当不能正常工作的节点正常工作后，依然可以从模型中获取模型中的最新数据。

2）模型中已有4个节点1，2，3，4.当节点4（ip：192.168.112.130）不正常工作（如Fig20），但是在node4上的web服务依然可以正常获取（如Fig21）.

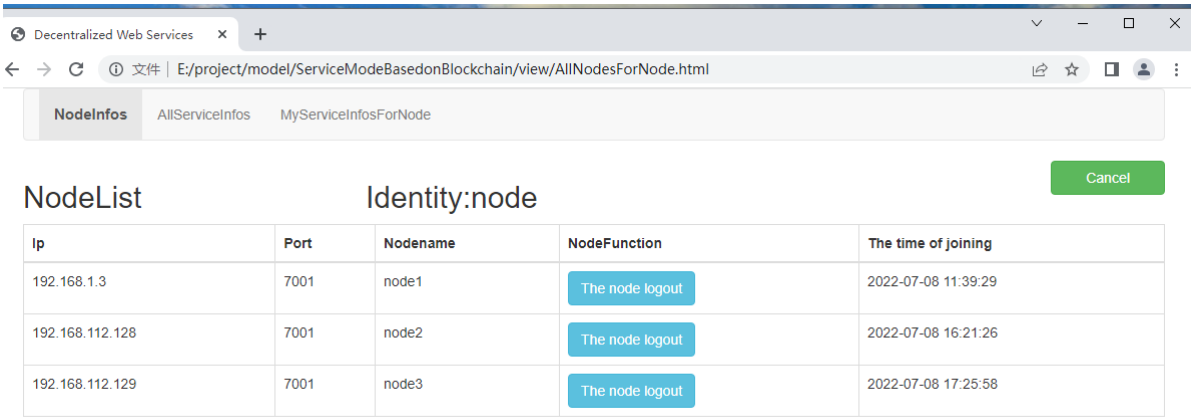

Fig20

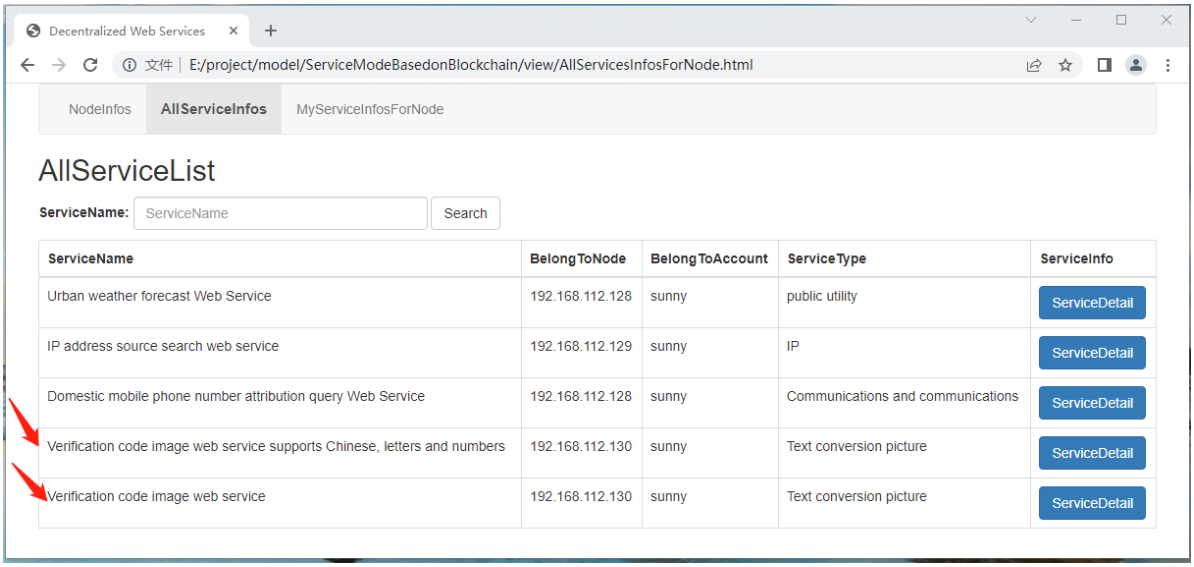

Fig 21

3）在node4不能正常工作情况下，在node3上发布1个新的web服务（如Fig22）

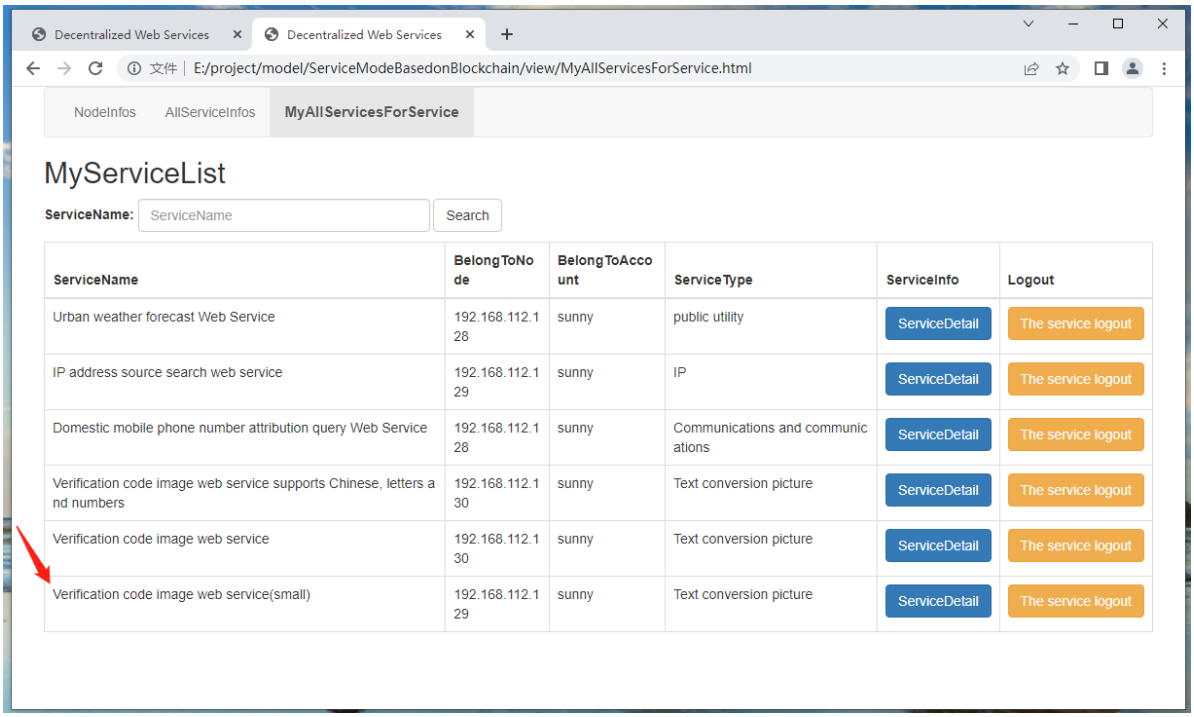

Fig22

4)当node4正常工作后（如Fig23），主机4以节点身份访问模型后。node4会从模型中获取最新的数据，新的web服务数据会显示在node4的web服务信息列表界面（如Fig24）

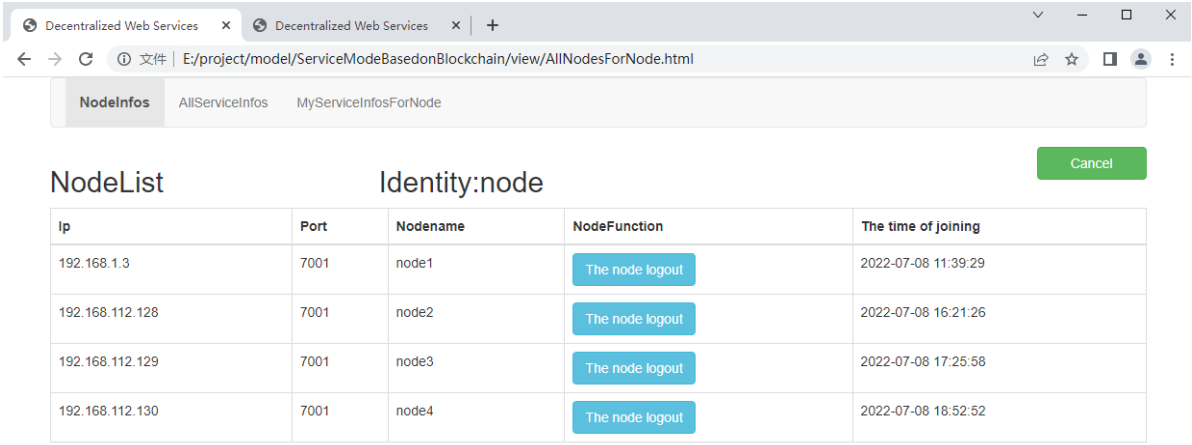

Fig23

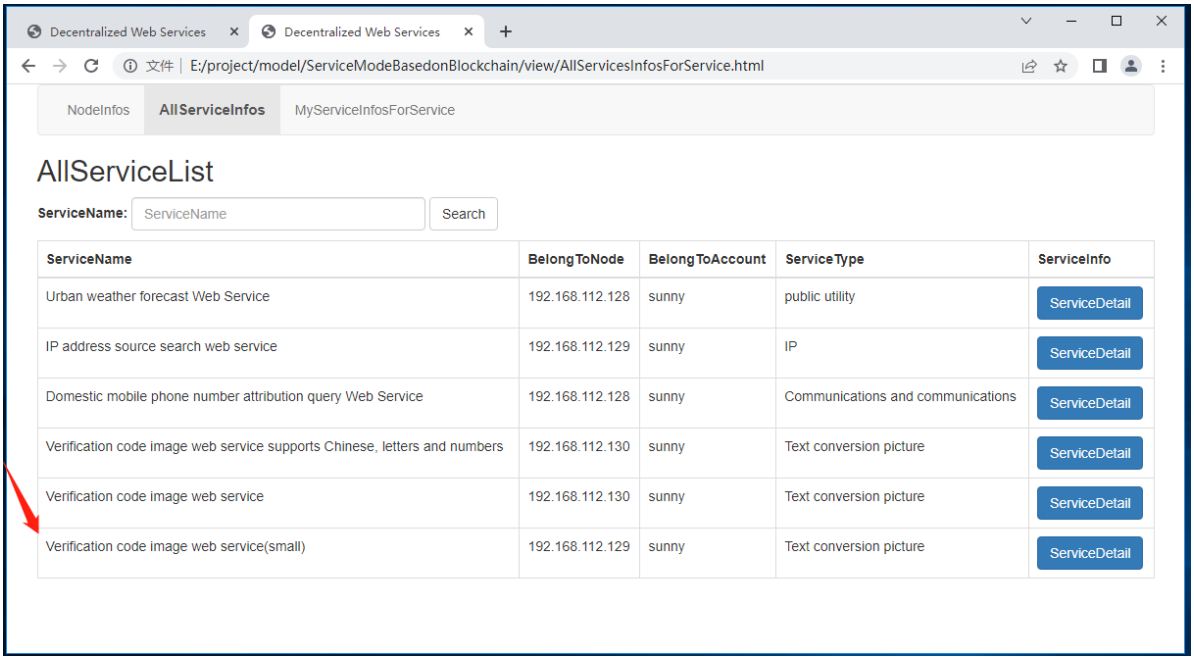

Fig24

4.12 节点对于发布在自己上的web服务具有一键退出权限

1）当一个节点退出模型后，发布在该节点上的所有web服务也会退出模型。

2）在node4（ip：192.168.112.130）上已有2个web服务（如Fig25），当node4退出模型后，发布在node4上的web服务也会退出模型（如Fig26）

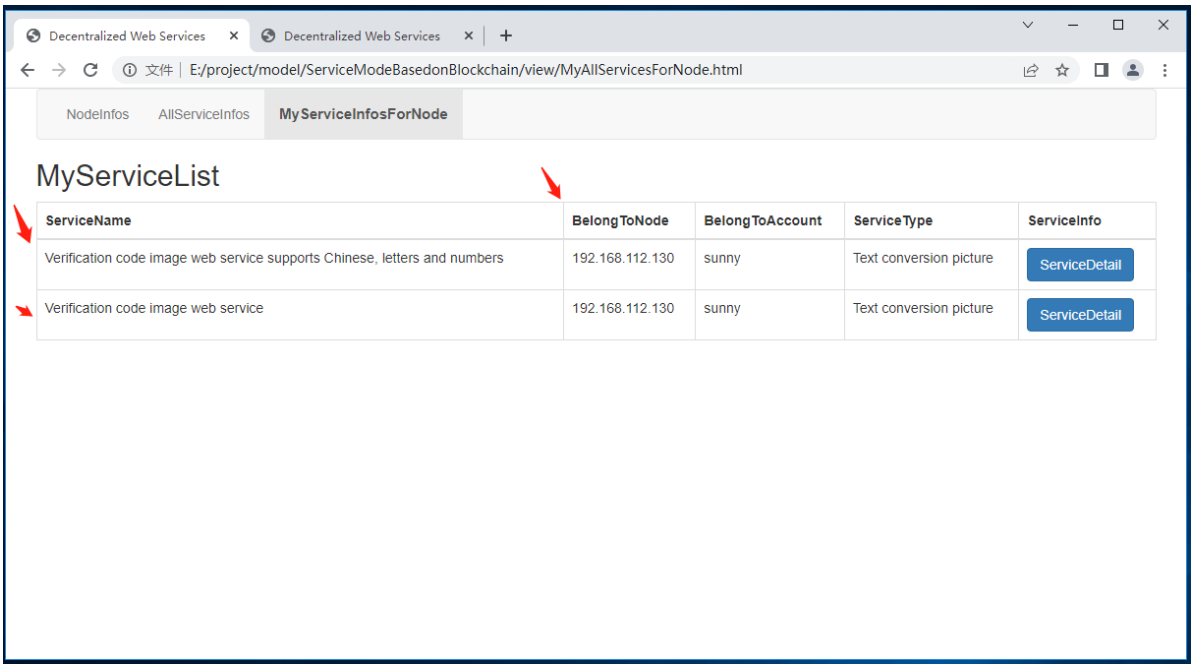

Fig25

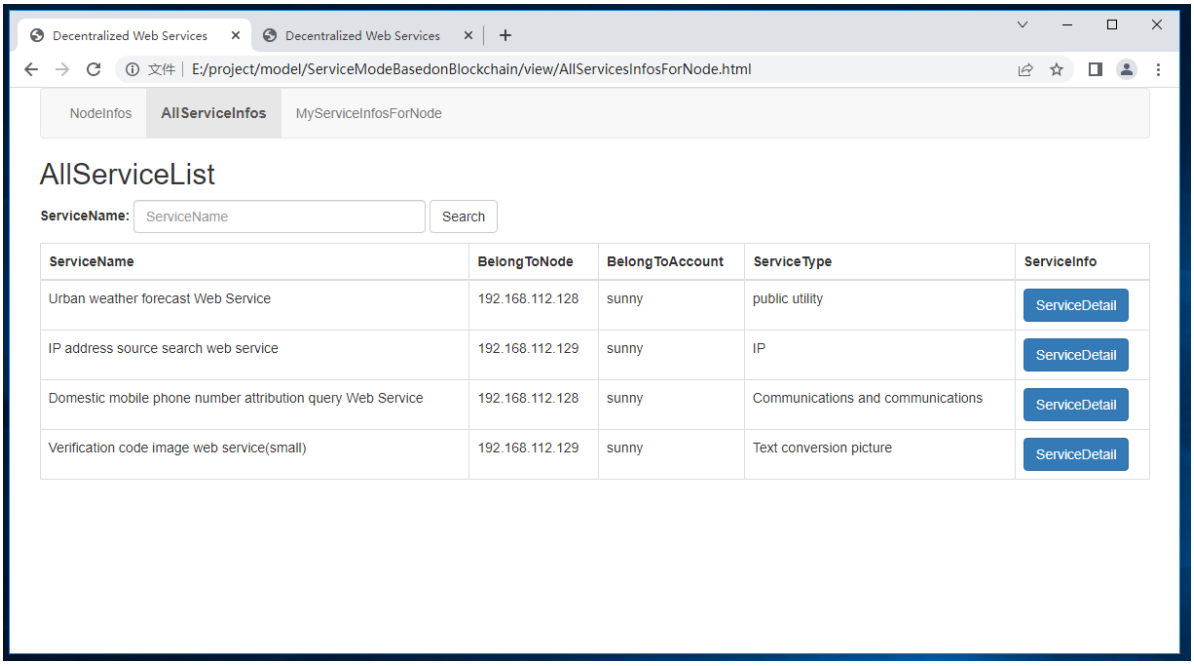

Fig26

5. 总结

经过实验，验证了本文设计的基于区块链的web服务治理模型，解决了web服务去中心化治理和缺乏信任的问题，数据的分布式存储，提升了服务治理的自治性、数据的安全性，并证明了该模型实施的可行性。