A Cooperation Model Towards the Internet of Applications

Zhe Zhang, Hailong Sun,Richong Zhang, Xudong Liu, Xu Peilong
School of Computer Science and Engineering
Beihang University
Beijing, China
{zhangzhe,sunhl,zhangrc,liuxd,xupl}@act.buaa.edu.cn

ABSTRACT

As Internet is changing from a network of data into a network of functionalities, a federated Internet of applications is a natural trending topic, where every application can cooperate with each other smoothly to serve users. Cooperation can be regarded as integrating applications of different providers for users. However, existing integration techniques do not pay enough attention to multiple participants including application providers and end-users. In this study, we advocate a global cooperation model of Internet of applications for all the participants. Specifically, we propose an intermediary based model to realize the cooperation among applications. With this model, on the one hand, users can be greatly facilitated to cooperatively use applications from various providers to meet their individualized requirements; on the other hand, providers can easily enable their own applications to interact with those from other providers. Thus, the federated Internet of applications is easier to be achieved than using existing solutions. In addition, we implement the model and show some case studies which demonstrate the effectiveness of this model. In our vision, such a model is the beginning to bring the socialized relationships behind applications into the digital world.

1. INTRODUCTION

With the development of Internet, an increasing number of providers deliver services through web applications [1] [2] or mobile applications [3] [4]. These applications are sufficing diverse demands of users.

Since users usually access various naturally related applications for tasks, automatically transferring information among applications will greatly facilitate users. For application providers, their applications may only serve a subset of functionalities in a task by themselves and need the help of other applications. Thus, in the view of both user and application provider, well cooperated applications do matter. However, cooperation among applications is still not well established [5] today, even though the requirements do exist. We need a mechanism to smooth the cooperation of the applications, such that the federated Internet of applications can be achieved, where information among application is seamless transferred if needed and users can regard the entire Internet as a unified service.

There are various works related to the aforementioned problem [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]. Internet scale service integration is a rising topic derived from service computing. Works like Internet Service Bus (ISB) [6], Complex Event Processing (CEP) [7] and Message Oriented Middleware (MOM) [8] build sophisticated techniques for service composition over Internet. There are also some End-user Development works like IFTTT[1] and mashup [9]. However, cooperation over the Internet involves end-users and application providers. A single participant (either application provider or end-user) cannot undertake the responsibility of cooperation comprehensively. As a result, participants are usually imposed on intolerable efforts to complete tasks. Existing solutions do not take such facts into consideration, so they are either user unfriendly or developer unfriendly. For example, it is developer unfriendly to meet user's diverse cooperation requirements, and it is user unfriendly to develop dependable services and maintain complex developmental level concepts. These problems can be commonly seen in existing solutions.

In this study, we propose a multi-participants cooperation model to avoid the ignorance of relationships among participants. In this model, responsibilities are reasonably allocated to participants, and an intermediary is presented to coordinate their works. Therefore, the efforts (that can be measured by the complexity of actions and the objects concerned in the process of development) of each participant are acceptable in more cases for cooperation. Specifically, application providers submit their interfaces and requirements to the intermediary, and end-users determine the final cooperation connections with the help of the intermediary. Multiple participants are involved in this configuration process. Then, by leveraging an uniformed identification mechanism and a communication protocol, applications query the intermediary to establish connections matching each end-user's preference automatically.

The contributions of this paper are as follows: First, we propose a multi-participants model as a new approach for the

[1]IFTTT, A service allowing users to create connections with "if this than that" statements

Internet of applications. Second, we show that the model has better effects on cooperation problem over the Internet compared with existing works. Third, we implement a system prototype with most important features, with which we show the practicability of the model by a few real demos implemented with our system prototype.

The rest of the paper is organized as follows. Section 2 introduces and classifies the related work. Section 3 introduces the model we proposed. In Section 4, we analyse some cases to demonstrate the effectiveness of our model compared with existing works. Section 5 introduces the conceptual architecture and requirements for each part in it. Section 6 describes our implementation of the model and shows two demos. Section 7 discusses the future works and forecast of the system and Section 8 summaries this paper.

2. RELATED WORKS

Lots of works are related to cooperation among applications. We classify the approaches depending on the roles involved in the integration process. Related roles are I (Intermediary Service), U (User) and P (Application Provider). Initially, solutions concerning all of the roles involved in a cooperation problem will be better.

2.1 IU solutions

End-user development solutions, like IFTTT [10], Zapier [11] and mashups [9], allow end-users to develop some functionalities based on existing services with the help of a platform, count as IU; These solutions usually focus on leaving the applications unchanged to utilize more application. As a consequence, applications know nothing about the global information, and application providers are not involved in the integration.

2.2 P solutions

Most traditional SOA solutions [12] belongs to this type. These solutions need a developer to define all the workflow procedures. Even some have intermediary, they just offer deterministic connections indicated by developers.

It is also popular to integrate other sites in the client pages, by embedding HTML/JavaScript snippets provided by some platform like website, such as Facebook's 'like' button. These solutions take advantage of communication protocols to access other's API directly or intermediary middleware without contribution to connection establishment, which still needs the developer to indicate which service they want to invoke.

2.3 IP solutions

The intermediary based services of IP solutions providing flexible connections among applications in different administrative domains, but the user is not involved for integration. Since SOA are evolving to integrate and connect applications on the Internet, some of them [6][7] provide more powerful intermediary to automatically adapt the services. They become IP solutions.

In the area of Computer Supported Cooperative Work (CSCW), there are some works [18] provide middleware to simplify the development of CSCW applications. There is also term Internetware and related work [19] that independently discribing the software development over Internet. They provide platform for cooperation management, coutings as IP solutions.

And topic/event based Pub/Sub systems like PubSubHubBub [13] and PLAY [14] allowing applications to publish and subscribe event with the intermediary, are such solutions, too. Most of them do not distinguish events for different user. Although some projects consider the end-user, we still do not think the end-user can actively develop the connections. Take the most similar one for example, PLAY allow the user to make privacy rules to enable/disable the events related to themselves, the user cannot be counted as involved in the integration — They cannot change the connections, and the control is weak that some connections may not be noticed sometimes.

2.4 IUP solutions

These solutions involve all the IUP characters in the development stage. Existing works are usually based on an operating system providing an eco-system. The entities of their cooperation problem are in their operating system other than Internet applications.

Filename extension [15] for Windows, Intent [16] for Android, and WebIntent [17] for browsers are such solutions for 'applications' in other domain (Native application for PC, Phone or front-end Web page), which means their cooperation happens within a single terminal. They allow applications to provide some requirements and enable users to determine the cooperation relationship. Our model share the same vision of building a cooperation mechanism for user-oriented operating system, and adopt a similar mechanism, but our model is for applications over the Internet.

The position of the user is promoted compared with IP. They determine the connections among applications finally. Compared with IU, the applications know and make contributions to the connections, extended their ability and awareness of the global Internet of applications.

3. MODEL

3.1 Problem Definition

To clarify the problem and our model, we define some terms first.

The application cooperation problem to be solved in this study can be formulated as: Given the applications and users, how to determine and execute the connections among applications for each user, such that the total efforts of the participants will decrease for the process, and the avaliable connections is controllable and meet the functional demands for each participant.

3.2 Model Description

In our model, every participant is involved in the connection establishment. The duty of application provider is to provide the information they require and can handle. And the duty of user is to determine the actual connection by assistance of an intermediary knowing more information.

Table 1: Terms

Application	Autonomous software entity accessible from Internet.
User	End-users who finally be served by applications.
Participant	A human controlled entity within an administrative domain, namely, end-user and application providers.
Connection	A cooperation relationship of two applications for a particular user that information may produced by one and consumed by another.
Userspace	The status of the Internet of applications for a user, including the connections for the user, the information transferring over them, and even the public status of user and applications. Intuitively, it is the personalized projection of Internet of applications for the user.

1. Roles and Duty
   The roles in our model include application provider, user and intermediary. They will create the connections in each user's userspace toghether.

   Application providers provide preferences that their application directly concern, typically the interface and some partial intents of application for the other end of a connection.

   Users determine the connections. Since the Internet of applications aimed to serve user, they are the ones know whether the connection meets their requirement. Users' preferences vary greatly due to personal interests or network environment, and it is a role that cannot make real-time reaction. This fact makes our model to adopt a solution that the connection is determined before actual communication. There is a special type of application represents as a browser or other kinds of clients, such that user can be involved in the activities of userspace, send or receive information to the userspace.

   Intermediary is a special entity to collect the preferences of applications and users to coordinate their cooperation. So each participant actually interacts directly with the intermediary first. It is a centralized services set that carries four major duties: 1) allowing application providers provide application definitions; 2) giving recommendation on connections between applications; 3) assisting users to confirm recommendations; 4) supporting applications to query the confirmed connection and finish the final invocation.

2. Cooperation Process

   The cooperation process happens in two stages: the configuration stage and the execution stage. In the configuration stage, intermediary collects the preferences of all the participants. As shown in Fig.1, application providers publish or update the information of their applications to the intermediary; users choose the applications via the intermediary. By confirming the recommendation of intermediary or connecting some applications manually, users finally determine the connection.

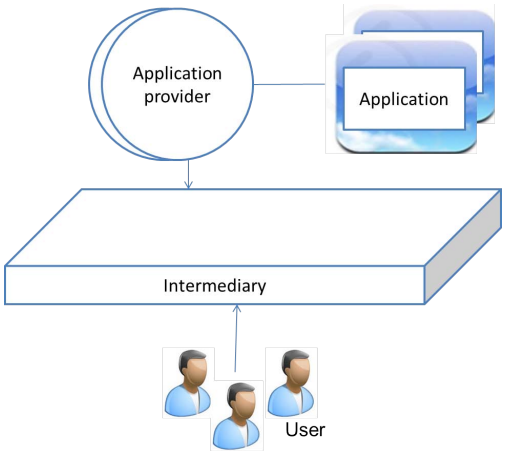

Figure 1: Configuration Stage

After this stage, the userspace (particularly, the connections of the user) will be changed. Fig.2 shows the situation that the intermediary supporting the userspaces and applications cooperate with different connections in different userspaces. In the execution stage, 1) application (including the special application representing user) raises requests by querying the connection from intermediary; 2) application executes the final communication by itself or through other services in case they can't do it. This design is compatible with activities unaware of the intermediary, gives applications the control of connections and alleviates the burden of intermediary; 3) application on both sides can check connections in the userspace to guarantee the control over connections related to them.

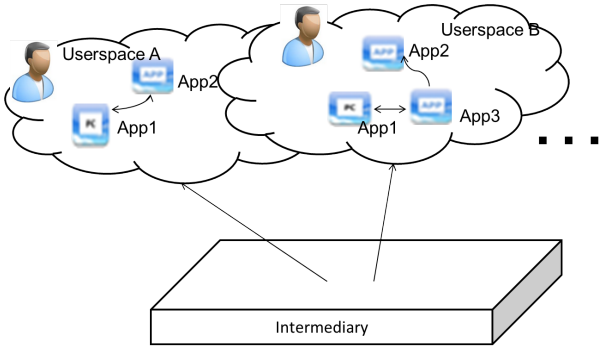

Figure 2: Execution Stage

The construction for tasks is evolving in the cooperation process. Some tasks are done through more than one connection, and the confirmed connections can be used in different tasks. Thus, a sole participant does not intent to build a task in advance. Tasks are implicitly built as users confirm the connections and applications provide new functionalities even raise new requests.

Under the coordination of the intermediary, applications are aware of the userspace, and communicate as user's prefer-

ences.

4. COMPARATIVE STUDY

We will use some metrics to show this model can promote the cooperation over applications. In our opinion, the effort each participant has to pay, and the quality and quantity of cooperation tasks the model can achieve, plays essential roles to show the effectiveness of a solution.

In this section, we classify the cooperation tasks into several types, and take the number of types a solution can handle as the quantity metric. The cooperation types considered in this study include 'Request Task', 'Data Processing Task' and 'Event Aware Task'. The major requirement for the connections in these tasks belongs to the sender, user and receiver respectively. Although there are also requirements of other participants that will affect the overall precision, if a solution fails on major requirement, it does not compelete the task.

1. Request Task: An application requests functional results of other applications. Thus, the most determinant requirement of the connection in the task belongs to the sender of the request. As an example, we take the case of retrieving a picture from other applications.
2. Data Processing Task: Users want to process their data with arbitrary application if only the data format is correct, such as 'Star Google Reader items and store them to Evernote'. The user is the only one who understand what exactly the connection should be like.
3. Event Aware Task: In these tasks, the sender applications need to trigger events and events listener applications take the events as triggers to start their logic. In this situation, the sender applications are only responsible for trigger the events, and the events listener applications require the entire the connection.

We use *Precision* that how the connections meet the actual demands of all the participants as a quality metric. It would determine whether the solution is reliable. Efforts are separated for each participant according to the complexity of actions and developing concerning objects. We use *User Effort* to represent the effort for the user to establish a connection, and use *Application Effort* for the application provider. In this paper, the precision metric and effort metric are not quantifiable metrics. We label 'High', 'Moderate', 'Low' intuitively.

Then, we compare our model (IUP) with existing works (IU, P, IP) to show the effectiveness by analysing the precision and efforts under each task.

4.1 Analysis

4.1.1 Request Task

The first case that we study is that Photoshop online[2] retrieves a picture from SkyDrive[3] or Flickr[4]. In this case, the connection should meet the requirement of request sender Photoshop online at first.

For IU solutions, their platform cannot affect active requests inside an application. This task is unlikely to be achieved.

For P solutions, Photoshop online should directly communicate with SkyDrive or Flickr. The options are fixed in the application. It is impossible for Photoshop online to know all the possible cooperations, so SkyDrive may be missed in the options. So the precision may not affected as it may not meet the user's requirements. Since the user does nothing but choose one among options in the cooperation, the User Effort is low. The Application Effort is high as it should collect and maintain the cooperation partner.

For IP solutions, a powerful intermediary is there to help the establishment of connections. Photoshop online would retrieve the picture from the intermediary, and the intermediary automatically choose a suitable service that can provide a picture. Nonetheless, the intelligence of intermediary is not likely to figure out the user even the application's wish, leading to a relatively low precision. The effort of the user is even lower than P solution in this case. The effort of application provider is moderate since it just needs to provide some descriptions of the request.

For our multi-participants model — an IUP solution. Photoshop online, SkyDrive and Flickr offer their definition at the intermediary. Users confirm if the connection meets their requirement. The recommendation is based on application provider's information, and they can check the connections at runtime, so the connection meets application's requirement, too. As a result, the precision is high. User has to manage the applications, so there are some acceptable moderate efforts. For application providers, the effort is the same as IP solutions.

4.1.2 Data Processing Task

The second case is about the situation that a user saves the stared Google Reader[5] items to Evernote[6]. Only small group of users will be interested in this task, so the major requirement belongs to the users.

For IU solutions, user should develop an action that if Google Reader stars a new item, then the item will be sent to Evernote. The connection has perfect precision for the user as it is same as user expected. But if Google does not intend to share their data with some competitors, this connection does not meet the requirement of Google, so the precision is affected in the view of the application provider. It takes users some advanced efforts (such as developing rules for IFTTT) to prepare the task that leads to high User Effort. For application providers, they do not need to spend extra efforts.

For P solutions, such requirements are almost impossible to be sufficed since the application providers won't develop such functionality for a small group of people.

[2]The online version of famous photo editor Photoshop
[3]SkyDrive is the cloud storage service of Microsoft
[4]Flickr is a online photo management and sharing application
[5]A News Reader provided by Google
[6]An online notebook

Table 2: Effectiveness Comparation on Request Task

	IU	P	IP	IUP
Precision	N/A	Moderate	Low	High
User Effort	N/A	Low	Low	Moderate
Application Effort	N/A	High	Moderate	Moderate

Table 3: Effectiveness Comparation on Data Processing Task

	IU	P	IP	IUP
Precision	High/Moderate	N/A	N/A	High
User Effort	High	N/A	N/A	High/Moderate
Application Effort	Low	N/A	N/A	Moderate

For IP solution, Google Reader and Evernote are not likely to be connect either, even with the help of an intermediary.

For our IUP solution, Google Reader would announce it may trigger 'star item' event, while Evernote may receive 'record content' requests. The user may still has to adapt the connection manually as IU solution does, or can get a recommendation from the intermediary. So the effort is high or moderate accordingly. Precision and application effort metric is as usual.

4.1.3 Event Aware Task

In the final case, we assume such a scenario: Bob lives in Beijing, and he bought a book at Amazon, later on, he has to leave Beijing to Shanghai. He books a flight and hotel at booking.com. He hopes that Amazon can send the book to his hotel in Shanghai. It is the receiver application Amazon's requirement that determine whether the connection makes sense.

For IU solutions, Bob should develop an action that if he books at booking.com then Amazon changes his order. As analysed in Data Processing Task, the precision is perfect as usual, user effort is high and application effort is low.

For P solutions, Amazon has to make agreements with all the possible applications that may know that the user changes his/her location, and ask them to notify Amazon about that. The precision is ok for applications, but for a particular user, he may want to leave their friend to receive the book, thus the precision for the user is not perfect. User do not need to do anything here so their effort is low, but the agreements with other applications are not acceptable for an application, so the application effort is high.

For IP solutions, Amazon can subscribe events that represent location changing, and bookings.com will trigger an event matching that. The only difference with P solutions is that they do not need to make agreements with each other directly, so despite that the Application Effort is moderate now, other metrics are the same.

For our IUP solution: As usual, Amazon announces that they are interested in user location changed event, and bookings.com announces that it will trigger such events. The precision is better than P/IP solution because the user determines the connection finally. Again, it is high precision and moderate effort for each participant.

4.2 Result

Overall, IU solutions usually have high precision since the user can infer the requirement of applications. P/IP solutions have relatively lower precision due to the absence of users. And IP solution has even worse precision sometimes since intelligent intermediary is not as precise as application provider's explicit determination. Our model gives connection control to every participant, leading to a high precision.

As for efforts, since a single participant is hard to deal with all the information about the connection. IU and P solutions allocate this job to one participant, leading to high and intolerable effort against popularity. IP solutions allocated them to different applications and intermediary, alleviated their efforts. Our IUP model allocates them to all the participants, to make everyone able to accept the efforts.

Overall, our model can balance the allocation of efforts and offer precision under more types of tasks.

5. CONCEPTUAL ARCHITECTURE

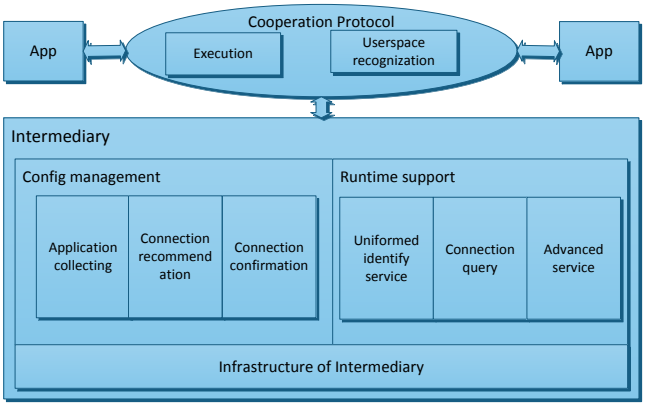

Figure 3: Conceptual Architecture for Realizing Model

To implement such a model, Fig.3 shows the concept architecture of what we have to do. We need a userspace recognizing protocol to make userspace be aware of each Application, and an execution protocol for final cooperation between applications; The protocol works with intermediary services. We need application collecting, connection confirmation and connection recommendation for the configuration stage; we

Table 4: Effectiveness Comparation on Event Aware Task

	IU	P	IP	IUP
Precision	High	Moderate	Moderate	High
User Effort	High	Low	Low	Moderate
Application Effort	Low	High	Moderate	Moderate

need a uniformed identify service and connection query service to support the protocol; Finally, we need a powerful infrastructure to support the running of intermediary services.

5.1 Cooperation Protocol

The protocol is the rule the participants should follow in our model.

The protocol have such requirements:

1. The protocol is able to instruct the execution and identify the userspace.
   This is the functional requirement. The definition of application only provides information they directly care about and the connection information should be able to instruct data transmission automatically among the userspace.

2. The protocol can handle as many types of cooperation as possible.
   The connections are diverse: 1) some connections just need an event notification, and some need a response to a request, some types of connection need more than one round-trip; 2) the communication protocol between applications is also diverse. So we need a protocol that can handle this situation.

3. The protocol can ensure the pravicy and security will not worse than not supporting cooperation.
   Since each application serves lots of users, and different application should share the same userspace for a user, a uniformed identify service is needed. The userspace is transitive, which means that when an application gets some information from the userspace, it can get aware of which userspace this information is from. This feature will enable the applications to cooperate in the back-end seamless for the user.
   However, this feature may introduce new security problems. For example, how to prevent an application forging a request to another application that the user did not permit as connection? Think about if a user tries an application and the application gets aware of the userspace by adopting the uniformed identify service, it automatically publishes ads to your facebook. It is terrible. So we need to ensure the security while offering convenient.

5.2 Config Management and Runtime Support

They are functional units for the intermediary. The intermeidiary responds for helping the participants to work under the model. For the Config Management component, the requirements are:

1. The Config Managment can help collecting the applications.
   This feature is necessary since the model need application providers to provide their interests as a step of the configuration stage.

2. The Config Management can give recommendation to users to help them with the decision.
   The recommendation for connection is not easy. Possible connections are neither arbitrary nor specific. For example, a photo processing application would never process a user login event, while many applications need a resource processing request handled by a 'share' action of an SNS application. This requirement is hard to be perfect.

3. The Config Management can help user with the decision confirmation
   This is also a necessary step. There are many chances for connection confirmation. For example, the can happen after user choicing an application. Or application can ask user to confirm some connection at runtime.

For the Runtime Support component, the requirements are:

1. The Runtime Support can support the requirement of protocol in the runtime.
   The protocol have several steps such as querying the connections, identifying the userspace that need the help of intermediary. This is the functional requirement for them.

2. The intermediary can offer advanced services to enhance the cooperation.
   Current technique is not enough for all possible cooperation for a seamless federated Internet of applications. Such as back-end applications cannot get the status of user and interact with them. So this component is introduced to enhance such functionalities.

5.3 Infrustructure

There are some problems affect the requirement for infrustructure.

The infrustructure should have such features:

1. The system is able to scale up and have acceptalbe delay for each request.
   The intermediary tends to be the infrastructure of the Internet of application. That means the stress would be as heavy as the DNS system. This is an ancient challenge about throughput and response time. Obviously, the intermediary should be a distributed system,

and the data storage and access policy heavily depends on the data access pattern of the functionality.

2. The system is hosted by different vendors. And the whole system can work even if a single vendor failed. The problem raises when we need someone to operate the intermediary. Besides the technical reliability of the infrustructure, a service may be shutdown by a company due to varieties of reason, and the application using it would face trouble. This situation has happened many times these years. Application providers won't relay their functionality over a facility that has lower reliability against themselves.

3. Users and applications can access any vendor's node and have consistant experience.
   Due to the last feature, users and applications may choice any vendor as the entry of the system. And as users travel, they may change the entry. To ensure the whole infrustructure work as one system, this feature is necessary.

6. IMPLEMENTATION

We implement a system for the model to verify some of our ideas. The system, including the intermediary, the protocols

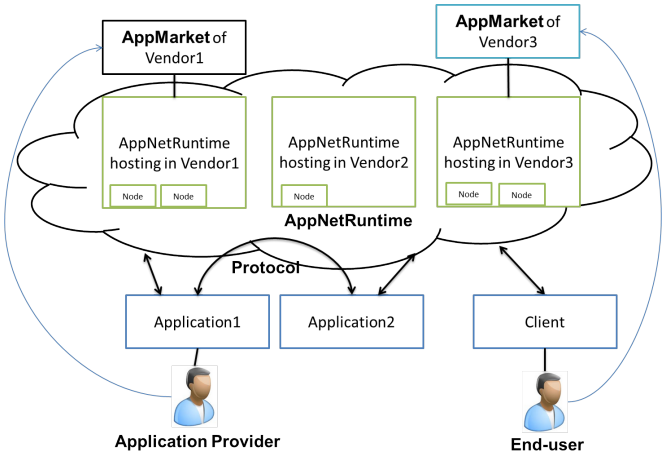

Figure 4: System Architecture

for the connection decision and the userspace recognition, and some other auxiliary work, implements the functionality and features discussed in Section 5 preliminarily;

6.1 Cooperation Protocol

Protocols are shown as bold arrows in Fig.4.

In particular, the connections are defined as *Routing* stored in the userspace after configuration stage at AppMarket. A Routing includes: userspace it belows to, the data format provided by source and the converting rules to receive APIs of different protocols. While each real cooperation will generate a *RoutingInstance* contain the information matching a *Signal* (The request or event sent from sender applications) and Routings. The details are showed in Table 5.

For pravicy and security requirements, we exploit an extended OAuth [20] protocol to access the knowledge of userspace. This protocol provides a secured third-party resource access mechanism by introducing some tokens as user's authorization, and the generation of authorization token is allowed to be extended. So we can add some information in the runtime supporting protocol to leverage it. Each application has their AppID. They would finally get an accesstoken after some authorization processes, which represent the userspace.

A typical process of the protocol runs as follow: When an application wants to raise a request to the userspace, it will query the routing by its request name and accesstoken first, getting a RoutingInstance that with an ID and some other userspace data required by the receiving API. Therefore it executes as the RoutingInstace instructed. When the receiver application received the request and what to know which permitted userspace this request in, it can get the accesstoken by the one-time usable ID of the RoutingInstance, which it the so-called extension for authorization. Moreover, application can check the RoutingInstance to refuse serving request that they not mean to serve, providing control of connection for application providers on both sides.

6.2 Intermediary

In our implementation, the infrustructure and functional components are coupled. So we introduce them together.

As mentioned, the infrustructure of our intermediary is in a distributed fashion. The intermediary is operated by different vendors. Besides the requirements mentioned in Section 5, we also take some factors about promotion of the system. A global distributed core infrastructure with consistency is required. But we try to minimize it to response realtime requests and keep a high-level reliability. And we try to provide some independent freedom for each vendor. Because it is significantly attractive for a vendor to obtain an entry and data access benefited from the intermediary, the independence between intermediaries of vendors enables them to have their private space to provide differential services. To take advantages of independence while keeping the whole intermediary united, the Config Management and Runtime Support components are built on different infrustructure, named as AppMarket and AppNetRuntime respectively.

6.2.1 AppMarket

AppMarkets are the services of Config Management. And they are portal web sites as configuration interface of Internet of applications for users. They can be separately provided by the vendors as Fig.4 shows, as long as they submit the final confirmed connections into the AppNetRuntime system. The reliability of AppMarkets is not that critical since if one portal dies, user can turn to another. Each AppMarket can maintain its own applications pool to control the quality of applications in the pool and offers advanced services for the user to make use of the entire system.

Currently, we provided an AppMarket that any application can be posted onto it. And recommendation and confirmation is done while user install applications in the AppMarket.

For the recommendation requirement, we use the 'named signal' and adapter mechanism for connection inferring: As pub/sub system did, this mechanism require an application

Table 5: Format of RoutingInstance

Item	Description	
ID	ID for this request	
Signal	Data of current sender application	
	Signalname	Name of the Signal
	Type	Request or Event, Request wait synchronously while Event not
	AppID	ID of sender Application
	Params	Data in key-value pair
Listener	Description of APIs that can accept the Signal	
	url	Interface of the API
	AppID	ID of receiver Application
	Type	RequestListener or EventListener
	Params	Contract of data format
	Adapter	The mapping strategy between Params in Signal and in Listener

to bind the connection to a topic predefined by some participants (Signal). With the help of adapter converting the parameter format, the data in the signal can be passed to the application. These information is propagated to fields *Signalname*, *Params* and *Adapter* fields in the RoutingInstance(Table 5).

6.2.2 *AppNetRuntime*

AppNetRuntime is the service of Runtime Support. It supports communication among applications and keeps the connection information of userspaces. Runtime service requires extreme reliability and performance since the requests happen many times after a connection is established. AppNetRuntime is a lightweight core fundamental infrastructure. AppNetRuntime nodes of different hosting vendors should share the code and data to make the service a centralized global system.

Vendors of intermediary are relatively static configured by neutral organization such as WWW, but in-administration domain nodes cluster is dynamic. The non-centralized distributed architecture is organized in a two-tier manner. As Fig.4 shows, the boxes of global tier (within the cloud) represent AppNetRuntime cluster hosting at vendors. Users can choose a vendor to store their userspaces as trust base where they are willing to keep their data. Meanwhile, the nodes (small boxes within an AppNetRuntime cluster) within the tier of a particular vendor are dynamic and work together to ensure the technical reliability and performance of the vendor.

The data is organized in userspace. A user choice vendor and all the information in the userspace is stored there. When applications access the userspace from other vendor, the infrustructure will redirect the request to proper vendor. In the future, we can adapt backups and optimize cache mechanism among vendors.

6.3 Auxiliary Works

We provided clients that can enable applications to interact with the user in more powerful ways,such as bookmarklet[7] and Android client. They can act as applications that provide interface interacting with user all the time. As observed in Fig.4, they obey the protocol as applications. This is part of the advanced services described in Section 5.

[7]A bookmarklet is a bookmark stored in a web browser that contains JavaScript commands to extend the browser's functionality.

6.4 Demos

We present two impressive demo scenes to show the convenience of a federated Internet of applications, and how they are achieved by our implementation.

6.4.1 *Demo 1:Status Synchronized among SNS*

Some people participate in more than one social network today, they want to keep their online status synchronized. We implement this synchronization on RenRen and Weibo[8]. This is a typical Event Aware Task.

Fig.5 shows the ideal procedure of the cooperation:

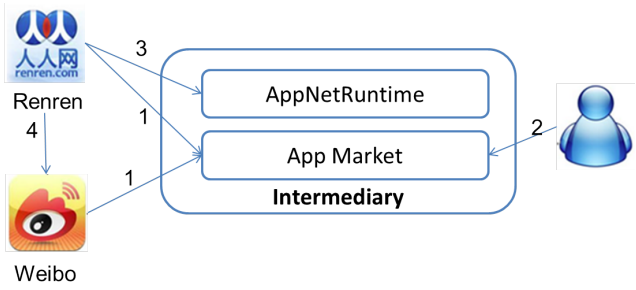

Figure 5: Procedure of Demo 1

1. RenRen and Weibo submit their application definition to the AppMarket, indicating that they can trigger and receive the 'update status' event;

2. The user installs the applications and confirms the suggested connections in the AppMarket;

3. When user updates his/her status at RenRen.com, RenRen triggers the 'update status' event to AppNetRuntime and gets the Routing.

[8]RenRen is a social network and Weibo is a microblog in China

4. RenRen invokes the receiver API of Weibo according to the Routing.

5. Weibo gets the request, regains the userspace, and updates the user's status at Weibo.

However, RenRen and Weibo do not support our model yet. To enable existing applications that are not officially supporting our model, we can implement stub services following the protocol in our system for them. In this demo, we provide stub services to monitor and update the status of real RenRen and Weibo.

6.4.2 *Demo 2: Powerful Web Resource Explorer*

We often see resource on the web and want to handle them with other applications, for example, we want to open a doc file link, or share a webpage to our SNS. The demo is about how to process resources on web pages with other applications.

There are some functional requirements more than cooperation, so a new application is needed, but we will show our system grant great ability of interaction to the application. We develop an application called Web Resource Explorer (WRE) to analyse the elements on a web page, and call proper applications to handle the resources in the elements. The application is loaded through a snippet of JavaScript code on a web page. It wants to be automatically loaded after any web page is loaded. So it will interest on the 'Page Load' event triggered by our client. And for 'open' request of a doc link, Zoho[9] application can receive it.

All the task types are shown in the demo: Data Processing Task (the user opens a doc with Zoho), Event Aware Task (WRE is aware of the 'Load Script' event) and Request Task (WRE executes its script on client).

The steps are similair to Demo 1:

1. The defination of each participant varies: Client of browser have a built-in event 'Page Load' (When a new page is loaded, the client will trigger the event to AppNetRuntime) and a receiver API 'Load Script' (Allow other application to provide a snippet of JavaScript onto current webpage); Web Resource Explorer indicates a recommendation to connect the 'Page Load' event and 'Load Script' receiver with their parameters; Zoho is lucky that their existing API is enough for handling a 'open' request and it do not need the userspace, so we just offer a definition file announcing it receive 'open' request other than providing stub service.

2. As usual, the user confirms the suggested connections.

3. When a new page is loaded, the client will load the script of Web Resource Explorer.

4. Web Resource Explorer analyses the elements of the page and figures out the elements can raise 'open' requests. When the cursor floating over the element, an 'Open' button will show up.

5. When user click on the button, Web Resource Explorer will open the doc file with Zoho according to the current userspace with the help of the client API. Fig.6

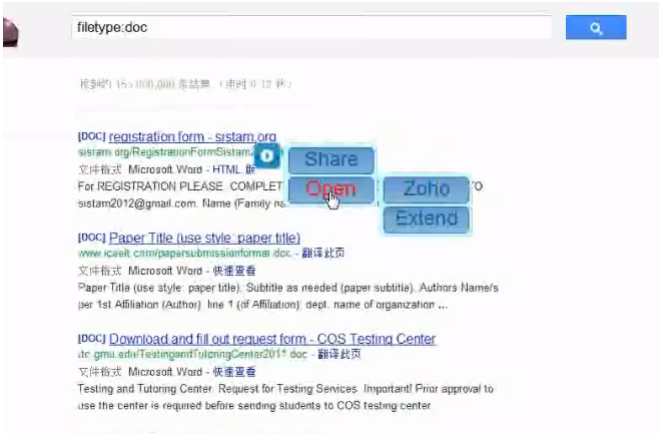

Figure 6: WRE run on Google

shows the snapshot of the Web Resource Explorer running on Google search result page. This demo shows an application can gain more powerful abilities with our system enhanced by clients, which offers more special information and functionalities in the userspace.

7. DISCUSSION

We can say our model is promising since the popularity is greatly affected by the highest effort of any participant and the trustiness of users caused by precision. If the effort of someone is too high, the chance of cooperation will greatly drop. If the precision is not perfect, participant would refuse to adopt it in many situations. When the connections we support can cover more situation and the chance to become realistic is improved, the federated Internet of application is likely to be achieved.

We treat the problem in a different view compared with existing solutions that can be used to cooperate applications over the Internet. In our perspective, the cooperation problem over the Internet is not a purely technical problem. Social natures like knowledge, trust and interest of each participant do matter in the federated Internet of application. We believe that as the IT is digitalizing our world, the relationship in the real world has to be considered while building technical infrastructure. Both technical and nontechnical solutions are involved to solve the problems. Actually, we treat our model as a higher level solution aiming at promoting an eco-system that the relationship of the user and applications providers are considered. The affection of this philosophy on cooperation mechanism is presented in our model: We encourage each participant provides their knowledge and allow them to control the connections. Participant related objects (Application, User) are first class objects in the system and protocols. This approach leads our model more effective than existing solutions (IU, P, IP) for cooperation problem for Internet applications, which derives from pure technical software development fields. Work focusing on incomplete participants certainly cannot beat our model in the view of cooperation.

[9]A web-based office suite, www.zoho.com

The cooperation problem over the Interent is also different with the problem on other fields. Cooperation entities over the Internet usually serves all the users in one service, while on end-user terminal, they are naturally isolated for each user. We provide userspace to group them together. The cooperation over Internet will be more powerful and complex since the entities are numerous and they have powerful computation ability. In our vision, like other IUP solutions in other fields which are based on operating system, the intermediary in our model would play an important role for user-oriented Internet operating system that supports applications running over the federated Internet of Applications. Such idea is mentioned before [21] and we are trying to clarify and build it. This model about cooperation is merely an aspect of the operating system, which may be the foundation of more areas of the future Internet. So our work is essentially a solution deriving from the idea of an operating system to solve the cooperation problem over the Internet.

Despite the preliminary implementation can be improved, the application of such model, even the application of the Internet operating system, will change the cooperation pattern between applications, generating new data and new behaviour patterns for further research activities. The intermediary itself is also a new kind of software with multiple vendors involved, requiring more researches to follow up.

Nevertheless, whether it really work depends on many other factors we do not know or cannot control yet. The future of the Internet is created other than forecasted.

8. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a multi-participants model to federate the Internet of applications. Realizing that the Internet of applications is evolving with various participants, we paid our attention to reduce the efforts of each participant by introducing a new cooperation model over the Internet, and built a system to verify its preliminary effect. Though it requires a paradigm shifting while developing cooperative Internet applications, we believe such shifting will be the foundation of the future Internet of applications as the nature had changed.

The following works should be conducted by both industry and academic. There should be some organization to carry it out and do the experiment in the public, and the challenges in the model can be solved better.

Acknowledgement

We would like to thank Fei Wang, Yijian Li, Wencong Xiao who help us implemented the system, thank Xu Wang, Ji Yuan and Kai Wang who help us with the paper writing. This work was supported partly by China 863 programs (No. 2012AA011203 and No. 2011AA01A202), and partly by National Natural Science Foundation of China (No. 61103031).

9. REFERENCES

[1] Maria Maleshkova, Carlos Pedrinaci, and John Domingue. Investigating web apis on the world wide web. In *Web Services (ECOWS), 2010 IEEE 8th European Conference on*, pages 107–114. IEEE, 2010.

[2] ProgrammableWeb. Programmableweb - mashups, apis, and the web as platform, 2013. http://www.programmableweb.com/.

[3] Apple. Appstore, 2013. http://store.apple.com/.

[4] Google. Google play, 2013. https://play.google.com/store.

[5] PETHURU Raj. Enriching the âĂŸintegration as a service paradigmâĂŹfor the cloud era. *Cloud computing–principles and paradigms. Wiley, Hoboken*, pages 57–96, 2011.

[6] Donald F Ferguson. The internet service bus. In *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*, pages 5–5. Springer, 2007.

[7] Alejandro Buchmann and Boris Koldehofe. Complex event processing. *it-Information Technology*, 51(5):241–242, 2009.

[8] Edward Curry. Message-oriented middleware. *Middleware for communications*, pages 1–28, 2004.

[9] Xuanzhe Liu, Yi Hui, Wei Sun, and Haiqi Liang. Towards service composition based on mashup. In *Services, 2007 IEEE Congress on*, pages 332–339. IEEE, 2007.

[10] Jason Stearns. Automate tasks with ifttt [quick tip]. 2012.

[11] Darren McCarra. Zapier: Add trigger and action accounts. 2012.

[12] Michael P Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-oriented computing: State of the art and research challenges. *Computer*, 40(11):38–45, 2007.

[13] Brad Fitzpatrick, Brett Slatkin, and Martin Atkins. Pubsubhubbub core 0.3–working draft. *Project Hosting on Google Code, available at http://pubsubhubbub. googlecode. com/svn/trunk/pubsubhubbub-core-0.3. html*, 2010.

[14] Života Janković, CIM Grupa doo Serbia, Roland Stühmer, Nenad Stojanovic, and Yannis Verginadis. Pushing dynamic and ubiquitous interaction between services leveraged in the future internet by applying complex event processing.

[15] Wikipedia. Filename extension, 2013. [Online; updated 29-April-2013].

[16] Erika Chin, Adrienne Porter Felt, Kate Greenwood, and David Wagner. Analyzing inter-application communication in android. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 239–252. ACM, 2011.

[17] P Kinlan. Web intents. specification, dec. 2010, 2012.

[18] Devdatta Kulkarni, Tanvir Ahmed, and Anand Tripathi. A generative programming framework for context-aware cscw applications. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 21(2):11, 2012.

[19] Fuqing Yang, Jian Lü, and Hong Mei. Technical framework for internetware: An architecture centric approach. *Science in China Series F: Information Sciences*, 51(6):610–622, 2008.

[20] Dick Hardt. The oauth 2.0 authorization framework. 2012.

[21] Tim OâĂŹReilly and John Battelle. Web squared: Web 2.0 five years on. *Special Report, Web*, 2, 2009.