Caching Mechanism in Peer-to-Peer Networks through Active XML

Abdullah Alrefae, Eric Pardede, Binh Viet Phan
Department of Computer Science and Computer Engineering
La Trobe University, Melbourne VIC 3083, Australia
{afalrefae@students, e.pardede@, vbphan@students,}latrobe.edu.au

***Abstract*—Active XML (AXML) is a new markup language framework that provides the ability to exchange intensional data. The intensional data is defined in the AXML documents as calls to local or remote Web services. AXML aims to improve the task partition and distribution in P2P networks. However, if one peer joins more than one P2P network and its services are invoked frequently by a large number of peers, this peer can become a bottleneck and requires large computational resource.**

In this paper we show that a caching mechanism in P2P networks based on the AXML technology can be a promising solution because AXML properties will simplify the caching procedures in these networks. We propose three layers of caches that will be located in: the bottleneck service provider, nominated peer(s) in the P2P network and the service requester.

***Index Terms*—Active XML, Web Service, Peer-to-Peer**

I. Introduction

Active XML (AXML) [13, 9] is a new declarative framework that aims to improve the performance of data and service integration over the web. AXML project is based on AXML document, which is syntactically a valid XML document with some parts of the data that are defined intensionally as embedded calls to Web services beside other parts of the data that are defined extensionally as in normal XML documents [9]. The embedded service calls give AXML documents the flexibility to become tools for distributed systems to cooperate with each other.

AXML documents can make calls to the external services that defined in other peers, and provide the capability to control the activation of these calls by appending some parameters to manage the validity of the results and to schedule the invocation of the service [12]. Therefore, AXML can be a powerful tool to integrate data and services over the web.

AXML is put to work in P2P architecture [2], which is a decentralized-based approach that employs the huge process and storage abilities available in the distributed resources over the Internet by directly exchanging services among them. Consequently, in P2P networks each peer can act as a server by providing some Web services and as a client by invoking other peer services [9].

However, in case one of the peers involves in more than one P2P network and its services are invoked frequently by a huge number of peers, the peer will be a bottleneck and its response can become inefficient and slow. In addition, the large number of requests to the same service can defect the network conditions by increasing the network traffic.

In this paper, we will address these issues by using a data caching technique, which is based on storing the XML/AXML documents that are resulted from materializing the embedded calls in different peers that requested the data. We adopt the idea in [4] that is based on nominating one peer in each group of peers that are interested in similar service. The nominated peer will take the responsibility to connect to the actual service provider and provide the service results for other peers. The caching peer will be selected based on different considerations like, better computing and storing abilities. We will extend this idea to any interest or location based P2P networks by nominating one peer in each network to be the caching peer. In addition, in order to guarantee the coherence of the cached data with the service updates, we will propose three caching layers located in the service provider, the nominated peer(s) in the P2P network and the service requester.

This caching mechanism will reduce the bottlenecks on service providers and will improve the network conditions by reducing the number of transactions over the Web. Furthermore, the data caching technique will improve the nested calls materialization by materializing the calls in the nominated peer and provide the final form of data to the other network peers. Moreover, this caching mechanism will lead to other positive impacts on the service invocation, such as accelerating the call response, reducing the connection cost, and facilitating the local queries over AXML documents.

We will show that caching mechanism can be a promising solution in the AXML system (networks that used AXML technology) because AXML technology simplifies the caching procedures over the P2P networks by providing the capability to guide the requester peers to one of the peers that cached the service results, by sending the path of these caching peers as a service call to the requester to connect them directly. Furthermore, AXML technology enables the network peers to have scheduled transactions to guarantee the coherence of the cached data by using the service call attributes that control the activation of the service call and the validity

doi:10.4304/jsw.7.3.508-515

of the results.

This paper is structured as follows. Section 2 gives an overview of the AXML project. In this section, we will briefly describe AXML documents and services, as well as some properties that simplify the process of managing the cached data in the AXML peer. In section 3, we will discuss the bottleneck problem that may be caused by frequent service invocation, and how it may affect the network conditions. We will describe the proposed three layers of caching in section 4, and we will explain what data will be cached in each layer and the motivation. Section 5 will give a possible scenario for the transaction of the cached data between the three layers and the qualitative analysis for the proposed caching mechanism by highlighting the areas that will be improved by this solution. Finally, we will conclude our work in section 6.

II. AXML Project: An Overview

According to [9], the bases of the AXML project are XML document and Web service. The AXML project gives the XML document the ability to have intensional calls to invoke the external Web services. Therefore, any AXML system that based on exchanging AXML documents has the following characteristics that differentiate it from normal XML system and makes it a powerful project for data integration [9]:

- The possibility to manage calls activation and the lifespan of the resulted data by appending some parameters to determine when the service call should be activated and for how long the results must be considered valid.
- Support the services with intensional data by providing AXML services that accept data with intensional input/output data.
- Allow continues services by supporting the use and the creation of Web services that may return a stream of answers, when such action is required.

A. AXML Documents

The basic content of the AXML system is the AXML document which is an extension of the traditional XML document with embedded calls to Web services to get some required data, beside other parts of the data that are defined explicitly in the document as in normal XML document. AXML documents are syntactically valid XML documents; therefore, standard XML tools can be used to exchange AXML documents [14]. For instance, SOAP and WSDL specifications are used to implement intensional service calls, whereas XPath and XQuery Languages are applied to query AXML documents.

The intensional data in the AXML documents are defined in a service call (<*sc*>) element that consists of the name of the server that provides the Web service and the name of the service that will be invoked. In addition, the service call element may handle, in specific mark-up, some parameters that are required to call the Web service [9]. Following is a sample of a service call element:

```
<weather>
   <sc service="forecast@weather.com">
     <city>
      melbourne
     </city>
   </sc>
</weather>
```

The process of activating the embedded service call to invoke the Web service and returning the result in the document is called *materialization* [9]. In this process, SOAP protocol is used to invoke the service by sending a request with a SOAP message that contains parameters values and the server will reply to this call by sending a SOAP message that contains the result to enrich the document [13]. Following is a possible result for the previous service call:

```
<weather>
   <temperature>22</temperature>
</weather>
```

AXML documents provide the following advantages for P2P environment. First is the dynamicity because the provided information in AXML document will change when the content of the Web service is changing and the user refresh the embedded call. Second, obtaining information directly by the users from anywhere is improving the independency and the knowledge of the users by giving them the ability to generalize the use of the service [14].

B. AXML Services

AXML services are Web services capable of exchanging AXML documents. In other words, they accept AXML documents as input parameters, and return AXML documents as results [8].

AXML provides a mechanism to integrate, store and query data through AXML services distributed over different peers. Each of these peers use WSDL to describe its services and has the ability to call its local services and also services in other peers by using SOAP protocol. Since AXML documents are syntactically valid XML documents, any Web service that invoke and describe functions with XML input/output can be called through AXML documents. Therefore, any of the various existed online services (such as, services in Amazon, E-bay, etc.) can be included [9]. The standard way to define an AXML service is by using XQuery, as described in the following sample [9]:

```
let service Forecast($x) be
for $a in document("weather-news.axml")
            /weather-apps/weather-news,
    $b in $a//weather/tenday
where $a@name=$x
    return <f> {$b/city-name} {$b/weather-
                forecast} </f>
```

The query will use the *weather-news.axml* document from AXML documents repository. When any peer invoke this service, the value of *$x* will be determined in the service call parameters.

C. Service Call Activation in AXML

The activation of the service call can be determined by two attributes, *mode* and *frequency*, attached to the service call element. The frequency is mostly defined as time interval (for example, every week), moment in time (for example, 1st of January), or triggered by modifying the included document [9]. A sample of the frequency attribute can be:

```
<sc frequency="weekly"> ...</sc>
```

Whereas, the mode can be immediate, to activate the service call whenever it is expired, or lazy to activate the service call only when its result needed, such as when the service call result is needed to evaluate a query over an AXML document [9].

In addition, the service call can be activated from the client side (pull mode) or from the server side (push mode). The pull mode is used for normal service operation upon client request, and has two types, implicit pull (in the lazy mode) or explicit pull (based on time frequency) [12]. In the push mode, the server determines when to push the data to the client. Mostly, this mode is used in the subscription to the continuous services when the server sends a stream of information synchronously to the subscribers. However, the continuous service can also invoke declaratively, by specifying query parameters for service activation, such as, time frequency of the responses, the limitation of the message size, and the clarification of the changes either by sending sequential versions, by sending script of the changes only, or by publishing the changes and sending a notification [12].

We use the push mode to exchange service results, particularly where the caching peer can push the cached data to the peers that interested in the service, whenever the caching peer obtained new service results. In addition, the caching peer can publish a notification whenever it obtained updated result, to give the other network peers the choice to pull this updates by activating the corresponding service call.

D. AXML Result Lifespan

The lifespan control of the resulted data can be managed by attaching a validity attribute to the data node in AXML documents. When the validity period expired, the resulted node will be removed from the document. The resulted node may come with *expiresOn* attribute to notify the user with the validity of the information [6].

Moreover, the caller can overwrite the validity of the result by attaching the attribute *valid* to the service call element. The valid attribute can contain values such as 10 days, 1 month, etc. It can have value of zero, to consider the result valid at the invocation time only. It can also have values of unbounded, to consider the result valid forever or until it is deleted [12].

III. Problem Description

In this section, we will describe some limitations of current AXML system in terms of the potential of high network traffic. Initially, we list the type of service calls that can be embedded in AXML documents [11] and the steps to activate them:

1. Calls to bring data only once (when invoked).
2. Calls to continuous service that will send a stream of answers (for example, subscription to an RSS feed service).
3. Calls to update an existing part of the document (refreshing previously fetched data).

In this paper, we improve the performance of the 2nd and the 3rd types of embedded calls. Both of them involve frequently requested data.

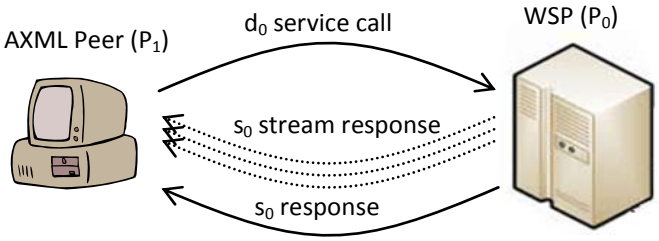
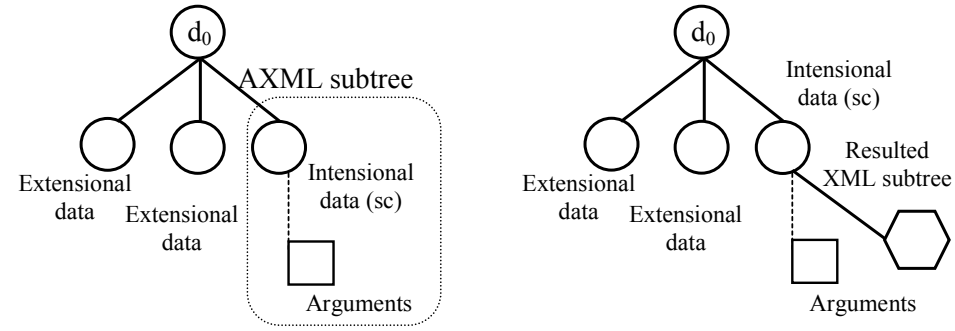


Figure 1 Service Call Activation Steps

In Fig.1, we show the steps of service call activation, where the Web service provider (P_0) provides the service s_0 (i.e. $s_0@P_0$). AXML peer (P_1) has an AXML document d_0 (shown in Fig. 2) that has an embedded service call (sc) to the service $s_0@P_0$. When the sc is activated, the following steps will occur [8]:

1. P_1 sends a copy of AXML subtree (see Fig. 2a) that holds the sc node and its children (arguments), to P_0, to evaluate s_0 on these arguments.
2. P_0 evaluates s_0 on the requesting inputs, and responses to this call by sending an XML subtree containing the result.
3. P_1 receives the XML subtree and encodes it in d_0, as a sibling of the sc node (see Fig. 2b).
4. In case $s_0@P_0$ is a continuous service, step 1 will occur once only, while the steps 2 and 3 will occur repeatedly, Consequently, P_1 will receive a stream of answers synchronously as a response and will encode them all in d_0 as siblings of the sc node. A special indication "end-of- stream" (or *eos*) [7] will be returned to notify the end of the service.

This simple illustration shows that an AXML system can be a powerful tool for integrating data and services over the web by using the embedded calls to invoke other services in different AXML peers or any online services, thus, support the distributed computation.



a. AXML document before materialization b. AXML document after materialization

Figure 2 AXML Document Before and After Materialization

However, the AXML system that is based on importing some of the AXML document's contents from other local or remote repositories or by invoking some local services or services in other peers can lead to negative impacts on the network conditions. Invoking a Web service that provided in an AXML peer or any Web server, by a large number of AXML peers, can lead to the bottleneck issue and the delay in data transmission by increasing the traffic. For example, in case the service $s_0@P_0$ is a common service that is required frequently by a large number of AXML peers, the activation steps will take place each time any of the requester peer activates the AXML document that hold a call to this service (for example, the document d_0).

Furthermore, some of the service invocations are only resulted from applying some applications on the AXML document, where in such cases the service call results are only secondarily required in order to get a fully materialized document. For instance, applying a query on an AXML document requires triggering some of the embedded service calls that related to the query. In addition, in the desire of changing the AXML document type to any other type, such as XML or HTML, all the embedded service calls need to be materialized first.

Moreover, some continues services (service with a subscription), where the service provider keeps sending the information to the subscriber peers, can lead to a costly propagation, especially when the subscribers are in remote locations, and can cause bottleneck and traffic issues as well.

In addition, in some cases, the server that provides the service is not responding, or the requester peer lacks the access to the service provider (for example, due to expiration of the subscription). Hence, the requester peer will not be able to import the intensional parts of the document from the other peer. In another word, they cannot materialize the service call to invoke the service.

In order to address these problems, we propose three layers of data caching mechanism, described in next section. However, the following issues may also arise because of using the data caching mechanism:

- Where to locate the caches (the copies of the documents)?
- How to guarantee the coherence of the caches in distributed environment such as the Internet?
- How to inform and specify to other peers to use the "appropriate" AXML repositories?

IV. Caching Layers

We are proposing data caching mechanism on three layers that can optimize the AXML workflow by addressing some of the network problems identified in the previous section. Our proposal also solves some other issues and questions that are related to applying the data caching mechanism in the AXML system. The three proposed layers of the caching mechanism based on their locations are: (i) caching data in the Web service provider (WSP); (ii) caching data in a nominated peer at the P2P network, and (iii) caching data in the AXML peer (see Fig. 3).

A. Data Caching Selection

In the first level, we can cache the data in the WSP, especially for common web servers that are frequently called by a large number of AXML peers, for example *weather.com*. The typical data cached in this level is as follows

- The result of the local services that usually called by the AXML peers and the result of the continuous services that have a large number of subscribers. Some validity attributes that are provided by the AXML system can be attached to the cached results to guarantee the coherence of the caches. In consequence, instead of activation of a particular service every time it is invoked by an AXML peer, the service provider will use the cached results to reply for these calls to reduce the computation tasks and save the processing capabilities for other tasks.
- The result of the intensional data in the AXML documents that the service provider sends as a response for continuous services callings based on an agreed schema between the sender and receiver. Thus, the subscribers in the service will receive a fully materialized document (in other words, document without nested calls). Therefore, the AXML system performance will be improved by reducing the

number of peers that materialize the nested call. The nested calls invocation will be done by WSP instead of by each AXML peer that receive the nested call as a part of the service result.

- In addition, the WSP will store the paths of the peers that have been nominated to cache the results of the intensional data in each P2P network. Afterwards, the WSP can redirect the new requests to one of the nominated peers or send a list of these peers as a result for the service call to give the requester the choice to call one of them.

The second level is caching data in one of the network peers (nominating a peer is based on the better computing abilities and faster internet connection [4]). Because the scale of the P2P networks grows, the centralized server (WSP), who is involved in large number of these networks, will become a bottleneck. Therefore, distributing the service results (copies of XML/AXML documents) on different peers, who are already interested in the provided data, can solve the bottleneck issue and reduce the network traffic. As described in the previous level, the WSP will store a list of the peers that have been chosen to store the service calls results and provide it for other peers who are also interested in this service. Hence, when a peer wants to subscribe in a service or request a service frequently, it sends an AXML subtree with the intensional data and its arguments to the WSP server and waits for the server to reply with the search results including a list of supplying peers (based on location, interest, security, less jam, and other considerations). Subsequently, the connection will be performed directly between the requesting peer and the supplying peer (nominated peer).

This caching level can improve the network connection as the data will be easily available for the peers. In addition, this will also improve the implementation of the nested calls, as the nominated peer will implement the nested calls in the service result to use them locally, and then send the fully materialized document if the requester peer ask for that.

The third level is caching data in each AXML peer, for instance, a personal computer. Obviously, we must take into account the storage capability for the peers. Therefore, the data that needs to be cached here is the result of the intensional data in the important AXML documents, by giving the user the ability to define important documents or services.

However, the materialization of the intensional data needs an Internet connection, at least if the requested service defined in another peer. Consequently, the results of the intensional data that have been defined as important will be updated automatically whenever the peer connects to the Internet and the result is expired and the latest copies of the data will be cached locally.

Accordingly, the local query implementation can be improved (speed up) because implementing any query on AXML document requires materialization of the related service calls in the document. Furthermore, in order to change an AXML document to any other format (other types of documentation), all service calls must be activated. Therefore, the cached data can be used to query or change the AXML document even if the peer is not connected to the Internet.

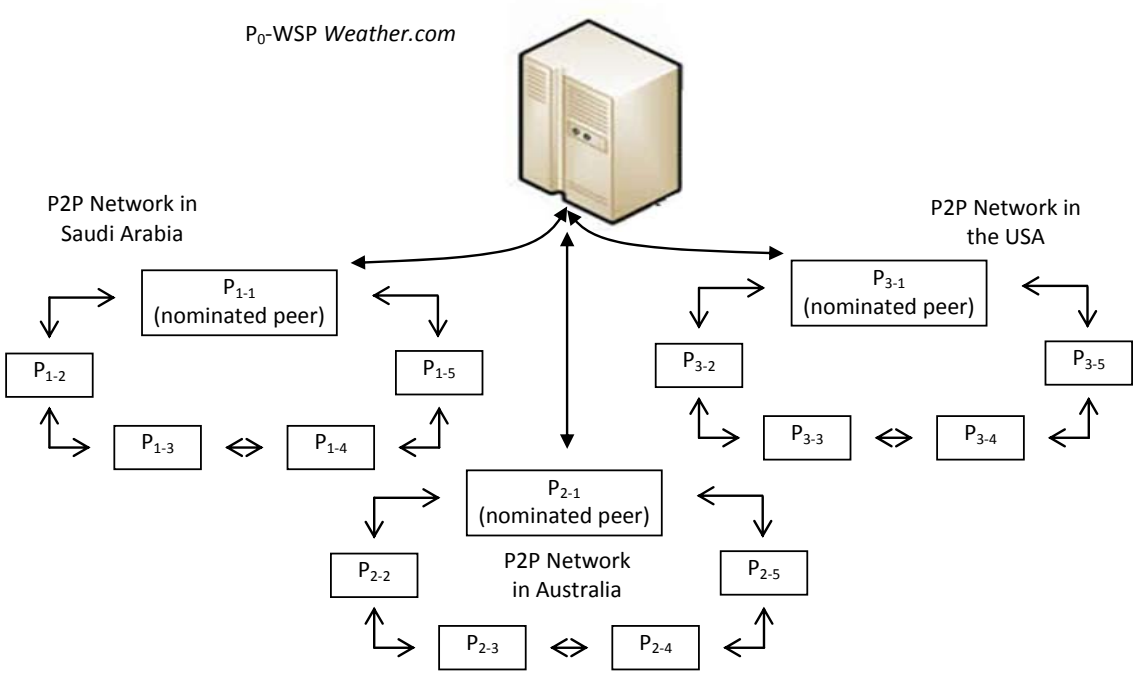


Figure. 3. Three Level of Caching Layers in AXML System

B. AXML Peers Components

In this section, we will explain the components of the AXML peer and how they cooperate with each other, in order to invoke a service or answer a call.

As we can see in Fig 4, the AXML peer is able to invoke services in other AXML peer and any service uses SOAP protocol for the description and invocation. Whereas, any SOAP client can invoke the AXML peer services [10]. The components of the AXML peer are (extended from [3]):

- *SOAP Wrapper*: gives the peer the ability to connect with any SOAP peer by encoding the request and the response to SOAP messages.
- *AXML Service Definitions*: define the AXML services provided on the peer.
- *AXML Document Storage*: stores AXML documents. This component will store the resulted data the peer desired to cache.
- *Evaluator*: activates the service calls embedded in the AXML documents by seeking the desired service locally in the AXML Service Definitions or invoke a remote Web service. The evaluator will also enrich the documents by the returned answer. In addition, it receives and executes the requests for any of the services provided by the AXML peer by sending a query to the XQuery Processor.
- *XQuery Processor*: implements evaluator query and reads the updates in AXML documents.
- *Cache Manager*: checks the validity of service calls result that cached by the peer in the AXML Document Storage and activates the corresponding service call to get the latest copy of the data. The Cache Manager will use a timer to schedule the update of the cached information. In addition, if the AXML peer is a nominated peer that provides the results of remote services to other peers, the XQuery Processor will ask the Cache Manager to get the required cached documents. This last component guarantees the coherence of the cached copies of information, by reactivating the corresponding service call whenever the result is expired even if the information is not required locally at that point of time.

VI. Case Study and Analysis

In this section, we show a possible scenario for caching a popular Web service and how to invoke the cached copies from the nominated peer. Afterwards, we will analysis the performance of the proposed framework.

A. Case Study

Fig. 5 shows a WSP *weather.com,* which provides weather forecast for any location worldwide. This WSP will be involved in a large number of P2P networks around the world. For example, networks A, B, and C are P2P networks, and these three networks are location-based (in other words, each network connecting peers in particular country, for instance, Australia, Saudi Arabia, and United states respectively). In addition, the network peers are newspapers websites, all of them showing weather forecasts for the large cities in the newspaper's home country. For example, in Australian network, the peers are Herald Sun, Daily Telegraph, The Advertiser, The Age, etc.

Instead of having separate connections between *weather.com* and each of these newspaper websites and having to send continuous calls to get weather forecast for Australian cities (for example, Adelaide, Brisbame, Melbourne, and Sydney), one of the peers (for example, Herald Sun) will be nominated to connect the WSP directly and cache the resulted data in a data store. This path will be stored in the WSP to redirect other peers to call Herald Sun website to request weather information about the mentioned Australian cities. Therefore, if Daily Telegraph website, for example, wants to invoke the forecast service, the following service call will be sent to the weather.com:

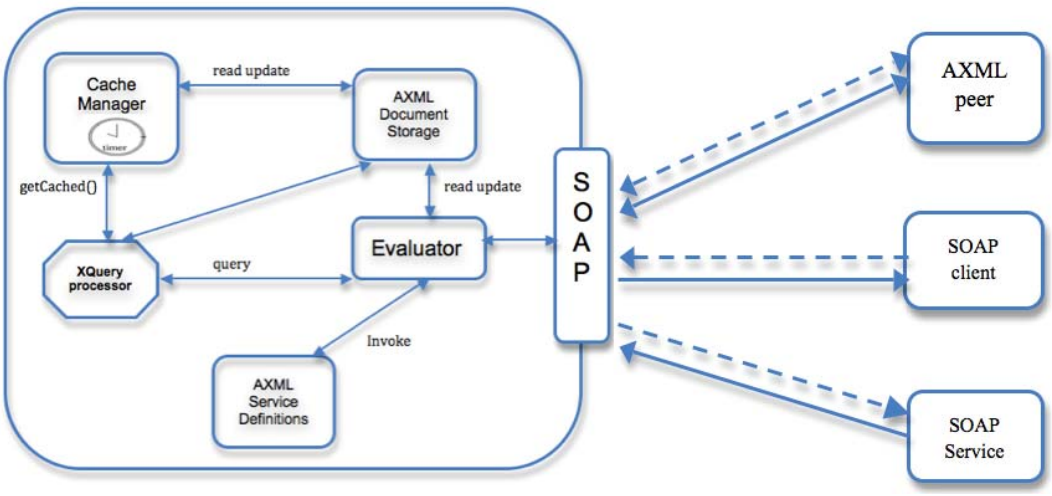


Figure 3 AXML Peer Components

```
<weather>
  <axml:call service="forecast@weather.com">
    <country>Australia</country>
      <city>Adelaide</city>
      <city>Brisbane</city>
      <city>Melbourne</city>
      <city>Sydney</city>
   </axml:call>
</weather>
```

However, as the Herald Sun website is a subscriber in the service and it is the nominated peer in the network to cache this service, the weather.com will reply to this call by sending an intensional data that has a call to the cached information in the Herald Sun. For instance:

```
<weather>
  <sc>
heraldsun.com/getCached(forecast(Adelaide,
Brisbane, Melbourne, Sydney)) </sc>
</weather>
```

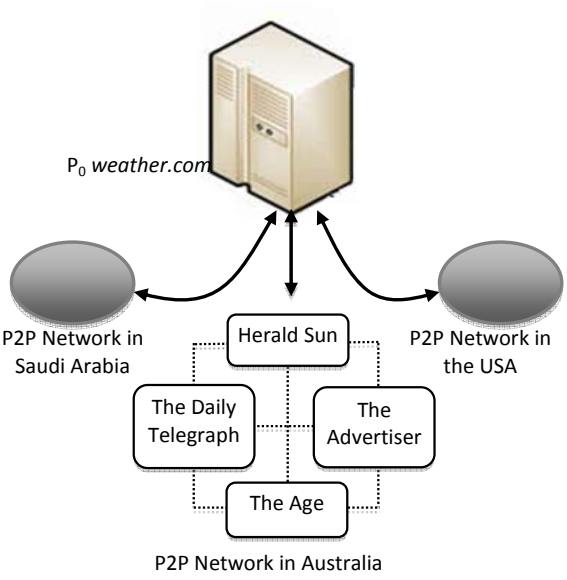


Figure 4. Case Study Illustration

Afterwards, Daily Telegraph server will use this service call to get the forecast information for the mentioned cities from the Herald Sun website. Moreover, the weather server may send to the super node (Herald Sun) some intensional data. In other words, requesters can make other calls to invoke some services defined in other different peers, such as a call to *map@bing.com* to bring the map for the required city. Subsequently, the super node will materialize the resulted nested calls to use it locally. Furthermore, the results for the nested calls will also be cached to provide them to other non-AXML peers, who cannot accept the intensional data, and also the AXML peers in the network to reduce the pressure on the actual service providers.

B. Analysis

This section will highlight the strengths of the proposed levels of caches and the functionalities that will be improved by applying such mechanism.

Request Latency. As some of the data in AXML documents is defined intensionally as embedded calls, these calls will be fired to request the corresponding service and replace the intensional data with resulted information. This request can be affected by the network traffic that can lead to response delay of the service provider. However, caching some copies of the frequently requested data in different peers that are interested in the service will improve the performance of the service requests.

Computational Tasks. When a number of AXML peers invoke a Web service, the service provider needs to activate this service and send its result to the requesters in each time the service is being requested. Caching the service results in the Web service provider and scheduling frequent activations to update the cached results can reduce the computation tasks of this service by using the cached documents to response to service calls. In addition, the WSP can redirect the service request to one of the peers that cache the corresponding service result.

Source Capability. The results of the Web service invocations can have other intensional service calls to other local or remote services. However, non-AXML peers cannot deal with the intensional data. In other words, they cannot fire the embedded service call to invoke the required service. Hence, if a non-AXML peer wants to receive data from a WSP, the data sent has to be fully materialized for the requester. Materializing the nested calls that can be embedded in the service result by the WSP and cache the results for of these nested calls can improve the performance of serving the non-AXML peers.

Query Over AXML Documents. In order to query over AXML documents, the intensional data that related to this query needs to be materialized. This is also required if we need change the type of the document from AXML document to any other type, like XML or HTML because for any reason the requesters do not accept intensional data. If the peer that has the AXML document cannot invoke the particular service, either because the peer is missing the right to access the service at that time or the service provider is not responding, the query services and the changing of type will be disabled for this document. Therefore, caching the latest copy of the important service result in the AXML peer or extract these results from the super peer (nominated peer) will solve the issue of the inability to invoke the actual service.

VII. CONCLUSION

The AXML system can be a powerful tool for integration data and services over the web. However, this system has the potential to negatively affect the network conditions. The proposed caching mechanism that is based on distributing the service information to various servers can be an appropriate solution to minimize these

impacts. In this paper we propose caching mechanism that uses three caching layers.

The caching mechanism will be an applicable technique in P2P networks that use the AXML system because AXML system provides appropriate characteristics to facilitate guidelines for the requesters of the services to the appropriate cache by embedding the path to this cache intensionally and exchange them between the network peers.

REFERENCES

[1] A. Ghitescu and E. Taroza, "Active XML documentation", *http://www.activexml.net/axmlv2/resources/axmldoc.pdf*. Jul. 2008.

[2] D. S. Milojicic, V. Kalogeraki, R. Lukose, Nagaraja, K., Pruyne, J., B. Richard, S. Rollins, and Z. Xu, "Peer-to-Peer Computing" *Tech. Report Hewlett- Packard HPL-2002-57*, Jul. 2003

[3] L. Helouet and A. Benveniste, "Distributed Active XML and Service Interfaces", *Tech. Report INRIA 00429433*, Oct. 2009.

[4] M. Marczewski and T. Pankowski, "Data caching in data integration systems based on AXML Technology", *Proc. Database and Expert Syst,. Workshop*, pp. 794-798, 2007.

[5] T. Milo, "Peer-to-peer Data Integration with Active XML" *Proc. Asian Computing Science Conf. (ASIAN),* pp. 11-18, 2005.

[6] M. Xiong, K. Ramamrithan, A.A. Stankovic, D. Towsley, and R. Sivasankaran, "Scheduling Transactions with Temporal Constraints: Exploiting Data Semantics". *IEEE Trans. Knowl. Data Eng.* 14 (5), pp.1155-1166, Sep. 2002.

[7] S. Abiteboul, I. Manolescu, and S. Zoupanos, "OptimAX: optimizing distributed continuous queries", *Proc. Intl. Conf. Data Eng. (ICDE)*, pp. 1564-1567, 2008.

[8] S. Abiteboul, I. Manolescu, and E. Taropa, "A Framework for Distributed XML Data Management", *Proc. Intl. Conf. Extending Database Tech. (EDBT)*, pp.1049-1058, 2006

[9] S. Abiteboul, O. Benjelloun, and T. Milo, (2008). "The Active XML project: an overview", *Very Large Database Journal* 17(5), pp 1019-1040, Aug. 2008.

[10] S. Abiteboul, O. Benjelloun, I. Manolescu, T. Milo, and R. Weber, "Active XML: Peer-to-peer Data and Web Services Integration", *Proc. Intl. Conf. Very Large Databases (VLDB)*, pp.1087-1090, 2002

[11] Active XML home page, *http://www.activexml.net*, Nov. 2010

[12] The Active XML Team, "Active XML Primer", *Tech. Report. GemoReport 307*, Jul. 2002

[13] T. Milo, S. Abiteboul, B. Amann, O. Benjelloun, and F. D. Ngoc, "Exchanging Intensional XML Data." *ACM Trans. Database Syst.* 30(1), pp.1-40, Mar. 2005.

[14] Y. Zhu, "Containment and Equivalence of Active XML documents" *Proc. Intl. Coll. on Computing, Communication, Control and Management*, pp. 590-594, 2008.