Managing Semi-Structured Data Using XPath, XML Trees and Tree Tuples over a Wireless Network

Hesham Elzentani
Faculty of Informatics and Computing
Singidunum University
Belgrade, 11000, SERBIA
hesham342002@gmail.com

Mladen Veinović
Faculty of Informatics and Computing
Singidunum University
Belgrade, 11000, SERBIA
mveinovic@singidunum.ac.rs

Goran Šimić
Faculty of Informatics and Computing
Singidunum University
Belgrade, 11000, SERBIA
gsimic@singidunum.ac.rs

Abstract

XML is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable, the increasing availability of heterogeneous XML informative sources has raised a number of issues concerning how to represent and manage semi-structured data. Although XML sources can exhibit proper structures and contents, differently annotated XML documents may in principle encode related semantics due to subjective definitions of markup tags. Discovering ontology knowledge to infer semantic organization of XML documents has become a major challenge in XML data management. XPath query language is used for selecting nodes and compute values from XML document, tuple the XML tree and address the problem of XML data clustering according to structure with lexical ontology knowledge. SOAP is a protocol specification for exchanging structured information in the implementation of web services in computer networks, in literature it relies on HTTP for message negotiation and transmission. Experiment on real XML database gives evidence that our proposed approach is highly effective in tuples and clustering of XML database, also aims to study the effect of digital signature on the SOAP messages. So this paper proposes two models consist of a Server and a Client connected to each other through a Wireless router over TCP protocol, XML data base is a sample of DBLP with size of 10 MB loaded on the Server and SOAP messages are used to communicate between both terminals. Finally we compared results between No. of replayed nodes, time and XPath queries, also compared the effects of unsigned and signed SOAP messages.

Keywords: *Digital Signature, SOAP message, XML trees, XML tuples, XPath.*

1. INTRODUCTION

XML is touted as the driving-force for representing and exchanging data on the Web. Indeed, the semi-structured and self-describing physiognomy of XML makes it feasible to model a broad variety of data as XML documents, in order to fulfill the promises of the next generation Web. In such a context, a challenge is inferring semantics from XML documents according to the available syntactic information, namely structure and content features. This has several interesting application domains, such as integration of data sources and query processing, that can be seamlessly generalized to any kind of semi-structured data. As a fundamental exploratory data mining task, clustering represents the natural solution to discover common characteristics and specific facets exhibited by XML documents. However, the complexity intrinsic to semi-structured data requires nontrivial effort to define an effective clustering framework. Extracting significant features, modeling document structures and contents, defining an appropriate notion of homogeneity between documents are only some of the issues to be addressed [7].

2. A BRIEF INTRODUCTION TO XPATH EXPRESSIONS

The XML Path Language, is a query language for selecting nodes from an XML document. In addition, XPath may be used to compute values (e.g., strings, numbers, or Boolean values) from the content of an XML document. XPath was defined by W3C. The XPath language is based on a tree representation of the XML document, and provides the ability to navigate around the tree, selecting nodes by a variety of criteria. For example, the XPath expression ***/book/chapter/section*** navigates from the root of a document (designated by the leading slash "/") through the top-level "*book*" nodes, to their "*chapter*" child nodes, and on to their child nodes named "*section*". The result of the evaluation of the entire expression is the set of all the "*section*" nodes that can be reached in this manner. Furthermore, at each step in the navigation the selected nodes can be filtered using qualifiers. A qualifier is a boolean expression between brackets that can test the existence or absence of paths. So if we ask for ***/book/chapter/section[citation]*** then the result is all "*section*" elements that have a least one child element named "*citation*" [2][3][4][5].

3. XML TREES AND PATHS

A *tree T* is a tuple $T = <r_T, N_T, E_T, \lambda_T>$, where $N_T \subseteq \mathbb{N}$ denotes the set of nodes, $r_T \in N_T$ is the distinguished root of *T*, $E_T \subseteq N_T \times N_T$ denotes the (acyclic) set of edges, and $\lambda_T : N_T \mapsto \sum$ is a function associating a node with a label in the alphabet $\sum$. Let Tag, *Att*, and *Str* be alphabets of tag names, attribute names and strings, respectively. An *XML tree XT* is a pair $XT = <T, \delta>$, such that:

1) *T* is a tree defined on the alphabet $\sum = Tag \cup Att \cup \{S\}$, where symbol $S \notin Tag \cup Att$ is used to denote the ***#PCDATA*** content model;
2) given $n \in N_T$, $\lambda_T(n) \in Att \cup \{S\} \Leftrightarrow n \in Leaves(T)$;
3) δ: $Leaves(T) \mapsto Str$ is a function associating a string to a leaf node of *T*.

An *XML path p* is a sequence $p = s_1.s_2.\ldots.s_m$ of symbols in $Tag \cup Att \cup \{S\}$. Symbol s_1 corresponds to the tag name of the document root element. An XML path can be of two types: *tag path*, if $s_m \in Tag$, or *complete path*, if $s_m \in Att \cup \{S\}$. We denote as P_{XT} the set of complete paths in *XT*.

Let $XT = \langle T, \delta \rangle$ be an XML tree, and $p = s_1.s_2.\ldots.s_m$ be an XML path. The application of *p* to *XT* identifies a set of nodes $p(XT) = \{n_1, \ldots, n_h\}$ such that, for each $i \in [1..h]$, there exists a sequence of nodes, or *node path*, $np_i^p = [n_{i_1}, \ldots, n_{i_m}]$ with the following properties:

1) $n_{i_1} = r_T$ and $n_{i_m} = n_i$;
2) $n_{i_{j+1}}$ is a child of n_{i_j} , for each $j \in [1..m\text{-}1]$;
3) $\lambda(n_{i_j}) = s_j$, for each $j \in [1..m]$.

Moreover, we say that the application of a path to an XML tree yields an *answer*, which is defined depending on the type of path. In case of a tag path *p*, the answer of *p* on *XT* is exactly the set of node identifiers *p(XT)*, that is $A_{XT}(p) \equiv p(XT)$. For a complete path *p*, the answer of *p* on *XT* is defined as the set of string values associated to the leaf nodes identified by *p*, that is $A_{XT}(p) = \{\delta_T(n) \mid n \in p(XT)\}$ [7][9].

4. XML TREE TUPLES

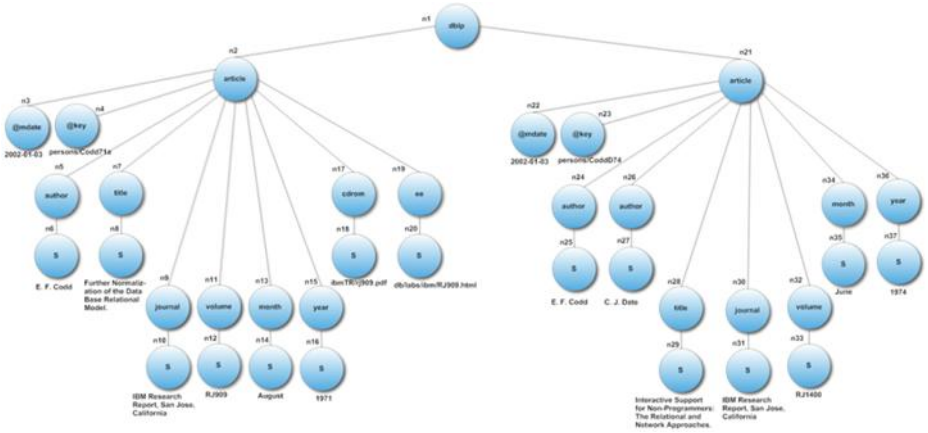
Tree tuples resemble the notion of tuples in relational databases and have been proposed to extend functional dependencies to the XML setting [8][9]. In a relational database, a tuple is a function assigning each attribute with a value from the corresponding domain. According to [8], we provide the following definition.

4.1 Definition

Given an XML tree *XT* , a tree tuple T is a maximal subtree of *XT* such that, for each (*tag or complete*) path *p* in *XT* , the answer $A_T(p)$ contains at most one element. We denote as T_{XT} the set of tree tuples from *XT*. Intuitively, a tree tuple is a (sub) tree representation of a complete set of distinct concepts that are correlated according to the structure semantics of the original tree. Moreover, tree tuples extracted from the same tree maintain identical structure while reflect different ways of associating content with structure as they can be naturally inferred from the original tree. For example consider the XML tree shown in Figure 1, which represents two journal articles from the DBLP (year 2013) archive. Any internal node has a unique label denoting a tag name, whereas each leaf node is labeled with either name and value of an attribute, or symbol ***S*** and a string corresponding to the ***#PCDATA*** content model. Path answers can be easily computed: for example, path ***dblp.article.title*** yields the set of node identifiers $\{n_7, n_{28}\}$, whereas path ***dblp.article.author.S*** yields the set of strings {'E. F. Codd', 'C. J. Date'}. Three tree tuples can be extracted from the example tree (see Figure 2). One tree tuple is extracted starting from the left subtree rooted in the *dblp* element. Two tree tuples are instead extracted starting from the right subtree rooted in *dblp*, as in this subtree there are two paths ***dblp.article.author***, each of which yields a distinct path answer corresponding to the articles' author [7][9].

```
<dblp>
 <article mdate="2002-01-03" key="persons/Codd71a">
  <author>E. F. Codd</author>
  <title>Further Normalization of the Data Base Relational Model.</title>
  <journal>IBM Research Report, San Jose, California</journal>
  <volume>RJ909</volume>
  <month>August</month>
  <year>1971</year>
  <cdrom>ibmTR/rj909.pdf</cdrom>
  <ee>db/labs/ibm/RJ909.html</ee>
 </article>
<article mdate="2002-01-03" key="persons/CoddD74">
  <author>E. F. Codd</author>
  <author>C. J. Date</author>
  <title>Interactive Support for Non-Programmers: The Relational and Network Approaches</title>
  <journal>IBM Research Report, San Jose, California</journal>
  <volume>RJ1400</volume>
  <month>June</month>
  <year>1974</year>
 </article>
</dblp>
```

(a)



(B)

FIGURE 1*: Example of DBLP XML document and its tree*

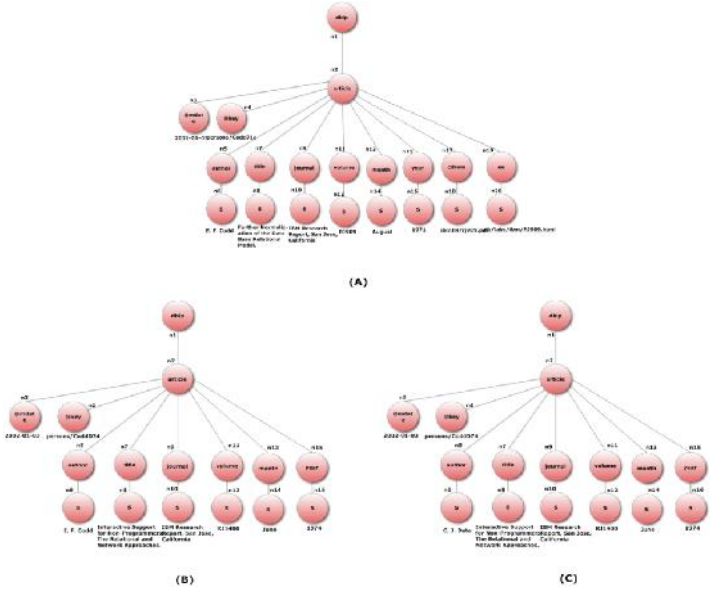


FIGURE 2*: The tree tuples extracted from the XML tree of Figure 1(b).*

5. A BRIEF INTRODUCTION TO DIGITAL SIGNATURE

XML Signature is a joint effort between W3C and IETF, its add authentication, data integrity, and support for nonrepudiation to the data that they sign. XML signature has been designed to both account for and take advantage of the Internet and XML. A fundamental feature of XML Signature is the ability to sign only specific portions of the XML tree rather than the complete document. This flexibility will also be critical in situations where it is important to ensure the integrity of certain portions of an XML document, while leaving open the possibility for other portions of the document to change An XML signature can sign more than one type of resource. Signature validation requires that the data object that was signed be accessible. XML signature itself will generally indicate the location of the original signed object, which can be referenced by a URI within the XML signature, reside within the same resource as the XML signature, be embedded within the XML signature or has its XML signature embedded within itself. The following listing illustrates the structure of XML Digital Signature [1]:

<Signature>

<SignedInfo>

<CanonicalizationMethod/>

<SignatureMethod/>

<Reference>

<Transforms>

<DigestMethod>

<DigestValue>

</Reference>

<Reference/> etc.

</SignedInfo>

<SignatureValue/>

<KeyInfo/>

<Object/>

</Signature>

6. SIMPLE OBJECT ACCESS PROTOCOL

SOAP is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks. It has three major characteristics: Extensibility (security and WS-routing are among the extensions under development), Neutrality (SOAP can be used over any transport protocol such as HTTP, SMTP, TCP) and Independence (SOAP allows for any programming model). SOAP is a mechanism for inter-application communication between systems across the net-work, where system implementations can be written in arbitrary languages. SOAP messages are in XML format to allow the exchange of structured information. A SOAP message is a one-way transmission between SOAP nodes, from a SOAP sender to a SOAP receiver, but messages can be

combined to construct more complex inter-actions, such as request and response, and peer-to-peer conversations. A SOAP message is encoded as an XML document, consisting of an <*Envelope*> element, which contains an optional <*Header*> element, and a mandatory <*Body*> element. The <*Fault*> element, contained within the <*Body*>, is used for reporting errors (see Figure 3) [6].

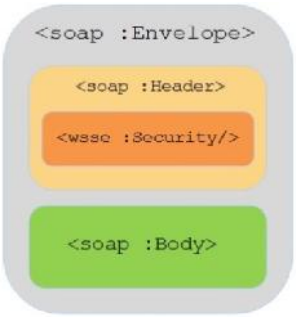


FIGURE 3*: SOAP Message.*

7. SYSTEM SPECIFICATION

In our experiment we use a sample of XML database of DBLP (Computer Science Bibliography) of year 2013 with size of 10 MB, our models consists of a Server and a Client connected to each other through a Wireless router over TCP protocol, the both models was coded in Java using NetBeans IDE 7.4 Beta.

7.1 Server Specifications

Processor is Intel ® Core ™ i5-2430 M CPU @ 2.40 GHz 2.40 GHz, RAM of 6 GB, hard disk of 700 GB, operating system is Windows 7 Home Premium, Service Pack 1 (64-bit) and wireless adapter is Atheros AR9002WB-1NG.

7.2 Client Specifications

Processor is Intel ® Core ™ i3 CPU M370 @ 2.40 GHz 2.40 GHz, RAM of 2 GB, hard disk of 300 GB, operating system is Windows 7 Ultimate, Service Pack 1 (32-bit) and wireless adapter is Qualcomm Atheros AR9285 802.11b/g/n.

7.3 Wireless Router Specifications

Router type is TP-Link 150Mbps N Access Point - model No: TL-WR740N / TL-WR740ND, wireless standards is IEEE 802.11n*, IEEE 802.11g, IEEE 802.11b, antenna of 5 dBi Fixed Omni Directional, frequency of 2.4 - 2.4835 GHz and signal rate of 11n (up to 150Mbps-dynamic), 11g (up to 54Mbps - dynamic) and 11b (up to 11Mbps-dynamic).

8. SYSTEM ACTIVITY

We use ten different types of XPath queries (see Table 1) to measure and count query's time and number of replayed nodes. The models' activity are explained in the following steps:

1. XML DBLP database is loaded on the Server (DOM is used).
2. SOAP messages used to communicate between both terminals (Server and Client).

3. SOAP request message, which contains XPath query in its <*Body*> element, created on Client side and send it to the Server.
4. SOAP replay message, which contains the XPath answer's query in its <*Body*> element, created on Server side and send it to the Client.
5. At Client side we measure query's time (the time taken from SOAP request sent until SOAP replay received in milliseconds) and total number of replayed nodes for all queries.

Query	**XPath Expression**
Q1	/dblp/article/author[3]
Q2	/dblp/article[author[3]]/pages[last()]
Q3	//phdthesis[@mdate=’2012-04-18’]
Q4	/dblp/phdthesis/url
Q5	/dblp/inproceedings/*
Q6	/dblp/inproceedings[year>=1974]
Q7	descendant::book
Q8	//book/isbn
Q9	dblp/www[author and year]
Q10	dblp/www[author or year]

TABLE 1: Queries of XPath expressions.

9. SYSTEM IMPLEMENTATION

9.1 First Model

In the first model (see Figure 4) we use unsigned SOAP messages to communicate between Client and Server.

We can conclude from the results which are illustrated in Table 2 and Figure 5, a proportional relation between answers of XPath Queries and replayed time (Server answer time), in other words when the number of replayed nodes increased; the time also increased, that will depend on the tree tuples of XML Database and how its clustered using XPath.

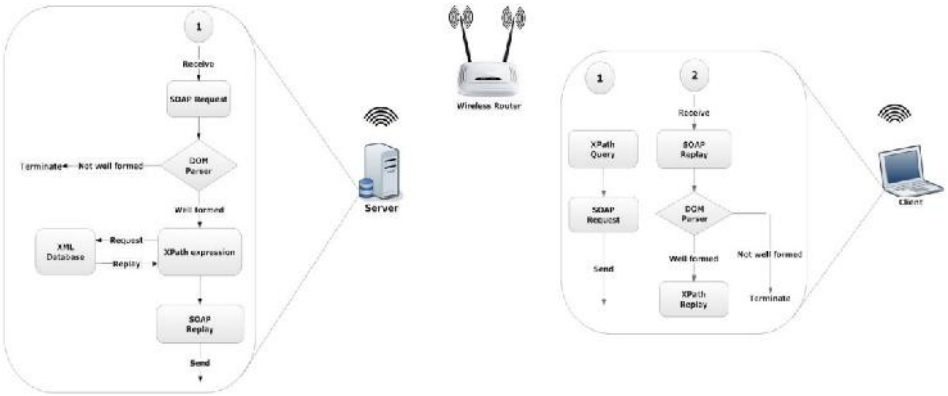


***FIGURE 4**: First model*

Query	Query answer of DBLP XML Database	
	Number of Replayed Nodes	Time (ms)
Q1	7583	530
Q2	6982	312
Q3	6020	328
Q4	3615	343
Q5	16	172
Q6	2	156
Q7	1	187
Q8	1	234
Q9	7	156
Q10	17	140

***TABLE 2**: Query's results of first model.*

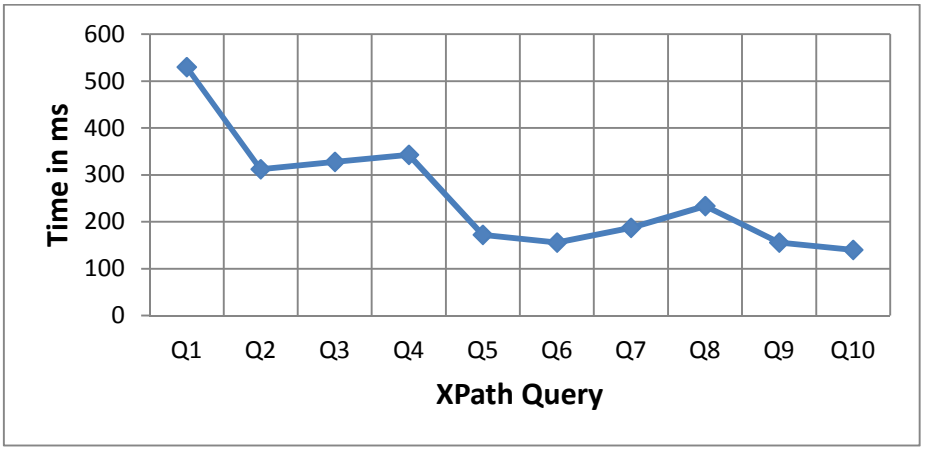


***FIGURE 5**: XPath Queries and replayed time of unsigned SOAP.*

9.2 Second Model

In the second model (see Figure 6) we use signed SOAP message to communicate between Client and Server, because its secure and guarantee that no one in the middle (third party) can alter the queries. We can conclude from the results which are illustrated in Table 3 and Figure 7, a proportional relation between answers of XPath Queries and replayed time (Server answer time), in other words when the number of replayed node increased; the time also increased, that will depend on the tree tuples of XML Database and how its clustered using XPath, also we see a little bit increasing in time because of signing and verifying a digital signature of SOAP message in both terminals (Server and Client).

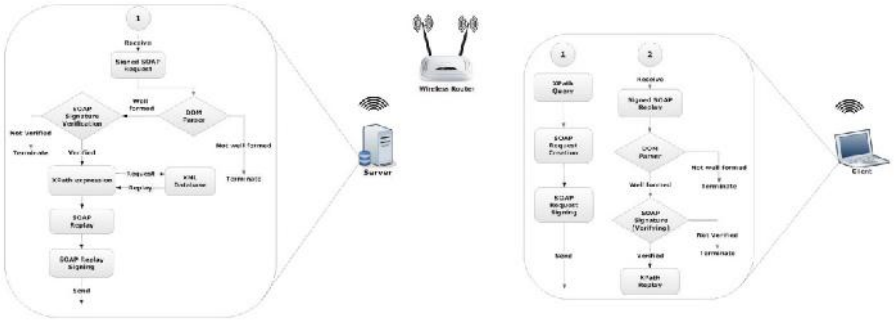


***FIGURE 6**: Second model.*

Query	Query answer of DBLP XML Database	
	Number of Replayed Nodes	**Time (ms)**
Q1	7583	858
Q2	6982	390
Q3	6020	374
Q4	3615	437
Q5	16	202
Q6	2	234
Q7	1	234
Q8	1	359
Q9	7	219
Q10	17	203

***TABLE 3**: Query's results of second model.*

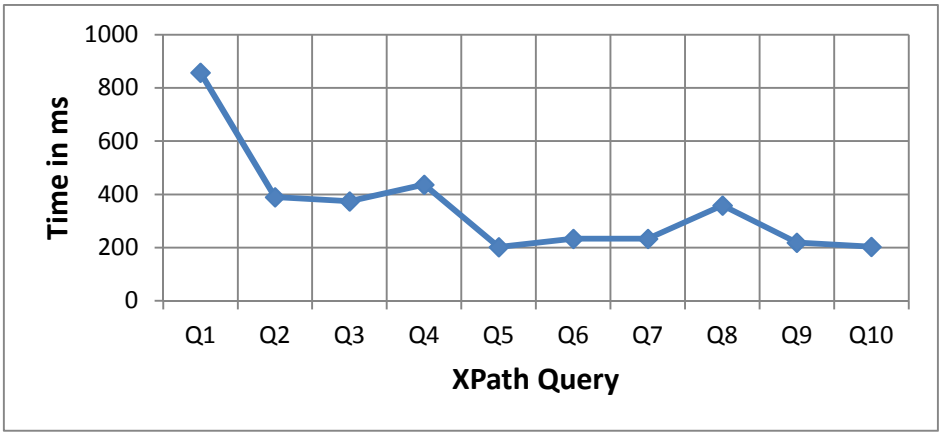


***FIGURE 7**: XPath Queries and replayed time of signed SOAP.*

9.3 COMPARISONS BETWEEN BOTH MODELS

Figure 8 shows little bit difference in time between both models, because of signing and verifying XML digital signature of SOAP messages will take a bit more time in the second model, but we can reduce this time using fast and high performance processors.

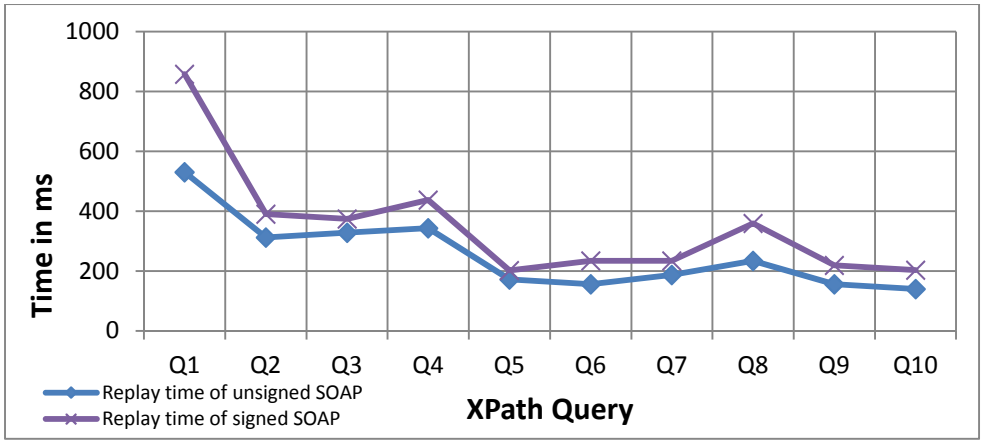


***FIGURE 8**: Comparison between results of both models.*

10. CONCLUSION

Extensible Markup Language (XML) has become a standard of data exchange and representation in many application. So in this paper we used a sample XML database from DBLP (Computer Science Bibliography of year 2013) of size 10 MB. Our models consists of a Server and a Client connected to each other through a wireless router over TCP protocol, XML DBLP database was loaded on the Server and SOAP messages (Unsigned and Signed SOAP messages) were used to communicate between both terminals, further, XPath query loaded on Client side and the Server replays for Client's query. The both models were completely coded in Java using NetBeans IDE 7.4 Beta. Results of both models showed that the number of replayed nodes (queries answers) are directly proportional to the queries' time, in other words the increasing in the replayed nodes would cause time to increase (take in account the time of signing and verifying XML digital signature of SOAP messages). Something more, the queries context themselves and the positions of nodes in the XML tree would affect queries' answer and their time.

From Table 2 and Table 3 we saw there were no changes in the number of replayed nodes for both models. Figure 8 showed a comparison between both models, we saw a little bit difference in time between both models, because of XML digital signature and signature verification of SOAP messages will take a bit more time in the second model, but we can reduce this time using fast and high performance processors.

11. REFERENCES

[1] Ed Simon, Paul Madsen, Carlisle Adams, "An Introduction to XML Digital Signatures", 2001.

[2] "XML and Semantic Web W3C Standards Timeline", February 04,2012.

[3] Bergeron, Randy, "XPath-Retrieving Nodes from an XML Document", October 31, 2000.

[4] Pierre Geneves, "Course- The XPath Language", October 2012.

[5] Pierre Geneves, "Logics for XML", December 2006.

[6] Sebastian Gajek, Lijun Liao and Jörg Schwenk, "Breaking and Fixing the Inline Approach".

[7] Andrea Tagarelli, Sergio Greco, "Toward Semantic XML Clustering".

[8] M. Arenas and L. Libkin, "A Normal Form for XML Documents", ACM Trans. Database Systems, 29(1):195–232, 2004.

[9] S. Flesca, F. Furfaro, S. Greco, and E. Zumpano, "Repairs and Consistent Answers for XML Data with Functional Dependencies", In Proc. Int. XML Database Sym-posium (XSym), pages 238–253, 2003.