Server-Centric P3P

Rakesh Agrawal   Jerry Kiernan   Ramakrishnan Srikant   Yirong Xu

Abstract

We propose a server-centric architecture for P3P that reuses database technology for implementation, as opposed to the prevailing client-centric implementations based on specialized engines. The server-centric implementation has several advantages including: setting up the infrastructure necessary for ensuring that web sites act according to their stated policies, allowing P3P to be deployed in thin, mobile clients that are likely to dominate Internet access in the future, and allowing site owners to refine their policies based on the privacy preferences of their users. Our experiments indicate that it performs significantly better than the sole public-domain client-centric implementation and that the latency introduced by preference matching is small enough for real-world deployments of P3P. We believe a good future direction for P3P would be to standardize on the server-centric architecture as an alternative to the current client-centric architecture.

1. Introduction

Platform for Privacy Preferences (P3P), developed by the World Wide Web Consortium (W3C), is the most significant effort underway to enable users to gain more control over what information a web-site collects. It provides a way for a web site to encode its data-collection and data-use practices in a machine-readable XML format, known as a P3P policy [7], which can be programmatically compared against a user's privacy preferences. A user may specify privacy preferences in APPEL[1], which provides an XML format for expressing preferences and an algorithm for matching preferences against policies [6]. A web site can have different privacy policies governing different parts of the site. P3P provides for a *Reference File* in which a site can set up associations between web pages and policies.

In this paper, we propose a server-centric architecture for P3P that reuses database technology, as opposed to the prevailing client-centric implementations based on specialized engines. The server-centric architecture has several advantages (discussed in Section 4) including: setting up the infrastructure necessary for ensuring that web sites act according to their stated policies, allowing P3P to be deployed in thin, mobile clients that are likely to dominate Internet access in the future, and allowing site owners to refine their policies based on the privacy preferences of their users. Our experiments indicate that the proposed server-centric architecture performs significantly better than the sole public-domain client-centric implementation [8]. These experiments also show that the proposed architecture has the necessary performance for it to be used in practical deployments of P3P.

We assume familiarity with the P3P specifications [6] [7]. We also assume familiarity with the basic concepts of XML. Throughout the paper, we use "element" and "attribute" as in the XML specification.

This position statement is an abbreviated version of [2], which has been accepted for publication in the 19th International Conference on Data Engineering (ICDE 2003).

[1]P3P does not require that APPEL be necessarily used as the language for expressing privacy preferences. However, we are not aware of any other viable alternative.

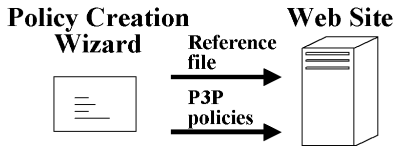


Figure 1. Creation and Installation of Policies (Client-Centric)

2. Current Client-Centric Architecture for P3P

A hypothetical architecture for implementing P3P has been described in [9]. There are two parts to deploying P3P. Web sites first create and install policy files at their sites (see Figure 1). Then as the users browse a web site, their preferences are checked against a site's policy before they access the site (see Figure 2).

There are two prominent implementations of the above architecture: Microsoft IE6 and AT&T Privacy Bird.

3. Server-Centric Architecture for P3P

We propose a server-centric architecture for deploying P3P as an alternative to the prevailing client-centric architecture. In this architecture, a website deploying P3P first

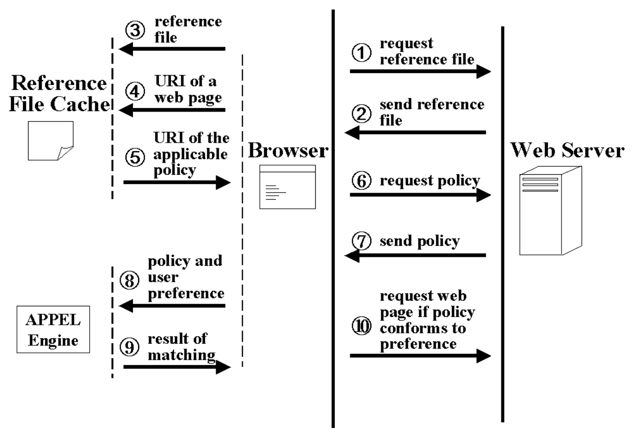


Figure 2. Policy-Preference Matching (Client-Centric)

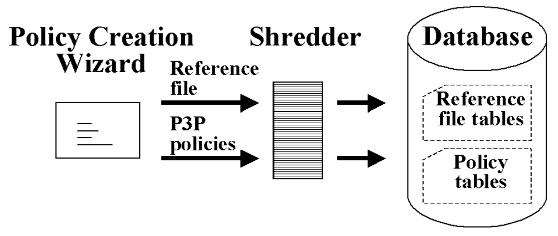
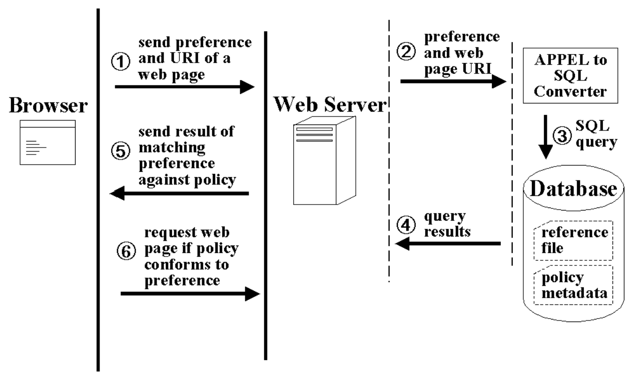


Figure 3. Creation and Installation of Policies (Server-Centric)



Browser
Web Server
APPEL to SQL Converter
Database
① send preference and URI of a web page
⑤ send result of matching preference against policy
⑥ request web page if policy conforms to preference
② preference and web page URI
③ SQL query
④ query results
reference file
policy metadata

Figure 4. Policy-Preference Matching (Server-Centric)

installs its privacy policies in a database system as shown in Figure 3. Then database querying is used for matching a user's preferences against privacy policies as shown in Figure 4.[2]

We envision three variations of this architecture:

1. Convert privacy policies into relational tables [3] [5], and convert an APPEL preference into an SQL query for matching.
2. Store privacy policies in relational tables, define an XML view over them [3] [5], and use an XQuery [4] derived from an APPEL preference for matching.
3. Store privacy policies in a native XML store and use an XQuery derived from an APPEL preference for matching.

Due to space constraint, this paper only discusses the first architecture in which the policies are stored in relational tables and APPEL preferences are converted into SQL queries for matching. See [2] for the results of our study of the latter two architectures. In the experiments reported therein, these architectures did not fare well compared to the SQL variant, indicating yet untapped performance enhancement opportunities in the XML stores.

3.1. Other Alternatives

There are two orthogonal dimensions in the space of choices for implementing P3P:

1. What engine should be used for matching a preference against a policy? Should it be a specialized engine (e.g. a native APPEL engine) or should it be a general purpose engine (e.g. a database engine)?
2. Where should the matching take place? Should it happen at the client or the server?

	Client	Server
Specialized engine	Current	?
Database engine	?	Proposed

Figure 5. Architectural Choices

Figure 5 shows the decision matrix. The current P3P deployments are using specialized APPEL engines to do preference matching at the clients. It is possible to continue to use a specialized engine, but move the matching to the server. However, this choice is less attractive as we lose the benefits of using the database engine for matching. Similarly, it is possible to do the checking at the client, but

[2]We are assuming that the client preferences will continue to be expressed in APPEL and they will be translated into database queries before the matching takes place. This translation step may become unnecessary should the proposed architecture catch on. For in that case, database queries may replace APPEL for representing privacy preferences and the GUI tools for generating preferences may directly generate database queries.

use database querying. This alternative has the advantage of avoiding the need for a specialized engine. However, it will require moving the database tables from the server to a light-weight main memory database system in the client, which is also not very attractive. We have, therefore, focussed on the alternative of using database querying at the server.

4. Advantages and Disadvantages

The following are some of the advantages of the server-centric architecture of P3P over the client-centric architecture:

- The preference checking at the client leads to heavier clients, which is a problem for thin, mobile devices that are likely to dominate Internet access in the future. Our proposal allows for lean clients.
- An upgrade in P3P specification may require an upgrade in every client, which can be a couple of orders of magnitude greater effort than upgrading all the servers.
- As new privacy-sensitive applications emerge, they will each require building preference checking into them, rather than reusing checking done at the server.
- Site owners can refine their policies if they know what policies have a conflict with the privacy preferences of their users. The current architecture does not allow the site owners to obtain this information.

Using databases for preference matching (in the server-centric architecture) yields the following additional advantages:

- We are creating the infrastructure necessary for enhancing P3P with enforcement in the future. The privacy data tables built for checking preferences against policies may serve as meta data for ensuring that policies are followed.
- Specialized preference checking engines are reinventing querying. We, on the other hand, can reuse the proven database technology for checking preferences against policies.
- Policies of a website will not stay static forever. Versions of policies can be better managed using a database system than the current file system based implementations.

The server-centric architecture has some disadvantages too, including:

- There needs to be a greater amount of trust in the server. For example, the server can see the user's preferences. Similarly, the user has to trust the database software being used by the server, whereas the user can (in principle) choose the checking software used in the browser.
- By caching a reference file, the client may avoid some checks, assuming a user visits many pages that are governed by the same policy. On the other hand, it is possible to design a hybrid architecture in which the reference file processing is done at the client while the preference checking is done at the server.

5. Implementation Experience

We now present the experience from our implementation of the server-centric P3P architecture.

5.1. System Description

Our implementation consists of the following parts:

- *Database Schema for P3P:* We first define a schema for storing policy data in the relational tables. This schema contains a table for every element defined in the P3P policy. The tables are linked using foreign keys reflecting the XML structure of the policies.
- *Storing Policies in the Database:* A given P3P policy is parsed and shredded as a set of records in the tables.
- *Translation of APPEL into SQL Queries:* An APPEL preference may contain multiple rules. These rules are matched against a policy in the order in which they appear in the preference. We translate each rule into a SQL query.
- *Query Evaluation:* SQL queries corresponding to a preference are submitted to the database engine in order. The result of the query evaluation yields the action to be taken.

See [2] for the details of the implementation.

5.2. Experimental Setup

We conduct experiments to study the performance of our database implementation of P3P. Our experiments measured the time to match a P3P policy with an APPEL preference, first using a native APPEL engine and then using a database engine. Both the APPEL engine and the database engine were run on a Windows NT 4.0 server with dual 600 MHz processors and 512 MB of memory.

The APPEL engine we used is available from the Joint Research Center (JRC) [8]. To the best of our knowledge, it is the only APPEL engine currently available in public domain. The database system we used was DB2 UDB 7.2. The policy database was created under DB2's default settings, with the application heap size set to 4 MB.

5.3. Data Set

We used 29 P3P policies in our experiments. They were obtained by crawling the web sites of the Fortune

Preference	#Rules	Size (KB)
Very High	10	3.1
High	7	2.8
Medium	4	2.1
Low	2	0.9
Very Low	1	0.3
Average	4.8	1.9

Figure 6. JRC APPEL preferences

	APPEL	SQL		
	Engine	Convert	Query	Total
Average	2.63	0.08	0.08	0.16
Max	9.08	0.14	0.24	0.34
Min	0.98	0.04	< 0.001	0.04

Figure 7. Execution time for matching a preference against a policy (seconds)

1000 companies looking for P3P policies. We found 29 companies with P3P policies, including companies such as AT&T, IBM, McGraw Hill, and Progressive Insurance Group. Sizes of these policies vary from 1.6 to 11.9 KBytes, with the average size being 4.4 KBytes. These policies contained a total of 54 statements (about 2 statements per policy on average).

We used 5 APPEL preferences in our experiments. These preferences were taken from the JRC site [8] and constitute their test suite. JRC designed these preferences for different levels of sensitivity for privacy: Very High, High, Medium, Low, and Very Low. Figure 6 lists the sizes of preferences and the number of rules they contained.

5.4. Performance Results

5.4.1 Shredding

We measured the time needed for shredding each of the 30 privacy policies and storing the shredded policies into privacy tables in DB2 as per the schema defined in Section 5.1. The average shredding time was 3.19 seconds, with the maximum and minimum being 11.94 and 1.17 seconds respectively. Since a policy changes infrequently, the lifetime cost of shredding can be considered negligible.

5.4.2 Matching

Figure 7 shows the performance of matching preferences against policies for the the native APPEL and SQL implementations. Each preference was matched against every policy. The figure shows the average, maximum, and minimum in seconds for matching a preference against a policy. For the SQL implementation, we separate the time needed for converting APPEL into SQL (conversion time) and the time needed for matching (query time). The total time is the sum of conversion and query times. Figure 8 shows the performance numbers broken down per five preference types.

Preference	APPEL	SQL		
	Engine	Convert	Query	Total
Very High	2.65	0.09	0.08	0.17
High	2.68	0.10	0.14	0.24
Medium	2.66	0.13	0.14	0.27
Low	2.60	0.06	0.03	0.09
Very Low	2.54	0.04	< 0.01	0.05

Figure 8. Per-preference-type execution times for matching a preference against a policy (seconds)

The numbers shown in Figure 7 and 8 are the "warm" numbers. They reflect the time likely to be experienced in deployed systems. The system was warmed up by first matching an extra (artificial) preference and discarding this time. This factors out one-time costs such as the JVM loading the classes. The difference between the warm and cold average matching times was about 1.4 seconds for the native APPEL engine and 1 second for the SQL implementation. For the SQL implementation, we stopped and restarted DB2 after matching each preference to avoid any advantage due to DB2 query caching.

Several conclusions can be drawn from these figures. First is the surprisingly good performance of the SQL implementation when compared to the native APPEL engine. We would have been satisfied if the SQL implementation came close to the APPEL implementation. But the SQL implementation turns out to be more than 15 times faster, even with the conversion time included in the SQL numbers. If we just compare the matching time, the SQL implementation is 30 times faster. The latter is a meaningful comparison as it is not unreasonable to think of a P3P deployment in which the preference generation GUI tool produces preferences as a set of SQL statements.

To understand this large performance difference, we profiled the APPEL engine. Before matching a preference against a policy, the APPEL engine first augments every data element in the policy with the corresponding categories predefined in the P3P base schema (see Section 5.4.6 in [6]). We found that this augmentation accounts for most of the difference in performance. In a client-centric architecture, the APPEL engine running in the client has to incur this cost for every preference checking. Our SQL implementation, on the other hand, does this expansion while shredding the policy into relational tables, and incurs no corresponding cost at the time of preference checking. Since a policy changes infrequently, the cost of shredding amortized over a large number of matchings of different preferences against a policy can be considered negligible.

More important than the relative comparison is the absolute time needed for matching preferences against policies. Figures 7 and 8 show that the latency introduced by our SQL implementation for preference matching is more than acceptable for it to be used in practical P3P deployments.

6. Conclusion

We proposed a reference architecture for a server-centric implementation of P3P and showed that it can be implemented by reusing the database querying technology, as opposed to the prevailing client-centric implementations based on specialized engines. Not only does the proposed architecture have qualitative advantages, our experiments indicate that it performs significantly better than the sole public-domain client-centric implementation and that the latency introduced by preference matching is small enough for real-world deployments of P3P.

The biggest advantage of a server-centric architecture is that it sets us up for adding enforcement mechanisms to ensure that sites act according to their stated policies. The absence of enforcement mechanism is the major criticism of the current P3P standard. We have outlined the architectural elements needed in a data system for such enforcement in [1]. A good future direction for P3P would be to standardize on the server-centric architecture and add enforcement mechanisms.

References

[1] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic databases. In *28th Int'l Conference on Very Large Databases*, Hong Kong, China, August 2002.

[2] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Implementing P3P using database technology. In *19th Int'l Conference on Data Engineering*, Bangalore, India, March 2003.

[3] M. Benedikt, M. Fernandez, J. Freire, and A. Sahuguet. XML and data management. In *WWW-2002 Tutorial*, Honolulu, Hawaii, May 2002.

[4] S. Boag, D. Chamberlin, M. F. Fernandez, D. Florescu, J. Robie, J. Simeon, and M. Stefanescu, editors. *XQuery 1.0: An XML Query Language*. W3C Working Draft, April 2002.

[5] S. Chaudhuri and K. Shim. Storage and retrieval of XML data using relational databases. In *VLDB Tutorial*, Roma, Italy, 2001.

[6] L. Cranor, M. Langheinrich, and M. Marchiori. *A P3P Preference Exchange Language 1.0 (APPEL1.0)*. W3C Working Draft, February 2001.

[7] L. Cranor, M. Langheinrich, M. Marchiori, M. Presler-Marshall, and J. Reagle. *The Platform for Privacy Preferences 1.0 (P3P1.0) Specification*. W3C Recommendation, April 2002.

[8] JRC P3P Resource Centre. http://p3p.jrc.it.

[9] The World Wide Web Consortium. *P3P 1.0: A New Standard in Online Privacy*. Available from `http://www.w3.org/P3P/brochure.html`.