
World Wide Web, 4: 121–140 (2001)

A Web-Based Knowledge Network for Supporting Emerging Internet Applications

MINSOO LEE Minsoo.Lee@oracle.com
2op1087 Oracle Corporation, 200 Oracle Parkway, Redwood Shores, CA 94065, USA

STANLEY Y. W. SU and HERMAN LAM {su;hlam}@cise.ufl.edu
Database Systems R&D Center, University of Florida, Gainesville, FL 32611, USA

Abstract

Although the Internet and the World Wide Web technologies have gained a tremendous amount of popularity among people and organizations, the network that these technologies created is not much more than a multimedia data network. It provides tools and services for people to browse and search for data but does not provide the facilities for automatically delivering the relevant information for supporting decision-making to the right people or applications at the right time. Nor does it provide the means for users to enter and share their "knowledge" that would be useful for making the right decisions. In this work, we introduce the concept of a Web-based knowledge network, which allows users and organizations to publish, not only their multimedia data, but also their knowledge in terms of events, parameterized event filters, customizable rules and triggers that are associated with their data and application systems. Operations on the data and application systems may post events over the Internet to trigger the processing of rules defined by both information providers and consumers. The knowledge network is constructed by a number of replicable software components, which can be installed at various network sites. They, together with the existing Web servers, form a network of knowledge Web servers.

Keywords: knowledge network, knowledge publishing, event filtering, rule service, knowledge Web server

1. Introduction

In recent years, the Internet has been widely used by people and organizations throughout the world. People can communicate quickly and easily with almost anyone anywhere in the world by e-mails. Many business enterprises are aggressively putting their home pages on the Internet, and buying and selling things over the Internet has become very common. It is evident that we are experiencing an explosive growth in the usage of the Internet. But, at the same time, we recognize several limitations of the current Internet technology. These limitations are preventing us from developing more advanced applications on the Internet. The fundamental problem with the current Internet is that it merely creates a passive network of multimedia data. The Internet needs to become more active, collaborative, and intelligent. We believe that these three characteristics form the new infrastructure requirements for emerging applications on the Internet. The first requirement is to make the Internet "active." Web servers currently just return the data that was requested by a remote user. Ideally, they should have a mechanism to automatically react to external events that are of interest to users and spontaneously perform relevant operations instead of simply

acting as managers of passive data repositories. Examples of such external events are: the generation of new data can be an event that triggers notification of many users who may be interested in the new data, and the updating of a Web page may trigger the updating of replicated copies that have been cached in other Web servers. If such a mechanism is provided, the Internet infrastructure will require fewer human interactions. Transforming the Internet from a passive infrastructure to an active infrastructure is one of our goals. The second new requirement is to make the Internet "collaborative." The data on the Web servers are currently isolated from one another. No Web server can spontaneously disseminate data to other Web servers in order to establish a collaborative relationship. The Internet infrastructure needs to provide a mechanism for Web servers to contact other Web servers and push data to them automatically, instead of simply waiting for an HTTP request and providing data to a user. Therefore, we focus on the development of a data pushing mechanism in our framework. The third new requirement is to add "intelligence" to the Internet. Although adding new data into the Web and allowing the Internet community to access them have become a common practice, embedding decision-making procedures and rules into the Web servers is not supported at present. Such a capability will make the Internet infrastructure much more intelligent.

Recognizing the above new requirements, we propose a general framework and the architecture of a knowledge network. The knowledge network allows both consumers and providers of multimedia data to express their knowledge in forms of events, parameterized event filters, customizable rules and triggers that are associated with the data and the operations on the data. The contributed knowledge can be incorporated into the current data network. This knowledge is gathered and stored by software components, which are extensions to the Web servers. The intended contributions of this work can be summarized as follows. First, a new infrastructure, namely, the knowledge network, is designed and implemented to support and promote new and emerging applications on the Internet. The knowledge network infrastructure reduces the need for constant human attention and enables more machine processing to be exploited by integrating knowledge into the Internet. Second, an event-trigger-rule (ETR) model, a high-level language and its associated GUIs have been designed and developed to allow high-level specifications of events, parameterized event filters, rules, and triggers. Third, techniques for event filtering, event history management, event notification, the handling of customizable rules and dynamic rules, and parallel rule processing are also introduced. Together, they enable Internet-based collaborations and the development of distributed applications. Fourth, a set of Web-based tools and software components has been developed to provide value-added services, which complement the Internet and Web services. They include knowledge publishing, event registration, event filtering and notification, knowledge profile management, and management and processing of triggers and rules. The organization of the remainder of this paper is as follows. In Section 2, some related research on events and rules on the Internet are surveyed. In Section 3, our event-trigger-rule model is presented. In Section 4, the architecture and the features of the knowledge network are explained. Section 5 gives the implementation details of the component modules. In Section 6, an example scenario that demonstrates the usefulness of the knowledge network is provided. Finally, Section 7 gives a summary of this work together with suggestions for future work.

2. Related research

The key technologies that have motivated us to pursue our research are the publishing and sharing of data on the Internet, notification services via the Internet, and rule technology.

The technology to allow people and companies to publish their data started out as HTML [35]. HTML focuses on the display format and is suitable for human interactions. CGI, JavaScript, applets, and servlets [29] have been introduced to provide additional capabilities to HTML. XML [5] is more machine-friendly by using user-defined tags to specify the semantics of the data contained within each part of a document. By knowing the semantics of the tags, the XML document can be processed by a machine (or program), which can easily extract information from the document. XSL [27] specifies rules for displaying each tagged part of the XML document. XML relies on DOM (Document Object Model) [10], which models a document as a tree of objects. These objects altogether form a semi-structured document, which can be easily parsed and also navigated. Our work extends the publishing concept to include knowledge publication on the Internet.

The basic communication paradigm on the Internet so far has been based on the pull model of interaction. In the pull model of interaction, clients need to pull data from the server. Because every request needs to be explicitly initiated by the client, there is a limit on the amount of data that a user can browse and access in the Internet environment, in which an enormous amount of data exists. To remedy this problem, the push model began to gain interest in the research community. The push model of interaction allows subscribers to specify their interests in certain data. The server will then push the data of interest to the subscribers based on their preferences. This model of interaction makes it possible for subscribers to receive data in a timely fashion without additional effort. It also has an additional advantage of scalability when the same data needs to be disseminated to a large number of subscribers. Some early work on push-based systems were Teletext [1], Datacycle at Bellcore [17], and Boston Community Information System (BCIS) at MIT [13]. Push technology-based products include the Pointcast system [28], Marimba's Castanet [23], Netscape's Netcaster [24], and Backwebs polite agent [2].

Event notifications are also a form of the push-based technology. An event is a high-level specification of something of interest that happens, upon which the users and/or software systems that subscribe to the event want to be notified. Products that support event notification services have been introduced, such as the Keryx notification service [19] by KeryxSoft, Vitria's businessware communicator [34], and WebLogic events [3]. In order for the push technology to work, one of the problems to be solved is how to establish a good open standard as a basis for communication.

The concept of rules was originally introduced in the research areas of artificial intelligence and expert systems. The declarative and simple forms of rules were appropriate to specify knowledge, and these started out as the condition-action type of rules. The condition-action type of rules has the semantics of "when a condition is true, perform the action." Expert systems such as OPS5 [6] or CLIPS [12] use this type of rules. The rules were soon incorporated into databases to create a new category of databases, namely, active databases. Some examples of these active database systems are HiPAC [9], Starburst [15], Ode [11], Postgres [30], OSAM*.KBMS [31], Sentinel [7], and Ariel [16].

Event-Condition-Action (ECA) rules have been used in many of these systems. They are composed of three parts: event, condition, and action. The semantics of an ECA rule is, "When an event occurs, check the condition. If the condition is true, then execute the action." The event provides a finer control as to when to evaluate the condition and gives more active capabilities to the database systems. Rules can automatically perform security and integrity constraint checking, alert people of important situations, enforce business policies and regulations, etc.

Some initial approaches toward using rules to integrate servers on the Internet have been experimented. WebRules [4] is a framework developed at the Israel Institute of Technology. The WebRules server has a set of built-in events that can notify remote systems, and has a library of system calls that can be used in a rule to connect Web servers. It does not include concepts such as event publishing or filtering. However, it is one of the first attempts to use rules for the purpose of integrating Web servers. WebLogic [3] also includes a basic form of rules, which are called actions. These actions need to be provided to the WebLogic server at the time when an application is registering for an event. These actions are actually specified by program codes rather than by a high-level specification facility. Thus, the developer of the actions needs to deal with system-level issues.

3. Knowledge model

Our knowledge model, the ETR model, consists of events, triggers, and rules. The event-trigger-rule (ETR) model [20,21] is a generalization and extension of the event-condition-action (ECA) rule model used in active database management systems. The ETR model separates the specifications of events from those of rules. By allowing this separation, the events can be defined independently of the rules and vice versa, which is essential for a distributed environment. In the distributed environment, events can be defined on one site and rules can be defined on another site. The remote event can be associated to a local rule by a trigger specification, which links the event to the rule. Each element of the ETR model is discussed in the following subsections. The complete language syntax for the elements is described in [22].

3.1. Events

Events signal that some things of interest have happened on the Internet. They can signal that new data is made available on a site, a Web page has been modified, an application has executed on a site, etc. An event can have attributes and the data of these attributes is associated with the thing of interest that has happened. The data is passed to the subscribers of this event. The syntax of an event specification is as follows:

```
EVENT          event_name (type attr1, type attr2, . . . , type attrN)
[DESCRIPTION   description_text]
```

The EVENT clause specifies the event name along with the list of event attribute names and types. A simple example of such an event (E3) is shown in Figure 1.

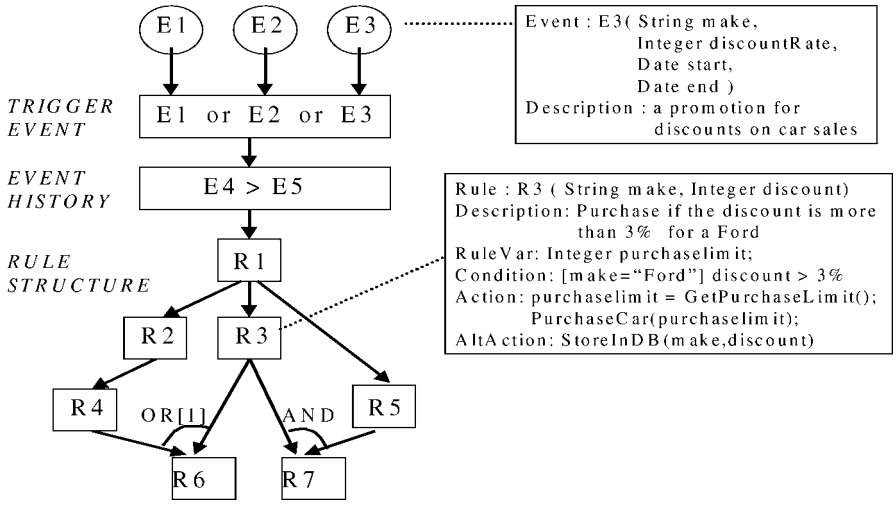

Figure 1. Example of event, trigger, and rule in the ETR model.

3.2. Rules

Rules in the ETR model are defined as Condition-Action-AlternativeAction (CAA) rules. CAA rules provide a very general way for specifying integrity and security constraints, business rules and policies, regulations that are relevant to the operation of a real or virtual enterprise. Each CAA rule represents a small granule of control and logic needed to enforce a constraint, business rule, policy, etc. A number of these rules, when executed in a certain order or structure, can represent a larger granule of control and logic. An example rule (R3) is shown in Figure 1.

A rule can have a number of parameters just like a procedure call and perform some desired operations. The list of parameter names and types are specified together with the rule name. When the rule is invoked, it first evaluates the CONDITION clause of the rule. If the result is true, the operations specified in the ACTION clause are executed. Otherwise, the operations specified in the ALTACTION clause are executed. The DESCRIPTION clause contains a text string describing what the rule does. The RULEVAR clause allows variables to be defined in a rule. Also, variables which need to be persistent are declared within this clause. Customizable rule variables, which can be assigned different values for different users (i.e., customizable rules), are also supported. The CONDITION clause is specified using a guarded expression. A guarded expression can be divided into two parts: the guard part and the condition expression part. The guard part is composed of a sequence of expressions, which are evaluated sequentially and are enclosed in brackets. If any of the expressions within the guard evaluates to false, the whole rule execution is skipped (i.e., the rule is not applicable). If all of the expressions evaluate to true, the condition expression part will be processed. The evaluation result of the condition expression decides whether to go to the ACTION clause or the ALTACTION clause. The reason for employing the guarded expression is that it allows for efficient and ordered processing of pre-requisite

conditions which must be satisfied in order for the rule processing to be meaningful. The ACTION clause and ALTACTION clause consist of a sequence of operations to be carried out.

3.3. Triggers

A trigger relates a structure of events with a structure of rules. Thus, it is composed of two main components: an event structure and a rule structure. An example of a trigger is shown in Figure 1. An event structure has two parts, namely, a TRIGGEREVENT part and an EVENTHISTORY part. The TRIGGEREVENT part specifies a number of events any of which, when posted, would initiate the evaluation of the EVENTHISTORY part. In Figure 1, the trigger events are E1, E2, or E3. If the event history evaluates to True, the structure of rules is processed. Otherwise, the structure of rules will not be processed. This separation of TRIGGEREVENT and EVENTHISTORY provides a way to specifically name the events (i.e., TRIGGEREVENT) that will trigger the evaluation of a more complex event expression (i.e., EVENTHISTORY). This is different from the event specification of some existing ECA rule systems in which, when a composite event is specified, all the events mentioned in the composite event expression implicitly become the trigger events. As a result, the composite event expression must be repeatedly evaluated even though some of its events should not have triggered its evaluation.

The structure of rules, given in the RULESTRUC clause, forms a graph structure. In the graph structure, rules can be executed sequentially, in parallel, or with synchronization points. The synchronization points can have AND conditions (where all of the rules should finish before the synchronization point) or OR[n] conditions (any n number of rules should finish before the synchronization point). Data from the attributes of the TRIGGEREVENT can be passed to the trigger parameters and then to the rules in a RULESTRUC. The trigger includes the specification of their mappings. Figure 1 shows an example of a complex trigger.

4. Architecture and concept of knowledge network

The infrastructure to support the knowledge network concept is composed of a large number of knowledge Web servers on the Internet. The knowledge Web server contains additional modules to extend the capability of the current Web servers. We will discuss the architecture of the knowledge Web server in the following subsection and then discuss the concept and key features of the knowledge network.

4.1. Architecture of the knowledge Web server

The general architecture of the knowledge Web server is shown in Figure 2. The figure shows three knowledge Web servers and their components. Knowledge Web server A is the provider of events E1 and E2, while knowledge Web servers B and C are subscribers of

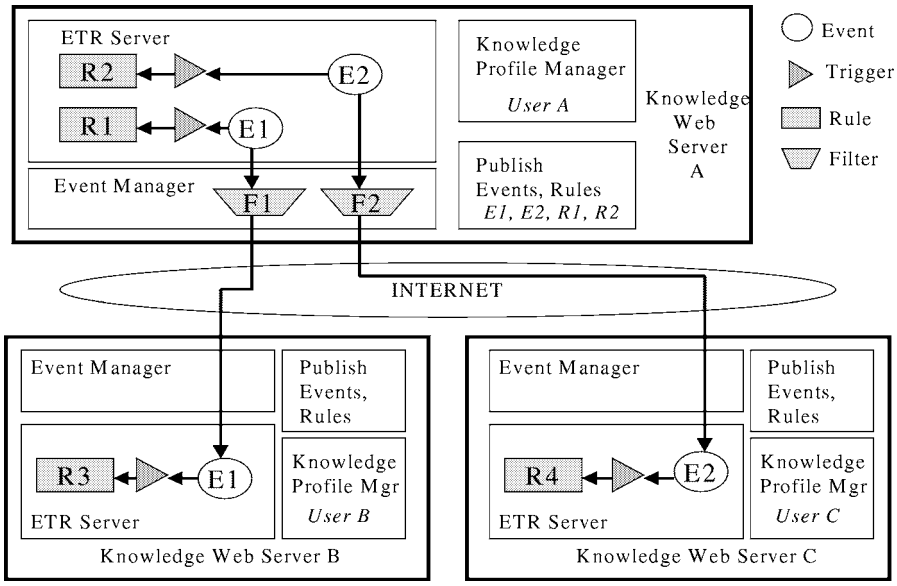

Figure 2. Overview of architectural components in the knowledge Web server.

the events. The figure shows the sequence of actions performed when the events are posted from knowledge Web server A.

Each knowledge Web server has an event manager, an ETR server, and a knowledge profile manager, which are additional components installed on a typical Web server. Because these components are installed on each Web server, the infrastructure has a symmetric architecture.

Each user who has data and applications on a web server has a knowledge profile that is persistently stored and maintained by the knowledge profile manager. The knowledge profile of a provider contains the events, triggers, and rules that were published by the provider. The knowledge profile of a subscriber contains the events that the user has subscribed to, and also the trigger and rules that were defined on the subscriber site. The definitions of the events, triggers, and rules are passed to the event manager and the ETR server. Since a user can play the role of a provider as well as the role of a subscriber, the knowledge profile manager keeps the corresponding profiles separately.

The event manager performs functions related to the registration, filtering and notification of events. The event manager provides an event registration facility to allow remote clients to register their interests by subscribing to certain events defined by a provider. During the event registration, subscribers can install event filters. These event filters provide selective subscription to a subset of the events. Figure 2 shows two example filters F1 and F2, which are installed into the event manager of the provider site by subscribers B and C, respectively. Also during the registration, values for customizable rules are provided by the subscribers. This enables subscribers of events to tailor the rules and install them into the ETR server on the provider site to meet their different needs. The event manager

provides an interface to allow the local applications to connect to it and generate an event. It is also responsible for performing event filtering before it notifies the subscribers. The event managers on different knowledge Web servers can communicate with each other. When an event manager of a subscriber receives an event notification from a remote event manager, it passes it to the local ETR server to initiate the processing of subscriber-side triggers and rules, as shown in Figure 2.

The ETR server performs the installation and processing tasks for triggers and rules in a knowledge Web server. The trigger and rule definitions that are input through the knowledge profile manager are provided to the ETR server and are transformed into internal data structures used for executing the triggers and rules. The ETR server also receives information from the event manager about the values for customizable rules that are provided by subscribers of events during the event registration process. Upon receiving an event, the ETR server can immediately identify the trigger related to the event, process the event history, and schedule the rules specified in the trigger for execution.

4.2. Key features of knowledge network

The key features of the knowledge network are: publishing knowledge, event filtering and notification, trigger and rule processing, and knowledge profile management. They correspond to the key concepts of the knowledge network and are described in more detail below.

4.2.1. Publishing knowledge. Publishing knowledge involves defining events, parameterized event filters, and provider-side rules. A knowledge provider can define new events that he/she would like to post over the Internet. The detailed textual descriptions about these events are provided on the provider's Web page. By publishing these events on a Web page, users on the Internet can browse the event descriptions, access the event registration form and subscribe to the event by registering themselves as subscribers to the event. Each event has its own event registration form. When publishing an event, the provider must be kept in mind that a subscriber may not be interested in subscribing to all the event instances that will be posted by the provider. For example, assume that there is an airfare special offer event being posted, but the subscriber is actually only interested in the airfares for flights that depart from Orlando, FL. If the provider allows the subscriber to give some values to indicate which subset of event instances he/she is interested in, a filter which screens out all the irrelevant event instances for that particular subscriber can be established. In order to support this, the provider must give the subscriber a parameterized filter. In other words, a filter with some undefined values is provided. For example, for the airfare special offer event, a parameterized filter on the city of departure can be defined. This will allow the subscriber to input (or choose from a list) the actual value of the city of departure that is desired. This kind of parameterized filter can be shown on the event registration form by displaying the city of departure with a blank box beside it. The subscriber can input the value Orlando into the blank box. The types of parameterized filters supported by our implemented system are:

- Equal: the subscriber should specify the exact value of the event attribute, which is of interest to the subscriber.
- Range: the subscriber should specify the minimum and maximum values of the event attribute that are of interest to the subscriber. This is only applicable to numerical type attributes. A similar concept was introduced in CoSent [8]. However the implementation is different from our approach.
- Greater (or less) than: the subscriber should provide a lower bound (or upper bound) of the event attribute. This is only applicable to numerical type attributes.
- Single selection: the subscriber should select one of the listed values that the provider has pre-defined.
- Multiple selection: similar to the single selection operator except that it allows the subscriber to select multiple values instead of a single value, meaning that the subscriber will receive event notifications if the attribute value associated with the event matches any of the selected values.

Provider-side rules are intended to allow subscribers of events to automatically execute rules on the provider's knowledge Web server when the subscribed event occurs. They are defined by the provider and can take the form of a customizable rule, which means that certain values in the rule are left for the subscribers to customize based on their individual needs. These provider-side rules are displayed within the event registration forms. During the registration process, the subscriber can select and customize the set of rules specified by the provider.

The knowledge network infrastructure can support this concept of publishing knowledge by providing the providers with an easy-to-use tool such as the knowledge profile manager. Through the GUI of the knowledge profile manager, the provider can define events, parameterized event filters, and provider-side rules as follows. First, the provider defines an event by specifying the event name and attributes. Second, one or more attributes of the event are selected. For each of the selected attributes, a filter type can be specified. Third, the provider can define customizable rules that have variables whose values can be customized by the subscribers. As a result of these steps, two XML files are created: an event filter template and provider-side rule template. Then, the provider only needs to create a link in a Web page that refers to the event manager with the names of these two XML files as input parameters to complete the knowledge publishing.

The event registration forms are generated based on the two XML files (i.e., event filter template and provider-side rule template) using a document-driven approach so that the provider neither needs to perform any coding for the forms nor understand how the underlying knowledge Web server works. When a subscriber clicks on the link, which was created as the last stage of the publishing described above, the event manager interprets the templates and dynamically creates an HTML form that displays the contents of the templates in a visual form to the subscribers. During the process of dynamically creating the event registration form, the parameterized filters are transformed into the visual elements, such as input boxes, drop-down boxes, and multiple radio buttons. Provider-side rules will also display input boxes for the customizable rule variables to allow the subscriber to assign values to them. After the subscriber fills in the form, the information will be provided

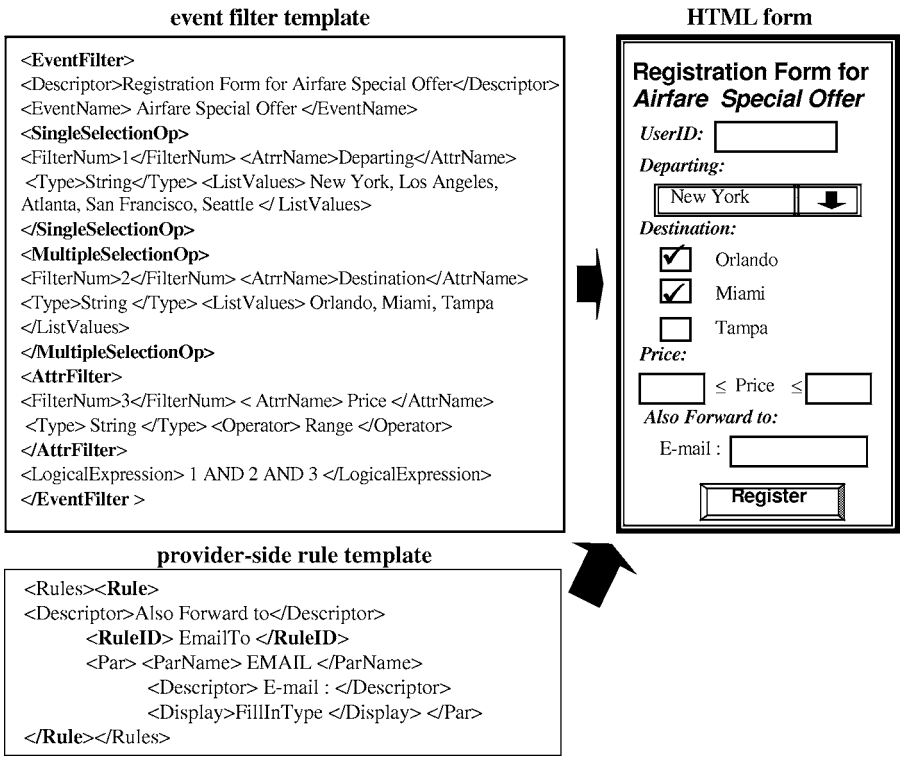

Figure 3. Example of dynamically generating the event registration form.

to the event manager for further processing. An example of the event filter template and provider-side rule template and its transformation for an airfare special offer event is shown in Figure 3.

4.2.2. Event filtering and pushing. In order to reduce the event notification traffic and allow the subscribers to receive exactly what they want, event filtering becomes an essential part of the knowledge network. Due to the fact that eliminating unneeded events as soon as possible would maximize the benefit of event filtering, we perform the filtering operation on the site of the event provider before it is sent out through the Internet.

As described in the previous subsection, the subscriber will register for an event and give information about him/herself such as user id and e-mail address. In an environment where static IP addresses are assigned and the subscriber always uses the machine designated as his/her knowledge Web server for browsing the Internet, the IP address of the subscriber's knowledge Web server can be obtained through the HTTP protocol by the event manager when the registration is performed. However, in the real world, due to dynamic IP addresses and subscribers moving around on different machines while browsing, this information should be explicitly provided by the subscriber in the form of a URL during registration instead of being automatically, but in some cases, incorrectly detected. The

subscriber will then input the values identifying the subset of event instances he/she desires to receive.

Several data structures and algorithms have been developed and implemented for matching a large variety of generated events against a large collection of user filters. We use an inverted index [32] data structure for equal, single selection, and multiple selection type filters that require an exact match of the generated event attribute value against the filter values specified by the subscribers. The mapping from an (event attribute, value) pair to the corresponding inverted list of users is implemented as a hash table.

During registration, the user can also specify a range type event filter. The inverted index is not suitable for range queries, which need to perform a lookup on intervals rather than discrete values. It is necessary to find a suitable data structure for storing range values and an efficient algorithm to search the range value to determine if a given value falls within these ranges. For example, assume that user U1 created a filter for the price range of [100, 200]. When the airfare special offer price is given by a travel agency, we want to determine if the given price falls within the filter range. If so, U1 should be notified. Using a modified version of 2–3 trees (or red–black trees) [18] is one way of solving the range search problem. For insertion, we use lexicographical sorting in order to put a key object into the corresponding node. The search and the split operations, which occur when an insertion takes place, is the same as the 2–3 tree except for the key comparison. Details of the algorithm are given in [14]. The search algorithm for this structure descends down the tree from the root. However, more than one subtree under a visited node may need to be searched. The search algorithm will eliminate irrelevant ranges and will examine only the ranges in which a given value falls. The time complexity for insertion and deletion of this structure is the same as for a 2–3 tree, which is $O(\log n)$ time where n is the number of items in the data structure. The time complexity for the search is $O(q + \log n)$, where q is any number in the following range $[0, n]$.

We have also implemented another data structure, range table structure, which is suitable for range searches. The range table structure is in the form of a table. Each entry of the table consists of two elements. The first one is a unique integer number (a minimum or a maximum value that belongs to some range), and the second one is a list of users. The entries in the table are sorted in ascending order. Two consecutive entries of the table store the start value and the end value of an interval. Assume that the integer number in the first entry is N, and the number of the second entry is M. In this case, the corresponding list of users stored in the first entry is the list of users who have registered for the value range $[N, M]$. The algorithms for performing insertion and deletion are shown in Figure 4. The time complexity for insertion and deletion of this structure is $O(n/2 + \log n)$ on the average. The time complexity for the search is $O(\log n)$. Comparing it to the modified 2–3 tree version, this structure has a better search time complexity. For situations where intensive event generation is present, the search operation will be dominant. By implementing and integrating both structures into the framework, the publisher may choose the suitable structure that will fit his/her application domain, depending on the frequencies of insert, search, and delete operations.

The pushing of events is performed by sending e-mail notifications or XML notifications to the subscribers. The XML notifications make the format interchangeable among vari-

```
RangeTableInsert ( low a, high b, user u ) {        // ** INSERT RangeTable **//
// a and b represent range boundary to insert
 ENTRY range_start =  BinarySearchInsert (a);
 ENTRY range_end =    BinarySearchInsert (b);
 FOR ALL ENTRIES J FROM range_start TO range_end { insert u into user list of J }

BinarySearchInsert ( value x ) {
   Binary Search for x value within the RangeTable;
   If exist, return entry location;
   If not exist, create item in appropriate position and return new item location; }
```

```
GetUsersFromRangeTable ( value v ) {          // ** SEARCH Range Table ** //
   Get entry J in RangeTable the largest value k where k <= v;
   If (entry J has value equal to v ) { Add the user list of entry J-1 to RESULT }
   Add the user list of entry J to RESULT;
   return RESULT; }
```

Figure 4. Algorithms used for insert and search for the range table.

ous systems. The event manager currently performs the pushing of the events using two types of communication: HTTP protocol and RMI. We are also investigating a hierarchical broadcasting mechanism to push the events on the Internet. If the multicasting protocol of the routers are enabled on the Internet, it would be straightforward to implement an event pushing mechanism that makes use of the IP multicasting protocol.

4.2.3. Trigger and rule processing. Trigger and rule processing is one of the important features of the knowledge network. It provides the means to embed knowledge into the Internet. Once an event occurs, the event can invoke the processing of triggers and rules, which capture the knowledge useful for making decisions, enforcing business policies, performing military command/control tasks, etc. In the knowledge network, the trigger and rule processing occur at both the provider and subscriber sites. Both of these triggers and rules are defined through their corresponding knowledge profile managers. On the provider site, a provider can login on his/her knowledge profile manager in the provider mode and define provider-side rules that are connected to a published event. Provider-side rules can be private (to the provider) or public (available for use to subscribers). The public provider-side rules become ready for execution when a subscriber selects them at the time of performing the event registration.

On the subscriber site, the subscriber can login on his/her own knowledge profile manager in the subscriber mode to view his/her subscribed events and define subscriber-side rules and triggers, which are linked to the subscribed events. Therefore, when an event occurs on the provider site, the event will first be filtered to identify the subscribers and invoke the relevant provider-side rules. Then, the event will be sent to the subscriber sites and invoke the processing of the subscriber-side triggers and rules.

The public provider-side rules allow the subscribers to make use of built-in facilities provided by the provider. They are customizable rules, which include customizable rule variables whose values are retrieved from a persistent store using the subscriber id as a key during the execution of the rules. The value of the customizable rule variable is populated when a subscriber inputs the customized value during an event registration.

Subscriber-side rules represent any type of knowledge that the subscriber wants to relate to the subscribed events. The trigger can be simple or very complex by including several triggering events, all of which are subscribed events. The event history specification of the trigger can model the complex relationships among the subscribed events. The rule structure enables the subscriber to specify the complex logic related to the rule processing. Parallel, sequential, and synchronized rules can be executed on the subscriber's site. Subscriber-side triggers and rules are private to subscribers since they are not published and are kept at the individual subscriber's sites. Each subscriber can freely define his/her own triggers and rules without worrying about giving out valuable information specified in them.

Replicated ETR servers perform the processing of the triggers and rules on both the provider and subscriber sites.

4.2.4. Knowledge profile management. An authorized user can have an account created for his/her knowledge profile on a knowledge Web server. The knowledge profile manager is the component that maintains the knowledge profiles. The user may have two modes of logging in: provider mode and subscriber mode. If granted to login with the provider mode, the user can publish events, triggers, and rules on the knowledge Web server. If granted to login with the subscriber mode, the user can define triggers and rules related to subscribed event definitions imported from remote knowledge Web servers. Users can also be granted authorization as both a provider and a subscriber on a knowledge Web server.

Knowledge profiles contain the meta-information (or definitions) about the events, triggers, and rules. The meta-information is provided to other components for transformation into executable forms. Maintaining knowledge profiles on different knowledge Web servers allows individuals' knowledge to be distributed among the knowledge Web servers instead of keeping all of the knowledge in a central repository on the Internet. Since the run-time components such as the ETR server and the event manager are also replicated at all sites, interactions between the knowledge profile manager and other run-time components can be carried out locally.

The knowledge profiles can be edited. In other words, event, trigger, and rule definitions can be modified or deleted. When published events are deleted, there exists the issue of notifying the subscribers of the deletion of the event so that the subscribers can delete it from their knowledge profiles. An approach to solving this problem could be to implement a system-level event that notifies the deletion of published events. When such an event occurs, a system-level rule on the subscriber sites will automatically clean up the event definitions.

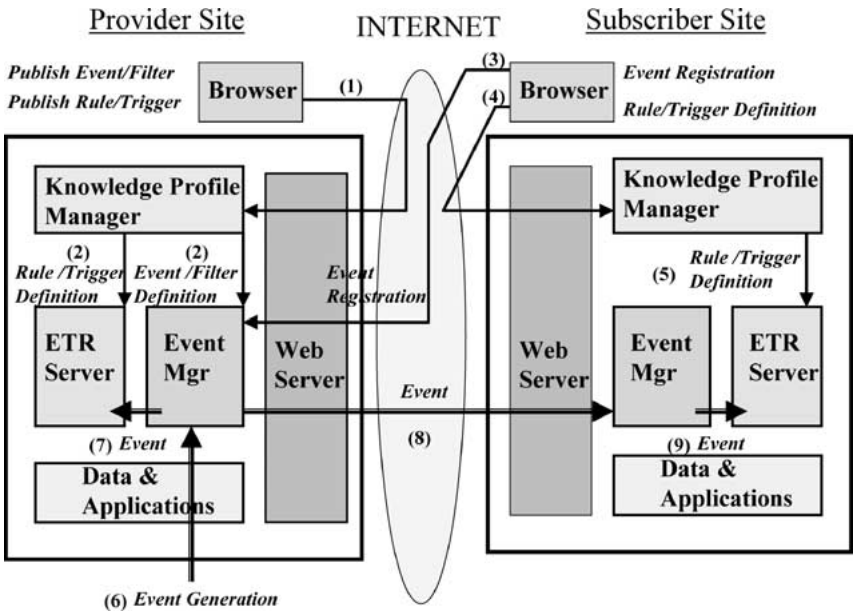

Figure 5. Steps for constructing and processing in the knowledge network.

4.3. Steps for constructing and processing the knowledge network

An overview of the steps involved in constructing the knowledge network and its processing is given here to explain the interactions among its components. Figure 5 shows the component interactions during the entire process. The steps are explained below:

1. Publishing knowledge by provider. In addition to publishing multimedia data through the Web, the provider publishes events, parameterized event filters, and provider-side rules using the knowledge profile manager.
2. Installation on provider site. The event and event filter definitions are given to the event manager. Provider-side customizable rule definitions are given to the ETR server.
3. Event registration by subscriber. A client (i.e., subscriber) accesses the Web page of the provider and registers for the subscription of an event and specifies the values needed for installing an event filter. Provider-side rules are also selected and customized. After the registration has been successfully performed, the event subscription information is automatically forwarded to the subscriber's site.
4. Trigger and rule definition by subscriber. The client accesses his/her own knowledge profile through the knowledge profile manager on his/her knowledge Web server and defines triggers and rules related to the subscribed events to be processed at the subscriber site.
5. Installation on subscriber site. The trigger and rule information is given to the ETR server and the necessary run-time codes are generated and installed based on their definitions.

6. Event generation and filter processing. An event is generated on the provider site (e.g., by some application or person) and is given to the local event manager. The event manager processes the filters and identifies the subscribers of the event.
7. Provider-side rule processing. The event is forwarded to the local ETR server to trigger any private provider-side rules. If a subscriber of the event had tied the event to a provider-side rule during the registration, the relevant rule is also executed on the provider's knowledge Web server.
8. Event notification. The event is posted over the Internet to the subscribers of the event and the event manager on the subscriber site receives it.
9. Subscriber-side rule processing. The event received is forwarded to the ETR server to execute any relevant subscriber-specified triggers and rules.

5. Implementation

All the components necessary for supporting the knowledge network concepts have been implemented and integrated to form a package that can be easily added onto any Web server. The code was developed using JDK 1.2.2, and the Web server used is the Apache Web server 1.3.9. The Jserv servlet engine and the Internet explorer browser 5.0 are also used. Since all the programs are written in Java, they can be easily deployed on any Web server of the existing Internet. Some details about the implementation of each component are given below.

The knowledge profile manager is composed of two major modules: an applet and a servlet. The applet forms the GUI part, which is downloadable to a browser and allows the users to interact with it. The applet displays different forms when running in the provider mode or the subscriber mode. The applet includes the event, trigger, and rule editor GUI. The trigger editor is specially designed to have two modes of input: a simple trigger editor for novice users and an advanced trigger editor for advanced users to input complex triggers with multiple triggering events, an event history, and a complex rule structure through a form-based GUI. The applet contacts the servlet to perform tasks that require interacting with other components or generating files in certain directories in the knowledge Web server. The servlet also includes a rule code generator, which generates Java codes in a specified directory for the ETR server to use. XML files containing event filter templates and provider-side rule templates (described in Section 4.2.1) are also generated by the servlet and stored into a specific directory for the event manager to use. The knowledge profile manager includes a metadata manager component, which is an RMI server that persistently stores the event, trigger, and rule definitions. The servlet contacts the metadata manager to persist the information entered through the applet GUI.

The event manager is implemented as an event registration servlet and event delivery RMI server/servlet combination. The event registration servlet includes the code for dynamically generating the event registration forms in HTML. It uses the XML DOM parser provided by IBM to interpret the event filter template and the provider-side rule template. The event registration servlet accepts the information that the subscriber entered into the event registration form and persistently stores it. The event delivery RMI server/servlet in-

cludes the filtering facility and the event delivery related codes. It uses the inverted index, modified 2–3 tree, and range table index structure (described in Section 4.2.2) for filtering. Also, functions for sending and receiving files are supported to communicate information such as the event definition in XML. The event delivery RMI server/servlet has APIs to send and receive event notifications. E-mail notification codes are also included.

The ETR server is implemented as an RMI server. It provides several APIs for the knowledge profile manager and the event manager to call. It uses hash tables to store the mapping between events and triggers. The ETR server includes a rule scheduler component that can perform parallel rule processing as well as sequential or synchronized rule processing as specified by the rule structure. The rules are executed as separate threads and there exist special data structures and algorithms based on two tables specifying the predecessor and successor of each node in the rule structure. The predecessor table keeps track of how many predecessor rules have been executed, while the successor table keeps track of the set of successor rules that are ready for execution once a rule has completed its execution. Once the predecessor condition is satisfied (i.e., the completed predecessor count is reduced to zero), these rules are ready to execute and are put into a queue for execution by a dispatcher. As soon as a rule completes its execution, the successor rules are looked up and, for those successor rules, the predecessor counts in the predecessor table are reduced. The Java class loader facility is used for the purpose of dynamically reloading rules that have been changed. This allows a rule to be changed while another instance of the rule with the previous definition is running. Thus, dynamic rules are supported. The ETR server has an extensible interface, which allows it to receive event notifications through different communication infrastructures. The current interface supports RMI, Orbix [36] and Vitria's communicator. The event-history-processor (EHP) server is also included as a module for the ETR server. It can also run as an independent RMI server. The EHP server uses an internal graph data structure to readily evaluate an event history expression. The ETR server contacts the EHP server when an event history expression is encountered in a trigger and obtains the result of the evaluation from it.

6. E-commerce scenario

To illustrate the usefulness of knowledge networks, we have developed an example e-commerce application to demonstrate the presented concepts and technologies: a business-to-business e-commerce scenario based on a company named "IntelliBiz." IntelliBiz performs the business-to-business e-commerce service by connecting suppliers of products/services to buyers who are seeking these products/services. They publish a list of suppliers and buyers on their company's home pages categorized by the products and services. A new supplier can go through a supplier registration process and give information about the types of product or service the company provides along with the company contact information. New buyers go through a separate registration process and give information about the product or service for which they are seeking. IntelliBiz provides the service to allow businesses (both suppliers and buyers) to easily find each other and forms a virtual marketplace of products and services at the business level.

IntelliBiz publishes a NewBuyer event. Suppliers can subscribe to this event about newly-registered buyers to obtain this important information in a timely manner. This event is posted when a new buyer registers with IntelliBiz. The event attributes encapsulate the buyer information. IntelliBiz also has a customizable rule (provider-side rule), NotifyBuyer, which will send an e-mail notification to a new buyer to introduce a registered supplier.

The e-CarSpeakers company, which specializes in selling audio speakers for cars, subscribes to the event NewBuyer. The filter installed requires that the requested product is speakers and the price range is more than $300. The NotifyBuyer rule is also selected on the provider side. On the e-CarSpeakers site, the trigger relates the NewBuyer event to the AlertMarketing rule. The AlertMarketing rule will store the buyer information into a database and also alert the marketing department about the new buyer. The capability of defining rules on each of the local Web sites allows local information or policies to be kept secure and undisclosed. When a new buyer MyAutos.com who is looking for car speakers registers with IntelliBiz and is willing to pay a price over $300, the provider-side rule NotifyBuyer is executed. This rule will send an e-mail to MyAutos.com introducing the e-CarSpeakers company. Meanwhile, the NewBuyer event will be sent over the Internet to the e-CarSpeakers site and automatically invoke the AlertMarketing rule.

7. Conclusion

In this paper, we have presented the concept, architecture, functions, implementation techniques, and an application example of a knowledge network. This work was motivated by the limitations we observed in the existing Internet and Web technologies for supporting the emerging applications such as e-commerce and enterprise integration. We propose to extend the existing Internet-Web infrastructure by adding event and rule services as a part of the information infrastructure. Events, rules and triggers, which relate events to the evaluation of event history and the activation of rules, can be used to capture human and enterprise knowledge in the Internet, making the Internet a knowledge network instead of a passive data network. The event, event filter, event history, and rule processing capabilities of the knowledge network offer very powerful and useful services to enable the timely delivery of relevant data and activation of operations.

The knowledge network can be realized by installing a number of knowledge Web servers at some selected sites in the Internet. Each knowledge Web server consists of the following components in addition to a Web server. An event manager performs event filtering, notification and management. An ETR server performs trigger processing, which includes the evaluation of event history (or composite events) and the activation of rules. Dynamic rules are supported by this component. Additionally, a knowledge profile manager manages the profile of users and/or enterprises registered with the server. It also provides a Web-based graphical user interface for specifying, storing, and installing knowledge into a knowledge Web server. The functions provided by these components extend the capabilities of the Internet infrastructure and are important for supporting collaborative and distributed applications built on the Internet.

There are some issues that need to be addressed in future research. First, there are security issues related to event delivery and rule triggering. Since event notifications can initiate the execution of rules, which may activate application systems, the potential damages caused by not having the proper security control on event and rule processing can be substantial. One approach to provide security in event delivery and rule processing for the subscriber is by embedding the event provider's identity into the events and use a public key/private key encryption mechanism to secure the event contents. When the subscriber registers for the event, the subscriber's public key could be provided. This would enable only the subscriber to decrypt the encrypted message that contains the event as well as identify the valid event provider's identity within the event, and therefore safely activate the rule processing. This is equivalent to using digital signatures with Secure Sockets Layer (SSL) [26]. This approach would of course have impact on the performance of the event delivery due to the additional overhead of encrypting and decrypting the events and comparing the identities of the event providers. For secured rule processing on the provider side, the provider should first make sure that the published provider-side rules do not include potentially disastrous operations. In order to verify the identities of subscribers, certificates could be required during the event registration time. These certificates can be issued by third party organizations such as VeriSign [33] or the organization itself can manage the certificate using tools such as Netscape certificate server [25]. However, this may complicate the registration process and require too much information from the subscriber. It may be more useful to simplify the registration process and make sure that only safely executable provider-side rules are published. Second, in this work, we assume that knowledge providers and consumers of an application domain use the same ontology. That is, the terms used in defining objects and their associated events and rules are understood by all users in that application domain. This assumption is not realistic because different users may have different interpretations and understanding of the same terms. These differences need to be resolved by some ontological mappings. A solution to this problem is to develop an ontology server for each business/application domain (e.g., automobile manufacturer, or toy retailer) to handle the translation of the terms and term relationships used in that domain. The translation rules used by the server may have to be derived gradually as discrepancies in term usages have been identified. It is possible that, through time, people in the same domain may come to an agreement that a set of standard terms be used. Third, rules contributed by different users and/or organizations may have contradictions, cyclic conditions, redundancies, and subsumptions among them. Techniques for validating the distributed knowledge base have to be introduced and applied to detect these problems. Fourth, event and rule libraries can be pre-established for different application domains so that all users can take advantage of their contents. For example, it will be useful to establish a library of events and rules for e-trading toys and another one for cars, etc., so that knowledge useful for different business domains can be captured and used by general users. Fifth, event notifications can be very time-consuming when a very large number of people and organizations have subscribed to the same events. Techniques for efficient delivery of events and the data associated with the events are needed.

Acknowledgement

This research is partially supported by the National Science Foundation, USA (Grant #EIA-0075284).

References

[1] M. Ammar and J. Wong, "The design of teletext broadcast cycles," *Performance Evaluation* 5(4), 1985, 235–242.
[2] BackWeb technologies, BackWeb, `http://www.backweb.com`, May 1999.
[3] BEA WebLogic, "WebLogic events," `http://www4.weblogic.com/docs/techoverview/em.html`.
[4] I. Ben-Shaul and S. Ifergan, "WebRule: An event-based framework for active collaboration among Web servers," *Computer Networks ISDN Systems* 29(8–13), 1997, 1029–1040; also in *Proc. of the 6th Internat. World Wide Web Conf.*, Santa Clara, CA, April 1997, pp. 521–531.
[5] T. Bray, C.M. Sperberg-McQueen, and J. Paoli, "Extensible Markup Language (XML)," W3C Recommendation, February 1998, `http://www.w3.org`.
[6] L. Brownston, R. Farrell, and E. Kant, *Programming Expert Systems in OPS-5: An Introduction to Rule-Based Programming*, Addison-Wesley: Reading, MA, 1985.
[7] S. Chakravarthy, E. Anwar, L. Maugis, and D. Mishra, "Design of sentinel: An object-oriented DBMS with event-based rules," *Inform. Software Technol.* 39(9), 1994, 555–568.
[8] W. W. Chu and W. Mao, "CoSent: A cooperative sentinel for intelligent information systems," in *Proc. of 2000 Internat. Conf. on Artificial Intelligence*, Las Vegas, NV, June 2000.
[9] U. Dayal, B. T. Blaustein, A. P. Buchmann, S. Chakravarthy, M. Hsu, R. Ledin, D. R. McCarthy, A. Rosenthal, S. K. Sarin, M. J. Carey, M. Livny, and R. Jauhari, "The HiPAC project: Combining active databases and timing constraints," *ACM SIGMOD Record* 17(1), 1988, 51–70.
[10] Document object model specification, `http://www.w3.org/DOM`, May 1999.
[11] N. H. Gehani and H. V. Jagadish, "ODE as an active database: Constraints and triggers," in *Proc. of the 17th VLDB Conf.*, Barcelona, Spain, 1991, pp. 327–336.
[12] J. C. Giarratano, *CLIPS User's Guide, Vol. 1, Rules*, NASA, Lyndon B. Johnson Space Center, Information Systems Directorate, Software Technology Branch: Houston, TX, 1993.
[13] D. Gifford, "Polychannel systems for mass digital communication," *CACM*, February 1990.
[14] M. Grueva, "A framework for event registration, filtering, and notification over the Internet," Master thesis, University of Florida, 1999.
[15] L. M. Haas, W. Chang, G. M. Lohman, J. McPherson, P. F. Wilms, G. Lapis, B. Lindsay, H. Pirahesh, M. Carey, and E. Shekita, "Starburst mid-flight: As the dust clears," *IEEE Trans. Knowledge Data Engrg.* 2(1), 1990, 143–160.
[16] E. N. Hanson, "The design and implementation of the ariel active database rule system," *IEEE Trans. Knowledge Data Engrg.* 8(1), 1996, 157–172.
[17] G. Herman, G. Gopal, K. Lee, and A. Weinrib, "The datacycle architecture for very high throughput database systems," in *Proc. of ACM SIGMOD Conf.*, San Francisco, CA, May 1987, pp. 97–103.
[18] E. Horowitz, S. Sahni, and D. Mehta, *Fundamentals of Data Structures in C++*, Freeman: New York, 1995.
[19] KeryxSoft, `http://keryxsoft.hpl.hp.com/`.
[20] H. Lam and S. Y. W. Su, "Component interoperability in a virtual enterprise using events/triggers/rules," in *Proc. of OOPSLA '98, Workshop on Objects, Components, and Virtual Enterprise*, Vancouver, BC, Canada, October 1998, pp. 47–53.
[21] M. Lee, S. Y. W. Su, and H. Lam, "Event and rule services for achieving a Web-based knowledge network," in *Proc. of 1st Asia-Pacific Conf. on Web Intelligence (WI-2001)*, Maebashi City, Japan, October 2001, accepted for publication.
[22] M. Lee, S. Y. W. Su, and H. Lam, "Event and rule services for achieving a Web-based knowledge network," Technical Report UF CISE TR00-002, University of Florida, 2000.

[23] Marimba, http://www.marimba.com, May 1999.
[24] Netscape, http://www.netscape.com, May 1999.
[25] Netscape, Netscape Certificate Server, http://home.netscape.com/certificate/v1.0/.
[26] Netscape, "SSL 3.0 specification," http://home.netscape.com/eng/ssl3/.
[27] Oasis, "The SGML/XML Web page—Extensible Stylesheet Language (XSL)," http://www.oasis-open.org/cover/xsl.html.
[28] PointCast, http://www.pointcast.com, May 1999.
[29] Servlet central, http://www.servletcentral.com/.
[30] M. Stonebraker and G. Kemnitz, "The postgres next generation database management system," *CACM* 34(10), 1991, 78–92.
[31] S. Y. W. Su, H. Lam, S. Eddula, J. Arroyo, N. Prasad, and R. Zhuang, "OSAM*.KBMS: An object-oriented knowledge base management system for supporting advanced applications," in *Proc. of ACM SIGMOD*, Washington, DC, May 1993, pp. 540–541.
[32] A. Tomasic, H. Garcia-Molina, and K. Shoens, "Incremental updates of inverted lists for text document retrieval," in *Proc. of the ACM SIGMOD Conf.*, Minneapolis, MN, May 1994, pp. 289–300.
[33] VeriSign, http://www.verisign.com/.
[34] Vitria Technology, "Vitria businessware communicator," http://www.vitria.com/products/index.html.
[35] W3 Consortium, "HyperText markup language," http://www.w3.org/MarkUp/MarkUp.html.
[36] White paper—OrbixTalk, IONA Technologies PLC, http://www.iona.com, 1997.