

Activated Online Auctioning

Using a Loosely-Coupled Request-Response Model

Scenera Research LLC

Cary, North Carolina

www.sceneraresearch.com

The World Wide Web primarily runs according to the rules of HTTP, a simple, flexible request-response protocol that expands well to suit the needs of the ever-growing Web. One need of the future that it doesn't meet today is the support of asynchronous messaging and loosely-coupled communication, which could be valuable in a wide variety of applications, including online auctioning. Currently, online auctions require users to manually request updates to keep up with changes in pricing and status. The Web of the future will automate those update requests and their responses, so users always have real-time information. Resources exist to perform this feat by adding request-response services to an existing messaging infrastructure supporting a publish/subscribe protocol model. That messaging infrastructure currently supports presence services.

Overview

Request-response message exchange is just what its name implies: it consists of two messages, a request and a response. A requestor sends a request message to a responder that receives it, processes it, and replies. This is one of the simplest and most common forms of communication, and it is supported by protocols as universal as Hypertext Transfer Protocol (HTTP) and Simple Mail Transfer Protocol (SMTP).

Searching and browsing on the Web are performed by continuous request-response message exchanges. When a user wants to see a particular resource on the Web, such as a Web page, image, or other type of file, she enters a URL or clicks a hyperlink. This results in her browser putting forth an HTTP request for the resource. The requestor waits for a specific response from the network device identified in the request, such as a server hosting a Web application. Then the request is processed by the receiver, a response is sent, and the requested resource, such as a Web page, is displayed by the requesting browser. With this process, two simple HTTP commands – request and response – are all that is needed to perform nearly all of the operations that occur on the World Wide Web today.

Because HTTP-based applications require clients to request information, information can be lost and performance can be negatively impacted when the application needs frequent information updates, as it does with online auctioning. To avoid problems on today's Web, auction sites like eBay and uBid maintain numerous servers and vast memory to store, manage, and continuously update all of the auction data they require for operation. Allowing the servers and memory needed for auctioning to be distributed on the Web would lessen the local resource requirements of auctioning sites.

Another limitation of current online auctioning sites is that they require users to manually update their auction data, such as user status and pricing information, by repeatedly renewing or refreshing their Web pages. Ideally, a user should be able to receive updated information without intervening: they should be able to connect to an auction Web site with pages that refresh automatically without the user or the browser constantly checking for updated information.

Automation of request-response is the key to these Web advancements, and we have designed a way to automate it that is as simple to implement as HTTP.

Presence Services and Principal-based Publish/Subscribe Protocols

Presence is a widespread service for determining, in real time, the online availability of people, applications, information, devices, and other resources needed for communication. Systems that store and provide presence information are known as “presence services” and applications that use information provided by presence are referred to as “presence-aware.”

Presence services are most commonly known to users through instant messaging (IM) applications, such as Yahoo Messenger, MSN Messenger, and AOL Instant Messenger. Instant messaging applications use presence services to let users know whether their “friends” or “buddies” are available on the IM network. Presence services can also provide each user’s status, such as “online,” “in a meeting,” or “at lunch.” Status can also provide instructions about contacting users by other media, such as cell phone, fax, or e-mail, the relative priority of each contact method, and detailed notes to specific users.

Presence is one type of service that can be provided using a principal-based publish/subscribe protocol model. Current principal-based publish/subscribe services are limited in that they only support one-way communication, from a publisher to a subscriber. However, many applications require two-way communications, and to do that they use the request-response protocol model. Current request-response protocols require a requestor to know the identity or address of a partner that is capable of providing a response, resulting in a tightly-coupled system. However, if one of the communications partners is not available, a tightly-coupled request-response system can break down or be halted. A looser coupling design may resolve this problem.

When we examined whether a request-response model that has the advantages of loose coupling could be created, we determined that it can. With further examination, we also determined that it should be more robust than current models and should reduce the heavy dependence users have on the Web search and directory lookup tools that exist today.

Creating the Right Protocol

As mentioned, the Web currently operates with HTTP, a request-response protocol that requires its information and service providers to wait for requests before they can respond. If the response does not arrive within a certain timeout period, the request is canceled or undergoes a time-out. This model works efficiently with some Web tasks, such as a browser sending a request to a server for a Web page and waiting for a reply in order to display the page. But most of life is asynchronous and loosely-coupled. For instance, participants in a face-to-face auction do not repeatedly ask for the current bid on an item for sale. They hear information announced by an auctioneer or see they it immediately when it

changes on an auction board. With the right protocol, an online auction can present relevant data as immediately and continuously as a live, face-to-face auction.

Publish/Subscribe with Presence

A publish/subscribe protocol is an asynchronous protocol, and presence services comprise one of the types of services that it supports. Together, publish/subscribe and presence services can provide specific information updates on the Web immediately when the information changes (1). The publish/subscribe protocols of presence services are based on a principal-based publish/subscribe protocol model described by the Internet Engineering Task Force (IETF) for providing presence services. The IETF model and the requirements for a principal-based publish/subscribe protocol are described in RFC 2778 and RFC 2779 (2, 3).

As defined in RFC 2778, “presence clients” are the users of a presence service or other publish/subscribe service, and presence has two distinct types of clients, the “presentity” and the “watcher.”

The presentity provides the information required for service and publishes it to a publish/subscribe service on behalf of an entity called a “principal,” which is a person or any client device that can communicate with a server. The information needed by each principal that the presentity represents is structured, and that structured information is referred to as a “tuple.” A tuple is a data object with two or more components, and is typically a User ID, User Status, Subscription, or other information needed to establish the service. Different principals can be represented by tuples with different structures, and those structures depend on the requirements of the publish/subscribe service that the tuple supports.

In a presence service, the published information is known as “presence information” and it includes the status of a user. Optionally, it can also include information such as the telephone and e-mail contact information used in instant messaging.

The second presence client is that watcher, and it represents a principal that is subscribed to, or watches, the tuple of a publishing principal. A change in the tuple indicates an update in the information, and when it sees a change the watcher pushes the update to the subscriber. A specific type of watcher is the poller, which updates data on a regular, or polling, basis. It keeps the information on the publish/subscribe service continuously up to date, as needed in online auctioning.

Publish/Subscribe with Request-Response

Most presence services, such as those used with instant messaging services, collect and share a principal’s (i.e. a user’s) status, communication address, some notes, and little else. Instant messaging applications use a separate protocol to deliver instant messages to their users. In fact, presence and the underlying publish/subscribe protocols used by presence services are seen as a service for setting up communication using some other protocol. This certainly has its place, but our view is that the principal-based publish/subscribe protocol model is a general purpose protocol capable of supporting many types of applications, including those that require a request-response interaction pattern.

An alternative to using multiple protocols to support request-response and asynchronous communications, including publish/subscribe interaction patterns, is to define or identify a protocol that consolidates the capabilities of multiple protocols. Both TCP/IP and HTTP are examples of protocols that

have consolidated a number of applications and protocols. With consolidation in mind, we defined a simple, efficient, request-response protocol that also has the capabilities of a principal-based publish/subscribe protocol.

HTTP is simple because it has few commands, and only two of them are needed to support most Web traffic: GET and POST. According to established IETF standards, the principal-based publish/subscribe protocol is similar to HTTP in that it is simple and focused on basic operations, as opposed to specific task operations. Additionally, a principal-based publish/subscribe protocol has only three basic commands:

- PUBLISH: A command that provides information for updating a tuple that represents a principal for which the PUBLISH command is sent.
- NOTIFY: A command that identifies changed information in the tuple. It is sent in response to a change in a tuple that represents a publishing principal to a subscribing or watching principal.
- SUBSCRIBE: A command sent on behalf of a watching principal that requests notifications in response to changes to an identified tuple of publishing principal.

As with HTTP, the simplicity of the publish/subscribe protocol model permits flexibility and expansion. With this flexibility, we created a request-response interaction model that uses the commands of the principal-based publish/subscribe protocol model. Supporting request-response communication over a principal-based publish/subscribe protocol provides a loosely-coupled request-response system that we describe using an online auctioning service.

An Example: Online Auctions

Online auctions are particularly suitable to automated request-response services because their users need to see real-time data on bids and status. During an online auction, sellers and buyers are all connected to an item for sale, and when changes are made to the price or status of the item, the sellers and buyers should all be able to see the changes simultaneously. With HTTP-based request-response, users must refresh their Web pages manually and repeatedly to see the changes. With presence, request-response, and publish/subscribe working together, the Web page refresh automatically. Users can see real-time bids, status, and other information on any device that can provide browser connection to the auction service.

Figure 1 shows the general configuration of a system that provides request-response over a publish/subscribe protocol. The system includes a publish/subscribe server that is configured to receive, store, and distribute information to and from various publish/subscribe clients configured to exchange information with one another via the publish/subscribe server.

The publish/subscribe server also can provide account services, payment services, and other services needed to support auctions.

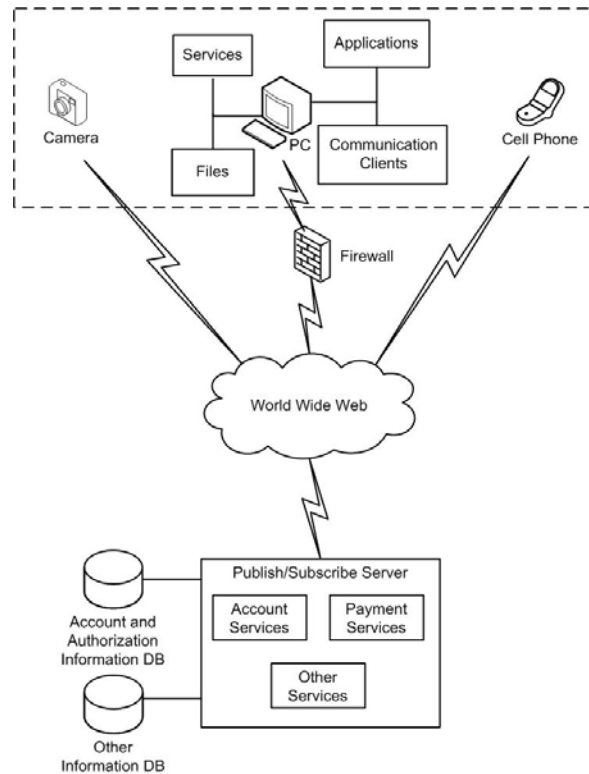


Figure 1. The general configuration of a real-time auctioning system that provides request-response over a publish/subscribe protocol.

Subscribing to an Auction

Typically, an online auction is initiated by a seller who wants to make merchandise or services available to buyers on the Web. In an automated Web system, sellers publish information about items that are for sale to a server. A seller sends the information to the publish/subscribe server and preferably receives confirmation that it arrived. From there, the information is available to anyone interested in participating in the auction for the item that is for sale. The seller can define the general buyer audience, such as “chess players,” “boats,” or “antiques,” but does not know who the specific members of the audience are. Or the participants in the auction can be a particular group of users such as friends of the seller or members of an organization such as antique wholesalers. This is termed a “loosely-coupled” connection. Once a buyer (subscriber) indicates an interest to engage in what the seller (publisher) is offering, the request-response connection can be tightened to provide a more direct connection between them, when needed.

When a potential buyer sees a published auction she is interested in, she subscribes to it. The publish/subscribe server is configured to receive a subscription to publish the required auction information and to provide her with an interface to the seller’s auction.

The server sends the buyer’s subscription to the seller using one of the publish/subscribe protocol’s commands. The simple NOTIFY command provides real-time updates and is used in applications where broadcasting the subscription within a group of interest watchers is appropriate. Broadcasting is open communication, much like a radio. The DIRECTED PUBLISH/NOTIFY command provides tighter coupling

and is used when greater security or privacy is needed between the publisher and the subscribers during the subscription, as is the case with online auctioning. It allows more direct request-response connections between the publisher and each subscriber.

To finalize the buyer/seller connection, the publisher and subscriber then share any additional information required for the auction to take place. Included in this setup is presence, which indicates the online user status of each auction participant. Once a subscription is established for a buyer, the auction can start. Other buyers can join by establishing subscriptions.

Using publish/subscribe-based request-response communication, the Web devices of buyers and sellers can also provide more advanced business tools. For instance, the device of a publisher or subscriber can be set up to automatically respond to the results of another subscription, such as the end of an auction that had no final purchase. A device can also have an interface that gathers information required to decide whether or not to publish or subscribe, then automatically publish an item or subscribe to another auction based on that decision. Many sophisticated associations can be used to enhance online auctioning when the publish/subscribe protocol and presence services are used that include request-response capability.

Placing Bids

When the seller's device receives a bid from a buyer, the Web page is updated by the protocol. The update includes a link that represents a tuple associated with the auction. All auction participants, publishers and subscribers, are notified of the bid immediately and automatically.

The final bid is confirmed for all auction participants according to whatever limits and conditions are set during subscription. When the seller closes the final bid and a buyer is selected, the purchasing process begins.

Purchasing

Purchasing can have several steps, from completing the purchase and placing the order to processing the shipment of the purchased item. Other services can be added to the automated process as needed, such as identity authorization, bank or credit authorization, shipping, inventory management, customer receipt, and customer notification throughout the process. All of these can use the request-response operations capabilities provided by the publish/subscribe service.

To transfer money from a financial account provider, such as a credit card company or a bank, the system can maintain a database of accounts for users who choose to be account holders. Purchasers do not need to be account holders; they can be employees, parents, guardians, or any representatives designated and defined by the account holder. The authorization rules for the different users can vary; for instance, they can have different payment limits or automatically ship items to different locations.

Additional Services

The system can be further developed to use online rosters and privacy lists to authorize and authenticate buyer access to the system or to prevent auctioneers from advertising to subscribers. Roster and privacy list data can be stored in a database, such as the account and authorization information database associated with credit account services.

The system can also include online context-sensitive help files with instructions that users need to join and participate in auctions and to purchase items.

Services can reside on the publish/subscribe server or on separate servers, as long as they can communicate properly through the network. The various types of services, their connection to the network, and advancements in their communication are vast (4).

A map of auction bidding and purchasing workflow, including auction opening and closing, tuple updates, and user authorizations, is shown in Figure 2.

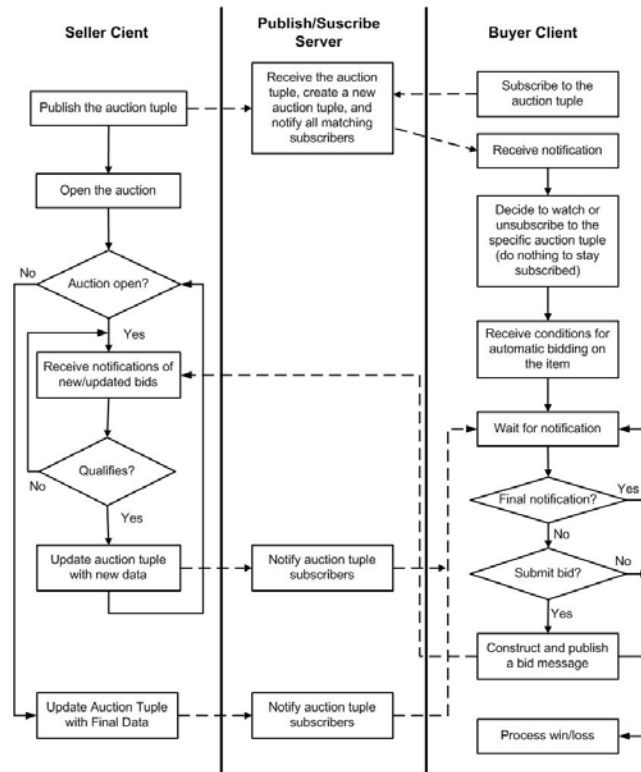


Figure 2. Updating auction tuple information and processing an auction subscription using a publish/subscribe protocol.

The Future of Asynchronous Workflow

The automated auctioning process exemplifies the future benefits of using loosely-coupled request-response, and it is clear to see its potential in a wide range of workflow settings. The system described above allows the development of enhanced applications, processes, and services with characteristics that are not available in conventional systems that use publish/subscribe protocols or request-response protocols alone.

Other online applications that can benefit from this system today are stock monitoring and purchasing operations, real estate searching and buying, workshop classes, chat rooms and social networks, networked libraries, and travel planning and scheduling, to name a few.

Working Asynchronously in a Synchronous Web

Asynchronous communications between buyers and sellers is not mandatory when it is not needed. The publish/subscribe protocol with presence and request-response is compatible with today's Web environment, so known synchronous protocols can be used with it as well. For instance, in a sales or auction system a response to an order may be sent automatically by e-mail using SMTP. In that scenario, the publish/subscribe protocol's transactions can be reserved for real-time communications that occur after an item is located, such as showing its status, pricing, bids, inventory, and expected delivery date.

The design of a presence, request-response, publish/subscribe protocol that works in the current HTTP-based Web environment is an on-going project in our organization (5). The simplicity and flexibility of this design are conducive to many adaptations, and their design and description are integral to many of our pursuits.

Acknowledgement

Scenera Research LLC thanks Julie Tomlinson for her research, writing, and editing.

References

1. R.P. Morris. 2009. [Waking Up the Web](#). White paper, Scenera Research LLC.
2. M. Day, J. Rosenberg, and H. Sugano. 2000. A model for presence and instant messaging. Internet Engineering Task Force, RFC 2778: <http://www.ietf.org/rfc/rfc2778.txt>.
3. M. Day, S. Aggarwal, G. Mohr, and J. Vincent. 2000. Instant messaging/presence protocol requirements. Internet Engineering Task Force, RFC 2779: <http://www.ietf.org/rfc/rfc2779.txt>.
4. R.P. Morris. 2006. Method, system, and data structure for providing a general request-response messaging protocol using a presence protocol. US Patent and Trademark Office, Patent Application 2006/0280166.
5. R.P. Morris. 2009. [The Evolution of HTTP](#). White paper, Scenera Research LLC.

Contact

Contact Scenera Research at info@sceneraresearch.com.