System and Method to Monitor and Transfer Hyperlink Presence

Method to monitor and transfer hyperlink presence information, including: transmitting a hyperlink presence monitor request to a web server; receiving an hyperlink presence information; and rendering a hyperlink based upon the hyperlink presence information. Optionally, the hyperlink presence information is represented as one of: attributes of a hyperlink element; HTML5 microdata; RDFa data; and XHTML message data. Optionally, the hyperlink presence information includes: a last-update information to indicate when the hyperlink presence was last updated; a last-status information to indicate a latest hyperlink presence status; a monitor information to indicate a URI that points to a resource to monitor for hyperlink presence; a status-list information to indicate a list of hyperlink presence status values; and a target information to indicate the hyperlink to be monitored.
FIG. 3

Dedicated Application
proxy/client
Heterogeneous networks and protocols
REST
web sites for storages
phone

Web Browser
proxy/client
REST
phone
SYSTEM AND METHOD TO MONITOR AND TRANSFER HYPERLINK PRESENCE

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Patent Application Ser. No. 61/471,507, filed on Apr. 4, 2011, the content of which is hereby incorporated by reference in its entirety.

BACKGROUND

[0002] 1. Field of the Invention
[0003] Embodiments of the present invention generally relate to hyperlink presence. More specifically, embodiments of the present invention relate to a system and method for monitoring presence information of a hyperlink, and rendering the presence information to a user.
[0004] 2. Description of the Related Art
[0005] An architectural style that underlies the Web is REpresentational State Transfer (“REST”). A web service that is compatible with REST is said to be “RESTful.” REST provides resource management and promotes architectural choices that include: 1) Addressability—each resource can be addressed by Uniform Resource Identifier (“URI”); 2) Connectedness—resources are linked to provide navigations; 3) Uniform Interface—all resources support a subset of a uniform interface between components, namely GET, PUT, DELETE and POST; 4) Statelessness—all requests to a resource contain all of information necessary to process the requests, and the servers do not need to keep any context about the requests; and 5) Layering: intermediate proxies between clients and servers can be used to cache data for efficiency. Because of its relative simplicity, compared to SOAP-based web services, REST web service paradigm is gaining popularity for use in implementing mobile device applications. Researchers have also compared the performance of the same SOAP and REST services implemented on mobile devices. REST was found to be more suitable to mobile devices since REST required less processing time and memory.

[0006] Constrained REST Environments is protocol that restricts HTTP when executing on low-end devices, such as 8-bit microcontrollers with up to 12 KB of memory. The Constrained REST Environments protocol binds HTTP to UDP for asynchronous messages. However, this approach is not suitable for mobile environments, because mobile phones do not have reachable IP addresses.

[0007] Researchers in the field have proposed using REST web service framework for a number of applications, such as: to implement mobile commerce spaces; to support services on mobile computing units in connection with semantic web technology; to expose IMS capabilities to Web 2.0 applications; to support person-to-person communication and collaborations over WiFi and 3G networks; and to provide a web service provisioning system having much lower overhead than a SOAP counterpart.

[0008] Researchers have also presented a REST architecture to render web interfaces on mobile devices in which the REST protocol is used to synchronize between applications.

[0009] Researchers have described an approach to adapt resource-oriented service framework to automatically control robots for industrial automation tasks. This approach uses a special protocol called MRROR to handle events. Performance of this framework was evaluated under three physical networks: wireless LAN, Bluetooth and ZigBee. The results showed that the REST framework had a lower overhead than SOAP-based Devices Profile for Web Services (“DPWS”).

[0010] Wireless Application Protocol (“WAP”) is a suite of protocols to connect wireless mobile phones to the Web. The typical WAP architecture includes a Content Server, a WAP Gateway, and mobile devices. When requested by the Content Server, the WAP Gateway uses Wireless Session Protocol (“WSP”), which is a binary version of HTTP, to transfer encoded Wireless Markup Language (“WML”) content to the mobile devices. However, modern smart phones rarely support WAP because they can interpret HTML directly.

[0011] None of the research discussed above has studied how to compact HTTP for mobile devices and how to expand HTTP to connect phones with each other to enable a collaborative endpoint network in a mobile phone environment and to link functions on the phones with enterprise applications.

[0012] Hyperlinks in hypertext can identify resource located in distributed hypertext systems. The most popular such system is the Web. As the Web converges with real-time communication and collaboration, uniform resource indicators (“URI”) (i.e., hyperlinks) are used to identify people in communication and collaboration. However, when a hyperlink is broken, there is no way for a user of the hypertext system to know when it will be available again unless the user constantly polls it. This polling process, similar to playing phone tag, is very inefficient and often leads to increased communication cost and time.

[0013] Presence is the availability and willingness of an entity to communicate. It has been used to reduce the uncertainty and cost of communications. However, there is currently no mechanism to associate presence information with hyperlinks in distributed hypertext systems. As a result, there is no efficient way for a user to keep track of the status of a resource pointed to by a hyperlink in real-time communication and collaboration over the Web. To address this issue, there is a need for a mechanism to monitor and display real-time presence information associated with hyperlinks in hypertext.

[0014] The HTTP 1.1 protocol provides status codes and response headers to indicate the status of a resource. For example, “404 Not Found” in response to a resource request indicates the resource does not exist. One of the response headers is “Retry-After” which indicates a resource is temporarily unavailable and suggests that the user agent retry the resource request sometime in the future. However, the “Retry-After” header does not guarantee that the resource will be available by the suggested time. Furthermore, the status codes and response headers are not available until the user agent requests it. The combination of these two factors forces the user agent into an inefficient polling mode, when it should get the status in real-time.

[0015] Some user agent, like a web browser, renders a hyperlink differently (e.g., using a different font, color, boldness, etc.) according to whether the hyperlink has been visited or not. Many web browsers keep track of a users’ browsing history, which is a set of visited hyperlinks and their dates. This browsing history indicates that the referenced resource was available at some time in the past.

[0016] Presence monitoring and updating protocols (e.g. SIP, XMPP etc.) have been developed and used in various communication systems, such as online chat, IM and IP phones (e.g., Skype, Google Talk, etc.). These protocols typi-
cally follows a subscribe-notify design pattern, in which the user agent subscribes to a presence server, which in turn notifies the subscribers the presence updates. The presence attributes and values are defined by the presence monitoring and updating protocols and can be extended by implementations if necessary.

[0017] Extensible Messaging and Presence Protocol ("XMPP") (RFC3920) supports efficient bidirectional XML streams between XMPP servers using two TCP/IP connections. This creates bidirectional notification flows between XMPP servers. However, XMPP protocol is not based on REST web services. Although BOSH (XEP-0124) uses HTTP long-polling to emulate bidirectional TCP streams, the established streams are not web resources that can be manipulated by HTTP. XMPP also supports a publish/subscribe extension (XEP-0060) to allow XMPP entities to subscribe and publish events to topics. But these subscriptions are unidirectional and not web resources.

[0018] None of the research discussed above has studied how to provide presence information for hyperlinks. Thus there is a need to provide a protocol that provides presence information or status for hyperlinks in general.

SUMMARY

[0019] As mobile phones become more and more advanced, they are replacing other devices, such as personal digital assistants ("PDAs") and notebook computers, as the next-generation PDA. Compared to other computing devices, the mobile phones offer a desirable combination of telephony functions (e.g., making and receiving phone calls), sensory functions (e.g., sound, camera, camcorder, location, acceleration, temperature, etc.) and communication networks (3G, WiFi, Bluetooth, WiMAX, etc.). More and more applications are being designed for mobile phones to take advantage of these capabilities, as evidenced by the popularity of iPhone and Android applications in the Apple App Store.

[0020] Embeddings in accordance with the present invention may include a web service approach to enable collaborative endpoint network for mobile phones and to expose their functions on mobile phones as REST web services, such that applications running remotely can monitor and control them in a near real-time manner. Collaborative endpoint network is an emerging area with applications in intelligent home network, etc. There are many motivating use cases to extend the collaborative endpoint network to mobile phones as a surveillance device to monitor and remind patients of their treatment, as well as find and locate medical professionals to treat them. A travel application is to use mobile phones as a virtual tour guide, and push relevant multimedia content to a visitor as he/she moves around a tourism site. Mobile phones are also an ideal device to keep track of traffic flows when the speed of many drivers can be obtained and aggregated automatically. All these applications require the ability to monitor and control one or more functions on the phone in near-real time, as these functions, such as location, may change frequently and in-time responses are needed.

[0021] An efficient approach to supporting these applications is to expose the fundamental functions on the phones as REST web services and make mobile phones as web service endpoints, so that services on mobile phones can be invoked and composed in different ways by different applications. This approach eliminates the need for each application to duplicate the same function. Making a phone into a web service endpoint enables the applications to interact with the phones in heterogeneous mobile environments, as web service is independent of transport protocols and programming languages. REST web service is easy to extend as it supports dynamic discovery through links. For example, to add a second camera on the phone into the services, we just need to implement a new camera resource and link it the main resource.

[0022] Two types of web services are SOAP-based and REST-based. Known implementations have chosen REST-based web services for mobile phones because of simplicity and close relationship with the architecture of the Web. However, the HTTP 1.1 protocol used in current REST web suffers several shortcomings. First, the HTTP 1.1 messages can be complex and large, while some features of HTTP 1.1 are not used in mobile phones. Second, HTTP 1.1 should be expanded to support multiple transport protocol bindings besides TCP/IP, in order to support REST services in heterogeneous environments. Therefore, a need exists to provide both compactness and expansion of the HTTP 1.1 protocol in order to support current REST services in a mobile phone environment. To address these issues, embodiments in accordance with the present invention construct a "Compact HTTP" protocol, which includes a small subset of HTTP 1.1 that keeps only the elements of HTTP 1.1 that are used to enable a collaborative endpoint network of mobile phones.

[0023] Embeddings in accordance with the present invention describe how Compact HTTP can be bound to multiple messaging protocols, in particular to XMPP and Short Message Service ("SMS"). These protocol bindings introduce asynchrony into REST to support event-driven REST web services on mobile phones. Furthermore, HTTP over XMPP in accordance with embodiments of the present invention introduces hyperlink presence into REST to mitigate the broken link issue that is important for mobile phones. Embeddings in accordance with the present invention also provide a security profile that is useful to afford flexible and quick setup of security contexts between services and clients.

[0024] Based on this protocol, embodiments in accordance with the present invention provide a lightweight web services framework on an Android mobile phone. Within this framework, a plurality of resources, including sound, camera, camcorder, location, power, motion, scheduler, and telephony manager are implemented as secured REST web services. The collaborative endpoint network framework also supports web storage (e.g., Google sites, YouTube, etc.) in order to upload recorded media for instant sharing and collaboration.

[0025] Embeddings in accordance with the present invention provide a method to monitor and transfer hyperlink presence information, including: transmitting a hyperlink presence monitor request to a web server; receiving an hyperlink presence information; and rendering a hyperlink based upon the hyperlink presence information. Optionally, the hyperlink presence information is represented as one of: attributes of a hyperlink element; HTML5 microdata; RDFa data; and XHTMLe message data. Optionally, the hyperlink presence information includes: a last-update information to indicate when the hyperlink presence was last updated; a last-status information to indicate a latest hyperlink presence status; a monitor information to indicate a URI that points to a resource to monitor for hyperlink presence; a status-list information to indicate a list of hyperlink presence status values; and a target information to indicate the hyperlink to be monitored.
Embodiments in accordance with the present invention provide a system to monitor and transfer hyperlink presence, including: a monitor module configured to transmit a hyperlink presence monitor request to a web server; a receiver configured to receive a hyperlink presence information; and a rendering module configured to render a hyperlink based upon the hyperlink presence information.

Optionally, the hyperlink presence information includes: a status-list information to indicate a last time when the hyperlink presence was updated; a last-status information to indicate a latest hyperlink presence status; and a monitor information to indicate a URI that points to a resource by which a user agent can monitor the hyperlink presence; a status-list information to indicate a list of hyperlink presence status that the hyperlink can have; and a target information to indicate the hyperlink to be monitored.

Optionally, the hyperlink presence information is represented as one of: attributes of a hyperlink element; HTML 5 microdata; RDF a data; and XHTML message data.

Optionally, the hyperlink presence information is received from a presence server that is separate from a server hosting the hyperlinked resource.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a message template pair, for a Compact HTTP request and response in accordance with an embodiment of the present invention;

FIG. 2 illustrates a phone interface for an operator to carry out steps of a process in accordance with an embodiment of the present invention;

FIG. 3 illustrates high level relations between client applications and REST services on the phones, in accordance with an embodiment of the present invention;

FIG. 4 illustrates a high level architecture of REST service framework on Android phone, in accordance with an embodiment of the present invention;

FIG. 5 illustrates a screenshot of web storage activity, in accordance with an embodiment of the present invention;

FIG. 6 illustrates a comparison of mean interceptor times, in accordance with an embodiment of the present invention;

FIG. 7 illustrates an overall web architecture to support hyperlink presence, in accordance with an embodiment of the present invention;

FIG. 8 illustrates track hyperlink access in accordance with an embodiment of the present invention;

FIG. 9 illustrates subscribing to a hyperlink presence through a web server and a presence server, in accordance with an embodiment of the present invention; and

FIG. 10 illustrates Transfer presence information across user agents, in accordance with an embodiment of the present invention.

The headings used herein are for organizational purposes only and are not meant to be used to limit the scope of the description or the claims. As used throughout this application, the word may is used in a permissive sense (i.e., meaning having the potential to), rather than the mandatory sense (i.e., meaning must). Similarly, the words “include”, “including”, and “includes” mean including but not limited to. To facilitate understanding, like reference numerals have been used, where possible, to designate like elements common to the figures. Optional portions of the figures may be illustrated using dashed or dotted lines.

DETAILED DESCRIPTION

Embodiments of the present invention generally relate to hyperlink presence. More specifically, embodiments of the present invention relate to a system and method based upon Representational State Transfer for monitoring presence information of a hyperlink, and rendering the presence information to a user.

As used herein in connection with embodiments of the present invention, the term “REST” refers to Representational State Transfer web services, as described below in further detail.

As used herein in connection with embodiments of the present invention, the term “RESTful” refers to a web service that is compatible with REST.

As used herein in connection with embodiments of the present invention, the term “R-Event” refers to a RESTful web service framework, and in particular to a RESTful web service framework which is usable to implement distributed event-based systems.

As used herein, the term “module” refers generally to a logical sequence or association of steps, processes or components. For example, a software module may comprise a set of associated routines or subroutines within a computer program. Alternatively, a module may comprise a substantially self-contained hardware device. A module may also comprise a logical set of processes irrespective of any software or hardware implementation.

As used herein in connection with embodiments of the present invention, the term “resource” refers to a special component that is addressable by URI and supports a uniform interface as defined by REST. Any information that can be named is a resource: a document or image, a temporal service (e.g. “today’s weather in Los Angeles”), a collection of other resources, a non-virtual object (e.g. a person), and so on. In other words, any concept that might be the target of an author’s hypertext reference fits within the definition of a resource. A resource is a conceptual mapping to a set of entities, not the entity that corresponds to the mapping at any particular point in time.

Compact HTTP

HTTP 1.1 has a set of very powerful features that can be too expensive to implement and support on resource-constrained mobile devices. Even though it is possible to run an HTTP 1.1 server on Android mobile phones, many of its features may never be utilized. Therefore, there is a need to develop a lightweight compact protocol, “Compact HTTP,” for mobile phones. Compact HTTP includes a subset of the HTTP 1.1 protocol, but yet is still useful to provide the endpoint network and collaborative applications of mobile phones. Protocol compaction as described herein is particularly useful for lightweight or otherwise less powerful devices.
[0050] For more powerful devices, either presently or as devices become more powerful in the future, compaction is less useful. For this reason, examples presented herein may illustrate HTTP messages in plain text, rather than binary, because the compact subset of HTTP 1.1 may grow to support further extensions.

[0051] Message Templates

[0052] To reach a compacted subset, embodiments in accordance with the present invention start from an empty feature set and add features to it as needed until the desired services are covered. For example, a Compact HTTP request and response template that follows HTTP 1.1 is shown in FIG. 1. All of the variables illustrated in FIG. 1, including "{operation}," "{path}," "{version}," "{status}," and "{reason}" are defined in HTTP 1.1. For Compact HTTP, the version is HTTP 1.1c. "{form}" is defined by HTML 4 (also known as HTML 1999), whose media type is "application/x-www-form-urlencoded." The differences between Compact HTTP and HTTP 1.1 are described below.

[0053] “Authorization” contains the access token for the message. In HTTP 1.1, “Authorization” is a request header. Embodiments in accordance with the present invention extend “Authorization” to responses because responses may be sent in a different connection. Therefore, the client needs to authorize a response before taking any action (e.g., to update a user interface).

[0054] “x-tid” is a new header to HTTP 1.1 that clients may use to correlate asynchronous responses and events to requests. A value for “x-tid” is set in a request and echoed in the responses.

[0055] The templates shown in FIG. 1 by design omit some headers considered important for REST services, such as the ETag response header. This is because the resources of the mobile device that Compact HTTP is designed to run on tend to have small representations that can be updated and transmitted without checking the versions. A representation of a resource is a stream of bytes returned by the resource. Example representations include a HTML document, JPEG or XML document. In web architecture, a user agent (e.g., a web browser) may cache the representation of a resource to reduce latency and network traffic. When the user agent requests the latest version of the representation, it may use ETag, in combination with other HTTP headers, to ask the server to send the representation only if it is different from what has been cached. In embodiments in accordance with the present invention, such checking is not used. The representation may be, for instance, as measured by the maximum size of an HTTP form. Embodiments in accordance with the present invention also omit content negotiation headers in favor of using URI.

[0056] Message Exchange Patterns

[0057] Typical HTTP messages follow a request-response pattern. Services following this pattern are atomic as the service is complete once the response is sent. However, many services on the mobile phones are multi-step instead of atomic. For example, to control a camera to take a picture involves the following steps: 1) adjust focus; 2) take shot; 3) upload picture to a web site. Modeling this type of services as atomic would not be a scalable approach because: 1) it would make the service stateful, which violates the REST statelessness principle; and 2) it makes a client less responsive to service state changes and it is much more expensive to recover from faults.

[0058] Embodiments in accordance with the present invention model the multi-step services as one request followed by multiple responses, in which intermediate responses indicate service progression while the final one indicates service completion. To separate them, the status for the intermediate responses should be 202 Accepted or 206 Partial Content, while the final response should be 200 OK or 201 Created. Status 206 is used only with GET in HTTP 1.1, and here we extend it to any request to convey the current service state as partial content. All asynchronous responses are correlated to its request by x-tid for both atomic and multi-step services.

[0059] Another type of message exchange pattern is event subscription and notifications. For example, a client can subscribe to a phone’s location tracking service to receive notifications about location changes of the phone. In this pattern, a subscription (sent by a client as PUT) is followed by unknown number of notifications (sent by the service as POST). A client should be able to tell which notifications are from which subscriptions, e.g., cancel, etc. To support this feature, the x-tid of notifications would echo the x-tid of the subscription request.

[0060] Design Patterns

[0061] The Compact HTTP does not specify how to design the resources to support these message exchange patterns. Embodiments in accordance with the present invention follow the REST service design patterns, including session, event subscription, multi-resource and multi-state, which are common in real-time communication services.

[0062] Compact HTTP Bindings

[0063] In conventional REST web services, there is a basic and implicit assumption that a HTTP server has a public IP address. However, this is not possible in mobile environment, because a mobile phone is typically behind its provider’s NAT gateway and its private IP address is not reachable from outside. Enterprise applications that control and monitor phones are also behind corporate firewalls. This creates difficulty for many real-time applications in which two-way messaging is required. On the other hand, many 3G mobile phones can join different communication networks over different protocols such that they are reachable without IP addresses. Instead of IP address, a phone can be addressed by a phone number (e.g., SMS protocol), an email address (e.g., SMTP protocol) or Jabber ID (“JID”) (e.g., XMPP).

[0064] Therefore, to support REST services in these heterogeneous environments, it is advantageous to decouple HTTP from TCP/IP. In addition to TCP/IP, it is both convenient and advantageous to treat these messaging protocols as “transport” for HTTP. This approach makes the REST services invariant to the protocol changes as the mobile phone connects to different networks. This allows embodiments in accordance with the present invention to keep the same services while optimizing their performance over available networks and protocols. For example, time sensitive messages may be transmitted over TCP/IP or XMPP, and noncritical messages over SMS or SMTP. Furthermore, HTTP over XMPP can bring presence information into REST architecture. Presence information is known as a status indicator that conveys ability and willingness of (in this situation) a hyperlinked resource to respond to a communication. In particular, we can assign presence to hyperlinks to address the issue of broken links in the Web.

[0065] A tenet of the web service paradigm is that web services should function independently of the transport pro-
Protocol used for providing the web service. Separation of protocol messages and transport is a consequence of that tenet. SOAP based web services embrace this approach by defining SOAP bindings to HTTP (see, e.g., WSDL 2001), to XMPP (see, e.g., SOAP/XMPP 2005), and to JMS (see, e.g., SOAP/ JMS 2009). However, HTTP presently has bindings only to TCP/IP and UDP. Embodiments in accordance with the present invention extend HTTP bindings with other message transport protocols by developing the process and modular components to extend the HTTP bindings. In particular, the binding of HTTP with XMPP is discussed below in detail.

[0066] URI Scheme

[0067] In REST services, any resource must have a URI that identifies where the resource is and how to contact it. Because we want to bind HTTP to different transport protocols, we adopt a two-level URI scheme according to RFC-3986, where the first level URI identifies the HTTP information and the second level URI identifies the transport information:

\[
\text{uri}_1 = \text{http://uri}_2/... \\
\{\text{uri}_2\} = \text{URI}
\]

For example, the format to address a resource \( x \) with HTTP over SMS to a phone number is:

[0069] \text{http://sms:5555/x}

Or alternatively, the format to address a resource \( x \) with HTTP through a SMTP gateway is:

[0071] \text{http://smtp:5555@example.com/x}

To address the same resource with HTTP over XMPP, the format is:

[0073] \text{http://xmpp:joe@example.com/x}

If the phone has an IP address, the URI for the resource would be:

[0075] \text{http://123.4.5.6/x}

To communicate with a resource with \( \text{uri}_1 \), a client first establishes a communication channel according to \( \text{uri}_2 \) and then transmits HTTP messages over the communication channel. This process is elaborated in the next section using XMPP as an example.

[0077] XMPP

[0078] XMPP architecture consists of XMPP servers and clients. To exchange messages, clients establish a TCP/IP connection to the same or federated XMPP servers. Federated servers are a network of XMPP servers connected by the XMPP server-server protocol. Open XMPP servers, such as Google Talk, provide users free accounts with federated identities. For instance, a user can use a valid Gmail account to login to the Google Talk server.

[0079] To connect to a XMPP server, a XMPP client typically needs to know the following information: 1) XMPP host and port (e.g., talk.google.com:5222); and 2) XMPP Service (e.g., gmail.com). This information is not included in URI scheme because the URI scheme is independent of which XMPP server is used. Therefore, the host/port/service information can be saved in a configuration file associated with a user. Once the user logs into the XMPP server, the connection identified by the saved host/port/service information is used to transmit all HTTP messages.

[0080] Chat service is among the base protocols of XMPP and most XMPP clients support this feature. Therefore, embodiments in accordance with the present invention transmit Compact HTTP messages as chat over XMPP. The template for HTTP request and response over XMPP is:

```xml
<message>
  <from>{from_jid}/to={to_jid}>
  <body>Compact HTTP message</body>
  </message>
</message>
```

[0081] Hyperlink Presence

[0082] While browsing the Web, a user may encounter a broken hyperlink, which may occur in certain circumstances when the resource that embeds the links is not aware of the linked resource. When the web server shuts down, or the linked resources that are hosted on the web server are moved or deleted, the link becomes broken. In this situation, it is difficult for the client to know when the link will be restored unless the client polls the server constantly, which creates unnecessary network traffic.

[0083] In accordance with embodiments of the present invention, HTTP over XMPP provides a solution to the problem of broken hyperlinks by using presence services provided by the XMPP layer. If a XMPP client listens for presence updates from a JID address, it can assign presence to a hyperlink containing that address in near real-time. For example, to assign presence to a hyperlink: \text{http://xmpp: someone@gmail.com/x}, the client just needs to monitor the presence of \text{someone@gmail.com}. By knowing the link presence, the client can avoid fetching or polling the broken links all together. The presence services also gives the REST services an ability to change their presence, for example in case of temporary maintenance, while keeping the clients informed in a timely fashion.

[0084] SMS

[0085] SMS is a very common feature in most mobile phones. There are several ways to send and receive SMS messages depending on the networks. In cellular networks that provide SMS, two phones can exchange SMS messages directly. Outside the cellular networks, a client may use a SMS gateway that exposes particular protocols, such as SMTP to send and IMAP to retrieve SMS messages. For example, this approach is available in the T-Mobile™ network, which assigns each G1 phone an email address that ties with its number, for example: [number]@t-mobile.net.

[0086] To bind to SMS, the Compact HTTP messages are contained in the User Data (“UD”) segment of the SMS Protocol Description Unit (“PDU”). To bind to SMTP, the HTTP message may be contained in the email body. Examples are illustrated below in greater detail.

[0087] Security

[0088] Security is an important issue in hosting web services on mobile phones because the open access to resources on the phone may be misused by malicious clients. It is typical to deploy layered security mechanisms from protocol up to applications such that a breach at a lower level can be defended by the layer above. Here, security mechanisms at Compact HTTP, transport protocols, services and applications are employed, as described below in greater detail.

[0089] Secure Messages

[0090] In mobile environments, a HTTP message may travel through different networks in different protocols. Therefore, it is desirable to employ end-to-end message level
security. A symmetric key based security protocol may be used to set up security contexts between two parties.

A design goal is to allow a user who operates the phone to quickly grant and reject clients and service requests. Symmetric key is chosen because our applications are aimed at trusted users who can exchange secret keys easily. For example, the phone and the client may be managed by the same user.

For convenience, the user that configures the phone is referred to as “operator” and the user that configures the application is referred as “director.” The protocol includes the following steps:

1) The operator and director agree upon a secret passphrase P and enter it into the phone and the client respectively.

2) The director creates an access token T1 and tells it to the operator.

3) The operator enters client’s URI A and T1 into the phone and creates an access token T2. The token is sent by a “join” message encrypted by P to URI A as follows:

```
PUT operators/{phone} HTTP/1.1c
Authorization: T1
x-tid: {number}
token=T2
```

4) The client decrypts the message with P, store T2 and sends an encrypted response message that indicates acceptance.

5) The phone receives and decrypts the response and activates the security context, which contains (A, P, T1, T2).

6) Any subsequent message from the client to the phone will contain T2 and be encrypted with P. The phone will decrypt any message from A with P and checks it against T2. Any response message to A will contain T1 and be encrypted with P.

7) The operator deactivates the security context by sending a “Delete” message to the client. The corresponding security context becomes invalid on the phone and client. An example of a “Delete” message is:

```
DELETE operators/{phone} HTTP/1.1c
Authorization: T1
x-tid: {number}
```

Embodiments in accordance with the present invention also secure access to sensitive data collected by the services, such as captured images and videos, by restricting storage of the sensitive data. For example, instead of returning the content to the authenticated client directly, the system stores it locally or stores it in a web storage site and provides a link to the client, who can retrieve it with another set of credentials.

Prototype System

Based on the protocols described above, a prototype system embodiment in accordance with the present invention has been developed that hosts a set of REST resources on T-Mobile G1 phones running Android 1.6 in T-Mobile cellular network. FIG. 3 illustrates relationships of the prototype system between various client applications and the phones, in which REST services are accessible to clients of different kinds over the heterogeneous networks.

FIG. 4 illustrates a high-level REST server architecture on an Android phone. There are three types of Android services in the framework: transport services 410, REST services 420 and storage service 430. The Android services may comprise components developed from the Android software development kit (“SDK”) and the Java SDK. Typically, transport services 410 and a portion of REST services 420 not within REST framework 421 are developed from the Android SDK, whereas storage services 430 are developed from the Java SDK. The core REST framework 421, including the security package 424, is also not developed from an Android packages and can be run in any Java runtime. REST framework 421 may rely upon at least one resource 422 that is developed from the Java SDK and at least one resource 423 that is developed from the Android SDK.

Transport services 410 are responsible for listening and sending messages over a transport protocol, such as SMS or XMPP. For XMPP transport, we used a XMPP client library compiled for Android. For SMS, we used Android SmsManager. If an incoming message is intended for the REST Service 420, it is forwarded to the REST Service 420 as an Intent on Android platform.

The REST Service 420 is an Android Service that contains the REST framework 421. Upon starting, the REST Service 420 registers an Intent Listener. Upon receiving a message encapsulated in the Intent, the Intent Listener looks up the security context 424 for the message and invokes the interceptor chain 425 in the REST framework 421 to process the message.

Interceptor chain 425, within REST framework 421, contains an incoming interceptor chain, a pivot, and an outgoing interceptor chain. The incoming chain consists of three interceptors to 1) decrypt; 2) de-serialize; and 3) authorize the message. If any of these interceptors fails, the message is discarded and an error message is returned. If the message is a request, the pivot will invoke the corresponding resource 422 or 423; if the message is a response, the pivot forwards it to the Android Notification Service. The outgoing chain consists of four interceptors to 1) encode; 2) serialize; 3) encrypt; and 4) deliver the responses. The chains can also be invoked by a resource 422 or 423 to control another resource 422 or 423. For example, a scheduler resource controls the camera by going through the same incoming chain for security reasons.

For message encryption and token generation, we used Java Crypto packages (javax.crypto and java.security) with password based encryption algorithm.
"PBEWithMD5AndDES." The encrypted messages are encoded as Base64 strings for transmission.

The REST resources in our approach can use web storages provided by Google to upload captured audio (Google Docs), image (Picasa) and video (YouTube) contents, so that they can be instantly shared for collaboration. The storage service is an abstraction of local and web storages with three methods: login(account), logout( ) and save(url, content). It uses a set of HTTP clients to upload multimedia contents to the designated server and publish the services to a web proxy (e.g. Google Sites). The phone interface to manage web storage is illustrated in FIG. 5.

### Implemented Resources

Table 1 lists the major resources developed within our collaborative endpoint network framework. Each resource is described by one table that defines its path, service, operations and response patterns, where "a" stands for atomic or "m" stands for multi-step.

#### TABLE 1

<table>
<thead>
<tr>
<th>Method</th>
<th>Path</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>sound/control</td>
<td>Retrieve current state</td>
</tr>
<tr>
<td>PUT</td>
<td>camera/control</td>
<td>Record or playback sound</td>
</tr>
</tbody>
</table>

### Implemented Clients

Two types of clients were implemented: the Android phone (both client and service) and a dedicated Java desktop client. The Android phone client can control and monitor other Android phones running REST services using HTTP over SMS or XMPP. The Java client, based on open source Smack 3.1.0 XMPP library (Smack API 3.1.0), controls and monitors REST services using HTTP over XMPP only.

### EXPERIMENTAL RESULTS

Performance of the REST based collaborative endpoint network framework is a consideration. On the server side, we measured the total processing time (from entering the incoming chain to leaving the outgoing chain) as well as the processing time of individual interceptors. On the client side, we measured the round-trip latency (which includes the network latency and XMPP library processing time). To obtain the time, we used the XMPP Java client on a desktop computer (Dual Core CPU 3.60 GHz with 2 GB RAM) connected to the Internet to send 52 HTTP messages over XMPP to each REST service on a T-Mobile HTC G1 phone (Firmware 1.6, Build DMD64) registered in T-Mobile cellular network. The measurements on the phone were collected using Android TimeLogger utility and on our Java desktop client using Java System.nanoTime function. The time performances are summarized in Table 2 and the message sizes are listed in Table 3.

#### TABLE 2

<table>
<thead>
<tr>
<th>Process</th>
<th>Client</th>
<th>Server</th>
<th>Encrypt</th>
<th>Decrypt</th>
<th>Deserialize</th>
<th>Serialize</th>
<th>Authorize</th>
<th>Endorse</th>
<th>Pivot</th>
<th>Deliver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>1400.6</td>
<td>821.7</td>
<td>554.3</td>
<td>3858.3</td>
<td>100</td>
<td>20</td>
<td>40</td>
<td>201</td>
<td>291</td>
<td>254</td>
</tr>
<tr>
<td>Std</td>
<td>1.3</td>
<td>0.2</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>Min</td>
<td>1400.6</td>
<td>821.7</td>
<td>554.3</td>
<td>3858.3</td>
<td>100</td>
<td>20</td>
<td>40</td>
<td>201</td>
<td>291</td>
<td>254</td>
</tr>
<tr>
<td>Max</td>
<td>6000.0</td>
<td>3000.0</td>
<td>1000.0</td>
<td>0</td>
<td>2000</td>
<td>100</td>
<td>300</td>
<td>2000</td>
<td>2000</td>
<td>2000</td>
</tr>
</tbody>
</table>

#### FIG. 6 illustrates the mean times spent in the interceptors. The graph shows that, on average, encryption, decryption and pivot are the top three time consuming components, taking up approximately 69% of total server time. Notice that the pivot time is the mean time spent by the resources to execute HTTP methods, which is outside the control of the framework. Table 3 shows that relatively small messages are sufficient to support a variety of basic services such as those referred to in Table 1.

#### TABLE 3

<table>
<thead>
<tr>
<th>Type</th>
<th>Encrypted</th>
<th>Decrypted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>113.36</td>
<td>77.93</td>
</tr>
<tr>
<td>Std</td>
<td>28.6</td>
<td>21.54</td>
</tr>
<tr>
<td>Min</td>
<td>88</td>
<td>50</td>
</tr>
<tr>
<td>Max</td>
<td>192</td>
<td>138</td>
</tr>
</tbody>
</table>
least that the proposed approaches and architecture are feasible, efficient, and extensible.

[0123] Embodiments Using Hyperlink Presence

[0124] The browsing history provided by web browsers does not indicate the current status of visited resources. Many other hypertext user agents, such as an Email client, a PDF Reader, and MS Word, do not provide such browsing history at all. Furthermore, these user agents do not provide a mechanism to transfer the presence information of a hyperlink to other user agents. When a user emails a hyperlink to someone else, the status of the hyperlink is unknown to the recipient. It is difficult for the recipient to know if the hyperlink is correct or has worked without this information.

[0125] The current presence protocols can only be used with a particular type of resources defined by the protocol. For example, SIP can only provide presence for resources identified by SIP URI, whereas XMPP only for resources identified by JID. These protocols and systems cannot associate presence information with resources identified by other URI schemes, for example HTTP. A user agent of these presence protocols (e.g. a GoogleTalk client) only understand a single protocol. Therefore, none of these presence protocols by itself is sufficient to provide presence for hyperlinks in general.

[0126] FIGS. 7-10 present several embodiments using hyperlink presence, each embodiment having a number of structural elements. It should be understood that a reference number assigned to a link between structural elements may refer to either the communication path itself or to the message (s) carried by the communication path. Multiple links between a pair of structural elements may refer to multiple communication paths, or may refer to a single communication path that carries multiple messages. The distinction that is intended when describing the embodiment will be apparent from the surrounding text.

[0127] FIG. 7 illustrates a simplified web architecture 700 in accordance with an embodiment of the present invention. Web architecture 700 provides for an association of resource presence information with its hyperlink in a hypertext document, and displays the hyperlink differently according to the presence information. The number and type of presence information states may vary from one hyperlink to another. Web architecture 700 includes a client 710 and a web server 720. Client 710 may include a user agent mark-up module 701 (e.g. web browser) that is able to describe hyperlink presence in hypertexts (e.g. HTML page) so a user agent can monitor it accordingly. User agent mark-up module is separate from the monitor module 702 and the render module 703, and is part of the user agent. Client 710 may further include a monitor module 702 that functions as a presence watcher for user agent 701. Client 710 may also include a render module 703 to display hyperlink presence information from monitor module 702 and user agent mark-up module 701. Client 710 may also include a hyperlink transfer module 704 to pass hyperlink presence information across user agents.

[0128] User agent mark-up module 701, monitor module 702, render module 703 and hyperlink transfer module 704 may be in communicative contact with one another within client 710. In operation, a monitoring function within user agent mark-up module 701 communicates with web server 720. The monitoring function informs web server 720 by use of monitor message 731 that client 710 wants to receive presence updates from web server 720. Web server 720 responds by use of an update message 732 when there is a change in a presence status of a monitored link (e.g., link has become unavailable, or an unavailable link has become available). The update message 732 is received by a render module 703, within client 710, which is able to interpret the update message 732 and render or otherwise update the display of affected hyperlinks in order to reflect the status of the presence information that was supplied by web server 720. Link 733 denotes a conceptual relation between the hyperlink 704 and the resource it points to in 720. This method described herein does not make distinction between a fault in the web server, as opposed to a fault in the communication path to the web server. An objective of the method is to provide presence for a hyperlink. Since a hyperlink identifies a resource via a communication link or conceptual relation, its presence may be affected by both the state of the resource and the communication link. This message traffic is described below in greater detail.

[0129] The mark-up module 701 is based on a hyperlink-presence data model that describes the presence information for any hyperlink with the following properties:

[0130] 1) last-update: the last time when the presence was updated;

[0131] 2) last-status: the latest presence status;

[0132] 3) monitor: a URI that points to a resource by which a user agent can monitor the presence;

[0133] 4) status-list: list of presence status that the hyperlink can have; and

[0134] 5) target: the hyperlink to be monitored.

[0135] To be consistent with W3C Server-Sent Events and WebSocket protocols, a single URI should be used to monitor hyperlink presence. Therefore, the listener URI is not needed.

[0136] This hyperlink-presence data model can be represented in hypertext (HTML or XHTML) in several ways by introducing a new vocabulary (i.e., custom message attributes and the set of values for those attributes) for the hyperlink-presence data model. A presence update message can include presence status in the status-list (e.g. one of s1, s2 and s3) and any other status information, such as link failure, related to the monitored hyperlink.

[0137] In this data model, only the “target” and “monitor” properties are required and all others are optional. The values of “target” and “monitor” may be the same.

[0138] An embodiment in accordance with the present invention for transmitting the hyperlink-presence data model may represent the properties as attributes of a hyperlink element, as shown below in Example 1:

```html
&lt;a href="http://www.example.com/resource"
    last-update="2011-03-10T08:30:10"
    last-status="s2"
    monitor="http://www.example.com/monitor"
    status-list="s1 s2 s3">
    A resource
&lt;/a&gt;
```

Example 1

Representing Properties as Attributes of a Hyperlink Element

[0139] The value of each attribute in this embodiment becomes the value of the corresponding property and the href value of &lt;a&gt; becomes the value of target property.
Another embodiment for transmitting the hyperlink-presence data model is to use Hyper-Text Markup Language version 5 ("HTML 5") microdata, as illustrated below in Example 3 within the "<div>..." and "</div>" tag pair. Microdata is known as a customized name/value (or attribute/value) pair used to define a vocabulary for a specialized purpose.

Example 2

Representing Properties as HTML5 Microdata

Another embodiment is to use Resource Description Framework—in—attributes ("RDFa"), which is a set of XHTML attributes to augment visual data with machine-readable hints. RDFa adds a set of attribute-level extensions to XHTML for embedding rich metadata within Web documents. Example 3 below illustrates usage of RDFa to transmit the hyperlink-presence data model.

Example 3

Representing Properties as RDFa Data

Monitor module 702 is a presence watcher module in user agent 701, whose tasks include: 1) track hyperlink access; and 2) subscribes and receives presence events from a web server 720. In some cases, web server 720 may be a gateway to a separate presence server, such as a SIP or XMPP presence server.

FIG. 8 illustrates a system and process to track hyperlink access, in accordance with an embodiment of the present invention. Message 801 conveys the click action of a user clicking a hyperlink. Message 801 results in user agent 810 sending a request 802 to a web server 820 for web server 820 to send information, including presence information, related to the hyperlink clicked on by the user. Message 803 is a response is received by user agent 810 from web server 820. This information is sent to presence watcher module 811 by way of message 805, where presence watcher module 811 tracks the information. Presence watcher module 811 communicates with presence model module 812 by way of message 806, in order to find the corresponding presence model. Finding the corresponding presence model refers to a process of matching or correlating the presence information returned by web server 820 against predefined presence models stored in memory.

A presence watcher is responsible to monitor the presence of many hyperlinks. It does so by maintaining a presence model for each hyperlink that exposes presence.

The presence model is an abstract data structure of five properties describing the presence information of a hyperlink. This model can be represented using one of the three mark-up methods in examples described above.

Message 803 contains presence information, and is tracked by presence watcher module 811. Presence watcher module 811 communicates with presence model module 812 by way of message 807, in order to update the presence model based on the response.

FIG. 9 illustrates a system 900 and process to subscribe to the presence information of a hyperlink. Message 901 conveys the action of a user clicking on a hyperlink or otherwise indicating a hyperlink that is to be monitored for presence information. Message 901 results in presence watcher 911 communicating with presence model 913 by way of message interface 902 in order to extract a presence data model for the hyperlink to be monitored.

By way of a message on interface 904 running between presence watcher 911 and a monitor resource 921, presence watcher 911 establishes an HTTP connection to the monitor resource 921. Monitor resource 921 may be hosted by a web server 920. Presence watcher 911 also registers, by way of a message on interface 903, a listener function 912 to process presence events received from monitored resource 921 over connection 906 (described below in further detail).

Presence watcher 911 sends, by way of a message on interface 904, a HTTP request following the Server-Sent Event protocol to the monitor resource 921, in order to request receiving presence information identified by [filter] about {target}. An example of this message is:

```
GET /monitor?target={target}&filter={filter} HTTP/1.1
Host: www.example.com
Accept: text/event-stream
```

Monitor resource 921 processes the request received on interface 904. Monitor resource 921 provides an answer to the request by use of a message on interface 905 running between monitor resource 921 and presence watcher 911. Event stream 906 is then established by W3C Server-Sent Events protocol. The event stream may also be established by W3C WebSocket protocol. If necessary, web server 920 will subscribe to a presence server 930 in order to obtain presence information of target resource 931. Subscribing would be by way of sending a message on interface 924 running between web server 920 and presence server 930, and receiving a response from presence server 930 by way of a message on interface 925 running between presence server 930 and listener resource 922. If presence server 930 grants the request, a positive response message is sent by use of interface 905 running between listener resource 922 and presence watcher 911.

If and when the presence status of the target resource 931 changes, web server 920 sends presence events to the user agent 910 over interface 906.

Next, presence watcher 911 updates the presence model 913, by use of a message on interface 908 with the
events. Presence watcher 911 also invokes, by use of a message on interface 907, the registered listener module 912 to process the presence events. Finally, upon the user's request, the presence watcher 911 disconnects the event stream 906 in order to stop presence updates.

[0153] Render Module
[0154] An embodiment of render module 703 in accordance with the present invention may use JavaScript to render hyperlink presence in a web browser such that different presence status can be distinguished and presented to the user. It is left to the browser and web application implementation regarding how such hyperlink presence information is presented to the user, for instance BY A CHANGE IN font, foreground color, background color, boldness, italic, underline, bounding box, blink, popup, or the like. Another embodiment is to display a notification window (or icon, etc.) anchored at the hyperlink. Yet another embodiment is to play a distinct sound. Regardless of what type of display is used to indicate changes to the hyperlink presence information, the detailed presence information can be obtained and viewed by some user input, for example, by hovering over the hyperlink or clicking the notification window.

[0155] Transfer Module
[0156] FIG. 10 illustrates a transfer 1000 of hyperlink presence between a local user agent 1001 and a remote user agent 1002, in accordance with an embodiment of the present invention. The user agents 1001, 1002 use the format illustrated in FIG. 10 to export and import the presence model. In the transfer 1000, hyperlink presence is transferred between local user agent 1001 and remote user agent 1002 by encoding the last-update and last-status properties with the hyperlink to be transferred. XHTML template anchor elements are used to represent these properties, as shown below, in order to make sure that different user agents can interpret the properties. Transfer from local user agent 1001 to remote user agent 1002 may be a transfer 1003 via a clipboard (e.g., Windows clipboard or other operating system clipboard), or a transfer 1004 via the Internet. The choice of one transfer method over another depends on whether a first agent and a second agent are collocated in the same computer. If the first agent and the second agent are in the same computer with shared memory, such as a web browser and an email client running on a PC, then the shared memory (e.g., clipboard) can be used to transfer the hyperlink presence. If the two agents are running on different computers with network connections, such as two email clients on two different PCs, then a network transfer is used.

[0157] FIG. 10 describes how a snapshot of presence information about a hyperlink obtained in one user agent can be shared with another user agent, who may or may not monitor the presence as described in the previous figures. A typical transfer XHTML message is shown below.

```xml
<a href="xref URI">monitor=[monitor] last-update="xref dateTime" last-status="xref status">AnchorText</a>
```

[0158] While the foregoing is directed to embodiments of the present invention, other and further embodiments of the present invention may be devised without departing from the basic scope thereof. It is understood that various embodiments described herein may be utilized in combination with any other embodiment described, without departing from the scope contained herein. Further, the foregoing description is not intended to be exhaustive or to limit the present invention to the precise form disclosed. Modifications and variations are possible in light of the above teachings or may be acquired from practice of the present invention.

[0159] No element, act, or instruction used in the description of the present application should be construed as critical or essential to the invention unless explicitly described as such. Also, as used herein, the article "a" is intended to include one or more items. Where only one item is intended, the term "one" or similar language is used. Further, the terms any or followed by a listing of a plurality of items and/or a plurality of categories of items, as used herein, are intended to include "any of," "any combination of," "any multiple of," and/or "any combination of multiples of" the items and/or the categories of items, individually or in conjunction with other items and/or other categories of items.

[0160] Moreover, the claims should not be read as limited to the described order or elements unless stated to that effect. In addition, use of the term "means" in any claim is intended to invoke 35 U.S.C. §112, ¶ 6, and any claim without the word "means" is not so intended.

What is claimed is:

1. A method to monitor and transfer hyperlink presence information about a hyperlinked resource, comprising: transmitting a hyperlink presence monitor request to a web server;
   receiving an hyperlink presence information; and
   rendering a hyperlink based upon the hyperlink presence information.

2. The method of claim 1, wherein the hyperlink presence information comprises an availability of a hyperlink.

3. The method of claim 1, wherein the hyperlink presence information comprises a change in an availability of a hyperlink.

4. The method of claim 1, wherein the hyperlink presence information comprises an availability of a hyperlink, the method further comprising: receiving a change in the availability of the hyperlink.

5. The method of claim 1, further comprising: transmitting the hyperlink presence information to a user agent.

6. The method of claim 1, wherein the hyperlink presence information comprises:
   a last-update information to indicate a last time when the hyperlink presence was updated;
   a last-status information to indicate a latest hyperlink presence status;
   a monitor information to indicate a URI that points to a resource by which a user agent can monitor the hyperlink presence;
   a status-list information to indicate a list of hyperlink presence status that the hyperlink can have; and
   a target information to indicate the hyperlink to be monitored.

7. The method of claim 1, wherein the hyperlink presence information is represented as attributes of a hyperlink element.

8. The method of claim 1, wherein the hyperlink presence information is represented as HTML5 microdata.

9. The method of claim 1, wherein the hyperlink presence information is represented as RDFa data.

10. The method of claim 1, wherein the hyperlink presence information is represented as XHTML message data.
11. The method of claim 1, wherein the hyperlink presence information is received from a presence server.
12. The method of claim 1, further comprising: matching the hyperlink presence information to a presence model.
13. The method of claim 1, wherein rendering a hyperlink based upon hyperlink presence information further comprises playing a sound based upon the hyperlink presence information.
14. A system to monitor and transfer hyperlink presence information about a hyperlinked resource, comprising:
   a monitor module configured to transmit a hyperlink presence monitor request to a web server;
   a receiver configured to receive an hyperlink presence information; and
   a rendering module configured to render a hyperlink based upon the hyperlink presence information.
15. The system of claim 14, wherein the hyperlink presence information comprises one of an availability of a hyperlink and a change in an availability of a hyperlink.
16. The system of claim 14, wherein the hyperlink presence information comprises an availability of a hyperlink, the method further comprising: receiving a change in the availability of the hyperlink.
17. The system of claim 14, further comprising: transmitting the hyperlink presence information to a user agent.
18. The system of claim 14, wherein the hyperlink presence information comprises:
   a last-update information to indicate a last time when the hyperlink presence was updated;
   a last-status information to indicate a latest hyperlink presence status;
   a monitor information to indicate a URI that points to a resource by which a user agent can monitor the hyperlink presence;
   a status-list information to indicate a list of hyperlink presence status that the hyperlink can have; and
   a target information to indicate the hyperlink to be monitored.
19. The system of claim 14, wherein the hyperlink presence information is represented as one of: attributes of a hyperlink element; HTML5 microdata; RDFa data; and XHTML message data.
20. The system of claim 14, wherein the hyperlink presence information is received from a presence server that is separate from a server hosting the hyperlinked resource.