EXHIBIT AA

Claim Chart

(*Ikeda, Tsutsumitake, Bird, and Pearson*)

Exhibit AA – Claim Chart for U.S. Patent No. 8,407,722

I. GROUNDS OF UNPATENTABILITY

Ground	Claim(s)	Statute(s)	Prior Art
1	1, 6, 7, 14-17, 20-23, 26-29, and 32-35	103	*Ikeda*, *Tsutsumitake*, and *Bird*
2	1, 6, 7, 14-17, 20-23, 26-29, and 32-35	103	*Ikeda*, *Tsutsumitake*, and *Pearson*

A. Prior Art Relied Upon

Primary References
Ex. 1004 ("Ikeda"): Ikeda (U.S. Patent Publication No. 6,999,991) was filed on July 24, 2000 and issued on February 14, 2006. Ikeda claims priority to Japanese Patent Application JP11-310254, which was filed on October 29, 1999. Ikeda is prior art at least under 35 U.S.C. § 102(e).
Ex. 1005 ("Tsutsumitake"): Tsutsumitake (U.S. Patent No. 6,480,883) was filed on June 29, 1999 and issued on November 12, 2002. Tsutsumitake claims priority to Japanese Patent Application JP10-199587, which was filed on June 30, 1998. Tsutsumitake is prior art at least under 35 U.S.C. § 102(e).
Ex. 1006 ("Bird"): Bird (European Patent Application EP1043671) was filed on March 17, 2000 and published on October 11, 2000. Bird claims priority to Great Britain Application 9906321, which was filed on March 19, 1999 and Great Britain Application 9906321, which was filed on June 4, 1999. Bird is prior art at least under 35 U.S.C. § 102(a) and (e).
Ex. 1007 ("Pearson"): Pearson (U.S. Patent No. 6,990,591) was filed on December 22, 1999 and issued on January 24, 2006. Pearson claims priority to U.S. Provisional Application 60/166,272, which was filed on November 18, 1999. Pearson is prior art at least under 35 U.S.C. § 102(e).

B. Claim Chart

1. Independent Claim 14 and Dependent Claims 15-17

Requester notes that the first claim shown below is claim 14, rather than claim 1. Requester has started with claim 14 because it is the narrowest method claim, which allowed Requester to show disclosure in the prior art of the majority of the limitations found in the other method claims.

Claim 14	
14[P]. A method comprising:	Whether or not the preamble is limiting, Ikeda teaches, or at least renders obvious, *a method* (e.g., push service processing method). Ikeda discloses a push service processing method. *Ikeda* (Ex. 1004), 1:6-8.

	See, e.g., Ikeda The present invention relates to a push service system and a **push service processing method** for processing push service in a network using IP (Internet protocol). Ikeda at 1:6-8. It is an object of the present invention to provide a push service system and a **push service processing method** in which resources in a network are utilized efficiently without increasing traffic on the network, and information required by users can be obtained quickly and expandability is high. Ikeda at 1:58-62. The present invention relates to a push service system and a **push service processing method** for processing push service in a network using IP (Internet protocol). Ikeda at [Abstract].
[14.1(a)] providing, using a processing device of an input source, a data representation to a client device, different from the input source, coupled to a routing network, wherein the data representation includes at least one live object recognizable by the client device, and	Ikeda in view of Tsutsumitake renders obvious *providing, using a processing device of an input source* (e.g., processing section of data servers), *a data representation* (e.g., providing a web page) *to a client device, different from the input source* (e.g., user terminals), *coupled to a routing network* (e.g., Internet), *wherein the data representation includes at least one live object recognizable by the client device* (e.g., web page includes an updateable element that is capable of being displayed by the client and/or recognizable by the client as being "updateable"). Ikeda describes a push system that includes data servers which contain information that is accessed by user terminals (e.g., client devices) via the Internet (e.g., a routing network). *Ikeda* (Ex. 1004), 19:26-28, 19:48-52, 20:21-23, Fig. 15. Ikeda teaches that the data servers include processing sections (e.g., processing devices). *Ikeda* (Ex. 1004), 19:67-20:20. A POSITA would have understood that these data servers, which are computers, would have used processing sections to carry out operations, including providing data and information to client devices. *Shamos Decl.* (Ex. 1003), ¶¶ 48, 23-25 (citing *Bowman-Amuah*, *Brendel*, *Tsutsumitake*, *Hassett*, *Todd*). And this is supported by Tsutsumitake. *Tsutsumitake* (Ex. 1005), 5:36-38, 6:16-18. Thus, Ikeda teaches that data servers (*input sources*), using a processing section (e.g., *processing device*), provide information to user terminals that is different from the data servers (e.g., *client devices*), coupled to the Internet (e.g., *a routing network*). However, Ikeda does not provide express examples of where or when such a push system would have been used.

	A POSITA would have looked to other references describing push systems to determine where / when to implement Ikeda's push system. *Shamos Decl.* (Ex. 1003), ¶¶ 49-60. Tsutsumitake is one such reference. Tsutsumitake describes a push system that operates to update an element (e.g., a live object) of a web page. *Tsutsumitake* (Ex. 1005), 1:7-12, 3:20-26, [Abstract], 10:10-15. Specifically, Tsutsumitake teaches that information providing servers provide information to clients in the form of web pages. *Id.* at 1:22-28. Tsutsumitake's clients receive these web pages and display them to users. *Id.* Thus, Tsutsumitake describes a content providing server (e.g., an input source) using a CPU (e.g., a processing device of the input source) to provide a web page (e.g., a data representation) to a client that is different than the content providing server (e.g., a client device). A POSITA would have understood that Tsutsumitake's web pages include live objects. *Shamos Decl.* (Ex. 1003), ¶¶ 51-52. The '722 Patent teaches that "an 'object' is any datum or data at the client 114 that can be individually identified or accessed," such as "elements of web pages such as text characters and strings, images, frames, tables, audio, video, applets, scripts, HTML, XML, and other code forming the web page, variables and other information used by applets, scripts and/or code, URLs embedded in the web page, etc." *'722 Patent* (Ex. 1001), 6:66-7:6. The '722 Patent teaches that the "[p]roperties of a live object can be dynamically updated in real-time at the client 114." *'722 Patent* (Ex. 1001), 6:63-66. Tsutsumitake discloses that portions of a webpage (i.e., a live object) may be updated dynamically when the client receives an event from the server. *Tsutsumitake* (Ex. 1005), [Abstract]. In more detail, Tsutsumitake teaches that an object of the invention is to provide a real-time information transmission system that allows information updated on the server side to be efficiently reflected on the client side. *Tsutsumitake* (Ex. 1005),3:27-33. Tsutsumitake provides one example of a web page that monitors the state of current in a specific control device in a plant system. *Tsutsumitake* (Ex. 1005), 13:31-34, Fig. 7. The page displays a current value display section and a state display section. *Id.* at 13:34-37, Fig. 7. An HTML expression of the page shows that the values in both sections are individually identified in the HTML and are individually accessed for updating because they are updated based on different rules. *Id.* at 14:4-15, Fig. 8, 13:38-55, 13:58-65. Both of these sections are updated by the server in real time. *Tsutsumitake* (Ex. 1005), 13:56-57. Thus, a POSITA would have understood these two values to be elements (e.g., objects) of a web page. *Shamos Decl.* (Ex. 1003), ¶ 52. The elements on Tsutsumitake's web pages are "live" because they can be individually

	accessed, are individually identified, and are updated in real-time at the client by the information provider. *Id.* Accordingly, Tsutsumitake further teaches that the webpage (e.g., data representation) includes at least one updateable element (e.g., live object). Tsutsumitake's updateable elements (e.g., live objects) are recognizable by the client device. Tsutsumitake teaches that the client processes and displays the web pages to a user, including the elements of the web page. *Tsutsumitake* (Ex. 1005), 13:27-45, Fig. 7. Thus, because the client device is capable of displaying the web page, including the web page elements (i.e., live objects) to the user, the client device recognizes these elements. Additionally, or alternatively, to the extent "recognizable by the client device" means that the client device is capable of recognizing that an element or object is "live," Tsutsumitake also teaches this. Tsutsumitake teaches that the web page include "event requests," which are associated with elements of a web page. *Id.* at 13:38-55, 14:4-16. When a web page includes an event request, then Tsutsumitake's client provides the event request to the event request unit 113. *Id.* at 9:60-67. Thus, Tsutsumitake's client recognizes that a webpage includes an updateable element when it detects an event request on a web page. *Shamos Decl.* (Ex. 1003), ¶ 53. In the proposed Ikeda-Tsutsumitake combination, Ikeda's push system would have been implemented in the context of providing and updating web pages (which is an application taught by Tsutsumitake). *Shamos Decl.* (Ex. 1003), ¶ 54. In the proposed combination, Ikeda's data servers would have stored data relating to web pages and data relating to changes to the web pages. *Id.* Ikeda's user terminals would have requested a web page from the data servers, just as Tsutsumitake's clients requested web pages from information providing servers. *Id.* Ikeda's data servers would have provided these web pages to the clients (using the CPU of the server) and, upon receipt, the client would have displayed the web page, including updateable elements of a web page (i.e., live objects), to a user. *Id.* The display of the web page and elements of the web page, as well as the ability of the client to process update events from the server destined for the object, demonstrate that the elements are recognizable as live objects by the client device. *Id.* That is, as implemented with Tsutsumitake's teachings, Ikeda's client would have recognized the inclusion of an "event request" in a web page that indicates the presence of an updateable element. *Id.* Just as Ikeda's user terminals registered for updates to data held by servers, in the proposed combination, Ikeda's user terminals would have registered for updates to web pages. *Id.*; *see also* claim [14.1(b)], *infra* (discussing registration in detail). Similarly, just as Ikeda's push system provided information about updating to user terminals through agents, in the proposed combination Ikeda's push system would have provided updates to elements of the web pages. *Shamos Decl.* (Ex. 1003), ¶ 55; *see also* claim [14.1(b)], *infra* (discussing sending update

messages in detail). And just, as Ikeda's user terminals obtain the contents of the updated data according to Ikeda's teachings, in the proposed combination, Ikeda's user terminals would have processed the received information to update the web page elements. *Shamos Decl.* (Ex. 1003), ¶ 55; *see also* claim [14.5], *infra* (discussing updating properties of live objects).

A POSITA would have modified Ikeda with Tsutsumitake for the reasons set forth in the Request. *See* Request, Section I.E.2.a.; *Shamos Decl.* (Ex. 1003), ¶¶ 46-60, 23-25 (citing *Bowman-Amuah*, *Brendel*, *Tsutsumitake*, *Hassett*, *Todd*).

See, e.g., Ikeda

> **FIG. 4B shows data which are used when the user terminals access to the data servers for the request to obtain the contents of the updated information**. The user terminals 5 to 7 hold a data server address D21 represented by IP or machine name and domain name, data type information D22 about ID or updated information which is the contents of data, and an access key D23 to data type such as ID, password, credit card No. and the like. The data server address D21 and the data type information D22 are essential information.

Ikeda at 9:8-17.

> According to one aspect of this invention, a plurality of user terminals transmit information of the user terminals and requested various information to some of not less than one agents so as to execute the registration process. **A plurality of data servers have various information** and post information about updating of various information to the agents. When the posted information about updating is the information about updating which is requested by the registered user terminals, the not less than one agents post the information about updating to the registered user terminals. The user terminals which have received the post access directly to the data servers which hold contents of the updated information via the network based on the posted information about updating, and obtain the contents of the updated information.

Ikeda at 2:11-26.

> Most of the conventional push services are realized by automatically accessing from a user terminal to a predetermined type of information per constant time. That is, **a function for accessing to a data source including the predetermined information per constant time is provided to the user terminal**,

and update information of the data source is obtained from the accessed results.

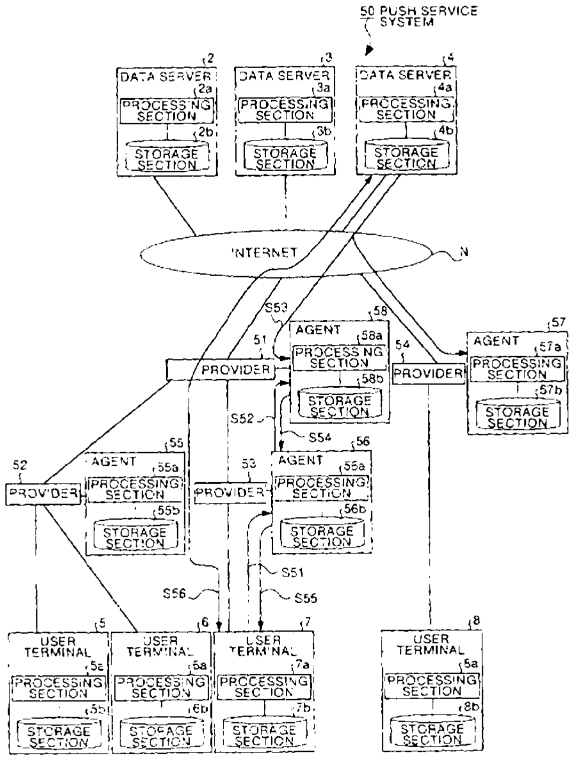
In another push services, user terminals are registered in a data source including various data, and when the data source is updated, **the data source directly provides the updated data to the respective user terminals.** In this case, the data source itself has the function for providing distribution service.

Ikeda at 1:24-36.

FIG. 15 is a diagram showing the structure of the push service system according to the seventh embodiment of the present invention. As shown in FIG. 15, in the push service system 50, **a plurality of data servers 2 to 4 and a plurality of providers 51 to 54 are connected to the Internet N**. The providers 51 to 54 locally connect the agents 55 to 58.

Ikeda at 19:24-29.

FIG.15



Ikeda at Fig. 15.

Further, when the information about updating is information about updating which is requested by the registered user terminal 7, the agent 56 posts the information about updating to the user terminal

	7 (S55). When the user terminal 7 receives the information about updating, the user terminal 7 accesses directly to the data servers 2 to 4 which have posted the information about updating via the Internet N so as to obtain contents of the updated information (S56). As a result, a series of the push service process is executed. The agents 55 to 58 have processing sections 55 a to 58 a and storage sections 55 b to 58 b respectively. **The data servers 2 to 4 have <u>processing sections 2 a to 4 a</u> and storage sections 2 b and 4 b respectively**. The user terminals 5 to 8 have processing sections 5 a to 8 a and storage sections 5 b and 8 b respectively. Ikeda at 19:57-20:4. ***<u>See, e.g., Tsutsumitake</u>*** The present invention relates generally to a **client/server type system wherein an information providing server provides information upon request from an information display client,** and more particularly to a real-time information transmission system for realizing novel and advantageous information transmission from an information providing server to an information display client. In these years, the WWW (World Wide Web) has permeated through societies, and various information is provided via the WWW. **The WWW is a distributed-type information-provision/information-display system using the Internet, and it comprises servers (information providing servers) and clients.** As is well known, a worldwide information network using hypertexts including images and sounds has been constructed by the WWW and utilized by many people. **In general, the server provides information to the client in units of a document called "page".** The page is an electronic file described in a format called HTML (Hypertext Markup Language). The client is normally realized with use of a software called "browser" for information display (information browsing), and the browser accesses the server to acquire and display pages including images and sounds. In the general use of the WWW, an address (called URL) of a page to be displayed is designated in the browser, and the browser issues a request to the server. The server returns the page corresponding to the address to the browser. **It should be noted here that the server performs nothing unless it receives a request from the browser. This means that even if pages stored**

> **in the server have been updated, the browser is unable to immediately know the change.**
>
> **For example, while the browser is displaying pages such as news and weather reports, which are updated in real time, the browser is not informed of the fact that the pages have been updated on the server side**. That is, the content of the updated page is not reflected on the display screen of the browser side.

Tsutsumitake at 1:7-43; *see also id.* at 7:66-8-6, 8:19-23, 9:42-67 (discussing the environment in greater detail).

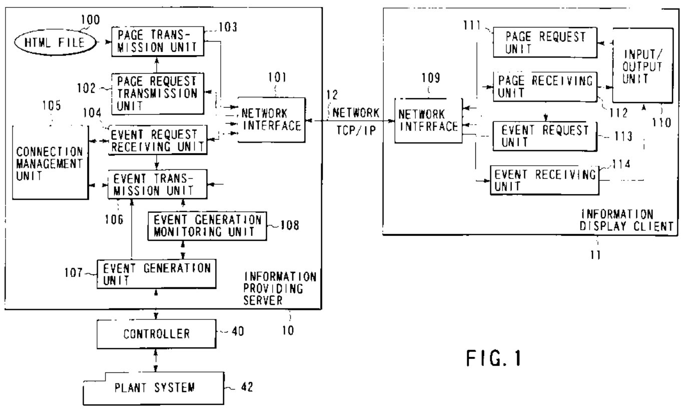
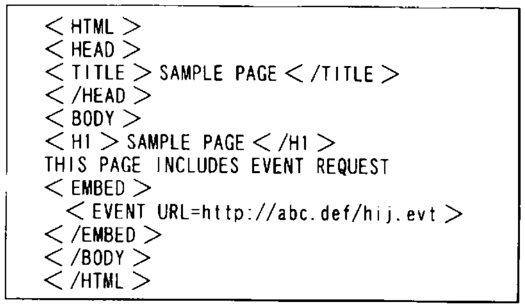


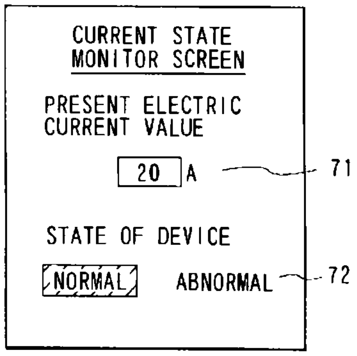
FIG. 1

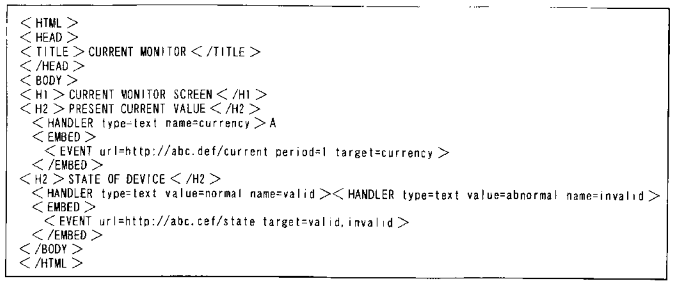
Tsutsumitake at Fig. 1.

> In the present embodiment, the **format of the page sent from the server 10 in response to the page request from the client 11 is a normal HTML file format**. Of course, another format may be adopted.
>
> FIG. 2 shows an example of the HTML file format. In this example, an event request to the server is included in the page. **The event request is expressed by an attribute "URL" in a tag "EVENT" in a tag "EMBED". The format of expression of the event request is not limited to this. It may be assumed, for example, that the event request is not positively described in the HTML file but is expressed as a link to another page, and the event request is described in the page of the destination link and is sent to the server 10 from the HTML file at the originating point.**

Tsutsumitake at 10:16-29.

	 ``` <HTML> <HEAD> <TITLE> SAMPLE PAGE </TITLE> </HEAD> <BODY> <H1> SAMPLE PAGE </H1> THIS PAGE INCLUDES EVENT REQUEST <EMBED> <EVENT URL=http://abc.def/hij.evt> </EMBED> </BODY> </HTML> ``` FIG. 2 Tsutsumitake at Fig. 2. **A page request receiving unit 102 receives a page request sent from the client 11 via the network interface 101, analyzes it, and informs a page transmission unit 103 of a page to be sent to the client 11.** Suppose that a format called URL (Uniform Resource Locator), which is generally used in the WWW, is applied to the page request. **The page transmission unit 103 transmits an HTML file 100 stored in the server 10 to the network 12 via the network interface 101**. Information to be transmitted is not limited to the HTML file (100), and may be freely chosen one such as image or voice. Tsutsumitake at 8:39-49. FIG. 7 shows an example of the screen image of the browser. **This example relates to a page for monitoring the state of current in a specific control device in the plant system 42. A page comprising a current value display section 71 indicating a value of electric current at present and a state display section 72 indicating the state of a device (control device) is displayed by the browser**. An event request is described on this page. If the page is displayed by the browser, the event request described on the page is sent to the server 10 which provides this page. In this example, **the event requested on this page includes an event indicating a present value of an electric current, which is sent from the server 10 at intervals of one second, and an event indicating the state (normal/abnormal) of the device, which is sent from the server 10 asynchronously.**

	The current value in the specific device in the plant system 42 is monitored by the server 10 every second through the controller 40. Specifically, the event generating unit 107 or event generation monitoring unit 108 generates the event of the present value of electric current at intervals of one second. In addition, the state (normal/abnormal) of the control device is monitored by the server 10 through the controller 40. The event generating unit 107 or event generation monitoring unit 108 generates events of the state (normal/abnormal) of the control device non-periodically. **These generated events are transmitted in real time to the client 11.** **The browser (client 11) processes a plurality of events** (both the event indicating the present value of current and the event indicating the state of the device being transmitted in real time) sent from the server 10 in response to one event request issued in accordance with the displayed page, and reflects on the screen the event processing results on the current value display section 71 or device state display section 72. **The event processing results can be reflected on the page screen image (event screen image)** by using the Java (Applet) of Sun Microsystems, JavaScript of Netscape Communications, or Dynamic HTML of Microsoft. Needless to say, the method is not limited to these. Tsutsumitake at 13:31-14:3.  CURRENT STATE MONITOR SCREEN PRESENT ELECTRIC CURRENT VALUE 20 A 71 STATE OF DEVICE NORMAL ABNORMAL 72 FIG. 7 Tsutsumitake at Fig. 7. FIG. 8 shows an HTML expression of a page, on which the screen display in FIG. 7 is based. **In the example of FIG. 8, the event request to the server 10 is designated by a tag “EVENT” on the page. This tag includes attributes such as a time interval (“period”) of events to be sent to the client 11 and a target**

	("target") for reflecting the received event on the screen. A tag "HANDLER" is introduced as a tag for reflecting the event on the screen. This tag includes attributes such as a format ("type") for screen display and a name ("name"). The same name as the name attribute is designated in the target attribute in the tag "EVENT". This example of expression is merely an example, and other formats may be used.
	
	According to the browser screen in FIG. 7 of the electric current state monitoring page in the client 11, **the user can monitor the present electric current value of the specific control device of the plant system at intervals of one second, without performing a display update operation, for example, by depressing a reload button. In addition, the user can monitor the change in state (normal/abnormal) of the control device.** As has been described above, since the server continuously returns responses (generated events) to a single event request from the client 11 while the connection is being maintained, the information transmission can be efficiently carried out without increasing a communication load between the client 11 and server 10.
	Tsutsumitake at 14:4-28.
	
	
	`< HTML >`
	`< HEAD >`
	`< TITLE > CURRENT MONITOR < /TITLE >`
	`< /HEAD >`
	`< BODY >`
	`< H1 > CURRENT MONITOR SCREEN < /H1 >`
	`< H2 > PRESENT CURRENT VALUE < /H2 >`
	`  < HANDLER type-text name=currency > A`
	`  < EMBED >`
	`    < EVENT url=http://abc.def/current period=1 target=currency >`
	`  < /EMBED >`
	`< H2 > STATE OF DEVICE < /H2 >`
	`  < HANDLER type=text value=normal name=valid >< HANDLER type=text value=abnormal name=invalid >`
	`  < EMBED >`
	`    < EVENT url=http://abc.cef/state target=valid, invalid >`
	`  < /EMBED >`
	`< /BODY >`
	`< /HTML >`
	
	FIG. 8
	Tsutsumitake at Fig. 8.

[14.1(b)] causing the client device to respond to the live object of the data representation by determining an object identifier (ID) of the live object and to register for updates of the live object with the routing network, such that	Ikeda in view of Tsutsumitake renders obvious *causing the client device* (e.g., user terminals) *to respond to the live object of the data representation* (e.g., web page including an updateable element) *by determining an object identifier (ID) of the live object* (e.g., Event ID or event request URL) *and to register for updates of the live object with the routing network* (e.g., submit a registration request), *such that registering the client device with the routing network provides client connection information* (e.g., address of the agent, and "terminal information D15 such as equipment type and the like, and another terminal information D16 held by the users such as equipment type, telephone number," and address) *to a node* (e.g., agent) *in the routing network* (e.g., coupled to the Internet).

registering the client device with the routing network provides client connection information to a node in the routing network; and	Ikeda teaches that user terminals transmit registration requests to a low level agent of the routing network. *Ikeda* (Ex. 1004), 19:40-44, 20:10-14. The registration requests include a type of data for which the terminal is registering. *Id.* at 19:40-44. Additionally, the registration request includes the address of the agent, and "terminal information D15 such as equipment type and the like, and another terminal information D16 held by the users such as equipment type, telephone number," and may include an address of the user terminal. *Id.* at 19:40-44, 8:58-9:1. Such information is connection information because it indicates to which agent the client device is connected and an address to which information for the client may be sent. Accordingly, Ikeda teaches that a client device registers for updates (e.g., registers for updates) of information from a data server, which includes providing client connection information (e.g., address of the agent, and "terminal information D15 such as equipment type and the like, and another terminal information D16 held by the users such as equipment type, telephone number," and address) to an agent (e.g., node) in the routing network (such that registering the client device with the routing network provides client information to an agent in the routing network). Ikeda does not expressly teach that the registration request is sent in response to receiving the live object of the data representation and does not expressly teach that the client determines an object ID of the live object. However, Tsutsumitake teaches that, in response to receiving a web page including an updateable element from a server, a client computer determines an object ID and also registers for updates to the updateable element. *Tsutsumitake* (Ex. 1005), 9:60-10:6, 11:43-55. In more detail, Tsutsumitake teaches that when a webpage includes updateable elements, the web page includes an "event request." *Id.* at 10:20-29. When an event request is included in the page, the client recognizes the event and submits the event request, which at least includes a URL. *Id.* at 9:60-10:6. Thus, Tsutsumitake teaches causing a client device (e.g., client) to respond to a live object of the data representation(e.g., updateable element of a web page) by registering for updates. Tsutsumitake further teaches that the client determines an object ID in response to receiving and recognizing an event request associated with an updateable element. First, Tsutsumitake teaches that a registration request from a client includes an event ID and also a connection ID. *Id.* at 11:43-55, Abstract, 12:64-13:5. A POSITA would have understood that the event ID is determined by the client device to allow the client device to register. *Shamos Decl.* (Ex. 1003), ¶¶ 62-63. Thus, Tsutsumitake's event ID at least renders obvious an object identifier as claimed. *Id.* at ¶ 63. Second, Tsutsumitake teaches that event requests are "expressed by an attribute

	'URL' in a tag 'EVENT' in a tag 'EMBED.'" *Tsutsumitake* (Ex. 1005), 10:20-29. A POSITA would have recognized that such a URL is an identifier for an updateable element responsive to an event. *Shamos Decl.* (Ex. 1003), ¶ 63. Because Tsutsumitake teaches that the URL is then used for registration (*Tsutsumitake* (Ex. 1005), 9:60-10:6), a POSITA would have understood that this URL is an object identifier. *Id.* at ¶ 63. For example, the URL would have served as the event ID. *Id.* Modified in light of Tsutsumitake's teachings, Ikeda's client, upon receiving a web page from a server, would have recognized that the web page included an event request. *Id.* at ¶ 64. Upon recognizing the event request, Ikeda's client would have submitted a registration request to the routing network. *Id.* The client would have determined an event ID of the event request (e.g., the URL) and included this in the registration request, along with client connection information. *Id.* The registration request submitted in the Ikeda-Tsutsumitake system would have included the same types of information included in Ikeda's registration request (i.e., a data type and client connection information), as well as Tsutsumitake's event ID. *Id.* A POSITA would have made the proposed modification for the reasons set forth in the request. *See* Request, Section I.E.2.b.; *Shamos Decl.* (Ex. 1003), ¶¶ 61-68, 23-25 (citing *Bowman-Amuah*, *Brendel, Tsutsumitake, Hassett*, *Todd*) 26 (citing *Schwartz*). ***<u>See, e.g., Ikeda</u>*** As for a summary of a series of the push service process, for example **the user terminal 7 registered information such as an address of the user terminal 7 and information such as a data type to be requested into the agent 56 of the provider 53 (S51)**. The agent 56 serves as the user terminal for the high hierarchical agent 58. The agent 56 registers the user terminal 7 and requests the high-hierarchical agent 58 of the high-hierarchical provider 51 to register the user terminal 7 so as to register (S52). When various information which is held and managed by the data servers 2 to 4 is updated, the server 2 to 4 IP-multi-scan information about updating to the providers 51 and 54 so as to transmit the information about updating to the agents 58 and 57 (S53). When the information about updating is information about updating which is managed by the registered low-hierarchical agent, the agent 58 posts the information about updating to the agent 56 (S54). Ikeda at 19:40-56.

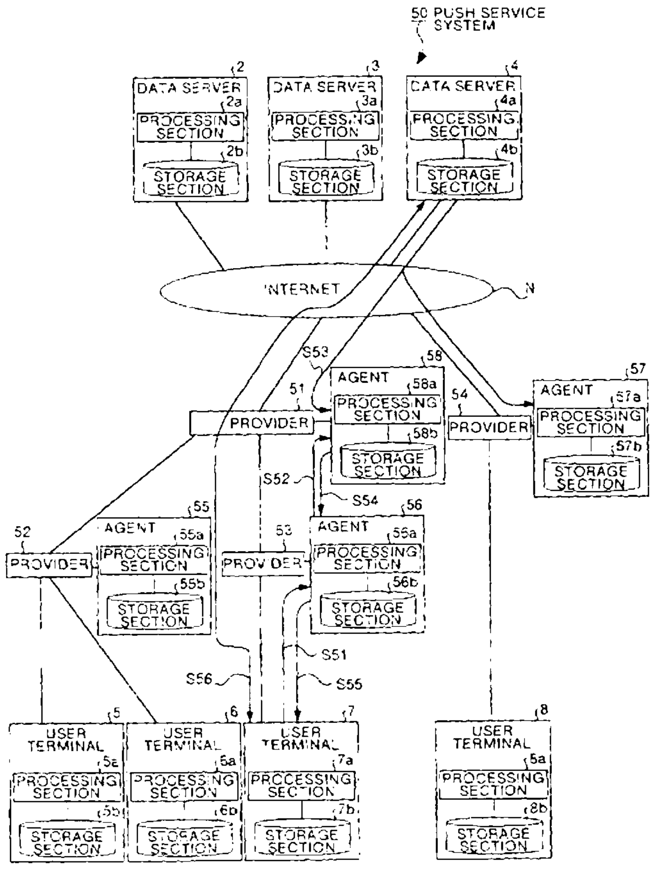
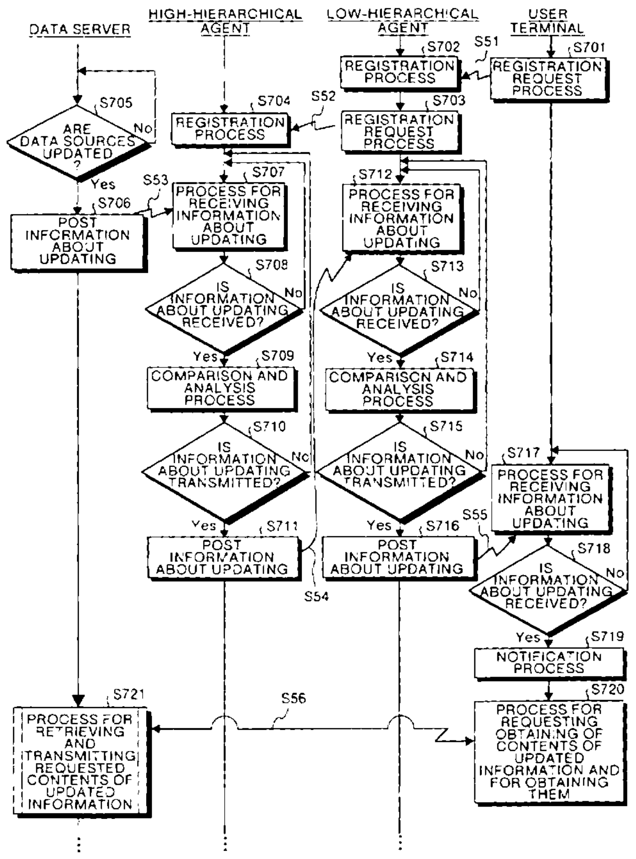
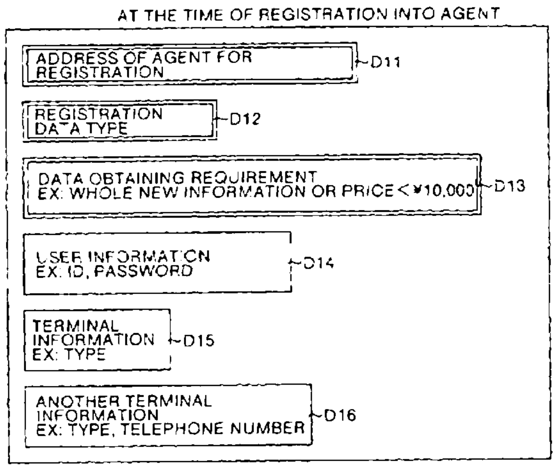
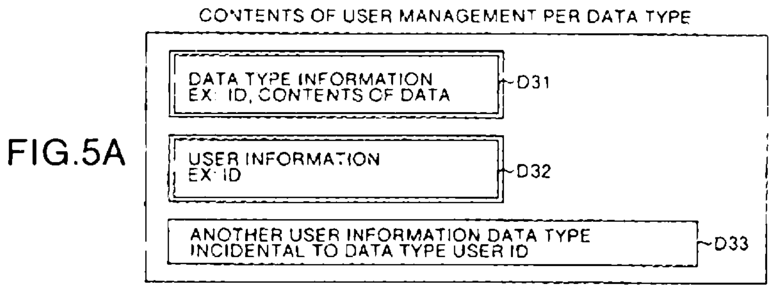
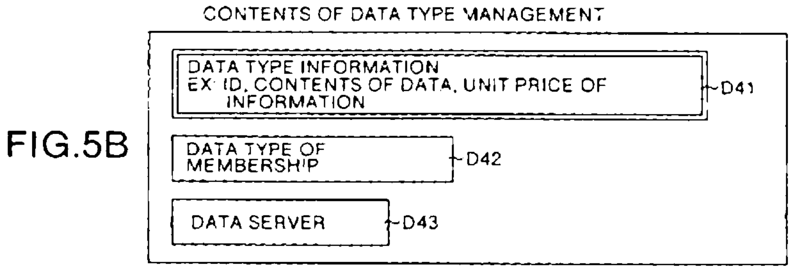
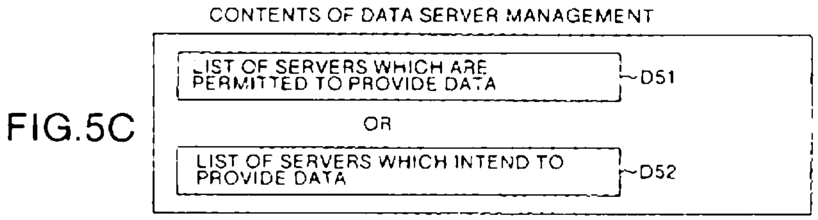
	FIG.15  50 PUSH SERVICE SYSTEM DATA SERVER PROCESSING SECTION STORAGE SECTION INTERNET N AGENT PROVIDER USER TERMINAL S51 S52 S53 S54 S55 S56 Ikeda at Fig. 15. A series of the push service process according to the seventh embodiment will be explained below with reference to the flow chart shown in FIG. 16. FIG. 16 is the flow chart showing the push service processing procedure performed by the push service system according to the seventh embodiment of the present invention. **As shown in FIG. 16, for example user terminals 5 to 7 request the low- hierarchical agents 55 and 56 of the providers 52 and 53 connected to the user terminals 5 to 7 to register the user terminals 5 to 7 (steps S701 and S51).** The low-hierarchical agents 55 and 56 which receive the registration request execute the registration process (step S702), and request the high-hierarchical agent 58 to register the registered contents (steps S703 and S52). On the contrary, the high-hierarchical agent 58 registers the low- hierarchical agents 55 and 56 which have been requested as to their registration (step S704). Ikeda at 20:5-20.

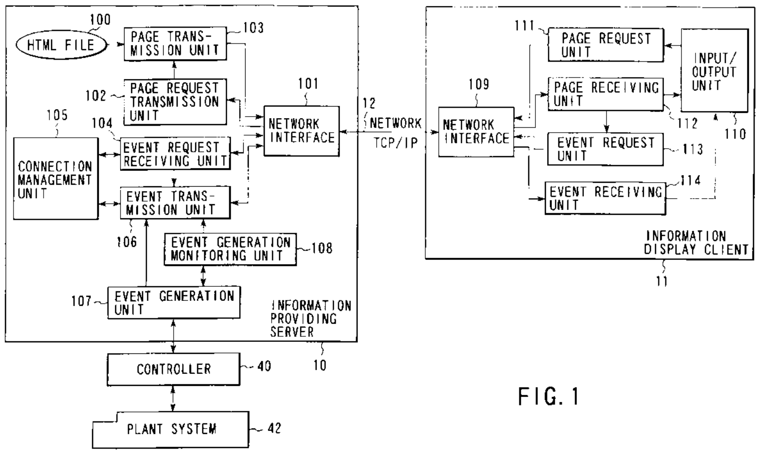
	FIG.16  DATA SERVER HIGH-HIERARCHICAL AGENT LOW-HIERARCHICAL AGENT USER TERMINAL REGISTRATION PROCESS REGISTRATION REQUEST PROCESS ARE DATA SOURCES UPDATED? POST INFORMATION ABOUT UPDATING PROCESS FOR RECEIVING INFORMATION ABOUT UPDATING IS INFORMATION ABOUT UPDATING RECEIVED? COMPARISON AND ANALYSIS PROCESS IS INFORMATION ABOUT UPDATING TRANSMITTED? NOTIFICATION PROCESS PROCESS FOR REQUESTING OBTAINING OF CONTENTS OF UPDATED INFORMATION AND FOR OBTAINING THEM PROCESS FOR RETRIEVING AND TRANSMITTING REQUESTED CONTENTS OF UPDATED INFORMATION Yes No S51 S52 S53 S54 S55 S56 S701 S702 S703 S704 S705 S706 S707 S708 S709 S710 S711 S712 S713 S714 S715 S716 S717 S718 S719 S720 S721 Ikeda at Fig. 16. Information about the push service process which is managed by the user terminals 5 to 7, the agent 1 and the data servers 2 to 4 will be explained below with reference to FIG. 4 to FIG. 6. FIG. 4A and FIG. 4B are diagrams showing data management structures in the user terminals. **When the user terminals 5 to 7 request the agent 1 to register the user terminals, data, for example, shown in FIG. 4A are required.** Ikeda at 8:51-57.

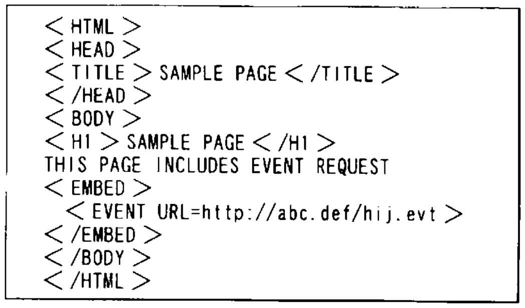
	 AT THE TIME OF REGISTRATION INTO AGENT ADDRESS OF AGENT FOR REGISTRATION D11 REGISTRATION DATA TYPE D12 DATA OBTAINING REQUIREMENT EX: WHOLE NEW INFORMATION OR PRICE<¥10,000 D13 USER INFORMATION EX: ID, PASSWORD D14 TERMINAL INFORMATION EX: TYPE D15 ANOTHER TERMINAL INFORMATION EX: TYPE, TELEPHONE NUMBER D16 FIG.4A Ikeda at Fig. 4A. FIG. 5A to FIG. 5C are diagrams showing the data management structures in the agent 1. As shown in FIG. 5A to FIG. 5C, the agent 1 manages the user terminals 5 to 7 and the data server 2 to 4 according to the data type. As shown in FIG. 5A, the agent 1 holds user management contents of each data type, the agent 1 manages a relationship between the data types and the user terminals 5 to 7. **The user management contents include a data type information D31 such as ID and obtained data contents, user information D32 such as user ID, and another user information D33 incidental to data types and user IDs.** The data type information D31 and the user information D32 are essential information which should be managed by the agent 1. **The ID in the data type information D31 is represented by a serial number, a hierarchical code and a number from new data type, and the data types can be managed hierarchically**. The data contents may be managed by using XML description so that the following hierarchical link relationships: <Group1 name=news> <Group2 name=sports> . . . are established. Ikeda at 9:18-39.

	 CONTENTS OF USER MANAGEMENT PER DATA TYPE FIG.5A DATA TYPE INFORMATION EX: ID, CONTENTS OF DATA D31 USER INFORMATION EX: ID D32 ANOTHER USER INFORMATION DATA TYPE INCIDENTAL TO DATA TYPE USER ID D33 Ikeda at Fig. 5A. FIG. 5B shows the data type management contents by means of the agent 1. A relationship between the data types and the data servers 2 to 4 is managed by the management contents. **The data type management contents include data type information D41 such as ID, data contents and information price, membership data type D42, data server D43 which holds the data type. The data type information D41 is essential information.** Ikeda at 9:40-47.  CONTENTS OF DATA TYPE MANAGEMENT FIG.5B DATA TYPE INFORMATION EX: ID, CONTENTS OF DATA, UNIT PRICE OF INFORMATION D41 DATA TYPE OF MEMBERSHIP D42 DATA SERVER D43 Ikeda at Fig. 5B. Further, as shown in FIG. 5C, the agent 1 holds data server management contents, and manages a server list D51 of servers which are permitted to provide data or a server list D52 of servers which intend to provide data as list table. Ikeda at 9:48-51.  CONTENTS OF DATA SERVER MANAGEMENT FIG.5C LIST OF SERVERS WHICH ARE PERMITTED TO PROVIDE DATA D51 OR LIST OF SERVERS WHICH INTEND TO PROVIDE DATA D52 Ikeda at Fig. 5C. **In other words, the user terminals 5 to 7 request the agent 1 to register the user terminals 5 to 7 using an address D11 of the**

	agent 1 in which the user terminals 5 to 7 should be registered, a registered data type D12 such as data type "news" and the like which is desired by the user terminals, a data obtaining requirement D13 such as "whole new information or price <10,000 yen" and the like, user information D14 such as ID and password, **terminal information D15 such as equipment type and the like, and another terminal information D16 held by the users such as equipment type, telephone number and the like**. In this case, the address D11, the registered data type D12 and the data obtaining requirement D13 are essential information. When the user information D14, the terminal information D15, and the another terminal information D16 are required by the agent 1, they are required when the agent 1 accesses to the data servers 2 to 4 as mentioned below. Ikeda at 8:58-9:7. ***<u>See, e.g., Tsutsumitake</u>*** The page request unit 111 issues the page request to a proper server 10 b (step A3). The page receiving unit 112 prepares to read a response from the server 10 (step A4) and waits until the page is sent from the server 10 (step A5). **The page receiving unit 112 analyzes the content of the page sent from the server 10 (step A6) to determine whether an event request is included in the page (step A7).** If an event request is not included, the page receiving unit 112 outputs the page to the input/output unit 110 (step A12). Thus, the content of the received page is displayed on the display of the input/output unit 110. **On the other hand, if the event request is included, the page receiving unit 112 informs the event request unit 113 of the URL of the event and causes the event request unit 113 to send the event request to the server 10 (step A8).** In this case, the page receiving unit 112 prepares to read a response (event) from the server 10 (step A9) to check whether the event has been sent from the server 10 (step A10). If the event is not sent, the page receiving unit 112 continues to wait. If the even has been sent, the page receiving unit 112 processes it (step A11). The display screen image of the input/output unit 110 is updated, depending on the result of processing of the event, as will be described later. Tsutsumitake at 10:53-11:8. **The structure of the client 11 will now be described.**

	A network interface 109 sends a page request or an event request from the client 11 to the sever 10 which is to be sent to the server 10 connected to the network 12, and receives a response from the server 10. The network interface 109 forwards the response from the server 10 to a page receiving unit 112 or an event receiving unit 114 according to the kind of the response. An input/output unit 110 comprises a keyboard, a display and a mouse. The URL of the page request is sent to a page request unit 111 by an operation using the input/output unit 110, for example, by directly inputting the URL to, e.g. the keyboard of the input/output unit 110, or by clicking a link on the page by means of the mouse. **If the address (URL) of the page is input to the page request unit 111 from the input/output unit 110, the input/output unit 110 requests the server 10 via the network interface 109 to transmit the page specified by the URL.** **The page receiving unit 112 receives a response from the server 10 to the page request, that is, the requested page, via the network interface 109**. The received page is displayed on the display of the input/output unit 110**. There is a case where the page received from the server 10 includes a description for issuing an event request to the server 10. In such a case, the page receiving unit 112 informs an event request unit 113 of the URL of the event request**. The event request unit 113 sends the URL of the event request to the server 10 via the network interface 109. Though not shown, the URL of the event request may not only be delivered from the page receiving unit 112, but also it may be directly input to the input/output unit 10 by the user. **The event receiving unit 114 receives the event sent from the server 10 on the basis of the request to the server 10 from the event request unit 113.** The received event is sent to the input/output unit 110, where necessary. For example, if the received event is an event indicating the update of the page in the server 10, the client 11 issues a page request to the server 10 and enables the updated page to be displayed on the display screen once again (i.e. the updated content being reflected on the display). In the present embodiment, the format of the page sent from the server 10 in response to the page request from the client 11 is a

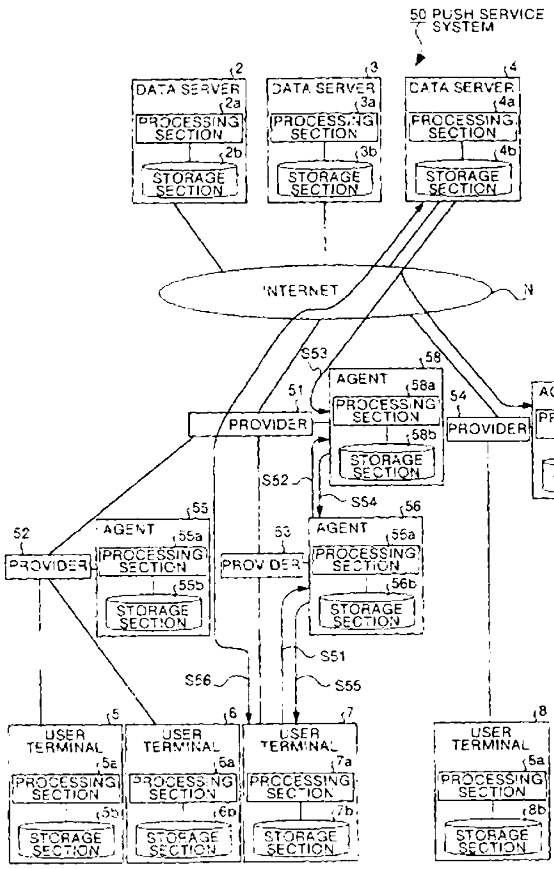
	normal HTML file format. Of course, another format may be adopted. Tsutsumitake at 9:42-10:19.  HTML FILE 100 PAGE TRANS-MISSION UNIT 103 PAGE REQUEST TRANSMISSION UNIT 102 101 NETWORK INTERFACE 105 104 EVENT REQUEST RECEIVING UNIT CONNECTION MANAGEMENT UNIT EVENT TRANS-MISSION UNIT 106 EVENT GENERATION MONITORING UNIT 108 107 EVENT GENERATION UNIT INFORMATION PROVIDING SERVER 10 CONTROLLER 40 PLANT SYSTEM 42 12 NETWORK TCP/IP 109 NETWORK INTERFACE 111 PAGE REQUEST UNIT PAGE RECEIVING UNIT 112 EVENT REQUEST UNIT 113 EVENT RECEIVING UNIT 114 INPUT/OUTPUT UNIT 110 INFORMATION DISPLAY CLIENT 11 FIG. 1 Tsutsumitake at Fig. 1. FIG. 2 shows an example of the HTML file format. **In this example, an event request to the server is included in the page. The event request is expressed by an attribute "URL" in a tag "EVENT" in a tag "EMBED".** The format of expression of the event request is not limited to this. It may be assumed, for example, that the event request is not positively described in the HTML file but is expressed as a link to another page, and the event request is described in the page of the destination link and is sent to the server 10 from the HTML file at the originating point. It is also possible to adopt a method wherein the **event request is generated dynamically from a certain program incorporated in the page** (e.g. "Applet" of Java developed by Sun Microsystems, or "JavaScript" developed by Netscape). Tsutsumitake at 10:20-29.

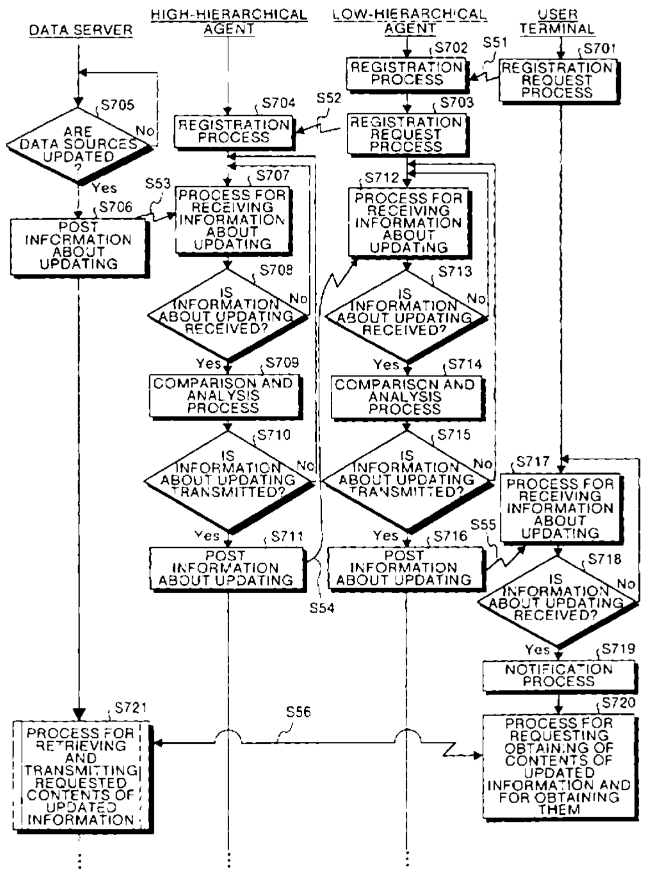
	 `< HTML >`
	`< HEAD >`
	`< TITLE > SAMPLE PAGE < /TITLE >`
	`< /HEAD >`
	`< BODY >`
	`< H1 > SAMPLE PAGE < /H1 >`
	`THIS PAGE INCLUDES EVENT REQUEST`
	`< EMBED >`
	`  < EVENT URL=http://abc.def/hij.evt >`
	`< /EMBED >`
	`< /BODY >`
	`< /HTML >`
	
	FIG. 2
	Tsutsumitake at Fig. 2.
	
	A real-time information transmission system can immediately transmit information updated on a server to a client, even if directly communication between the server and the client is not established. If the server receives a page request from the client, the server returns a requested page to the client. The same connection ID is added to the request from the client and to an associated response from the server. The client analyzes the page. If an event request is included in the analyzed page, the client adds another connection ID to the request and sends it to the server. **Using the connection ID, the server sends an event to the client each time the event has occurred. The client processes the event and reflects the processed result on a screen of a display**.
	Tsutsumitake at [Abstract].

[14.2] sending, using the processing device of the input source, an update message to the routing network, wherein the update message identifies the live object and contains update data that updates a property of the live object,	Ikeda (as modified by Tsutsumitake pursuant to claim [14.1] above) renders obvious *sending, using the processing device of the input source* (e.g., processing section of the data server), *an update message to the routing network* (e.g., sends information about updating information to agents coupled to the Internet), *wherein the update message identifies the live object* (e.g., includes a data type ID associated with an updateable element) *and contains update data that updates a property of the live object* (e.g., includes contents of updated data). Ikeda teaches that the data servers update information held and managed by the servers and provide messages about the updated information to routing network. *Ikeda* (Ex. 1004), 19:48-52, 20:21-23, 6:60-64. As discussed above, Ikeda's data servers included processing sections, which a POSITA would have understood performed the operations of the data server, including sending messages about updated information. *See* claim [14.1(a)], *supra*.

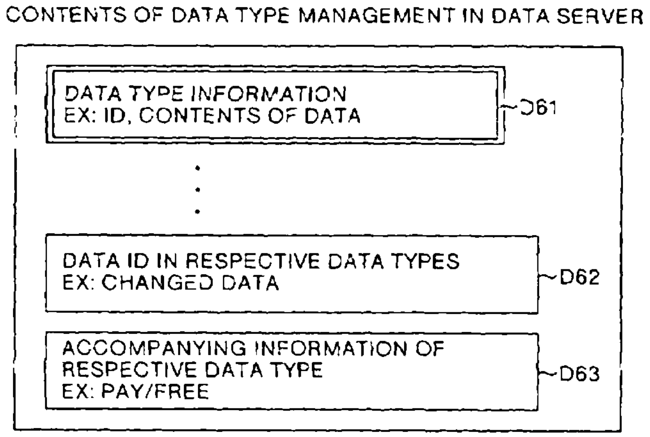
	As explained by Ikeda, the data servers include information for data stored within the server, this data included data ID, data contents, and changed data. *Id.* at 9:53-58, Fig. 6. Ikeda notes that at least data type information such as ID and data contents is essential information for realizing the push service. *Id.* at 9:58-60. Ikeda further teaches that forwarding is accomplished in the push service by agents comparing information about updating with information in their registries, such as data ID. *Id.* at 20:31-38. Thus, a POSITA would have recognized that the update message includes the data ID. *Shamos Decl.* (Ex. 1003), ¶ 70. Accordingly, Ikeda teaches that the update message identifies the data. Further, Ikeda teaches that agents may pass data contents and changed data directly to user terminals. *Id.* at 21:29-44. Although Ikeda teaches that the user terminals may use the information about updating to obtain the contents of the updated information from the data servers. (*Ikeda* (Ex. 1004), 19:60-64, 21:6-22), Ikeda also teaches that the contents of the updated information can also be passed through the agents to the user terminals. *Id.* at 21:29-43 (explaining that the contents of the updated information can be transmitted through agents according to the second and third embodiments), 11:55-65, 12:11-17 (explaining that in the third embodiment, data servers transmit information about updating, including the contents of updated information to agents). Thus, Ikeda teaches that the update message sent by the data server includes update data. Accordingly, Ikeda teaches *sending, using the processing device of the input source* (e.g., processing section of the data server), *an update message to the routing network* (e.g., sends information about updating information to agents coupled to the Internet), *wherein the update message identifies* updated data (e.g., includes a data type and ID associated with data) *and contains update data* (e.g., includes contents of updated data). However, Ikeda does not expressly teach that the data is a live object. As discussed above, in the proposed Ikeda-Tsutsumitake combination, Ikeda's push system would have been implemented (pursuant to Tsutsumitake's teachings) to provide web pages and provide updates to web pages, which included updateable elements (e.g., live objects). *See* claim [14.1(a)], *supra*. As discussed above, as modified by Tsutsumitake, the information held and managed by Ikeda's data servers would have been information for displaying web pages and updating web pages. *See* claim [14.1(a)], *supra*. Thus, in the proposed Ikeda-Tsutsumitake combination, the messages about updated information in Ikeda, when implemented according to Tsutsumitake's teachings, would have been messages regarding changes to data elements of the web page. *Shamos Decl.* (Ex. 1003), ¶ 71. This information would have included an identification of the live object, as taught by Ikeda. *Id.* Indeed, it is

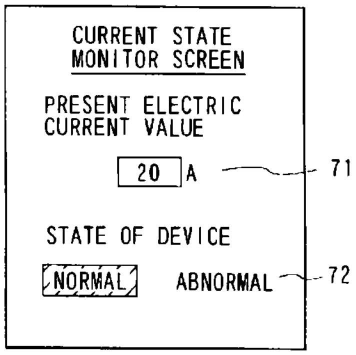
	necessary to identify which live object is to be updated. Accordingly, in the proposed combination, Ikeda's update message would have identified the live object. *Id.* Otherwise, the client would not be able to distinguish one update message from another. *Id.* Thus, the information about updating, includes an identifier (e.g., data ID) of the data and update data (contents of updated data). *Id.* Accordingly, the Ikeda-Tsutsumitake combination renders obvious *sending, using the processing device of the input source* (e.g., processing section of the data server)*, an update message to the routing network* (e.g., sends information about updating information to agents coupled to the Internet)*, wherein the update message identifies the live object* (e.g., includes a data type ID associated with an updateable element) *and contains update data* (e.g., includes contents of updated data). The Ikeda-Tsutsumitake combination further renders obvious the update data *updates a property of the live object.* Tsutsumitake teaches that changes to a data element update a property of the data element. For example, Tsutsumitake teaches that a current value is displayed as a number and this number is updated in real-time. *Tsutsumitake* (Ex. 1005), 13:31-37, 13:58-65, 14:14-28. The '722 Patent teaches that properties of live objects are "any modifiable data related to the object or referenced with respect to the object" and include properties affecting visual aspects, such as "content, color, typeface, size, formatting, or other attribute of text, images, or other data displayed by the object." *'722 Patent* (Ex. 1001), 7:35-38. Thus, because Tsutsumitake teaches changing an attribute of a text value on a web page (as opposed to re-sending the entire page), Tsutsumitake teaches updating a property of a live object. Thus, as modified by Tsutsumitake pursuant to claim [14.1(a)], *supra*, Ikeda's data servers would have sent information about updating (e.g., update messages) that included an identifier (e.g., data ID) of the updateable element (e.g., live object) and update data for the date element (data contents and/or changed data for the live object), which updates a property of the data element by visually changing the text value. *Shamos Decl.* (Ex. 1003), ¶ 72. A POSITA would have modified Ikeda with Tsutsumitake for the reasons discussed in the Request. *See* Request, Section I.E.2.c.; *Shamos Decl.* (Ex. 1003), ¶¶ 69-73, 46-60. ***<u>See, e.g., Ikeda</u>*** As for a summary of a series of the push service process, for example the user terminal 7 registered information such as an address of the user terminal 7 and information such as a data type

	to be requested into the agent 56 of the provider 53 (S51). The agent 56 serves as the user terminal for the high hierarchical agent 58. The agent 56 registers the user terminal 7 and requests the high-hierarchical agent 58 of the high-hierarchical provider 51 to register the user terminal 7 so as to register (S52). **When various information which is held and managed by the data servers 2 to 4 is updated, the server 2 to 4 IP-multi-scan information about updating to the providers 51 and 54 so as to transmit the information about updating to the agents 58 and 57 (S53).** When the information about updating is information about updating which is managed by the registered low-hierarchical agent, the agent 58 posts the information about updating to the agent 56 (S54). Ikeda at 19:40-56. FIG.15  Ikeda at Fig. 15. Meanwhile, the data servers 2 to 4 judge as to whether or not data sources of various information stored in the storage sections 2 b to 4 b are updated (step S705). **When the data sources are updated**

	(YES at step S705), the data servers 2 to 4 IP-multi-cast information about updating to the providers 51 and 57 via the Internet N and post the information about updating to the agents 58 and 57 (steps S706 and S53). Ikeda at 20:21-27. FIG.16  DATA SERVER HIGH-HIERARCHICAL AGENT LOW-HIERARCHICAL AGENT USER TERMINAL S701 REGISTRATION REQUEST PROCESS S51 S702 REGISTRATION PROCESS S703 REGISTRATION REQUEST PROCESS S52 S704 REGISTRATION PROCESS S705 ARE DATA SOURCES UPDATED ? No Yes S706 POST INFORMATION ABOUT UPDATING S53 S707 PROCESS FOR RECEIVING INFORMATION ABOUT UPDATING S708 IS INFORMATION ABOUT UPDATING RECEIVED? S709 COMPARISON AND ANALYSIS PROCESS S710 IS INFORMATION ABOUT UPDATING TRANSMITTED? S711 POST INFORMATION ABOUT UPDATING S54 S712 PROCESS FOR RECEIVING INFORMATION ABOUT UPDATING S713 IS INFORMATION ABOUT UPDATING RECEIVED? S714 COMPARISON AND ANALYSIS PROCESS S715 IS INFORMATION ABOUT UPDATING TRANSMITTED? S716 POST INFORMATION ABOUT UPDATING S55 S717 PROCESS FOR RECEIVING INFORMATION ABOUT UPDATING S718 IS INFORMATION ABOUT UPDATING RECEIVED? S719 NOTIFICATION PROCESS S720 PROCESS FOR REQUESTING OBTAINING OF CONTENTS OF UPDATED INFORMATION AND FOR OBTAINING THEM S56 S721 PROCESS FOR RETRIEVING AND TRANSMITTING REQUESTED CONTENTS OF UPDATED INFORMATION Ikeda at Fig. 16. The seventh embodiment described that the process corresponding to the process in the first embodiment is executed, but the process is not limited to this, and the process corresponding to the process in the second or third embodiment may be executed. That is, in the case where the process corresponding to the process in the second embodiment is executed, the agent 51 or the agents 55 to 56 access to the data serves 2 to 4 based on the information about updating posted by the data servers 2 to 4, and obtain the contents of the updated information. The obtained contents of the updated information is transmitted to the user terminals 5 to 7directly or via the agents 55 and 56. **Moreover, in**

	the case where the process corresponding to the process in the third embodiment is executed, the data servers2 to 4 transmit the contents of the updated information to the agent 51. The agent51 transmits the contents of the updated information to the agents 55 and 56, and the agents 55 and 56 transmit the contents of the updated information to the user terminals 5 to 7. Ikeda at 21:25-44. A third embodiment of the present invention will be explained below. In the second embodiment, instead of the user terminals 5 to 7, the agent 1 accesses to the data servers 2 to 4 and obtains the contents of the updated information, and transmits the contents of the updated information to the corresponding user terminals 5 to 7. **However, in this third embodiment, in the case where information is changed in the data servers 2 to 4, the data servers 2 to 4 transmit contents of the updated information to the agent 1. The agent 1 transmits the contents of the updated information to the user terminals 5 to 7**. Ikeda at 11:55-65. The data servers 2 to 4 judges as to whether or not data sources of various information stored in the storage sections 2 b to 4 b are updated (step S303). **When the data sources are updated (YES at step S303), the data servers 2 to 4 post contents of updated information including the information about updating to the agent 1 via the Internet N (steps S304 and S12 b).** Ikeda at 12:11-17, *see also id.* at Fig. 8. FIG. 6 is a diagram showing the data management structure in the data servers. As shown in FIG. 6, the <u>**data servers have the plurality of pieces of data type information D61 such as ID**</u> **and data contents for the push service process, and holds data IDs (D62) in the respective data types such as changing date and the like and accompanying information D63 of the respective data types such as pay/free.** <u>**The data type information D61 is essential information for realizing this push service process**</u>. Ikeda at 9:53-60.

	FIG.6  CONTENTS OF DATA TYPE MANAGEMENT IN DATA SERVER DATA TYPE INFORMATION EX: ID, CONTENTS OF DATA D61 DATA ID IN RESPECTIVE DATA TYPES EX: CHANGED DATA D62 ACCOMPANYING INFORMATION OF RESPECTIVE DATA TYPE EX: PAY/FREE D63 Ikeda at Fig. 6. ***See, e.g., Tsutsumitake*** FIG. 7 shows an example of the **screen image of the browser**. This example relates to a page for monitoring the state of current in a specific control device in the plant system 42. A page comprising a **current value display section 71** indicating a value of electric current at present and **a state display section 72** indicating the state of a device (control device) is displayed by the browser. An event request is described on this page. If the page is displayed by the browser, the event request described on the page is sent to the server 10 which provides this page. In this example, the event requested on this page includes an **event indicating a present value of an electric current**, which is sent from the server 10 at intervals of one second, and **an event indicating the state (normal/abnormal) of the device**, which is sent from the server 10 asynchronously. The current value in the specific device in the plant system 42 is monitored by the server 10 every second through the controller 40. Specifically, the event generating unit 107 or event generation monitoring unit 108 generates the event of the present value of electric current at intervals of one second. In addition, the state (normal/abnormal) of the control device is monitored by the server

	10 through the controller 40. The event generating unit 107 or event generation monitoring unit 108 generates events of the state (normal/abnormal) of the control device non-periodically. **These generated events are transmitted in real time to the client 11.** **The browser (client 11) processes a plurality of events (both the event indicating the present value of current and the event indicating the state of the device being transmitted in real time) sent from the server 10 in response to one event request issued in accordance with the displayed page, and reflects on the screen the event processing results on the current value display section 71 or device state display section 72.** Tsutsumitake at 13:31-65.  CURRENT STATE MONITOR SCREEN PRESENT ELECTRIC CURRENT VALUE 20 A 71 STATE OF DEVICE NORMAL ABNORMAL 72 FIG. 7 Tsutsumitake at Fig. 7.
[14.3] wherein a gateway device at the routing network is configured to identify a category of the update message based on the input source, to determine a node type to which the identified category maps, and to route the update message to the node,[1] having	Ikeda in view of Bird (Ground 1), or Ikeda in view of Pearson (Ground 2), renders obvious this limitation. The '722 Patent teaches that **[T]he nodes are assigned to one or more of M types, and mappings are created between message categories and node types**. **Each gateway keeps track of these mappings.** When a gateway receives messages from input sources, the gateway identifies the categories of the messages and routes the messages to the nodes of the types to which the categories are mapped. *'722 Patent* (Ex. 1001), 3:40-46. The '722 Patent explains that these mappings are maintained by the gateway and also by the nodes which use the mappings for forwarding purposes. *Id.* at [Abstract], 3:23-27, 3:42-43. The '722 Patent explains that category 1 messages are forwarded to type

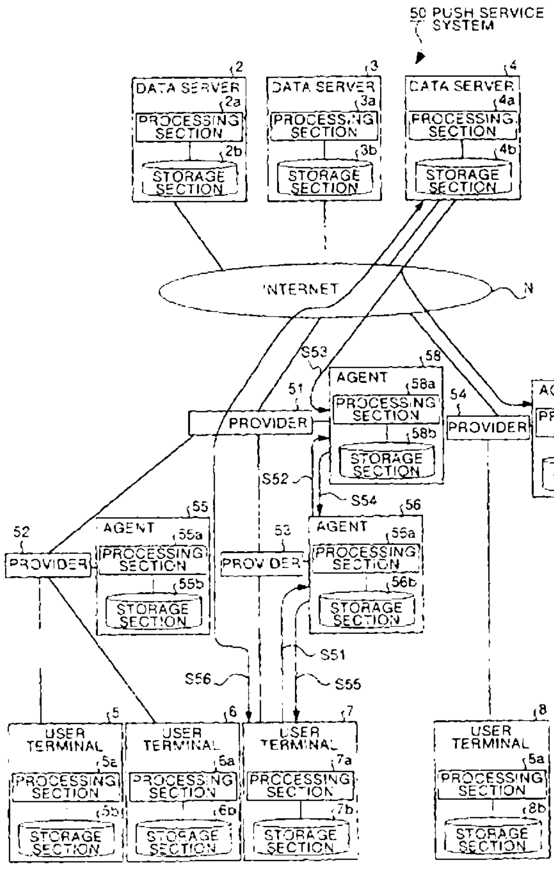
[1] Requester notes that "the node" has no antecedent basis and should therefore be found indefinite under § 112. However, for the purposes of this Request only, Requester interprets "the node" as "a node."

the node type, at the routing network,	1 nodes. *Id.* at 18:23-51. However, any type of mapping between categories and nodes is possible (e.g., multiple nodes may receive the same category of message, etc.). *Id.* Ikeda teaches that information about updating is routed through a hierarchy of agents (nodes). *Ikeda* (Ex. 1004), 19:16-23, 19:30-39, 3:31-61. First, Ikeda teaches that a high-level agent receives the information about updating and determines a type of data. *Id.* at 20:28-42. The high-level agent then determines whether the data type of the information about updating corresponds to a data type for which low-level agents are registered. *Id.* If so, the high-level agent routes the information about updating to the low-level agent. *Id.* at 20:42-47. Ikeda teaches that the forwarding of the update message is based on mappings maintained by the agents. For example, Ikeda teaches that client terminals register with low-level agents to receive specific data types. *Ikeda* (Ex. 1004), 19:40-47, 20:10-14; *see also* claim [14.1(b)], *supra.* Ikeda teaches that the low-level agents then register with high-level agents. *Ikeda* (Ex. 1004), 19:40-47, 20:14-20. These high-level agents maintain a mapping (e.g., information held by the agent) from the low-level agent to a type of data that the low-level agent is to receive. *Id.* at 20:31-68; *see also id.* at 9:18-47, Figs. 5A, 5B (disclosing data structures maintained by agents for maintaining mappings). Thus, just like the '722 Patent, which maintains mappings of nodes to data types, Ikeda also teaches maintaining mappings of nodes to data types. For example, just as category 1 data is forwarded to type 1 nodes in the '722 Patent, news data in Ikeda would have been forwarded to news-registered nodes in Ikeda. Accordingly, Ikeda teaches a high-level agent (*a gateway device at the routing network*) determines a type of data represented by the information about updating (*is configured to identify a category of the update message*) and determines which nodes are assigned to handle such data types and routes the information about updating to the determined node (*determine a node . . . to which the identified category maps, and to route the update message to the node . . . at the routing network*). Ikeda does not expressly teach that the category of the update message is based on input source. However, as explained below, Ikeda as modified by Bird (Ground 1), and/or Ikeda as modified by Pearson (Ground 2), renders obvious *identify[ing] a category of the update message based on the input source*. **<u>GROUND 1: Ikeda in view of Bird</u>** Ikeda in view of Bird renders obvious *wherein a gateway device at the routing network* (e.g., high-hierarchical agent 58) *is configured to identify a category of the update message* (e.g., the data type represented by the

	information about updating is compared with the data type registered by the user terminal so that a determination is made as to whether or not the data types coincide with each other or corresponding data type exists) *based on the input source* (e.g., the data type is the name of a publisher / sender, such as Ikeda's data sources), *to determine a node type to which the identified category maps* (e.g., judges as to whether or not the received information about updating should be transmitted to the low-hierarchical agents 55 and 56), *and to route the update message to the node, having the node type, at the routing network* (e.g., routes the information about updating to registered low-hierarchical agent 55 and/or 56). As discussed above, Ikeda teaches that a high-level agent (a gateway device at the routing network) determines a type of data represented by the information about updating (is configured to identify a category of the update message) and determines which nodes are assigned to handle such data types and routes the information about updating to the determined node (determine a node . . . to which the identified category maps, and to route the update message to the node . . . at the routing network). Bird discloses a publish/subscribe system in which a subscriber (such as a client device) registers to receive updated information from publishers. *Bird* (Ex. 1006), [0060]. Bird teaches that subscribers or clients specify a type or topic of data about which they wish to receive updates. *Id.* Bird expressly discloses that an input source (i.e., a sender or publisher of information) is one category from which a user may wish to receive information. *Bird* (Ex. 1006), [0060], [0003]. Bird further teaches that this is determined by reviewing a message header of a packet of information. *Id.* at [0044]. A POSITA would have modified Ikeda with Bird such that the name of a publisher of information (i.e., Ikeda's data sources) is one of the data types for which a client computer may register. *Shamos Decl.* (Ex. 1003), ¶¶ 74-81, 27-28 (citing *Miles, Major*). Modified as such, one type of information for which a user terminal would have requested registration would have been data corresponding to a specific input source. *Id.* at ¶ 76. As modified, the high-level agent would have determined a data type by looking to a sender identifier retrieved from the message header of a published message, as expressly taught by Bird. *Id.* (citing *Bird* (Ex. 1006), [0044]). This would then have been compared to the data types for which various low-level agents are registered to receive, as taught by Ikeda. *Id.* A POSITA would have modified Ikeda with Bird for the reasons set forth in the Request. *See* Request, Section I.E.2.d.; *Shamos Decl.* (Ex. 1003), ¶¶ 74-81, 25 (citing *Bowman-Amuah*, *Brendel*, *Tsutsumitake*), 27-28 (citing *Miles, Major*).

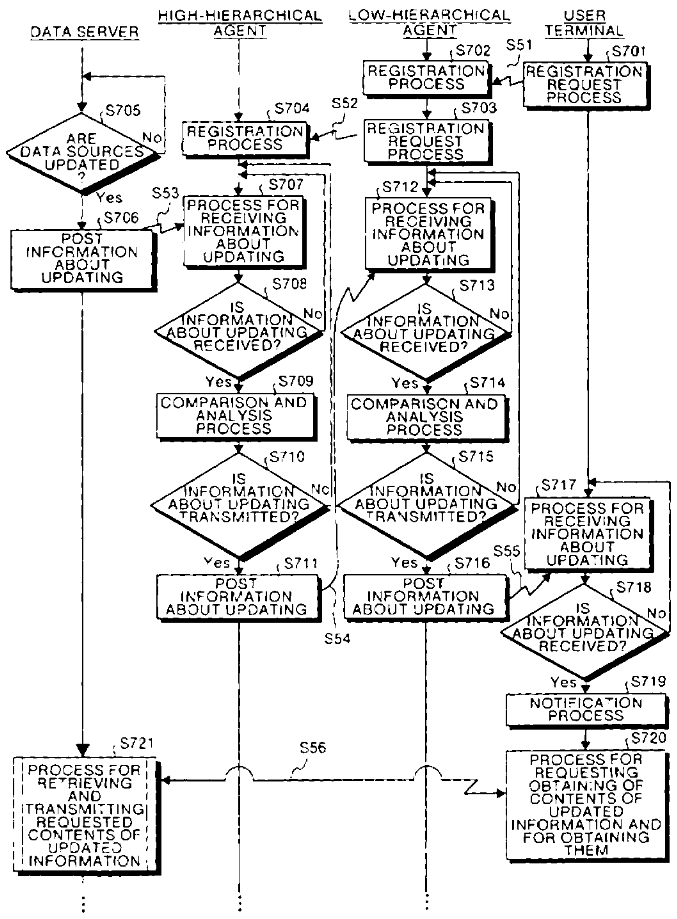
	<u>GROUND 2: Ikeda in view of Pearson</u> Ikeda in view of Pearson renders obvious *wherein a gateway device at the routing network* (e.g., high-hierarchical agent 58) *is configured to identify a category of the update message* (e.g., the data type represented by the information about updating is compared with the data type registered by the user terminal so that a determination is made as to whether or not the data types coincide with each other or corresponding data type exists) *based on the input source* (e.g., port number)*, to determine a node type to which the identified category maps* (e.g., judges as to whether or not the received information about updating should be transmitted to the low-hierarchical agents 55 and 56)*, and to route the update message to the node, having the node type, at the routing network* (e.g., routes the information about updating to registered low-hierarchical agent 55 and/or 56). As discussed above, Ikeda teaches a high-level agent (a gateway device at the routing network) determines a type of data represented by the information about updating (is configured to identify a category of the update message) and determines which nodes are assigned to handle such data types and routes the information about updating to the determined node (determine a node . . . to which the identified category maps, and to route the update message to the node . . . at the routing network). Pearson discloses that one method commonly employed in routing systems to categorize types of data was to look at a port number identifying a type of the data. *Pearson* (Ex. 1007), 18:41-19:18. A POSITA would have been familiar with port numbers and would have known that content providers (e.g., input sources), were assigned to specific port numbers based on the types of information that they produce. *Id.* at 18:62-19;13. Pearson teaches that this is determined by examining the header of a data packet. *Id.* at 18:41-19:18, 20:5-16. Accordingly, Pearson teaches determining a category of data based on an input source. *Shamos Decl.* (Ex. 1003), ¶ 101. A POSITA would have modified Ikeda with Pearson's teachings of determining a data type based on port number. *Id.* at ¶ 102. Specifically, Ikeda's high-level agent would have looked to a port number of an update message to determine a data type. *Id.* For example, Ikeda teaches that news is one type of data that a user terminal may register to receive. *Ikeda* (Ex. 1004), 8:58-9:1. A POSITA would have been familiar with port numbers and would have known that certain port numbers were used for routing specific types of data. *Shamos Decl.* (Ex. 1003), ¶¶ 102, 29-30 (citing *Greene*, *Hannel*, *RFC 1700*); *Pearson* (Ex. 1007), 19:62-20:13. In the context of news data, a POSITA would have known that port 119 was

	used to route network news transfer protocol data. *Shamos Decl.* (Ex. 1003), ¶¶ 102, 30 (citing *Hannel*, *RFC 1700*). Accordingly, a POSITA would have modified Ikeda to categorize data from port 119 as news data. *Id.* Then the update message would have been routed to the low-level agents that are registered to receive that type of data, as taught by Ikeda. *Id.* A POSITA would have modified Ikeda with Pearson's teachings for the reasons set forth in the request. *See* Request, Section I.F.2.d.; *Shamos Decl.* (Ex. 1003), ¶¶ 100-107, 25 (citing *Bowman-Amuah*, *Brendel*, *Tsutsumitake*), 29-30 (citing *Greene*, *Hannel*, *RFC 1700*). ***<u>See, e.g., Ikeda</u>*** Low hierarchical providers 52 and 53 are connected to the provider 51, and agents 55 and 56 are locally connected to the providers 52 and 53 respectively. Moreover, the providers 52, 53 and 54 connect the user terminals 5, 6, 7 and 8. As a result, the agent 55 is substitute for the user terminals 5 and 6, **the agent 56 is substitute for the user terminal 7, the agent 58 is substitute for the user terminals 5 to 7**, and the agent 57 is substitute for the user terminal 8. **Namely, the agent 58 and the agents 55 and 56 have the hierarchical relationship.** As for a summary of a series of the push service process, for example the user terminal 7 registered information such as an address of the user terminal 7 and information such as a data type to be requested into the agent 56 of the provider 53 (S51). The agent 56 serves as the user terminal for the high hierarchical agent 58. The agent 56 registers the user terminal 7 and requests the high-hierarchical agent 58 of the high-hierarchical provider 51 to register the user terminal 7 so as to register (S52). **When various information which is held and managed by the data servers 2 to 4 is updated, the server 2 to 4 IP-multi-scan information about updating to the providers 51 and 54 so as to transmit the information about updating to the agents 58 and 57 (S53). <u>When the information about updating is information about updating which is managed by the registered low-hierarchical agent, the agent 58 posts the information about updating to the agent 56 (S54)</u>.** Ikeda at 19:40-56.

	FIG.15  50 PUSH SERVICE SYSTEM DATA SERVER PROCESSING SECTION STORAGE SECTION INTERNET AGENT PROVIDER USER TERMINAL S51 S52 S53 S54 S55 S56 Ikeda at Fig. 15. A series of the push service process according to the seventh embodiment will be explained below with reference to the flow chart shown in FIG. 16. FIG. 16 is the flow chart showing the push service processing procedure performed by the push service system according to the seventh embodiment of the present invention. As shown in FIG. 16, for example user terminals 5 to 7 request the low- hierarchical agents 55 and 56 of the providers 52 and 53 connected to the user terminals 5 to 7 to register the user terminals 5 to 7 (steps S701 and S51). The low- hierarchical agents 55 and 56 which receive the registration request execute the registration process (step S702), and **request the high-hierarchical agent 58 to register the registered contents (steps S703 and S52). On the contrary, <u>the high-hierarchical agent 58 registers the low- hierarchical agents 55 and 56</u> which have been requested as to their registration (step S704).** Ikeda at 20:5-20.

> **The high-hierarchical agent 58 executes a process for receiving the information about updating (step S707),** and judges as to whether or not the information about updating is received (step S708). Thereafter, the **high-hierarchical agent 58 compares and analyzes the information about updating with information held by the agent 58 (step S709). In this comparison and analysis, the data type represented by the information about updating is compared with the data type registered by the user terminal so that a determination is made as to whether or not the data types coincide with each other or corresponding data type exists.**
>
> **Thereafter, the high-hierarchical agent 58 judges as to whether or not the received information about updating should be transmitted to the low-hierarchical agents 55 and 56 (step S710). When the judgment is made that the information about updating should be transmitted to the low-hierarchical agents 55 and 56 (YES step S710), the agent 58 posts the information about updating to the registered low- hierarchical agents 55 and 56 which have made the request (steps S711 and S54)**. When the information about updating is not received (NO at step S708) and is not transmitted (NO at step S710), the agent 58 goes to step S707 so as to repeat the process for receiving the information about updating.
>
> **The low- hierarchical agents 55 and 56 execute a process for receiving the information about updating from the high-hierarchical agent 58 (step S712),** and judge as to whether or not the information about updating is received (step S713). Thereafter, the agents 55 and 56 compare and analyze the information about updating with information held by the agents 55 and 56 (step S714). Thereafter, the agents 55 and 56 judge as to whether or not the received information about updating should be transmitted to the user terminals 5 and 6 and the user terminal 7 connected to the agents 55 and 56 respectively (step S715). When the judgment is made that the information about updating should be transmitted to the user terminals 5 to 7 (YES at step S715), the agents 55 and 56 post the information about updating to the registered user terminals 5 to 7 which have made the request (steps S716 and S55). When the information about updating is not received (NO at step S713) and is not transmitted (NO at step S715), the agents 55 and 56 go to step S712 so as to repeat the process for receiving the information about updating.

Ikeda at 20:28-21:3.

	FIG.16  DATA SERVER HIGH-HIERARCHICAL AGENT LOW-HIERARCHICAL AGENT USER TERMINAL S701 REGISTRATION REQUEST PROCESS S51 S702 REGISTRATION PROCESS S703 REGISTRATION REQUEST PROCESS S52 S704 REGISTRATION PROCESS S705 ARE DATA SOURCES UPDATED ? No Yes S706 POST INFORMATION ABOUT UPDATING S53 S707 PROCESS FOR RECEIVING INFORMATION ABOUT UPDATING S708 IS INFORMATION ABOUT UPDATING RECEIVED? S709 COMPARISON AND ANALYSIS PROCESS S710 IS INFORMATION ABOUT UPDATING TRANSMITTED? S711 POST INFORMATION ABOUT UPDATING S54 S712 PROCESS FOR RECEIVING INFORMATION ABOUT UPDATING S713 IS INFORMATION ABOUT UPDATING RECEIVED? S714 COMPARISON AND ANALYSIS PROCESS S715 IS INFORMATION ABOUT UPDATING TRANSMITTED? S716 POST INFORMATION ABOUT UPDATING S55 S717 PROCESS FOR RECEIVING INFORMATION ABOUT UPDATING S718 IS INFORMATION ABOUT UPDATING RECEIVED? S719 NOTIFICATION PROCESS S720 PROCESS FOR REQUESTING OBTAINING OF CONTENTS OF UPDATED INFORMATION AND FOR OBTAINING THEM S56 S721 PROCESS FOR RETRIEVING AND TRANSMITTING REQUESTED CONTENTS OF UPDATED INFORMATION Ikeda at Fig. 16. FIG. 5A to FIG. 5C are diagrams showing the **data management structures in the agent 1**. As shown in FIG. 5A to FIG. 5C, the agent 1 manages the user terminals 5 to 7 and the data server 2 to 4 according to the data type. As shown in FIG. 5A, the agent 1 holds user management contents of each data type, the agent 1 manages a relationship between the data types and the user terminals 5 to 7. **The user management contents include a data type information D31 such as ID and obtained data contents, user information D32 such as user ID, and another user information D33 incidental to data types and user IDs.** The data type information D31 and the user information D32 are essential information which should be managed by the agent 1. **The ID in the data type information D31 is represented by a serial number, a hierarchical code and a number from new data type, and the data types can be managed hierarchically.**

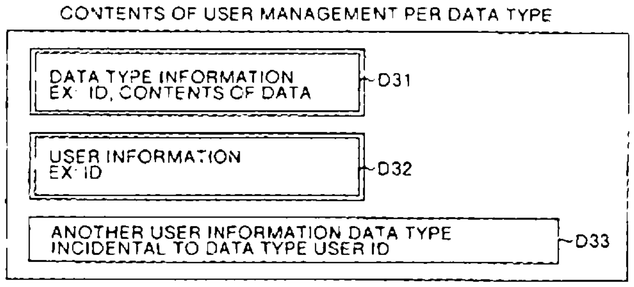
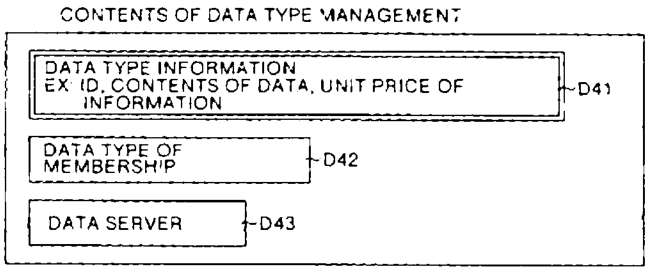
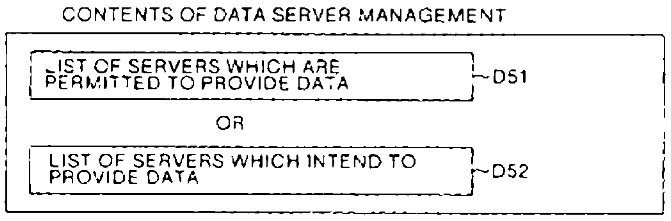
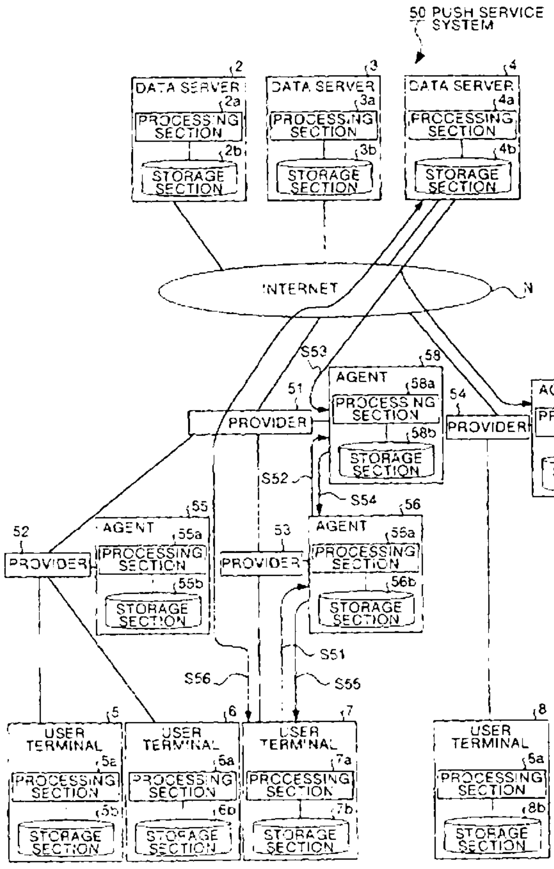
	The data contents may be managed by using XML description so that the following hierarchical link relationships: <Group1 name=news> <Group2 name=sports> . . . are established. Ikeda at 9:18-39. FIG.5A  CONTENTS OF USER MANAGEMENT PER DATA TYPE DATA TYPE INFORMATION EX: ID, CONTENTS OF DATA D31 USER INFORMATION EX: ID D32 ANOTHER USER INFORMATION DATA TYPE INCIDENTAL TO DATA TYPE USER ID D33 Ikeda at Fig. 5A. FIG. 5B shows the data type management contents by means of the agent 1. A relationship between the data types and the data servers 2 to 4 is managed by the management contents. **The data type management contents include data type information D41 such as ID, data contents and information price, membership data type D42, data server D43 which holds the data type. The data type information D41 is essential information.** Ikeda at 9:40-47. FIG.5B  CONTENTS OF DATA TYPE MANAGEMENT DATA TYPE INFORMATION EX: ID, CONTENTS OF DATA, UNIT PRICE OF INFORMATION D41 DATA TYPE OF MEMBERSHIP D42 DATA SERVER D43 Ikeda at Fig. 5B. Further, as shown in FIG. 5C, the agent 1 holds data server management contents, and manages a server list D51 of servers which are permitted to provide data or a server list D52 of servers which intend to provide data as list table. Ikeda at 9:48-51.

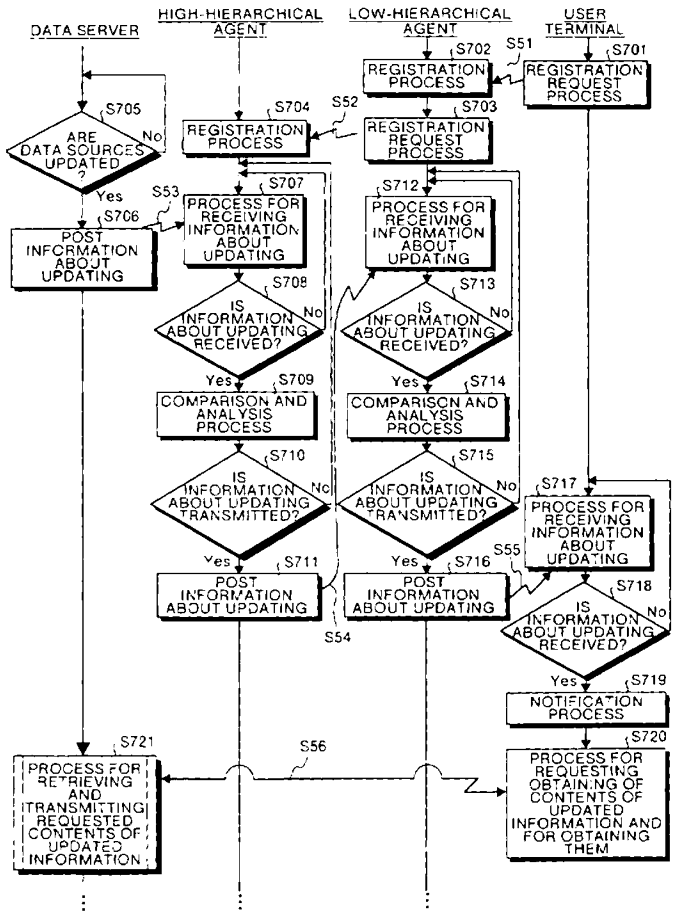
	FIG.5C  CONTENTS OF DATA SERVER MANAGEMENT LIST OF SERVERS WHICH ARE PERMITTED TO PROVIDE DATA D51 OR LIST OF SERVERS WHICH INTEND TO PROVIDE DATA D52 Ikeda at Fig. 5C. FIG. 6 is a diagram showing the data management structure in the data servers. As shown in FIG. 6, the **<u>data servers have the plurality of pieces of data type information D61 such as ID</u> and data contents for the push service process, and holds data IDs (D62) in the respective data types such as changing date and the like and accompanying information D63 of the respective data types such as pay/free. <u>The data type information D61 is essential information for realizing this push service process</u>**. Ikeda at 9:53-60. <u>*See, e.g., Bird*</u> **The message broker uses the topic identifier to check 210 records of subscriber requirements** - determining which subscribers have registered a requirement to receive messages related to this topic and determining what specific data requirements have been specified by those subscribers. As an example, **a first subscriber may have registered with the message broker with a requirement for the subscriber to be set a message whenever the message broker receives messages including <u>a topic identifier naming the specific publisher organisation</u>.** The subscriber may also have specified that all received messages from that publisher organisation should have their image content processed to generate a derived image showing classifications of crops identifiable from the captured raw image. Furthermore, the subscriber system's capabilities may impose requirements for particular image processing operations. One example is the pixel depth of the subscriber system display, which may be bilevel, 256 level greyscale, or 24 bit colour - mapped by the broker to one of a set of pixel depth classes (High, Medium or Low). Having recognised the presence of image content, the message broker accesses the system capabilities information and user-specified requirements and then generates a task definition which describes the operations to be performed to process the received message's image content in the required way. Bird at [0060].

	When a message is received by the message broker, a message analysis component 60 within the broker firstly investigates 180 the message header for a topic or subject identifier and/or identification of the sender, as is known in the art. A message publisher application may have identified a message topic or subject, such as by entering "STOCK/COMPUTERS/IBM" or "STOCK/AUTOS/GM" into a relevant field of the message header to indicate that the subject of the message is stock prices for IBM Corporation or General Motors respectively. **The message broker includes a rules engine 40 which compares 210 any topic identifier and sender identifier retrieved from the message header of a published message with a list of subscribers' information requirements to identify which subscribers wish to receive this message**. Bird at [0044]. Known publish/subscribe mechanisms rely on distribution lists, typically executing very simple logic to achieve required routing of highly structured, small transactional messages. **Subscriber applications can be registered to receive all messages which include a specific subject classification or keyword, such as a company name or a more specific subject, within the message header or within a particular content field of the structured message.** Message dictionaries hold information about the structure of messages, providing the message broker with a definition of the structure of all messages that will pass through it and so enabling unpacking of fields from the message content for inspection. The message broker queries the message header or a field within the content to retrieve the subject classification or keyword, and this is then compared with a list of subscriber applications' information requirements and any other stored rules for routing messages to determine which applications the message should be sent to. Bird at [0003]. ***See, e.g., Pearson*** Refer now to FIG. 9 for a discussion of the list 170 of simplified exemplary attack signatures. As described above in connection with FIG. 8, a communication device 106 constructed in accordance with the present invention is operative to store a list 170 of attack signatures, and to **compare an incoming communication to entries on the list t**o determine if a received communication constitutes a threat event and respond accordingly.

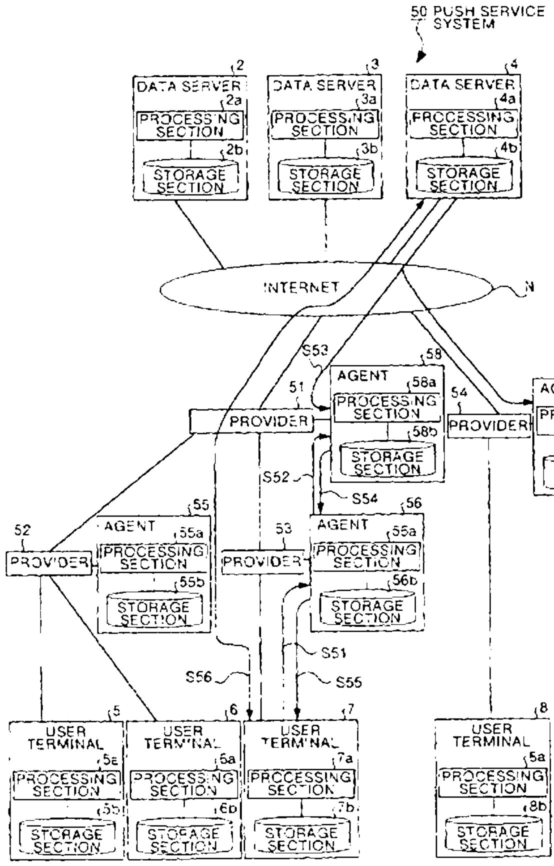
	The list 170 comprises a plurality of entries, preferably arranged in a predetermined sort order to facilitate rapid access by indexing or hashing. **Each exemplary entry on the list 170 comprises a header field 810**, a body field 820, and a two-bit priority field 826. The priority field 826 in the disclosed embodiment is 00=ignore, 01=low priority, 10=not assigned (unused), and 11=high priority. Those skilled in the art will understand that the priority field is set in accordance with user and/or predetermined remote monitoring system preferences, for example by establishing certain predetermined priorities for certain types of signatures via a high, medium, or low set policy (FIG. 4A), or by user setting of priority through the advanced options settings (FIG. 4B). **The header 810 may comprise an IP address 812 and a <u>port number 814</u>**. The IP address is the destination address of the communication device. **Port numbers appear after the IP address and typically indicate a particular type of data communication service**, some of which are known to be associated with security attacks, for example, a login host protocol on port 49. As will be known to those skilled in the art, **Internet data communication services are provided on a particular port number of a particular server that provides that service, with the port number identifying a type of data communication service. While most Internet data communication services have standard port numbers (e.g., port 21 is for ftp (file transfer protocol) services, port 23 is for telnet services, port 25 is for smtp (e-mail), port 80 is for http (hypertext transfer protocol used for the WWW), etc.)**, services can also occur on non-standard ports. Those skilled in the art will understand that standard port numbers can be obtained by reference to Internet Request for Comments (RFC) 1700, Assigned Numbers, by J. Reynolds, J. Postel, July 1994. The reader should appreciate that header 810 may comprise additional elements, such as source and destination IP addresses, source and destination ports, fragmented bits and data transfer protocols, etc., which are well known to those skilled in the art. Pearson at 18:41-19:18. Method 900 begins at step 902, where the **communication device 106 receives a communication that arrives at intrusion detector 160**. Method 900 continues from step 902 to step 904 where communication device 106 **disassembles the**

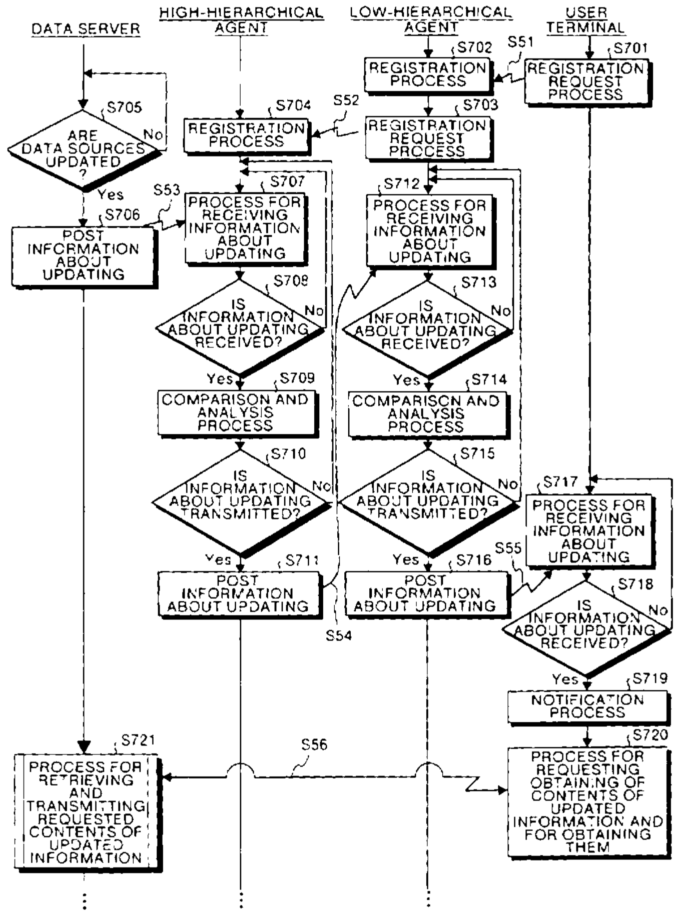
	communication to ascertain the communication's "header" information, such as the port number. Method 900 continues from step 904 to step 906, where the intrusion detector 160 **compares the communication's "header" information with entries in the list 170 of attack signatures,** and passes to decision block 910. If the information in the incoming communication **does not match an entry in the list 170 (e.g. the port number of the communication), control passes to step 912 and the communication is routed to its intended destination. However, if the information in the incoming communication matches an entry in the list 170, control passes to step 916 where the communication's body is disassembled. Control then passes to step 918.** Pearson at 20:5-16.
[14.4(a)] wherein the node is configured to identify the client device as a registered device and	Ikeda teaches, or at least renders obvious, *wherein the node is configured to identify the client device as a registered device* (e.g., the agents 55 and 56 judge as to whether or not the received information about updating should be transmitted to the user terminals 5 and 6 and the user terminal 7 connected to the agents 55 and 56 respectively). Ikeda teaches that, after receiving the information about updating from the high-level agent, the low-level agent determines whether the information about updating is information that is requested by any registered user terminals. *Ikeda* (Ex. 1004), 19:57-61, 20:51-61. Thus, the low-level agent (e.g., node) is configured to identify the client device as a registered client device. ***See, e.g., Ikeda*** Further, **when the information about updating is information about updating which is requested by the registered user terminal 7, the agent 56 posts the information about updating to the user terminal 7 (S55).** When the user terminal 7 receives the information about updating, the user terminal 7 accesses directly to the data servers 2 to 4 which have posted the information about updating via the Internet N so as to obtain contents of the updated information (S56). As a result, a series of the push service process is executed. The agents 55 to 58 have processing sections 55 a to 58 a and storage sections 55 b to 58 b respectively. The data servers 2 to 4 have processing sections 2 a to 4 a and storage sections 2 b and 4 b respectively. The user terminals 5 to 8 have processing sections 5 a to 8 a and storage sections 5 b and 8 b respectively.

	Ikeda at 19:57-20:4. FIG.15  50 PUSH SERVICE SYSTEM DATA SERVER PROCESSING SECTION STORAGE SECTION INTERNET N AGENT PROVIDER USER TERMINAL S51 S52 S53 S54 S55 S56 Ikeda at Fig. 15. The low- hierarchical agents 55 and 56 execute a process for receiving the information about updating from the high-hierarchical agent 58 (step S712), and judge as to whether or not the information about updating is received (step S713). Thereafter, the agents 55 and 56 compare and analyze the information about updating with information held by the agents 55 and 56 (step S714). **Thereafter, the agents 55 and 56 judge as to whether or not the received information about updating should be transmitted to the user terminals 5 and 6 and the user terminal 7 connected to the agents 55 and 56 respectively (step S715).** When the judgment is made that the information about updating should be transmitted to the user terminals 5 to 7 (YES at step S715), the agents 55 and 56 post the information about updating to the registered user terminals 5 to 7 which have made the request (steps S716 and S55). When the information about updating is not received (NO at step S713) and is not transmitted (NO at step

	S715), the agents 55 and 56 go to step S712 so as to repeat the process for receiving the information about updating. Ikeda at 20:51-21:3. FIG.16  DATA SERVER HIGH-HIERARCHICAL AGENT LOW-HIERARCHICAL AGENT USER TERMINAL S701 REGISTRATION REQUEST PROCESS S51 S702 REGISTRATION PROCESS S703 REGISTRATION REQUEST PROCESS S52 S704 REGISTRATION PROCESS S705 ARE DATA SOURCES UPDATED ? No Yes S706 POST INFORMATION ABOUT UPDATING S53 S707 PROCESS FOR RECEIVING INFORMATION ABOUT UPDATING S708 IS INFORMATION ABOUT UPDATING RECEIVED? S709 COMPARISON AND ANALYSIS PROCESS S710 IS INFORMATION ABOUT UPDATING TRANSMITTED? S711 POST INFORMATION ABOUT UPDATING S54 S712 PROCESS FOR RECEIVING INFORMATION ABOUT UPDATING S713 IS INFORMATION ABOUT UPDATING RECEIVED? S714 COMPARISON AND ANALYSIS PROCESS S715 IS INFORMATION ABOUT UPDATING TRANSMITTED? S716 POST INFORMATION ABOUT UPDATING S55 S717 PROCESS FOR RECEIVING INFORMATION ABOUT UPDATING S718 IS INFORMATION ABOUT UPDATING RECEIVED? S719 NOTIFICATION PROCESS S720 PROCESS FOR REQUESTING OBTAINING OF CONTENTS OF UPDATED INFORMATION AND FOR OBTAINING THEM S56 S721 PROCESS FOR RETRIEVING AND TRANSMITTING REQUESTED CONTENTS OF UPDATED INFORMATION Ikeda at Fig. 16.

[14.4(b)] to route the update message to the client device, and	Ikeda teaches, or at least renders obvious, [wherein the node is configured] *to route the update message to the client device* (e.g., the low level-agent posts the information about updating to the registered user terminals). Ikeda teaches that the low-level agent determines whether the information about updating is information that is requested by any registered user terminals. *Ikeda* (Ex. 1004), 19:57-61, 20:51-61. If yes, then the information about updating is forwarded to the user terminals registered to receive the information about updating. *Id.* at 19:57-61, 20:57-67. Thus, the low-level agents (e.g., node) is configured to route the information to the identified, registered client device. ***See, e.g., Ikeda*** **Further, when the information about updating is information about updating which is requested by the registered user terminal 7, the agent 56 posts the information about updating to the user terminal 7 (S55).** When the user terminal 7 receives the information about updating, the user terminal 7 accesses directly to the data servers 2 to 4 which have posted the information about updating via the Internet N so as to obtain contents of the updated information (S56). As a result, a series of the push service process is executed. The agents 55 to 58 have processing sections 55 a to 58 a and storage sections 55 b to 58 b respectively. The data servers 2 to 4 have processing sections 2 a to 4 a and storage sections 2 b and 4 b respectively. The user terminals 5 to 8 have processing sections 5 a to 8 a and storage sections 5 b and 8 b respectively. Ikeda at 19:57-20:4.

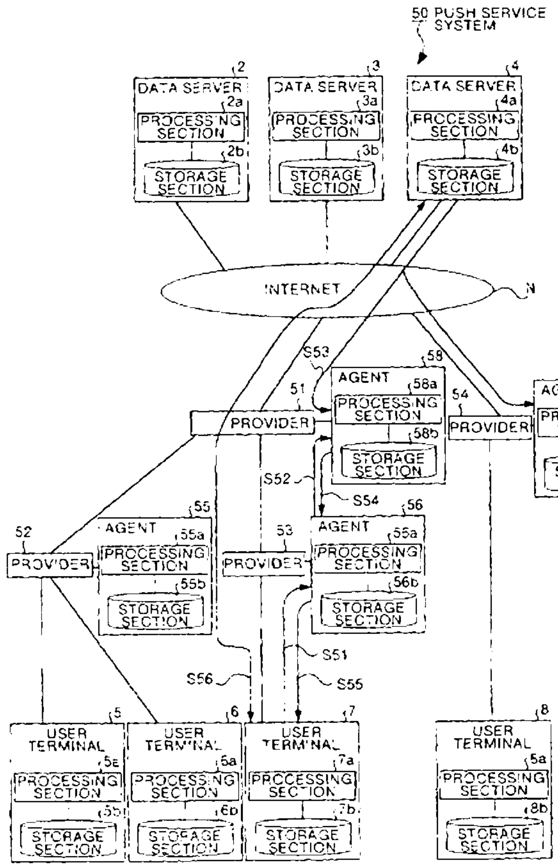
	FIG.15  50 PUSH SERVICE SYSTEM 2 DATA SERVER 2a PROCESSING SECTION 2b STORAGE SECTION 3 DATA SERVER 3a PROCESSING SECTION 3b STORAGE SECTION 4 DATA SERVER 4a PROCESSING SECTION 4b STORAGE SECTION INTERNET N S53 58 AGENT 58a PROCESSING SECTION 58b STORAGE SECTION 51 PROVIDER 54 PROVIDER 57 AGENT 57a PROCESSING SECTION 57b STORAGE SECTION S52 S54 55 AGENT 55a PROCESSING SECTION 55b STORAGE SECTION 52 PROVIDER 53 PROVIDER 56 AGENT 56a PROCESSING SECTION 56b STORAGE SECTION S51 S56 S55 5 USER TERMINAL 5a PROCESSING SECTION 5b STORAGE SECTION 6 USER TERMINAL 6a PROCESSING SECTION 6b STORAGE SECTION 7 USER TERMINAL 7a PROCESSING SECTION 7b STORAGE SECTION 8 USER TERMINAL 8a PROCESSING SECTION 8b STORAGE SECTION Ikeda at Fig. 15. The low-hierarchical agents 55 and 56 execute a process for receiving the information about updating from the high-hierarchical agent 58 (step S712), and judge as to whether or not the information about updating is received (step S713). Thereafter, the agents 55 and 56 compare and analyze the information about updating with information held by the agents 55 and 56 (step S714). Thereafter, the agents 55 and 56 judge as to whether or not the received information about updating should be transmitted tot he [sic] user terminals 5 and 6 and the user terminal 7 connected to the agents 55 and 56 respectively (step S715). **When the judgment is made that the information about updating should be transmitted to the user terminals 5 to 7 (YES at step S715), the agents 55 and 56 post the information about updating to the registered user terminals 5 to 7 which have made the request (steps S716 and S55).** When the information about updating is not received (NO at step S713) and is not transmitted (NO at step S715), the agents 55 and 56 go to step S712 so as to repeat the process for receiving the information about updating.

	Ikeda at 20:51-21:3. FIG.16  DATA SERVER HIGH-HIERARCHICAL AGENT LOW-HIERARCHICAL AGENT USER TERMINAL S701 REGISTRATION REQUEST PROCESS S702 REGISTRATION PROCESS S703 REGISTRATION REQUEST PROCESS S704 REGISTRATION PROCESS S705 ARE DATA SOURCES UPDATED? S706 POST INFORMATION ABOUT UPDATING S707 PROCESS FOR RECEIVING INFORMATION ABOUT UPDATING S708 IS INFORMATION ABOUT UPDATING RECEIVED? S709 COMPARISON AND ANALYSIS PROCESS S710 IS INFORMATION ABOUT UPDATING TRANSMITTED? S711 POST INFORMATION ABOUT UPDATING S712 PROCESS FOR RECEIVING INFORMATION ABOUT UPDATING S713 IS INFORMATION ABOUT UPDATING RECEIVED? S714 COMPARISON AND ANALYSIS PROCESS S715 IS INFORMATION ABOUT UPDATING TRANSMITTED? S716 POST INFORMATION ABOUT UPDATING S717 PROCESS FOR RECEIVING INFORMATION ABOUT UPDATING S718 IS INFORMATION ABOUT UPDATING RECEIVED? S719 NOTIFICATION PROCESS S720 PROCESS FOR REQUESTING OBTAINING OF CONTENTS OF UPDATED INFORMATION AND FOR OBTAINING THEM S721 PROCESS FOR RETRIEVING AND TRANSMITTING REQUESTED CONTENTS OF UPDATED INFORMATION S51 S52 S53 S54 S55 S56 Yes No Ikeda at Fig. 16. The seventh embodiment described that the process corresponding to the process in the first embodiment is executed, but the process is not limited to this, and the process corresponding to the process in the second or third embodiment may be executed. That is, in the case where the process corresponding to the process in the second embodiment is executed, **the agent 51 or the agents 55 to 56 access to the data serves 2 to 4 based on the information about updating posted by the data servers 2 to 4, and obtain the contents of the updated information**. **The obtained contents of the updated information is transmitted to the user terminals 5 to 7 directly or via the agents 55 and 56**. Moreover, in the case where the process corresponding to the process in the third embodiment is executed, **the data servers 2 to 4 transmit the contents of the updated information to the agent 51. The agent**

	51 transmits the contents of the updated information to the agents 55 and 56, and the agents 55 and 56 transmit the contents of the updated information to the user terminals 5 to 7. **When the data type which is managed by the high-hierarchical agent 51 is updated, the agent 51 automatically posts the updated data type to the low- hierarchical agents 55 and 56.** According to the seventh embodiment, the agent 51 is the high-hierarchical agent and the agents 55 and 56 are the low-hierarchical agents, namely, a plurality of agents are provided so as to have the hierarchical relationship. Accordingly, the labor and time required in the user terminals 5 to 7 and 45 to 47 can be reduced, and the desired contents of the updated information can be obtained quickly. Moreover, useless traffic on the Internet does not increase, and resources relating to the push service process can be reduced. Moreover, the expandability of the push service process can be heightened. Ikeda at 21:25-59.

[14.5] wherein the client device processes the update message upon receipt to update the property of the live object at the client device.	Ikeda in view of Tsutsumitake renders obvious *wherein the client device* (e.g., user terminal) *processes the update message upon receipt to update the property of the live object at the client device* (e.g., user terminal obtains contents of the updated information from the agents and updates the data element). *Ikeda* teaches that the information about updating is passed directly to the user terminal by the agents, which includes the updated data contents. *Ikeda* (Ex. 1004), 21:25-44, 11:55-65, 12:11-17. However, Ikeda does not expressly teach that the update data updates a property of a live object. Tsutsumitake teaches that changes to a data element update a property of the data element. *See* claim [14.2]. For example, Tsutsumitake teaches that a current value is displayed as a number and this number is updated in real-time. *Tsutsumitake* (Ex. 1005), 13:31-37, 13:58-65, 14:14-28. The '722 Patent teaches that properties of live objects are "any modifiable data related to the object or referenced with respect to the object" and include properties affecting visual aspects, such as "content, color, typeface, size, formatting, or other attribute of text, images, or other data displayed by the object." *'722 Patent* (Ex. 1001), 7:35-38. Thus, because Tsutsumitake teaches changing an attribute of a text value on a web page (as opposed to re-sending the entire page), Tsutsumitake teaches updating a property of a live object. As discussed above, as modified by Tsutsumitake, the received contents of the updated information would have been used to update the data element. *Shamos Decl.* (Ex. 1003), ¶¶ 82-83; *see* claims [14.1(a)], [14.2], *supra*. A POSITA would have modified Ikeda with Tsutsumitake for the reasons set forth in the Request. *See* Request, Section I.E.2.e., *supra*; *Shamos Decl.* (Ex. 1003), ¶¶ 82-84, 46-60. ***<u>See, e.g., Ikeda</u>*** > The seventh embodiment described that the process corresponding to the process in the first embodiment is executed, but the process is not limited to this, and the process corresponding to the process in the second or third embodiment may be executed. That is, in the case where the process corresponding to the process in the second embodiment is executed, the agent 51 or the agents 55 to 56 access to the data serves 2 to 4 based on the information about updating posted by the data servers 2 to 4, and obtain the contents of the updated information. The obtained contents of the updated information is transmitted to the user

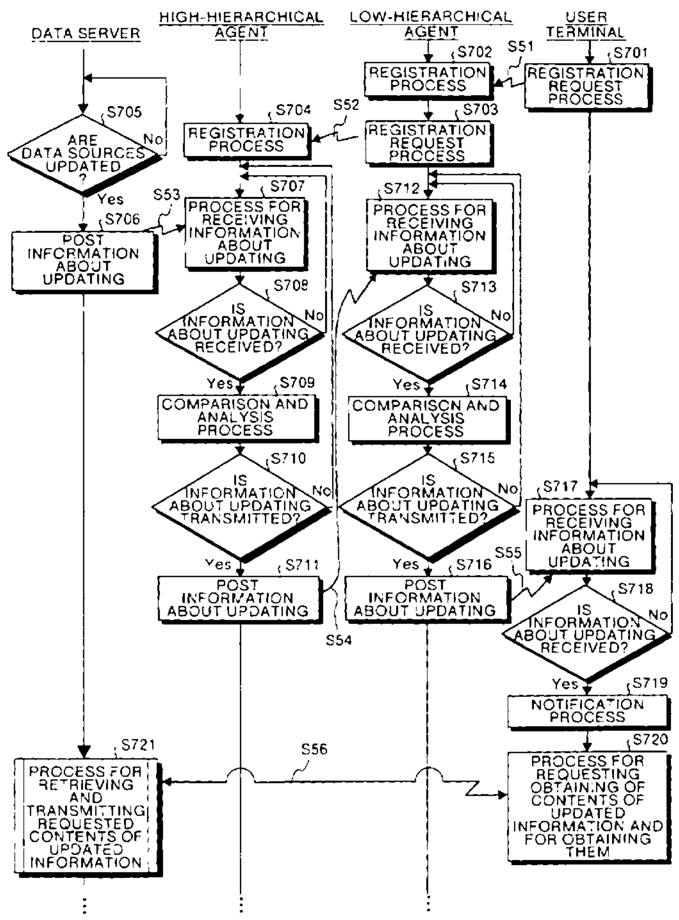
	terminals 5 to 7directly or via the agents 55 and 56. **Moreover, in the case where the process corresponding to the process in the third embodiment is executed, the data servers2 to 4 transmit the contents of the updated information to the agent 51. The agent51 transmits the contents of the updated information to the agents 55 and 56, and the agents 55 and 56 transmit the contents of the updated information to the user terminals 5 to 7.** Ikeda at 21:25-44. A third embodiment of the present invention will be explained below. In the second embodiment, instead of the user terminals 5 to 7, the agent 1 accesses to the data servers 2 to 4 and obtains the contents of the updated information, and transmits the contents of the updated information to the corresponding user terminals 5 to 7. **However, in this third embodiment, in the case where information is changed in the data servers 2 to 4, the data servers 2 to 4 transmit contents of the updated information to the agent 1. The agent 1 transmits the contents of the updated information to the user terminals 5 to 7.** Ikeda at 11:55-65. The data servers 2 to 4 judges as to whether or not data sources of various information stored in the storage sections 2 b to 4 b are updated (step S303). **When the data sources are updated (YES at step S303), the data servers 2 to 4 post contents of updated information including the information about updating to the agent 1 via the Internet N (steps S304 and S12 b).** Ikeda at 12:11-17.

	FIG.15  50 PUSH SERVICE SYSTEM DATA SERVER PROCESSING SECTION STORAGE SECTION INTERNET N AGENT PROVIDER USER TERMINAL S51 S52 S53 S54 S55 S56 Ikeda at Fig. 15. The user terminals 5 to 7 execute a process for receiving the information about updating (step S717), and judge as to whether or not the information about updating is received (step S718). **When the information about updating is received (YES at step S718), the user terminals 5 to 7 notify the information about updating (step S719). For example, the information about updating is displayed on the display sections, not shown, of the user terminal 5 to 7 or an alarm is sounded.** **Thereafter, the user terminals 5 to 7 request the data servers which hold contents of the updated information which is entity of the updated information as to obtaining of the contents of the updated information via the Internet N based on the information about updating.** The data servers retrieve the requested contents of the updated information and transmit the contents of the updated information to the user terminals which have made the request. The user terminals obtain the contents of the updated information (steps S720, S721 and S56). As a result,

a series of the push service process according to the seventh embodiment is executed.

Ikeda at 21:4-24.

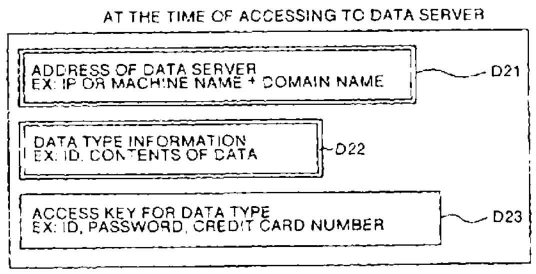
FIG.16


DATA SERVER
HIGH-HIERARCHICAL AGENT
LOW-HIERARCHICAL AGENT
USER TERMINAL
S701
REGISTRATION REQUEST PROCESS
S51
S702
REGISTRATION PROCESS
S703
REGISTRATION REQUEST PROCESS
S52
S704
REGISTRATION PROCESS
S705
ARE DATA SOURCES UPDATED ?
No
Yes
S706
POST INFORMATION ABOUT UPDATING
S53
S707
PROCESS FOR RECEIVING INFORMATION ABOUT UPDATING
S708
IS INFORMATION ABOUT UPDATING RECEIVED?
No
Yes
S709
COMPARISON AND ANALYSIS PROCESS
S710
IS INFORMATION ABOUT UPDATING TRANSMITTED?
No
Yes
S711
POST INFORMATION ABOUT UPDATING
S54
S712
PROCESS FOR RECEIVING INFORMATION ABOUT UPDATING
S713
IS INFORMATION ABOUT UPDATING RECEIVED?
No
Yes
S714
COMPARISON AND ANALYSIS PROCESS
S715
IS INFORMATION ABOUT UPDATING TRANSMITTED?
No
Yes
S716
POST INFORMATION ABOUT UPDATING
S55
S717
PROCESS FOR RECEIVING INFORMATION ABOUT UPDATING
S718
IS INFORMATION ABOUT UPDATING RECEIVED?
No
Yes
S719
NOTIFICATION PROCESS
S720
PROCESS FOR REQUESTING OBTAINING OF CONTENTS OF UPDATED INFORMATION AND FOR OBTAINING THEM
S56
S721
PROCESS FOR RETRIEVING AND TRANSMITTING REQUESTED CONTENTS OF UPDATED INFORMATION

Ikeda at Fig. 16.

FIG. 4B shows data which are used when the user terminals access to the data servers for the request to obtain the contents of the updated information. The user terminals 5 to 7 hold a data server address D21 represented by IP or machine name and domain name, data type information D22 about ID or updated information which is the contents of data, and an access key D23 to data type such as ID, password, credit card No. and the like. **The data server address D21 and the data type information D22 are essential information.**

Ikeda at 9:8-17.

	FIG.4B  AT THE TIME OF ACCESSING TO DATA SERVER ADDRESS OF DATA SERVER EX: IP OR MACHINE NAME + DOMAIN NAME D21 DATA TYPE INFORMATION EX: ID, CONTENTS OF DATA D22 ACCESS KEY FOR DATA TYPE EX: ID, PASSWORD, CREDIT CARD NUMBER D23 Ikeda at Fig. 4B. ***<u>See, e.g., Tsutsumitake</u>*** These generated **events are transmitted in real time to the client 11**. The **browser (client 11) processes a plurality of even**ts (both the event indicating the present value of current and the event indicating the state of the device being transmitted in real time) sent from the server 10 in response to one event request issued in accordance with the displayed page, and **reflects on the screen the event processing results on the current value display section 71 or device state display section 72.** Tsutsumitake at 13:56-65.

Claim 15	
15. The method of claim 14, wherein providing the data representation to the client device includes providing the live object that causes the client device to register with a client proxy of the routing network.	Ikeda teaches, or at least renders obvious, the additional limitation of claim 15. Both Ikeda in view of Tsutsumitake and Bird (Ground 1) and Ikeda in view of Tsutsumitake and Pearson (Ground 2) render obvious *[t]he method of claim 14*. *See* claim 14, *supra*. Ikeda in view of Tsutsumitake renders obvious *wherein providing the data representation to the client device* (e.g., providing the web page to the user terminal) *includes providing the live object that causes the client device to register with a client proxy of the routing network* (e.g., the updateable element is provided in the web page, the receipt of which causes the user terminal to register with a low-level agent, which is a substitute for the user terminal). As discussed above, Tsutsumitake teaches that a web page (i.e., data representation) that is provided to the client device includes events (i.e., live objects) which cause a client to register with the routing network. *See* claim [14.1(b)], *supra*. Ikeda teaches that a client registers with a low-level agent, which serves as a substitute for a user terminal. *Ikeda* (Ex. 1004), 19:30-39. Thus, Ikeda's low-level agent is a proxy of the routing network. *Shamos Decl.* (Ex. 1003), ¶¶ 85, 31 (citing *Albert*). Accordingly, as modified by Tsutsumitake to send a registration request in response to receiving a live object of a data representation (*see* claim [14.1(b)], *supra*), Ikeda's client registers with a client proxy of the routing network. ***<u>See, e.g., Ikeda</u>*** Low hierarchical providers 52 and 53 are connected to the provider 51, and agents 55 and 56 are locally connected to the providers 52 and 53 respectively. Moreover, the providers 52, 53 and 54 connect the user terminals 5, 6, 7 and 8. As a result, **the agent 55 is substitute for the user terminals 5 and 6, the agent 56 is substitute for the user terminal 7, the agent 58 is substitute for the user terminals 5 to 7**, and the agent 57 is substitute for the user terminal 8. Namely, the agent 58 and the agents 55 and 56 have the hierarchical relationship. Ikeda at 19:30-39.

Claim 16

16. The method of claim 14, wherein providing the data representation to the client device includes providing the live object that causes the client device to register with the node of the routing network.	Ikeda in view of Tsutsumitake renders obvious the additional limitation of claim 16. Both Ikeda in view of Tsutsumitake and Bird (Ground 1) and Ikeda in view of Tsutsumitake and Pearson (Ground 2) render obvious *[t]he method of claim 14*. *See* claim 14, *supra*. Ikeda in view of Tsutsumitake renders obvious *wherein providing the data representation to the client device* (e.g., providing the web page to the user terminal) *includes providing the live object that causes the client device to register with the node of the routing network* (e.g., the updateable element is provided in the web page, the receipt of which causes the user terminal to register with a low-level agent, which is part of the routing network). As discussed above, Ikeda teaches that a client registers with a low-level agent. *See* claims [14.1(b)], 15, *supra*. The low-level agent is a node of the routing network at least because it is a component that takes part in routing information through a network.

Claim 17	
17. The method of claim 14, wherein providing the data representation to the client device includes providing an activation module that is executed by the client device and that registers the live object with the routing network.	Ikeda in view of Tsutsumitake renders obvious the additional limitation of claim 17. Both Ikeda in view of Tsutsumitake and Bird (Ground 1) and Ikeda in view of Tsutsumitake and Pearson (Ground 2) render obvious *[t]he method of claim 14*. *See* claim 14, *supra*. Ikeda as modified by Tsutsumitake renders obvious, *wherein providing the data representation to the client device* (e.g., providing a web page to a user terminal) *includes providing an activation module that is executed by the client device* (e.g., an Applet or JavaScript program is incorporated in the web page and is executed by the client browser) *and that registers the live object with the routing network* (e.g., the Applet or JavaScript program performs the registration step when executed). As discussed above, Tsutsumitake teaches that, upon receiving a web page with an event request, a client device sends an event request registration to a data server. *See* claim [14.1(b)], *supra*. Tsutsumitake also teaches that this registration step may be performed by a program that is incorporated in the web page, such as an Applet or a JavaScript program. *Tsutsumitake* (Ex. 1005), 10:30-33. A POSITA would have been familiar with Applets and JavaScript programs and would have understood that such programs are executed by the browser of a client device. *Shamos Decl.* (Ex. 1003), ¶¶ 87, 32-33 (citing *England*). Thus, Tsutsumitake teaches that providing a web page to a client (e.g., *providing the data representation to the client device*) includes providing an Applet or JavaScript program that is executed by the client browser to register an

	event request (e.g., *includes providing an activation module that is executed by the client device and that registers the live object with the routing network*). A POSITA would have modified Ikeda with Tsutsumitake's teaching for the reasons set forth in the Request. *See* Request, Section I.E.3.; *Shamos Decl.* (Ex. 1003), ¶¶ 86-89, 46-60. ***<u>See, e.g., Tsutsumitake</u>*** **It is also possible to adopt a method wherein the event request is generated dynamically from a certain program incorporated in the page** (e.g. "Applet" of Java developed by Sun Microsystems, or "JavaScript" developed by Netscape). Tsutsumitake at 10:30-33.

2. Independent Claim 1 and Dependent Claim 6

Claim 1	Disclosure in Prior Art
1[P]. A method comprising:	Whether or not the preamble is limiting, Ikeda teaches, or at least renders obvious, *a method*. *See* claim 14[P], *supra*.
[1.1] receiving, using a processing device, an update message from an input source, the update message identifying a live object and containing data for updating a property of the live object;	Ikeda in view of Tsutsumitake renders obvious *receiving, using a processing device* (e.g., processing section of provider that includes an agent), *an update message* (e.g., an agent receives information about updating information) *from an input source* (e.g., data server), *the update message identifying a live object* (e.g., includes a data type ID associated with an updateable element) *and containing data for updating a property of the live object* (e.g., includes data contents). As discussed above, a POSITA would have modified Ikeda with Tsutsumitake to provide web pages and updates to web pages. *See* claim [14.1(a)], *supra*. As discussed above, Ikeda in view of Tsutsumitake renders obvious *sending, using the processing device of the input source* (e.g., processing section of the data server), *an update message to the routing network* (e.g., sends information about updating information to agents coupled to the Internet), *wherein the update message identifies the live object* (e.g., includes a data type ID associated with an updateable element) *and contains update data that updates a property of the live object* (e.g., includes contents of updated data). *See* claim [14.2], *supra*. Thus, Ikeda in view of Tsutsumitake renders obvious an input source sending an *update message identifying a live object and containing data for updating a property of the live object*. Ikeda teaches that the information about updating (e.g., update message) is received by a high-hierarchical agent. *Ikeda* (Ex. 1004), 21:21-38. Ikeda teaches that the high-hierarchical agent includes a processing section (e.g., a processing device), which a POSITA would have understood executes software to perform its functions, including receiving the information about updating. *Shamos Decl.* (Ex. 1003), ¶ 25 (citing *Bowman-Amuah, Brendel, Tsutsumitake*). ***<u>See, e.g., Ikeda</u>*** Meanwhile, the data servers 2 to 4 judge as to whether or not data sources of various information stored in the storage sections 2 b to 4 b are updated (step S705). **When the data sources are updated (YES at step S705), the data servers 2 to 4 IP-multi-cast information about updating to the providers 51 and 57 via the**

	Internet N and post the information about updating to the agents 58 and 57 (steps S706 and S53). **The high-hierarchical agent 58 executes a process for receiving the information about updating (step S707), and judges as to whether or not the information about updating is received (step S708).** Thereafter, the high-hierarchical agent 58 compares and analyzes the information about updating with information held by the agent 58 (step S709). In this comparison and analysis, the data type represented by the information about updating is compared with the data type registered by the user terminal so that a determination is made as to whether or not the data types coincide with each other or corresponding data type exists. Ikeda at 21:21-38; *see also id.* at Fig. 16. As for a summary of a series of the push service process, for example the user terminal 7 registered information such as an address of the user terminal 7 and information such as a data type to be requested into the agent 56 of the provider 53 (S51). The agent 56 serves as the user terminal for the high hierarchical agent 58. The agent 56 registers the user terminal 7 and requests the high-hierarchical agent 58 of the high-hierarchical provider 51 to register the user terminal 7 so as to register (S52). **When various information which is held and managed by the data servers 2 to 4 is updated, the server 2 to 4 IP-multi-scan information about updating to the providers 51 and 54 so as to transmit the information about updating to the agents 58 and 57 (S53).** When the information about updating is information about updating which is managed by the registered low-hierarchical agent, the agent 58 posts the information about updating to the agent 56 (S54). Further, when the information about updating is information about updating which is requested by the registered user terminal 7, the agent 56 posts the information about updating to the user terminal 7 (S55). When the user terminal 7 receives the information about updating, the user terminal 7 accesses directly to the data servers 2 to 4 which have posted the information about updating via the Internet N so as to obtain contents of the updated information (S56). As a result, a series of the push service process is executed. **The agents 55 to 58 have processing sections 55 a to 58 a and storage sections 55 b to 58 b respectively**. The data servers 2 to 4 have processing sections 2 a to 4 a and storage sections 2 b and 4 b respectively. The user terminals 5 to 8 have processing sections 5 a to 8 a and storage sections 5 b and 8 b respectively. Ikeda at 19:40-20:4; *see also id.* at Fig. 16.

[1.2] identifying a category of the update message based on the input source;	Ikeda in view of Bird (Ground 1), or Ikeda in view of Pearson (Ground 2) renders obvious this limitation. **<u>Ground 1: </u>Ikeda, Tsutsumitake, and Bird** Ikeda in view of Bird (Ground 1) renders obvious *identifying a category of the update message* (e.g., the data type represented by the information about updating is compared with the data type registered by the user terminal so that a determination is made as to whether or not the data types coincide with each other or corresponding data type exists) *based on the input source* (e.g., the data type is the name of a publisher / sender, such as Ikeda's data sources). *See* claim [14.3], *supra.* **<u>Ground 2: </u>Ikeda, Tsutsumitake, and Pearson** Ikeda in view of Pearson (Ground 2) renders obvious *identifying a category of the update message* (e.g., the data type represented by the information about updating is compared with the data type registered by the user terminal so that a determination is made as to whether or not the data types coincide with each other or corresponding data type exists) *based on the input source* (e.g., port number). *See* claim [14.3], *supra.*

[1.3] determining a node having a node type to which the update message is to be routed based on a mapping of categories of update messages to node types, the mapping controlling an amount of update message traffic through nodes of a routing network;[2]	Ikeda renders obvious *determining a node having a node type to which the update message is to be routed based on a mapping of categories of update messages to node types* (e.g., judges as to whether or not the received information about updating should be transmitted to the low-hierarchical agents 55 and 56)), *the mapping controlling an amount of update message traffic through nodes of a routing network* (e.g., decreasing useless traffic on the Internet). As discussed above, Ikeda teaches, or at least renders obvious, *a gateway device at the routing network* (e.g., high-hierarchical agent 58) *is configured to identify a category of the update message* (e.g., the data type represented by the information about updating is compared with the data type registered by the user terminal so that a determination is made as to whether or not the data types coincide with each other or corresponding data type exists) . . . *to determine a node type to which the identified category maps* (e.g., judges as to whether or not the received information about updating should be transmitted to the low-hierarchical agents 55 and 56). *See* claim [14.3], *supra*. Accordingly, because Ikeda teaches determining a node to which the identified category maps, Ikeda teaches determining a node having a node type to which the update message is to be routed based on a mapping of categories of update messages to node types. Ikeda further teaches, or at least renders obvious, that the mapping controls an amount of update message traffic through nodes of a routing network. For example, Ikeda expressly teaches that reducing traffic is one benefit of the system. Ikeda explains that the use of agents, which includes low-level agents (e.g., nodes having node types to which update messages are to be routed based on a mapping of categories of update messages to node types), reduces useless traffic on the Internet. *Ikeda* (Ex. 1004), 21:49-59. Indeed, a POSITA would have understood that by only transmitting information about updating to nodes registered to receive specific types of information, the amount of traffic through the nodes would be less than if information about updating were transmitted to all nodes regardless of registration. *Shamos Decl.* (Ex. 1003), ¶ 90. ***<u>See, e.g., Ikeda</u>*** However, the conventional push services where the access is executed per constant time, since a check is made as to whether or not the information in the data source is updated in each access, **there arises a problem that useless traffic on the network increase**.

[2] Requester notes that the phrase "the mapping controlling an amount of update message traffic through nodes of a routing network" recites nothing more than an intended use or intended result. Such statements should not be given patentable weight. *Boehringer Ingelheim Vetmedica, Inc. v. Schering-Plough Corp.*, 320 F.3d 1339, 1345 (Fed. Cir. 2003). However, as shown above, even if patentable weight is given to this limitation, Ikeda satisfies the limitation.

Ikeda at 1:37-41.

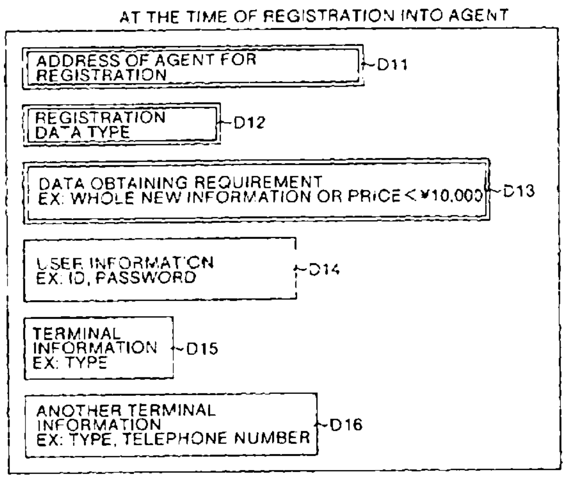
> According to the seventh embodiment, the agent 51 is the high-hierarchical agent and the agents 55 and 56 are the low-hierarchical agents, namely, a plurality of agents are provided so as to have the hierarchical relationship. Accordingly, the labor and time required in the user terminals 5 to 7 and 45 to 47 can be reduced, and the desired contents of the updated information can be obtained quickly. **Moreover, useless traffic on the Internet does not increase, and resources relating to the push service process can be reduced**. Moreover, the expandability of the push service process can be heightened.

Ikeda at 21:49-59.

> Information about the push service process which is managed by the user terminals 5 to 7, the agent 1 and the data servers 2 to 4 will be explained below with reference to FIG. 4 to FIG. 6. FIG. 4A and FIG. 4B are diagrams showing data management structures in the user terminals. **When the user terminals 5 to 7 request the agent 1 to register the user terminals, data, for example, shown in FIG. 4A are required.**

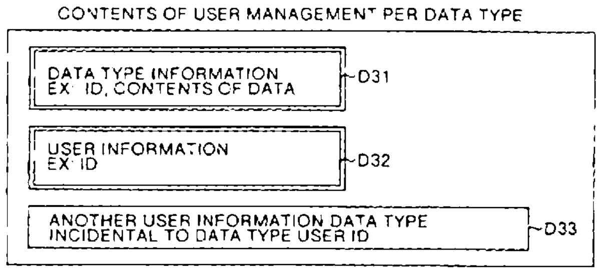
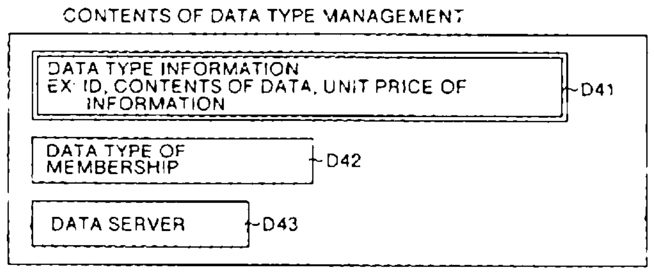
Ikeda at 8:51-57.

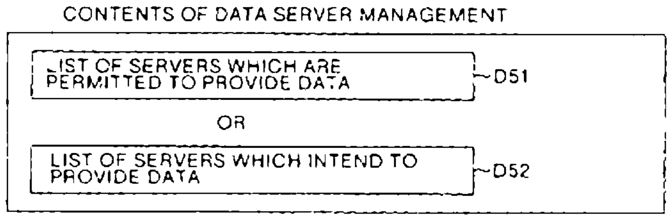
FIG.4A


AT THE TIME OF REGISTRATION INTO AGENT
ADDRESS OF AGENT FOR REGISTRATION
D11
REGISTRATION DATA TYPE
D12
DATA OBTAINING REQUIREMENT EX: WHOLE NEW INFORMATION OR PRICE<¥10,000
D13
USER INFORMATION EX: ID, PASSWORD
D14
TERMINAL INFORMATION EX: TYPE
D15
ANOTHER TERMINAL INFORMATION EX: TYPE, TELEPHONE NUMBER
D16

Ikeda at Fig. 4A.

> FIG. 5A to FIG. 5C are diagrams showing the data management structures in the agent 1. As shown in FIG. 5A to FIG. 5C, the agent 1 manages the user terminals 5 to 7 and the data server 2 to 4 according to the data type. As shown in FIG. 5A, the agent 1 holds user management contents of each data type, the agent 1 manages a relationship between the data types and the user terminals 5 to 7. **The user management contents include a data**

	type information D31 such as ID and obtained data contents, user information D32 such as user ID, and another user information D33 incidental to data types and user IDs. The data type information D31 and the user information D32 are essential information which should be managed by the agent 1. **The ID in the data type information D31 is represented by a serial number, a hierarchical code and a number from new data type, and the data types can be managed hierarchically**. The data contents may be managed by using XML description so that the following hierarchical link relationships: <Group1 name=news> <Group2 name=sports> . . . are established. Ikeda at 9:18-39.  CONTENTS OF USER MANAGEMENT PER DATA TYPE DATA TYPE INFORMATION EX: ID, CONTENTS OF DATA D31 USER INFORMATION EX: ID D32 ANOTHER USER INFORMATION DATA TYPE INCIDENTAL TO DATA TYPE USER ID D33 FIG.5A Ikeda at Fig. 5A. FIG. 5B shows the data type management contents by means of the agent 1. A relationship between the data types and the data servers 2 to 4 is managed by the management contents. **The data type management contents include data type information D41 such as ID, data contents and information price, membership data type D42, data server D43 which holds the data type. The data type information D41 is essential information.** Ikeda at 9:40-47.  CONTENTS OF DATA TYPE MANAGEMENT DATA TYPE INFORMATION EX: ID, CONTENTS OF DATA, UNIT PRICE OF INFORMATION D41 DATA TYPE OF MEMBERSHIP D42 DATA SERVER D43 FIG.5B Ikeda at Fig. 5B.

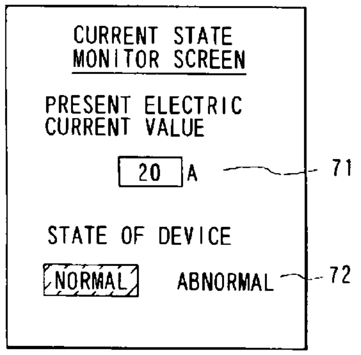
	Further, as shown in FIG. 5C, the agent 1 holds data server management contents, and manages a server list D51 of servers which are permitted to provide data or a server list D52 of servers which intend to provide data as list table. Ikeda at 9:48-51.  CONTENTS OF DATA SERVER MANAGEMENT LIST OF SERVERS WHICH ARE PERMITTED TO PROVIDE DATA D51 OR LIST OF SERVERS WHICH INTEND TO PROVIDE DATA D52 FIG.5C Ikeda at Fig. 5C. Further, a plurality of user terminals transmit information of the user terminals and requested various information to a high-hierarchical agent which is locally connected to the provider connecting the user so as to execute the registration process. The high-hierarchical agent transmits the information of the registered user terminals and requested various information to a higher-hierarchical agent so as to execute a registration process. A plurality of data servers have various information, and IP-multi-cast and post information about updating of various information or contents of the updated information to the highest-hierarchical agents via the provider. When the posted information about updating or contents of the updated information is the information about updating which is requested by the registered user terminals, the highest-hierarchical agents post the information about updating to the registered user terminals or the contents of the updated information to the low-hierarchical agents. When the posted information about updating and contents of the updated information is information about updating requested by the registered user terminals, the low-hierarchical agents which have received the post the information about updating to the registered user terminals or contents of the updated information to the low-hierarchical agents or the user terminals. When the low-hierarchical agents have lower-hierarchical agents, the posting process which is similar to the low-hierarchical agents is executed. The user terminals access to the data servers which hold the contents of the updated information based on the posted information about updating and obtain the contents of the updated information, or receive the posted contents of the updated information so as to obtain them. For this reason, the labor and time which are required for the user terminals can be further reduced, and the desired updated information contents can be obtained quickly. **Moreover, since the limited agents intervene**

	in the network, useless traffic on the network does not increase, and resources to be required for the push service process can be reduced. Particularly since the representative agent can execute the registration process for the agents flexibly, the expandability of the push service can be heightened. Ikeda at 26:31-27:3.
[1.4] routing, using the processing device, the update message to the node having the determined node type;	Ikeda teaches, or at least renders obvious, *routing, using the processing device, the update message to the node having the determined node type* (e.g., high-hierarchical agent routes the information about updating to registered low-hierarchical agent 55 and/or 56 using its processing section). *See* claim [14.3], *supra.*
[1.5] causing the node, through the update message, to determine a client, different from the input source, that has registered for updates of the live object;	Ikeda teaches, or at least renders obvious, *causing the node, through the update message, to determine a client, different from the input source, that has registered for updates of the live object* (e.g., upon receiving the information about updating the low-level agents 55 and 56 judge as to whether or not the received information about updating should be transmitted to the user terminals 5 and 6 and the user terminal 7 connected to the agents 55 and 56 respectively). *See* claim [14.4(a)], *supra.*
[1.6] causing the node to route the update message from the node to the client; and	Ikeda teaches, or at least renders obvious, *causing the node to route the update message from the node to the client* (e.g., upon determining a user terminal has registered for the update, the low level-agent posts the information about updating to the registered user terminals). *See* claim [14.4(b)], *supra.*
[1.7] causing the client to process the update message and to update the property of the lire object.[3]	Ikeda teaches, or at least renders obvious, *causing the client* (e.g., user terminal) *to process the update message and to update the property of the lire object* (e.g., user terminal obtains contents of the updated information from the agents and updates the data element). *See* claim [14.5], *supra.*

Claim 6	
6. The method of claim 1, wherein the property of the live object has a	Ikeda in view of Tsutsumitake renders obvious the additional limitations of claim 6. Ikeda in view of Tsutsumitake and Bird (Ground 1), or Ikeda in view of Tsutsumitake and Pearson (Ground 2), renders obvious *[t]he method of claim 1. See* claim 1, *supra.*

[3] The applicant filed a certificate of correction changing "of the lire" to "of the live." *'722 Patent* (Ex. 1001).

direct effect on a visual representation of the live object in a data representation, has an effect on an internal aspect of the live object and has no effect on the visual representation of the live object in the data representation, or has a direct effect on one aspect of the visual representation of the live object in the data representation and has no effect on other aspect of the visual representation of the live object in the data representation.	Ikeda in view of Tsutsumitake renders obvious *the property of the live object has a direct effect on a visual representation of the live object in a data representation* (e.g., the text value, such as number, of the updateable element of the web page is changed). Tsutsumitake teaches that web pages include data elements, such as text representing numerical values (e.g., current value of electric current). *Tsutsumitake* (Ex. 1005), 13:31-37. Tsutsumitake expressly teaches that a displayed value (visual representation) is changed. *Id.* at 13:58-65, Fig. 7. Accordingly, as modified by Tsutsumitake, Ikeda's update information would have updated a property of the live object, including having a direct effect on the visual representation of the live object on the web page. A POSITA would have made the proposed modification for the reasons set forth in the Request. *See* Request, Section I.E.4; *Shamos Decl.* (Ex. 1003), ¶¶ 91, 46-60, 69-73, 82-84. ***<u>See, e.g., Tsutsumitake</u>*** **FIG. 7 shows an example of the screen image of the browser**. This example relates to a page for monitoring the state of current in a specific control device in the plant system 42. **A page comprising a current value display section 71 indicating a value of electric current at present and a state display section 72 indicating the state of a device (control device) is displayed by the browser**. An event request is described on this page. If the page is displayed by the browser, the event request described on the page is sent to the server 10 which provides this page. In this example, the event requested on this page includes an event indicating a present value of an electric current, which is sent from the server 10 at intervals of one second, and an event indicating the state (normal/abnormal) of the device, which is sent from the server 10 asynchronously. The current value in the specific device in the plant system 42 is monitored by the server 10 every second through the controller 40. Specifically, the event generating unit 107 or event generation monitoring unit 108 generates the event of the present value of electric current at intervals of one second. In addition, the state (normal/abnormal) of the control device is monitored by the server 10 through the controller 40. The event generating unit 107 or

	event generation monitoring unit 108 generates events of the state (normal/abnormal) of the control device non-periodically. **These generated events are transmitted in real time to the client 11.** The browser (client 11) processes a plurality of events (both the event indicating the present value of current and the event indicating the state of the device being transmitted in real time) sent from the server 10 in response to one event request issued in accordance with the displayed page, **and <u>reflects on the screen the event processing results on the current value display section 71 or device state display section 72</u>.** The event processing results can be **reflected on the page screen image** (event screen image) by using the Java(Applet) of Sun Microsystems, JavaScript of Netscape Communications, or Dynamic HTML of Microsoft. Needless to say, the method is not limited to these. Tsutsumitake at 13:31-14:3.  CURRENT STATE MONITOR SCREEN PRESENT ELECTRIC CURRENT VALUE 20 A 71 STATE OF DEVICE NORMAL ABNORMAL 72 FIG. 7 Tsutsumitake at Fig. 7. According to the browser screen in FIG. 7 of the electric current state monitoring page in the client 11, **the user can monitor the present electric current value of the specific control device of the plant system at intervals of one second, without performing a display update operation, for example, by depressing a reload button. In addition, the user can monitor the change in state (normal/abnormal) of the control device.** As has been described above, since the server continuously returns responses (generated events) to a single event request from the client 11 while the connection is being maintained, the information transmission can be efficiently carried out without increasing a communication load between the client 11 and server 10.

	Tsutsumitake at 14:16-28.

3. Independent Claim 7

Claim 7	
7[P]. A routing network comprising:	Whether or not the preamble is limiting, Ikeda teaches, or at least renders obvious, *[a] routing network comprising* (e.g., a push service including various components coupled to the Internet). Ikeda teaches a push service system that routes information over the Internet, which is a routing network. *Ikeda* (Ex. 1004), 1:6-17, 19:14-29. Ikeda teaches that various components (e.g., servers, user terminals, and agents) are coupled to the Internet and perform the functions recited in claims [7.1]-[7.8(b)] below. *Id.* at 19:14-29. Thus, Ikeda teaches, or at least renders obvious, *a routing network*. ***<u>See, e.g., Ikeda</u>*** It is an object of the present invention to provide a push service system and a push service processing method in which **resources in a network are utilized efficiently** without increasing traffic on the network, and information required by users can be obtained quickly and expandability is high. Ikeda at 1:58-62. The present invention relates to a push service system and a push service processing method for **processing push service in a network using IP (Internet protocol)**. Ikeda at 1:6-8. In recent years, as **computer technique and IP network have developed, push service is widely provided in the Internet or intranet. The push service transmits data of a host apparatus in the network to a low-order apparatus of a user or the like** at timing different from that of a request from the low-order apparatus. To transmit the data from the host apparatus to the low-order apparatus when the low-order apparatus requests the data from the host apparatus is called pull. In this push service, it is important that data are distributed efficiently and quickly, and realization of the push service which distributes data efficiently and quickly is realized. Ikeda at 1:12-23.

	A seventh embodiment of the present invention will be explained below. In the sixth embodiment, **the agents are provided in the provider to be connected to the Internet N**, and the push service process is executed via the agents. However, in the seventh embodiment, a hierarchical relationship is given to the agents correspondingly to a hierarchical relationship of the providers so that the push service process is executed. FIG. 15 is a diagram showing the structure of the push service system according to the seventh embodiment of the present invention. As shown in FIG. 15, in the push service system 50, **a plurality of data servers 2 to 4 and a plurality of providers 51 to 54 are connected to the Internet N**. The providers 51 to 54 locally connect the agents 55 to 58. Ikeda at 19:14-29. Further, a plurality of user terminals transmit information of the user terminals and requested various information to a high-hierarchical agent which is locally connected to the provider connecting the user so as to execute the registration process. The high-hierarchical agent transmits the information of the registered user terminals and requested various information to a higher-hierarchical agent so as to execute a registration process. A plurality of data servers have various information, and IP-multi-cast and post information about updating of various information or contents of the updated information to the highest-hierarchical agents via the provider. When the posted information about updating or contents of the updated information is the information about updating which is requested by the registered user terminals, the highest-hierarchical agents post the information about updating to the registered user terminals or the contents of the updated information to the low-hierarchical agents. When the posted information about updating and contents of the updated information is information about updating requested by the registered user terminals, the low-hierarchical agents which have received the post the information about updating to the registered user terminals or contents of the updated information to the low-hierarchical agents or the user terminals. When the low-hierarchical agents have lower-hierarchical agents, the posting process which is similar to the low-hierarchical agents is executed. The user terminals access to the data servers which hold the contents of the updated information based on the posted information about updating and obtain the contents of the updated information, or receive the posted contents of the updated information so as to obtain them. For this reason, the labor and time which are required for the user terminals can be further

	reduced, and the desired updated information contents can be obtained quickly. **Moreover, since the limited agents intervene in the network, useless traffic on the network does not increase, and resources to be required for the push service process can be reduced**. Particularly since the representative agent can execute the registration process for the agents flexibly, the expandability of the push service can be heightened. Ikeda at 26:31-27:3. In recent years, as computer technique and IP network have developed, **push service is widely provided in the Internet or intranet**. The push service transmits data of a host apparatus in the network to a low-order apparatus of a user or the like at timing different from that of a request from the low-order apparatus. To transmit the data from the host apparatus to the low-order apparatus when the low-order apparatus requests the data from the host apparatus is called pull. In this push service, it is important that data are distributed efficiently and quickly, and realization of the push service which distributes data efficiently and quickly is realized. Ikeda at 1:12-23.
[7.1] a gateway device configured to:	Ikeda teaches, or at least renders obvious, *a gateway device* (e.g., high-hierarchical agent) *configured to* perform the functions recited in claims [7.2]-[7.4(b)] below. *See* claim [14.3], *supra*.

[7.2] receive an update message from an input source, the update message identifying a live object and containing data for updating a property of the live object,	Ikeda in view of Tsutsumitake renders obvious [a gateway device (e.g., high-hierarchical agent) configured to:] *receive an update message* (e.g., receives information about updating information) *from an input source* (e.g., data server), *the update message identifying a live object* (e.g., includes a data type ID associated with an updateable element) *and containing data for updating a property of the live object* (e.g., includes data contents). *See* claim [1.1], *supra.*
[7.3] identify a category of the update message based on the input source;	Ikeda in view of Bird (Ground 1), or Ikeda in view of Pearson (Ground 2) renders obvious this limitation. **Ground 1: Ikeda in view of Bird** Ikeda in view of Bird renders obvious [a gateway device (e.g., high-hierarchical agent) configured to:] *identify a category of the update message* (e.g., the data type represented by the information about updating is compared with the data type registered by the user terminal so that a determination is made as to whether or not the data types coincide with each other or corresponding data type exists) *based on the input source* (e.g., the data type is the name of a publisher / sender, such as Ikeda's data sources). *See* claim [1.2], *supra.* **Ground 2: Ikeda in view of Pearson** Ikeda in view of Tsutsumitake renders obvious [a gateway device (e.g., high-hierarchical agent) configured to:] *identify a category of the update message* (e.g., the data type represented by the information about updating is compared with the data type registered by the user terminal so that a determination is made as to whether or not the data types coincide with each other or corresponding data type exists) *based on the input source* (e.g., port number). *See* claim [1.2], *supra.*

[7.4(a)] determine a node having a node type to which the update message is to be routed based on a mapping of a categories of update messages to node types, the mapping controlling an amount of update message traffic through nodes of a routing network; [4] and	Ikeda in view of Tsutsumitake renders obvious [a gateway device (e.g., high-hierarchical agent) configured to:] *determine a node having a node type to which the update message is to be routed based on a mapping of a categories of update messages to node types* (e.g., judges as to whether or not the received information about updating should be transmitted to the low-hierarchical agents 55 and 56), *the mapping controlling an amount of update message traffic through nodes of a routing network* (e.g., decreasing useless traffic on the Internet). *See* claim [1.3], *supra.*
[7.4(b)] route the update message; and	Ikeda in view of Tsutsumitake renders obvious [a gateway device (e.g., high-hierarchical agent) configured to:] *route the update message. See* claim [1.4], *supra.*
[7.5] the node configured to:	Ikeda teaches, or at least renders obvious, *the node* (e.g., low-hierarchical agent) *configured to* perform the functions recited in claims [7.6]-[7.8(b)] below.
[7.6] receive the update message from the gateway device, wherein the node is configured to be mapped to the node type,	Ikeda teaches, or at least renders obvious, [the node (e.g., low-hierarchical agent) configured to:] *receive the update message from the gateway device, wherein the node is configured to be mapped to the node type* (e.g., the identified low-hierarchical agent receives the information about updating from the high-hierarchical agent).
[7.7] determine a client, different from the input source, that has registered for updates of the live object, and	Ikeda teaches, or at least renders obvious, [the node (e.g., low-hierarchical agent) configured to:] *determine a client, different from the input source, that has registered for updates of the live object* (e.g., the agents 55 and 56 judge as to whether or not the received information about updating should be transmitted to the user terminals 5 and 6 and the user terminal 7 connected to the agents 55 and 56 respectively). *See* claim [1.5], *supra.*

[4] Requester notes that the phrase "the mapping controlling an amount of update message traffic through nodes of a routing network" recites nothing more than an intended use or intended result. Such statements should not be given patentable weight. *Boehringer Ingelheim Vetmedica, Inc. v. Schering-Plough Corp.*, 320 F.3d 1339, 1345 (Fed. Cir. 2003). However, as shown above, even if patentable weight is given to this limitation, Ikeda satisfies the limitation.

[7.8(a)] route the update message from the node to the client,	Ikeda teaches, or at least renders obvious, [the node (e.g., low-hierarchical agent) configured to:] *route the update message from the node to the client* (e.g., the low level-agent posts the information about updating to the registered user terminals). *See* claim [1.6], *supra*.
[7.8(b)] wherein the client is adapted to process the update message and to update the property of the live object.	Ikeda teaches, or at least renders obvious, [the node configured to: . . . route the update message from the node to the client] *wherein the client* (e.g., user terminal) *is adapted to process the update message and to update the property of the live object* (user terminal obtains contents of the updated information from the agents and updates the data element). *See* claim [1.7], *supra*.

4. Independent Claim 20 and Dependent Claims 21-23

Claim 20	
20[P]. An apparatus comprising:	Whether or not the preamble is limiting, Ikeda teaches, or at least renders obvious *[a]n apparatus* (e.g., a push service system). Ikeda teaches a push service system, which is an apparatus. *Ikeda* (Ex. 1004), 19:24-29. ***<u>See, e.g., Ikeda</u>*** FIG. 15 is a diagram showing the structure of the **push service system** according to the seventh embodiment of the present invention. As shown in FIG. 15, in the push service system 50, a plurality of data servers 2 to 4 and a plurality of providers 51 to 54 are connected to the Internet N. The providers 51 to 54 locally connect the agents 55 to 58. Ikeda at 19:24-29. It is an object of the present invention to provide a **push service system** and a push service processing method in which resources in a network are utilized efficiently without increasing traffic on the network, and information required by users can be obtained quickly and expandability is high. Ikeda at 1:58-62. **A push service system** is provided with data servers which are connected to an Internet and post information about updating information in storage sections to an agent, an agent which is connected to the Internet and when the information about updating is information about updating requested by registered user

	terminals, post the information about updating to the registered user terminals, and the user terminals which are connected to the Internet and receives the information about updating from the agent and access to the data servers which have posted the information about updating so as to obtain contents of the updated information. Ikeda at Abstract.
[20.1(a)] an input source device configured to provide a data representation to a client device, different from the input source, coupled to a routing network, wherein the data representation includes at least one live object that is recognizable by the client device, and that	Ikeda in view of Tsutsumitake renders obvious *an input source* (e.g., data servers) *device configured to provide a data representation* (e.g., providing a web page) *to a client device, different from the input source* (e.g., user terminals), *coupled to a routing network* (e.g., Internet), *wherein the data representation includes at least one live object that is recognizable by the client device* (e.g., web page includes an updateable element that is capable of being displayed by the client and/or recognizable by the client as being "updateable"). *See* claim [14.1(a)], *supra*.
[20.1(b)] causes the client device to determine an object identifier (ID) of the live object to register for updates of the bye object[5] with the routing network, such that registering the client device with the routing network provides client connection information to the routing network,	Ikeda in view of Tsutsumitake renders obvious *caus[ing] the client device* (e.g., user terminals) *to determine an object identifier (ID) of the live object*(e.g., Event ID or event request URL) *to register for updates of the bye object* (e.g., submit a registration request) *with the routing network, such that registering the client device with the routing network provides client connection information to the routing network* (e.g., address of the agent, and "terminal information D15 such as equipment type and the like, and another terminal information D16 held by the users such as equipment type, telephone number," and address) *to a node* (e.g., agent) *in the routing network* (e.g., coupled to the Internet). *See* claim [14.1(b)], *supra*.

[5] The applicant filed a certificate of correction, changing "of the bye" to "of the live." *'722 Patent* (Ex. 1001).

[20.2] wherein the input source device is configured to route an update message to the routing network, wherein the update message identifies the live object and contains update data for updating a property of the live object,	Ikeda in view of Tsutsumitake renders obvious *the input source device* (e.g., data servers) *is configured to route an update message to the routing network* (e.g., sends information about updating information to agents coupled to the Internet), *wherein the update message identifies the live object* (e.g., includes a data type ID associated with an updateable element) *and contains update data for updating a property of the live object* (e.g., includes data contents). *See* claim [14.2], *supra.*
[20.3] wherein a gateway device at the network is configured to identify a category of the update message based on the input source, to determine a node type to which the identified category maps, and to route the update message to a node of the node type at the routing network,	Ikeda in view of Bird (Ground 1), or Ikeda in view of Pearson (Ground 2) renders obvious this limitation. **Ground 1: Ikeda in view of Bird** Ikeda in view of Bird renders obvious *a gateway device at the network* (e.g., high-hierarchical agent coupled to the Internet) *is configured to identify a category of the update message based* (e.g., the data type represented by the information about updating is compared with the data type registered by the user terminal so that a determination is made as to whether or not the data types coincide with each other or corresponding data type exists) *on the input source* (e.g., the data type is the name of a publisher / sender, such as Ikeda's data sources), *to determine a node type to which the identified category maps* (e.g., judges as to whether or not the received information about updating should be transmitted to the low-hierarchical agents 55 and 56), *and to route the update message to a node of the node type at the routing network* (e.g., routes the information about updating to registered low-hierarchical agent 55 and/or 56). *See* claim [14.3], *supra.* **Ground 2: Ikeda in view of Pearson** Ikeda in view of Tsutsumitake renders obvious *a gateway device at the network* (e.g., high-hierarchical agent coupled to the Internet) *is configured to identify a category of the update message based* (e.g., the data type represented by the information about updating is compared with the data type registered by the user terminal so that a determination is made as to whether or not the data types coincide with each other or corresponding data type exists) *on the input source* (e.g., port number), *to determine a node type to which the identified category maps* (e.g., judges as to whether or not the received information about updating should be transmitted to the low-hierarchical agents 55 and 56), *and to route the update message to a node of the node type at the routing network* (e.g.,

	routes the information about updating to registered low-hierarchical agent 55 and/or 56). *See* claim [14.3], *supra.*
[20.4(a)] wherein the node is configured to identify the client device as a registered device and to	Ikeda in view of Tsutsumitake renders obvious *the node is configured to identify the client device as a registered device* (e.g., the agents 55 and 56 judge as to whether or not the received information about updating should be transmitted to the user terminals 5 and 6 and the user terminal 7 connected to the agents 55 and 56 respectively). *See* claim [14.4(a)], *supra.*
[20.4(b)] route the update message containing the update data to the client device, and	Ikeda in view of Tsutsumitake renders obvious *route the update message containing the update data to the client device* (e.g., the low level-agent posts the information about updating to the registered user terminals). *See* claim [14.4(b)], *supra.*
[20.5] wherein the client device is configured to process the update message upon receipt to update the property of the bye[6] object at the client device.	Ikeda in view of Tsutsumitake renders obvious *the client device* (e.g., user terminal) *is configured to process the update message upon receipt to update the property of the bye object at the client device* (e.g., user terminal obtains contents of the updated information from the agents and updates the data element). *See* claim [14.5], *supra.*

Claim 21	
21. The apparatus of claim 20, wherein the live object of the data representation is configured to cause the client device to register with a client proxy of the routing network.	Ikeda in view of Tsutsumitake and Bird (Ground 1) or Ikeda in view of Tsutsumitake and Pearson (Ground 2) renders obvious *[t]he apparatus of claim 20. See* Claim 20, *supra.* Ikeda in view of Tsutsumitake further renders obvious *the live object of the data representation is configured to cause the client device to register with a client proxy of the routing network* (e.g., the updateable element is provided in the web page, the receipt of which causes the user terminal to register with a low-level agent, which is a substitute for the user terminal). *See* claim 15, *supra.*

Claim 22	
22. The apparatus of claim 20, wherein the live	Ikeda in view of Tsutsumitake and Bird (Ground 1) or Ikeda in view of Tsutsumitake and Pearson (Ground 2) render obvious *[t]he apparatus of claim 20. See* Claim 20, *supra.* Ikeda in view of Tsutsumitake further

[6] The applicant filed a certificate of correction, changing "bye object" to "live object." *'722 Patent* (Ex. 1001).

object of the data representation is configured to cause the client device to register with the node of the routing network.	renders obvious *the live object of the data representation is configured to cause the client device to register with the node of the routing network* (e.g., the updateable element is provided in the web page, the receipt of which causes the user terminal to register with a low-level agent, which is part of the routing network). *See* claim 16, *supra.*

Claim 23	
23. The apparatus of claim 20, wherein the received data representation includes an activation module that is configured to be executed by the client device and adapted to register the live object with the routing network.	Ikeda in view of Tsutsumitake and Bird (Ground 1) or Ikeda in view of Tsutsumitake and Pearson (Ground 2) renders obvious *[t]he apparatus of claim 20. See* Claim 20, *supra.* Ikeda in view of Tsutsumitake further renders obvious *the received data representation* (e.g., web page received by user terminal) *includes an activation module that is configured to be executed by the client device* (e.g., an Applet or JavaScript program is incorporated in the web page and is executed by the client browser) *and adapted to register the live object with the routing network* (e.g., the Applet or JavaScript program performs the registration step when executed). *See* claim 17, *supra.*

5. **Independent Claim 26 and Dependent Claims 27-29**

Claim 26	
26[P]. An article of manufacture including a computer-readable storage medium having instructions stored thereon, execution of which by a computing device causes the computing device to perform operations comprising:	Whether or not the preamble is limiting, Ikeda teaches, or at least renders obvious, *[a]n article of manufacture including a computer-readable storage medium* (e.g., a recording medium, which can be read by a computer) *having instructions stored thereon* (e.g., a program for allowing a computer to execute the method is recorded on the storage medium), *execution of which by a computing device causes the computing device to perform operations* (e.g., the program allows the computer to execute the method). Ikeda teaches a push service system. *Ikeda* (Ex. 1004), 19:24-29. Ikeda teaches that preferred embodiments of the system include "a recording medium onto which a program for allowing a computer to execute the method is recorded and which can be read by the computer according to the present invention." *Ikeda* (Ex. 1004), 6:47-52. Thus, Ikeda teaches that the system includes a computer-readable storage medium (e.g., a recording medium that is read by a computer) having instructions stored

	thereon (e.g., on which a program for executing a push service is stored), execution of which by a computing device causes the computing device to perform operations (e.g., a computer executes the instructions to perform the operations of the push system). ***See, e.g., Ikeda*** Preferred embodiments of a document directory distributed management system, a method of obtaining this system, and **a recording medium onto which a program for allowing a computer to execute the method is recorded and which can be read by the computer** according to the present invention will be explained below with reference to the drawings. Ikeda at 6:47-52. FIG. 15 is a diagram showing the structure of the **push service system** according to the seventh embodiment of the present invention. As shown in FIG. 15, in the push service system 50, a plurality of data servers 2 to 4 and a plurality of providers 51 to 54 are connected to the Internet N. The providers 51 to 54 locally connect the agents 55 to 58. Ikeda at 19:24-29. It is an object of the present invention to provide a **push service system** and a push service processing method in which resources in a network are utilized efficiently without increasing traffic on the network, and information required by users can be obtained quickly and expandability is high. Ikeda at 1:58-62. **A push service system** is provided with data servers which are connected to an Internet and post information about updating information in storage sections to an agent, an agent which is connected to the Internet and when the information about updating is information about updating requested by registered user terminals, post the information about updating to the registered user terminals, and the user terminals which are connected to the Internet and receives the information about updating from the agent and access to the data servers which have posted the information about updating so as to obtain contents of the updated information. Ikeda at Abstract.
[26.1(a)] providing, using a processing	Ikeda in view of Tsutsumitake renders obvious *providing, using a processing device of an input source* (e.g., processing section of data

device of an input source, a data representation to a client device, different from the input source, coupled to a routing network, wherein the data representation includes at least one live object that is recognizable by the client device, and that	servers), *a data representation* (e.g., providing a web page) *to a client device, different from the input source* (e.g., user terminals), *coupled to a routing network* (e.g., Internet), *wherein the data representation includes at least one live object that is recognizable by the client device* (e.g., web page includes an updateable element that is capable of being displayed by the client and/or recognizable by the client as being "updateable"). *See* claim [14.1(a)], *supra.*
[26.1(b)] causes the client device to respond to the live object by determining an object identifier (ID) of the live object to register for updates of the live object with the routing network, such that registering the client device with the routing network provides client connection information to the routing network; and	Ikeda in view of Tsutsumitake renders obvious *caus[ing] the client device* (e.g., user terminals) *to respond to the live object by determining an object identifier (ID) of the live object* (e.g., Event ID or event request URL) *to register for updates of the live object with the routing network* (e.g., submit a registration request), *such that registering the client device with the routing network provides client connection information to the routing network* (e.g., address of the agent, and "terminal information D15 such as equipment type and the like, and another terminal information D16 held by the users such as equipment type, telephone number," and address) *to a node* (e.g., agent) *in the routing network* (e.g., coupled to the Internet). *See* claim [14.1(b)], *supra.*
[26.2] sending, using the processing device of the input source, an update message to the routing network, wherein the update message identifies the live	Ikeda in view of Tsutsumitake renders obvious *sending, using the processing device of the input source* (e.g., processing section of a data server), *an update message to the routing network* (e.g., sends information about updating information to agents coupled to the Internet), *wherein the update message identifies the live object* (e.g., includes a data type ID associated with an updateable element) *and contains update data for updating a property of the live object* (e.g., includes data contents). *See* claim [14.2], *supra.*

object and contains update data for updating a property of the live object,	
[26.3] wherein a gateway device at the routing network is configured to identify a category of the update message based on the input source, to determine a node type to which the identified category maps, and to route the update message to a node of the node type at the routing network,	Ikeda in view of Tsutsumitake and Bird (Ground 1), or Ikeda in view of Tsutsumitake and Pearson (Ground 2) render obvious this limitation. **<u>Ground 1</u>** Ikeda in view of Tsutsumitake and Bird render obvious *a gateway device at the routing network* (e.g., high-hierarchical agent coupled to the Internet) *is configured to identify a category of the update message* (e.g., the data type represented by the information about updating is compared with the data type registered by the user terminal so that a determination is made as to whether or not the data types coincide with each other or corresponding data type exists) *based on the input source* (e.g., the data type is the name of a publisher / sender, such as Ikeda's data sources), *to determine a node type to which the identified category maps* (e.g., judges as to whether or not the received information about updating should be transmitted to the low-hierarchical agents 55 and 56), *and to route the update message to a node of the node type at the routing network* (e.g., routes the information about updating to registered low-hierarchical agent 55 and/or 56). *See* claim [14.3], *supra.* **<u>Ground 2</u>** Ikeda in view of Tsutsumitake and Pearson render obvious *a gateway device at the routing network* (e.g., high-hierarchical agent coupled to the Internet) *is configured to identify a category of the update message* (e.g., the data type represented by the information about updating is compared with the data type registered by the user terminal so that a determination is made as to whether or not the data types coincide with each other or corresponding data type exists) *based on the input source* (e.g., port number), *to determine a node type to which the identified category maps* (e.g., judges as to whether or not the received information about updating should be transmitted to the low-hierarchical agents 55 and 56), *and to route the update message to a node of the node type at the routing network* (e.g., routes the information about updating to registered low-hierarchical agent 55 and/or 56). *See* claim [14.3], *supra.*
[26.4(a)] wherein the node is configured to identify the client device as a registered device and to	Ikeda in view of Tsutsumitake renders obvious *the node is configured to identify the client device as a registered device* (e.g., the agents 55 and 56 judge as to whether or not the received information about updating should be transmitted to the user terminals 5 and 6 and the user terminal 7 connected to the agents 55 and 56 respectively). *See* claim [14.4(a)], *supra.*

[26.4(b)] route the update message to the client device, and	Ikeda in view of Tsutsumitake renders obvious [the node is configured . . . to] *route the update message to the client device* (e.g., the low level-agent posts the information about updating to the registered user terminals). *See* claim [14.4(b)], *supra.*
[26.5] wherein the client device is configured to process the update message upon receipt to update the property of the live object at the client device.	Ikeda in view of Tsutsumitake renders obvious *the client device* (e.g., user terminal) *is configured to process the update message upon receipt to update the property of the live object at the client device* (e.g., user terminal obtains contents of the updated information from the agents and updates the data element). *See* claim [14.5], *supra.*

Claim 27	
27. The article of manufacture of claim 26, wherein the live object of the data representation causes the client device to register with a client proxy of the routing network.	Ikeda in view of Tsutsumitake and Bird (Ground 1) or Ikeda in view of Tsutsumitake and Pearson (Ground 2) render obvious *[t]he article of manufacture of claim 26. See* claim 26, *supra.* Ikeda in view of Tsutsumitake further renders obvious *the live object of the data representation causes the client device to register with a client proxy of the routing network* (e.g., the updateable element is provided in the web page, the receipt of which causes the user terminal to register with a low-level agent, which is a substitute for the user terminal). *See* claim 15, *supra.*

Claim 28	
28. The article of manufacture of claim 26, wherein the live object of the data representation causes the client device to register with the node of the routing network.	Ikeda in view of Tsutsumitake and Bird (Ground 1) or Ikeda in view of Tsutsumitake and Pearson (Ground 2) render obvious *[t]he article of manufacture of claim 26. See* claim 26, *supra.* Ikeda in view of Tsutsumitake further renders obvious *the live object of the data representation causes the client device to register with the node of the routing network* (e.g., the updateable element is provided in the web page, the receipt of which causes the user terminal to register with a low-level agent, which is part of the routing network). *See* claim 16, *supra.*

Claim 29	
29. The article of manufacture of	Ikeda in view of Tsutsumitake and Bird (Ground 1) or Ikeda in view of Tsutsumitake and Pearson (Ground 2) render obvious *[t]he article of*

claim 26, wherein the received data representation includes an activation module executed by the client device and adapted to register the live object with the routing network.	*manufacture of claim 26. See* claim 26, *supra.* Ikeda in view of Tsutsumitake further renders obvious *the received data representation includes an activation module executed by the client device* (e.g., an Applet or JavaScript program is incorporated in the web page and is executed by the client browser) *and adapted to register the live object with the routing network* (e.g., the Applet or JavaScript program performs the registration step when executed). *See* claim 17, *supra.*

6. Independent Claim 32 and Dependent Claims 33-35

Claim 32	
32[P]. A non-transitory computer readable storage medium having instructions stored thereon, the instructions comprising:	Whether or not the preamble is limiting, Ikeda teaches, or at least renders obvious, *[a] non-transitory computer readable storage medium having instructions stored thereon* (e.g., a recording medium onto which a program for allowing a computer to execute the method is recorded and which can be read by the computer). Ikeda teaches that preferred embodiments of the system include "a recording medium onto which a program for allowing a computer to execute the method is recorded and which can be read by the computer according to the present invention." *Ikeda* (Ex. 1004), 6:47-52. Thus, because Ikeda teaches a recording medium that stores instructions that are executed by a computer, Ikeda teaches a non-transitory computer readable storage medium having instructions stored thereon. ***<u>See, e.g., Ikeda</u>*** Preferred embodiments of a document directory distributed management system, a method of obtaining this system, and **a recording medium onto which a program for allowing a computer to execute the method is recorded and which can be read by the computer** according to the present invention will be explained below with reference to the drawings. Ikeda at 6:47-52.
[32.1(a)] instructions for providing a data representation to a client device	Ikeda in view of Tsutsumitake renders obvious *instructions* (e.g., software executed by a processing section of a data server) *for providing a data representation* (e.g., providing a web page) *to a client device* (e.g., user terminals) *coupled to a routing network* (e.g., Internet), *wherein the data representation includes at least one live object recognizable by the client*

coupled to a routing network, wherein the data representation includes at least one live object recognizable by the client device, and	*device* (e.g., web page includes an updateable element that is capable of being displayed by the client and/or recognizable by the client as being "updateable"). *See* claim [14.1(a)], *supra.*
[32.1(b)] wherein the client device is configured to respond to the live object of the data representation by determining an object identifier (ID) of the live object to register for updates of the live object with the routing network, such that registering the client device with the routing, network provides client connection information to the routing network; and	Ikeda in view of Tsutsumitake renders obvious *the client device* (e.g., user terminals) *is configured to respond to the live object of the data representation by determining an object identifier (ID) of the live object* (e.g., Event ID or event request URL) *to register for updates of the live object with the routing network* (e.g., submit a registration request), *such that registering the client device with the routing, network provides client connection information to the routing network* (e.g., address of the agent, and "terminal information D15 such as equipment type and the like, and another terminal information D16 held by the users such as equipment type, telephone number," and address) *to a node* (e.g., agent) *in the routing network* (e.g., coupled to the Internet). *See* claim [14.1(b)], *supra.*
[32.2] instructions for providing, using a processing device of an input source, different from the client device, an update message to the routing network, wherein the update message identifies the live object and contains update data for	Ikeda in view of Tsutsumitake renders obvious *instructions for providing, using a processing device of an input source* (e.g., software executed by a processing section of a data server), *different from the client device* (e.g., user terminal), *an update message to the routing network* (e.g., sends information about updating information to agents coupled to the Internet), *wherein the update message identifies the live object* (e.g., includes a data type ID associated with an updateable element) *and contains update data for updating a property of the live object* (e.g., includes data contents). *See* claim [14.2], *supra.*

updating a property of the live object,	
[32.3] wherein a gateway device at the routing network is configured to identify a category of the update message based on the input source, to determine a node type to which the identified category maps, and to route the update message to a node of the node type at the routing network,	Ikeda in view of Tsutsumitake and Bird (Ground 1), or Ikeda in view of Tsutsumitake and Pearson (Ground 2) render obvious this limitation. **<u>Ground 1</u>** Ikeda in view of Tsutsumitake and Bird render obvious *a gateway device at the routing network* (e.g., high-hierarchical agent coupled to the Internet) *is configured to identify a category of the update message* (e.g., the data type represented by the information about updating is compared with the data type registered by the user terminal so that a determination is made as to whether or not the data types coincide with each other or corresponding data type exists) *based on the input source* (e.g., the data type is the name of a publisher / sender, such as Ikeda's data sources)*, to determine a node type to which the identified category maps* (e.g., judges as to whether or not the received information about updating should be transmitted to the low-hierarchical agents 55 and 56)*, and to route the update message to a node of the node type at the routing network* (e.g., routes the information about updating to registered low-hierarchical agent 55 and/or 56). *See* claim [14.3], *supra.* **<u>Ground 2</u>** Ikeda in view of Tsutsumitake and Pearson render obvious *a gateway device at the routing network* (e.g., high-hierarchical agent coupled to the Internet) *is configured to identify a category of the update message* (e.g., the data type represented by the information about updating is compared with the data type registered by the user terminal so that a determination is made as to whether or not the data types coincide with each other or corresponding data type exists) *based on the input source* (e.g., port number)*, to determine a node type to which the identified category maps* (e.g., judges as to whether or not the received information about updating should be transmitted to the low-hierarchical agents 55 and 56)*, and to route the update message to a node of the node type at the routing network* (e.g., routes the information about updating to registered low-hierarchical agent 55 and/or 56). *See* claim [14.3], *supra.*
[32.4(a)] wherein the node is configured to identify the client device as a registered device and to	Ikeda in view of Tsutsumitake renders obvious *the node is configured to identify the client device as a registered device* (e.g., the agents 55 and 56 judge as to whether or not the received information about updating should be transmitted to the user terminals 5 and 6 and the user terminal 7 connected to the agents 55 and 56 respectively). *See* claim [14.4(a)], *supra.*

[32.4(b)] route the update message containing the update data to the client device, and	Ikeda in view of Tsutsumitake renders obvious [the node is configured . . . to] *route the update message containing the update data to the client device* (e.g., the low level-agent posts the information about updating to the registered user terminals). *See* claim [14.4(b)], *supra.*
[32.5] wherein the client device is configured to process the update message upon receipt to update the property of the live object at the client device.	Ikeda in view of Tsutsumitake renders obvious *the client device* (e.g., user terminal) *is configured to process the update message upon receipt to update the property of the live object at the client device* (e.g., user terminal obtains contents of the updated information from the agents and updates the data element). *See* claim [14.5], *supra.*

Claim 33	
33. The non-transitory computer readable storage medium of claim 32, wherein the live object of the data representation causes the client device to register with a client proxy of the routing network.	Ikeda in view of Tsutsumitake and Bird (Ground 1) or Ikeda in view of Tsutsumitake and Pearson (Ground 2) render obvious *[t]he non-transitory computer readable storage medium of claim 32. See* claim 32, *supra.* Ikeda in view of Tsutsumitake further renders obvious *the live object of the data representation causes the client device to register with a client proxy of the routing network* (e.g., the updateable element is provided in the web page, the receipt of which causes the user terminal to register with a low-level agent, which is a substitute for the user terminal). *See* claim 15, *supra.*

Claim 34	
34. The non-transitory computer readable storage medium of claim 32, wherein the live object of the data representation causes the client device to register with the node of the routing network.	Ikeda in view of Tsutsumitake and Bird (Ground 1) or Ikeda in view of Tsutsumitake and Pearson (Ground 2) render obvious *[t]he non-transitory computer readable storage medium of claim 32. See* claim 32, *supra.* Ikeda in view of Tsutsumitake further renders obvious *the live object of the data representation causes the client device to register with the node of the routing network* (e.g., the updateable element is provided in the web page, the receipt of which causes the user terminal to register with a low-level agent, which is part of the routing network). *See* claim 16, *supra.*

Claim 35	
35. The non-transitory computer readable storage medium of claim 32, wherein the data representation includes an activation module that is executed by the client device and that is adapted to register the live object with the routing network.	Ikeda in view of Tsutsumitake and Bird (Ground 1) or Ikeda in view of Tsutsumitake and Pearson (Ground 2) render obvious *[t]he non-transitory computer readable storage medium of claim 32*. *See* claim 32, *supra*. Ikeda in view of Tsutsumitake further renders obvious *the data representation includes an activation module that is executed by the client device* (e.g., an Applet or JavaScript program is incorporated in the web page and is executed by the client browser) *and that is adapted to register the live object with the routing network* (e.g., the Applet or JavaScript program performs the registration step when executed). *See* claim 17, *supra*.