On a Web Mail System based on Web Agents

Tadachika Ozono[†], *Toramatsu Shintani*[†], *and Yujiro Fukagaya*[†]

Computer Science and Engineering, Graduate School of Engineering
Nagoya Institute of Technology
Gokiso-cho, Showa-ku, Nagoya 466-8555 JAPAN

Summary
In this paper, we propose an intelligent notification of a new e-mail, and describe an implementation of the notification system. There are two ways to read e-mails by using a computer. One way is a mail system on a user's personal computer. The other way is a web mail system on a web server. Using a web mail system is better for users because they can read e-mails by using other person's personal computers. A web mail system has a problem. A web mail system is low immediacy. Users have to look at a web page of a web mail system again and again to check new e-mails. To solve this problem, we implemented a web mail system WisdomMail. WisdomMail is a web application based on a web agent. WisdomMail has a function that shows a notification of new e-mails without special plug-ins in a web browser. This system uses the web agent framework MiSpider. MiSpider enables developers to implement a web agent having a persistent function, a message passing function, and a graphical user interface. By using MiSpider, a notification of a new e-mail appears on a web page that a user is browsing. A feature of this system is that a notification of a new e-mail appears on any web pages. Also, the notification system automatically adjusts notification timing and a position of a notification on a web page. By using the notification system, users can read new e-mails without looking at a web page of a web mail system. Finally, we evaluated the scalability of the notification system and show experimental results.

Key words:
E-Mail, Web mail system, Web agent

1. Introduction

In this paper, we propose a web mail system that shows an intelligent notification of a new e-mail. E-mail is one of the most successful computer applications [11]. There are two ways to check and read e-mails by using a computer. One way is a mail system (e.g., Microsoft Outlook [8], Mail.app [3]) on a user's personal computer. The other way is by using a web mail system (e.g., MSN Hotmail [9], Yahoo! Mail [12]) on a web server. Using a web mail system is better for users because they can check and read e-mails by using any person's computer with no configuration. However, a web mail system has a problem. A web mail system is low immediacy. Users have to look at a web page of a web mail system again and again to check and read new e-mails.

Manuscript received May 5, 2006.
Manuscript revised May 25, 2006.

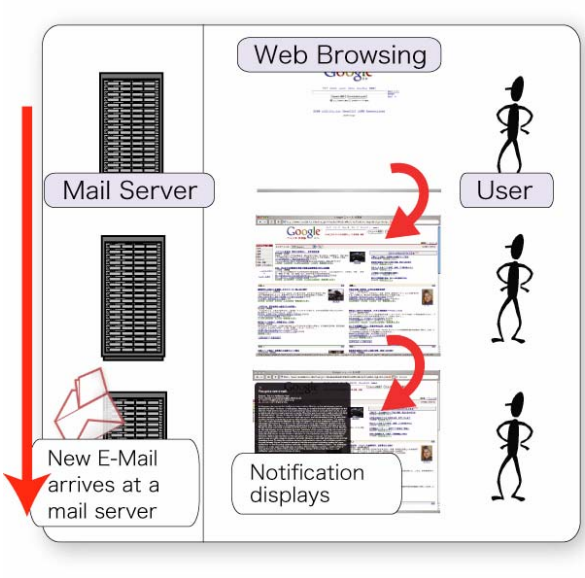

Fig. 1 A notification of new e-mail

To solve this problem, we implemented a web mail system WisdomMail. WisdomMail is a web application based on a web agent. WisdomMail has a function that shows a notification of new e-mails without special plug-ins in a web browser. The notification of a new e-mail is displayed on a web page, when a new e-mail arrived at a mail server. Figure 1 shows the snapshot of a notification of a new e-mail. The notification is displayed on a web page. The web page is formed from three frames. Users can configure the position that the notification displays. (a) The notification is displayed on the largest frame of the web page. In this snapshot, the notification displays at the fixing position (top, left) = (10px, 10px) .The notification contains a subject, a sender, a receiver, and a part of the content of the e-mail. And the notification contains an entry field. Users can easily reply to the new e-mail by inputting the entry field.

The notification system has five features as follow:

1. The notification system shows a notification of a new e-mail on any web pages without a reload operation. The notification system does not show a notification on

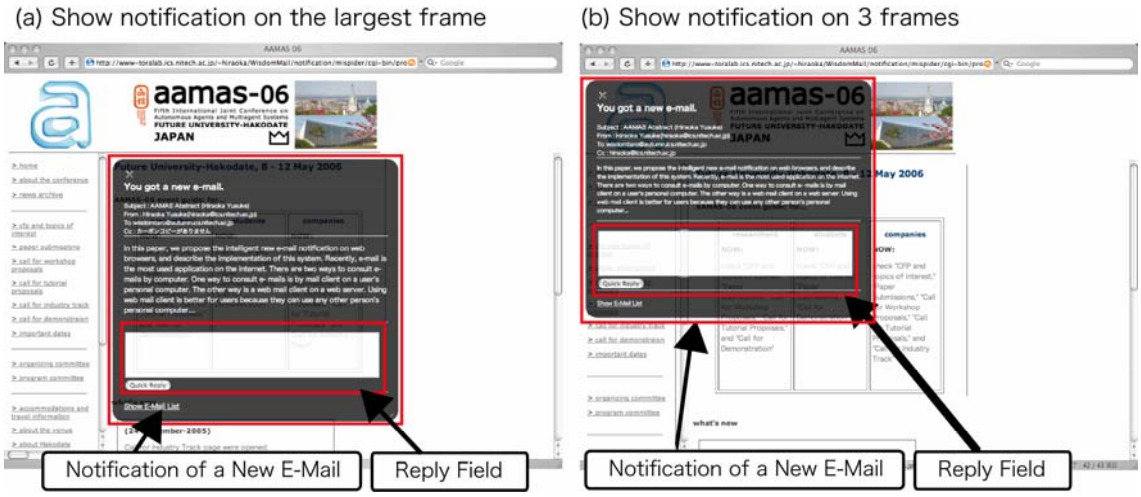

Fig. 2 A snapshot of a notification of a new e-mail on a web page.

a status bar of a web browser or a user's desktop, as existing systems. The notification system displays a notification on web pages, and a notification provides more information than existing systems.

2. Users can reply to a new e-mail by using an entry field of a notification easily.

3. The notification system automatically adjusts a notification area. For example, a notification is displayed on the empty area of a web page.

4. The notification system automatically adjusts notification timing. For example, the notification system does not show a notification while user scrolls of a web page, and inputs text to the entry field.

5. The new e-mail notification method operates without special plug-ins.

Users usually need to reload a web browser by using a reload button or a hot key to update information on a web browser. Recently, a service with an interface that updates information on a web page without a reload operation by using JavaScript has attracted creators of a web service. The implementation method of such a service is called Ajax [2]. Ajax uses an HTTP connection of JavaScript. This HTTP connection method enables a web service to exchange data between a web page on a web browser and a server without a reload operation of the web browser. By using Ajax, developers can create an interactive web service without a reload operation. Concretely, the Ajax method uses DHTML (HTML, JavaScript, and CSS) and a Java Script Object called XMLHttpRequest. However, by using the Ajax method, a developer needs to modify existing HTML files. It is difficult to implement feature 1: "The system displays a notification of a new e-mail on any web pages being viewed." We need a framework to implement a new e-mail notification system that is not independent of existing HTML files.

The notification system is implemented by using a web agent framework MiSpider [4] MiSpider enables developers to implement services without a reload operation on a web browser. Developers doesn't need change existing HTML files. MiSpider implements a persistent agent that has a message passing function on web pages. Information of MiSpider is never initialized by moving web pages. MiSpider is realized by using JavaScript.

We implemented a new e-mail notification method by using a communication method between a mail agent on a mail server and a MiSpider agent on a user's web browser. A mail agent conveys information of a new e-mail to a MiSpider agent when a new e-mail arrives at the mail server. A mail agent and a MiSpider agent communicate to adjust timing and an area that a notification displays. Thus, this new e-mail notification method doesn't obstruct user's web browsing.

This paper consists of seven sections. Section 2 shows the outline of MiSpider. Section 3 shows the outline of a web mail system called WisdomMail. In section 4, we describe the agent on the web browser and show an implementation method of the new e-mail notification. In section 5, we show experimental result of the system. Finally, we discuss features of the system in section 6 and show a conclusion in section 7.

2. A Web agent framework: MiSpider

2.1 An outline of MiSpider

MiSpider is the web agent framework that enables developers to implement a web agent having a persistent function, a message passing function, and a graphical user interface. Developers can implement web services based on agents by using the provided APIs.

The MiSpider service can login with the name of an agent that provides a service, a password, and a URL of a web page. By operating the above method, a user can use a provided service in web browsing.

To describe MiSpider's processes, developers use JavaScript. JavaScript enables developers to implement a process that responds to the operations of users. For example, MiSpider can handle mouse click event, scrolling event, and inputting event. MiSpider can process according to in these cases of the event. And MiSpider can show windows, images, and texts on the web page. Thus, JavaScript is suitable for agent implementation on a web browser.

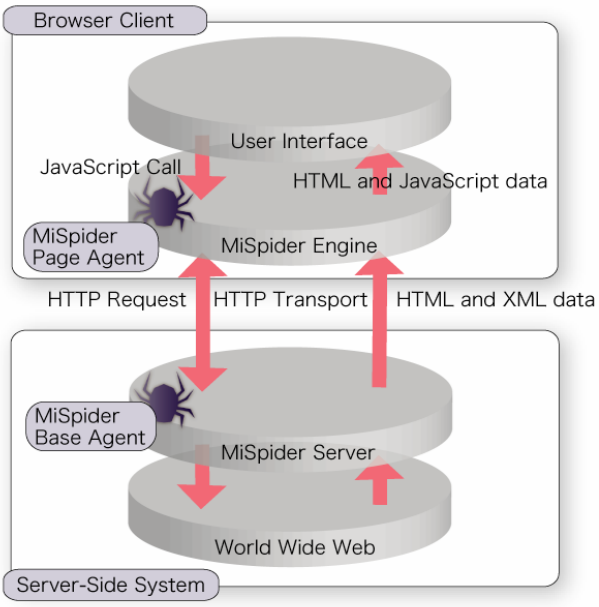

Fig. 3 MiSpider Architecture

Figure 3 shows an outline of MiSpider. MiSpider is comprised of a MiSpider Base Agent and a MiSpider Page Agent. A MiSpider Page Agent is an agent operating on a user's web browser. A MiSpider Base Agent is an agent operating on a web server. A MiSpider Base Agent communicates with a MiSpider Page Agent by using an HTTP connection by using an XMLHttpRequest object. A MiSpider Base Agent enables a MiSpider Page Agent to pass messages etc, by communicating between these two agents.

Users access a web page via a web proxy of MiSpider (implemented as CGI program) by using a web browser. A MiSpider Base Agent on a web proxy adds a MiSpider Page Agent to the web pages when these web pages pass the web proxy. A MiSpider Page Agent is implemented by using JavaScript. A MiSpider base agent adds a MiSpider Page Agent to web pages as an external JavaScript program.

Developers can specify that a service uses multiple MiSpider Page Agents to one MiSpider Base Agent. Developers can reduce CPU time, required memory capacity, and needed disk space when developers specify multiple MiSpider Page Agents to one MiSpider Base Agent. However, the response time of the MiSpider service is deteriorated by specifying multiple MiSpider Page Agent to one MiSpider Base Agent.

2.2 Implementation of MiSpider

Figure 4 shows the implementation of MiSpider framework. MiSpider framework consists of a JavaScript script file and a CGI program.

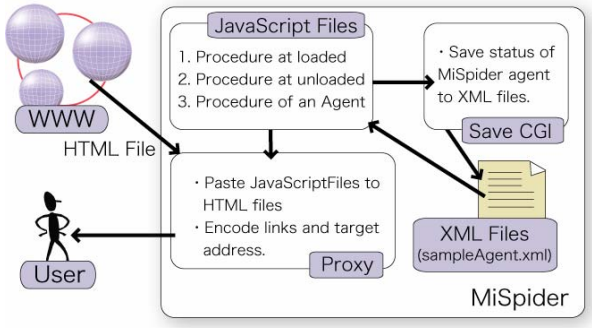

Fig. 4 Implementation of MiSpider

Users access web pages via a web proxy implemented as CGI. This web proxy adds a MiSpider JavaScript script file to HTML file. The web proxy rewrites link addresses of the web pages to link addresses attached an address of this web proxy. The web proxy also rewrites the HTML tag, for example, it rewrites `<FRAME>` to `<IFRAME>`. The added JavaScript contains three descriptions of processes as follow:

1. A description of a process when the user's web browser loads the web page.
2. A description of a process when the user moves to another web page.

3. A description of a process executed by a MiSpider Page Agent.

In the description 1., a MiSpider Page Agent (for example, agent name: sampleAgent) consults agent information by consulting a corresponding XML file (sampleAgent.xml). A MiSpider Page Agent converts the XML source to an object of the JavaScript file. In the description 3., a MiSpider Page Agent processes described action. In the description 2., a MiSpider Page Agent convert the current agent information to an XML source and sends a query to the CGI program to save the agent information on the XML file. This CGI program writes a received XML source to the XML file (Agent.xml). The persistence mechanism of a MiSpider Page Agent is implemented based on 1. and 2..

2.3 Message passing

A MiSpider Page Agents enrich web services by using message passing among agents on the World Wide Web. A MiSpider Page Agent can connect to any agent on the World Wide Web.

There is one problem when implementing message passing. In usual web services, a JavaScript program on a web browser needs a reload operation to connect to a server. A reload operation initializes an execution status of a JavaScript program. To solve this problem, a MiSpider Page Agent uses XMLHttpRequest object for a connection method without reload operation.

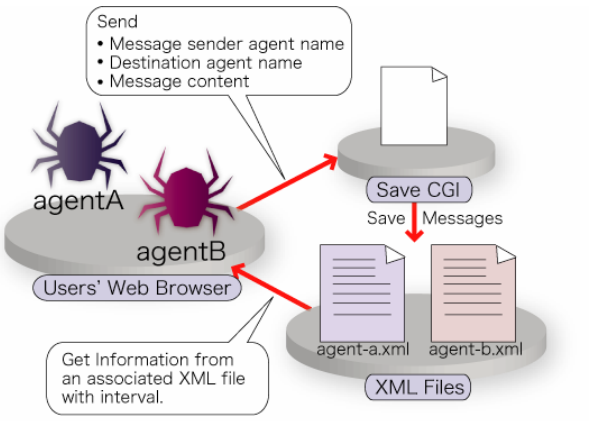

Fig. 5 Message passing method on MiSpider

Figure 5 shows the implementation method of message passing. When agentA sends a message to agentB, agentA sends the name of agentB and the message content to the CGI program. This CGI program saves the message content to XML file of agentB (agent-B.xml). Each agent accesses corresponding XML file periodically and renews agent information. AgentB gets the message from agentA by accessing the corresponding XML file(agent-B.xml).

2.4 Persistence

To make agents to process continuously via moving web pages, the MiSpider Page Agent saves agent information when unloading web pages and gets agent information when loading new web pages.

Description by using JavaScript

```
1 kbase.hoge_site.items[0].title = "item0";
  kbase.hoge_site.items[0].url =
        "http://www.hoge.co.jp/item0.html";
2 kbase.hoge_site.items[1].title = "item1";
  kbase.hoge_site.items[1].url =
        "http://www.hoge.co.jp/item1.html";
```

Description by using XML

```
<kbase>
 <knowledge>
  <name>hoge_site</name>
  <items>
1  <item>
    <title>item0</title>
    <url>http://www.hoge.co.jp/item0.html</url>
   </item>
2  <item>
    <title>item1</title>
    <url>http://www.hoge.co.jp/item1.html</url>
   </item>
  </items>
 </knowledge>
</kbase>
```

Fig. 6 Example of JavaScript object data description on MiSpider XML

The concrete technique is shown below. The MiSpider Page Agent accesses the corresponding XML file when a web browser loads a web page. The MiSpider Page Agent parses the XML source and converts the XML source to a JavaScript object "`kbase`." The MiSpider Page Agent converts the `kbase` to an XML source and sends the XML source to CGI. By using XML, developers can express JavaScript objects with a tree structure of XML.XML nodes, expressed as a singular tag name, expresses a JavaScript array objects. XML nodes, expressed as a plural tag name, express JavaScript objects. A leaf of the XML tree is treated as a string value. Figure 6 shows an example of correspondence between a JavaScript object and XML element. This example expresses the first and second elements of the JavaScript object "`kbase.hoge_site`".

Concretely, the enclosed XML element (1) rectangle corresponds with enclosed JavaScript object (1). Enclosed XML element (2) rectangle corresponds with enclosed JavaScript object (2).

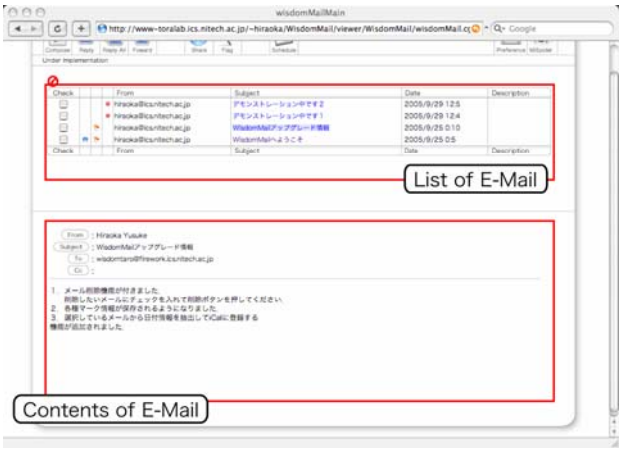

Fig. 7 Snapshot of web e-mail system WisdomMail

3. Web mail system: Wisdom Mail

Figure 7 shows a snapshot of a web mail system called WisdomMail. Users can read a list of e-mail, and contents of e-mail by using this e-mail list page. WisdomMail provides function that sends e-mail, classifies e-mails by using rules, and generates rules for classification automatically.

This system automatically renews an e-mail list without a reload operation, when a user receives new e-mails by using Ajax.

This system uses web proxy with a user ID and a password. To browse a web page, users connect to the web proxy. The proxy provides a web page with a MiSpider Page Agent named wisdomMail. Users can access the agent wisdomMail from any web page. Users can move to the web page of this web mail system by clicking on the agent wisdomMail on the web page.

Also, the wisdomMail agent provides a new e-mail notification function while users are browsing the web. Figure 2 shows a snapshot of this new e-mail notification. The notification is displayed at a fixing position (top, left) = (10px, 10px) on a consulted a web page. This notification contains a subject, a sender, a receiver, and a part of a content of the e-mail, and an entry field to reply.

3. E-mail processing agent

3.1 Architecture of e-mail processing agent

Figure 8 shows the architecture of e-mail processing agents on the new e-mail notification system. An e-mail processing agent is comprised of a mail agent on a mail server and a MiSpider Page Agent on a user's web browser. A mail agent communicates with a MiSpider Page Agent to show a notification. Each agent has different features on ability and event handling.

Features of a mail agent: Since a mail agent processes on a mail server, the mail agent can process with a fast processor and much memory. The mail agent also handles an event when a new e-mail arrives at the mail server. However, the mail agent needs to connect to the MiSpider Page Agent to get information on the user's web browser, such as position of an image.

Features of a MiSpider Page Agent: A MiSpider Page Agent processes on a user's web browser. Thus, the MiSpider Page Agent cannot process with a fast processor and much memory. On the other hand, the MiSpider Page Agent can handle information on the web browser in real time, such as mouse clicks, input strings on forms, web page scrolls, position of all object (images, strings forms, movies, etc.), and so on. Thus, the MiSpider Page Agent can get empty space on a web page by using position information. Also, the notification system is implemented as a function of a MiSpider Page Agent, so the MiSpider Page Agent can decide notification timing.

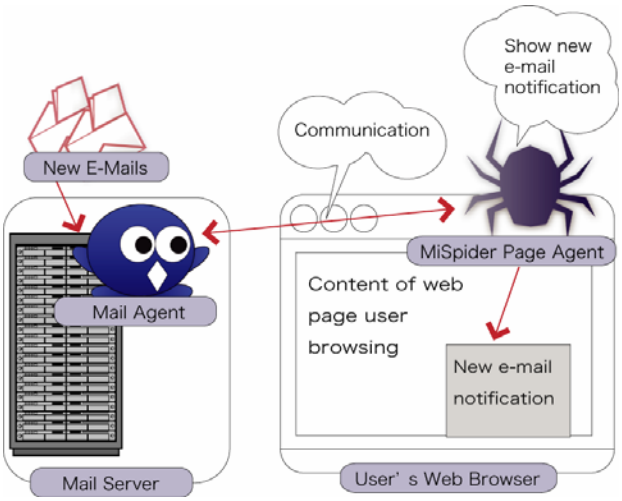

Fig. 8 Architecture of e-mail processing agent on the notification system

3.2 A new e-mail notification based on collaboration

The purpose of the e-mail notification system is to display many parts of an e-mail without blocking web browsing. To implement the notification system with such conflicting purposes, we set different utility functions to mail agent and a MiSpider Page Agent.

We considered the features of each agent to set utility functions. A mail agent's utility function has more utility value by using a larger area and a faster notification. A MiSpider agent's utility function has more utility value, without jamming to user browsing.

We show the utility function of a mail agent as follows:

$$\begin{aligned} f_{mailagent} &= quantity + timespan \\ 0 &\le quantity \le 1 \\ 0 &\le timespan \le 1 \end{aligned} \quad (1)$$

Value *timespan* is a linear monotone function decreases by the time from receiving a new e-mail to a notification. Also, value *quantity* is a linear monotone function increases by area (pt^2) of a notification.

We show the utility function of a MiSpider Page Agent as follow:

$$\begin{aligned} f_{mispider} &= event + invisible \\ event &= \{0,1\} \\ 0 &\le invisible \le 1 \end{aligned} \quad (2)$$

Value *event* is a function determined by a user's event. In user inputting to form, scrolling web pages, and dragging a mouse, *event* is 0. In other cases, *event* is 1. Value *invisible* is a linear monotone function that decreases by the number of hidden objects by a notification.

Next, we describe the flow of notification decisions of new e-mail.

1. A mail agent receives a new e-mail when the new e-mail arrived at a mail server.
2. A mail agent generates an e-mail summary (summary rate is over 0% and below 90% at 10% intervals). And, a mail agent calculates the necessary size to display each summary.
3. The mail agent sends necessity size and utility function for each summary to a MiSpider Page Agent. Figure 9 shows part of an example of XML message sent by a mail agent. This example describes the following of a new e-mail (1) ID, (2) subject, (3) sender, (4) receiver. Also, summary information is described by a format rounded by enclosed by rectangles (5) and (6). Rectangle (5) describes that the summary of content needs 20000 pt^2 area to display. Element "quantity" describes the quantity value of the utility value of the mail agent. Also, an element "timespan_gradient" describes the coefficient of the timespan function of the utility value.
4. MiSpider Page Agent keeps display area. To keep larger area, a MiSpider Page Agent search for the empty areas of a web page. To search for the empty areas of a web page, a MiSpider Page Agent gets positions and sizes of all elements on the web pages by using JavaScript.
5. The MiSpider Page Agent changes a notification area. MiSpider Page Agent shows a notification at a decided area and timing based on the sum of the utility values of the mail agent and utility value of MiSpider Page Agent, except in the case of *event* value of Eq. 2.

```
<kbase>
 <knowledge>
  <name>new_emails</name>
  <items>

   <item>
①  <identifier>new_email1124377005917</identifier>
②  <subject>A new e-mail notification</subject>
③  <from>hiraoka@ics.nitech.ac.jp</from>
④  <to>hiraoka@ics.nitech.ac.jp</to>
    <items>
⑤    <item>
       <area>20000</area>
       <content>
          This is a notification of new e-mail.....
       </content>
       <quantity>0.1</quantity>
       <timespan_gradient>-0.01</timespan_gradient>
     </item>
⑥    <item>
       <area>40000</area>
       <content>
       .........
       <item>
         ........
       </item>
       ..................................
    </items>
   </item>
   ..............................
  </items>
 </knowledge>
</kbase>
```

Fig. 9 Part of an example message that a mail agent send to a MiSpider Page Agent

Figure 10 shows a snapshot of a notification on a web using the above method. The web page has empty area in the right part of the in the right bottom frame. The notification is displayed on this empty area.

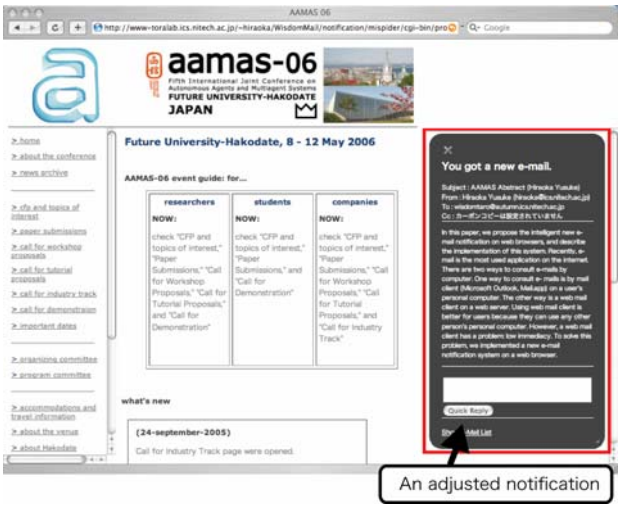

Fig. 10 Snapshot of adjusted a notification of an e-mail on a web page

5. Scalability experiments

We experimented with this system's scalability. Purpose of this experimentation is inspecting whether or not this system can offer service to many users. Concretely, we measured the span from when a mail server receives a new e-mail to the time when a notification is displayed while number of users increases.

In this experimentation, we didn't consider time to summarization process. Thus, we didn't use summarization module in this experimentation. We used this method with two MiSpider Base Agent. Half of users use the one MiSpider Base Agent and the other half of users uses the other MiSpider Base Agent. We used Safari2.0. In this experimentation, all users consulted Google web pages, and all users received a new e-mail at the same time. We measured the average time to display a notification of a new e-mail.

Figure 11 shows the experimental results. The vertical axis shows the number of users. The horizontal axis shows the measured average time. This system can displays a notification in about six seconds for 40 users. Thus, this system can easily display a notification in a community of 40 users.

Also, we did experimentation with 50 users. By using one MiSpider Base Agent, this system did not show a notification to 50 users. We solve this problem by additional MiSpider Base Agent. Two MiSpider Base Agent showed a notification to 50 users. Also, we plan the implementation of new checking algorithm. With present implementation, MiSpider Page Agent downloads the XML file at fixed interval to check new e-mails. We plan the implementation that the MiSpider Page Agent read the 1byte text file that changes by changing of XML file. With this implementation, the entire transfer quantity decreases, thus scalability of our systems improves.

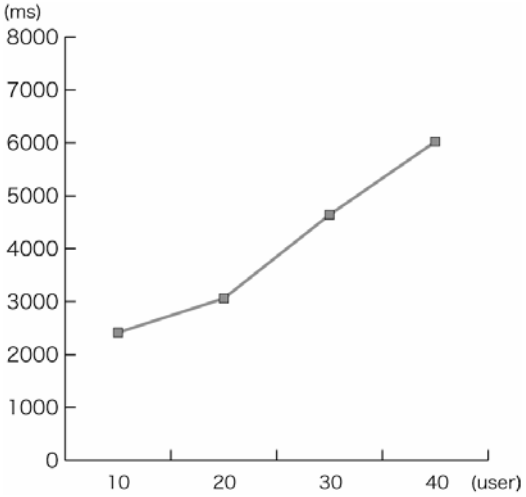

Fig. 11 Scalability experimental results

6. Discussion

We describe features of the notification system and future views as follows. The notification system is implemented by separating MiSpider Page Agent that displays a notification from a mail agent that generates contents to display. Thus, developers do not need to consider timing and an area of notification. Developers can implement a notification system for other purposes by changing agents that generate contents. For example, a notification system can inform an RSS feed on the web pages.

An e-mail has a priority value in this header. In the present implementation, we did not consider the important value of an e-mail. We will add such an importance value to the utility function of a mail agent. Then the notification system will notify an important e-mail by priority.

Table 1: Comparison among web mail systems

Function to compare	MSN Hotmail	GMail	Proposed system
1. A notification of a new e-mail	MSN Hotmail shows a notification by using MSN Messenger that needs installation.	GMail shows a notification by using an RRS feed. Users have to register address of RSS feed of GMail.	Proposed system shows a notification by using MiSpider. This system doesn't need special web browser plug-ins.
2. Content of a notification of a new e-mail	A notification contains number of new e-mails and information about a sender.	A notification contains number of new e-mails and part of content of a new e-mail.	A notification contains number of new e-mails and meta information of e-mails.
3. Area of a notification of a new e-mail	MSN Hotmail shows a notification by using a pop up window on user's desktop.	GMail shows a notification by using status bar of a web browser, when user using Safari.	Proposed system shows a notification on a web page user browsing.

A key aspect of next generation e-mail systems is to notify a user only when the incoming e-mails are relevant to the task at hand [10]. We plan that we implement the notification system that color of a notification changes based on the e-mail content. Then users can understand the kind of e-mails by color.

As follows, we describe related researches. Semantic Email [7] is a research to approve the interface of a web mail system. Semantic Email proposes a framework that has a flexible interface and is service based on a metadata attached to a sender. For example, Semantic Email can provide a service that displays event information from an event metadata of e-mail. The notification system, different from existing web mail systems, provides a new e-mail interface that displays a notification of a new e-mail on web pages being browsed. Many developers have proposed frameworks that provide interactive web services by using PUSH technology [6]. ActiveDesktop [1] shows a web browser by using push technology on a user's desktop. These systems need a special web browser, or plug-ins. The notification system, different from existing frameworks, doesn't need a special web browser, or a plug-in.

Also, Table 1 compares the notification system, MSN Hotmail, and GMail [5]. Our system does not need installation of special application software and web browser plug-ins. MSN Hotmail uses MSN Messenger to show a notification of new e-mails. Users have to install MSN Messenger on their computers. GMail uses RSS feed to show a notification of new e-mails. After registration, Users can monitor new e-mails by using a web browser with an RSS reader function. Users can use the notification system in limited computer environments such as schools, or Internet cafes. Next, we describe a content of a notification. An MSN Hotmail displays the number of new e-mails and names of senders. By using GMail RSS feed, a notification is displayed on a status bar of the web browser. Also, by using RSS reader function, users can read the contents of new e-mails. Our system displays a notification on a web page being browsed, thus the notification uses a vast area. Our system shows a notification with a subject, a sender, a carbon copy (cc), and contents of new e-mails with no moving web pages.

Conclusions

In this paper, we proposed a web mail system WisdomMail, and a notification of a new e-mails on a web page. We proposed a collaboration method among agents to adjust timing and an area of a notification of a new e-mail.

The notification system has the following five features:

1. The notification system displays a notification of a new e-mail on any web page being consulted. The notification system does not show a notification on a part of a web browser, as existing notification systems. The notification system displays a notification on the web browser, and so a notification has more information than existing systems.
2. Users can reply to a new e-mail by using an entry field of a notification.
3. The notification system automatically adjusts an area of notification.
4. The notification system automatically adjusts notification timing.
5. The notification system operates without plug-ins in a web browser.

The notification system collaboratively uses a mail agent on a mail server and a MiSpider agent on a user's web browser. The notification system reduces the disturbance of web browsing by collaboration between mail agent and a MiSpider agent.

Finally, we showed experimental results. The notification system shows a notification of a new e-mail to 40 users in about six seconds.

References

[1] Active desktop: http://www.microsoft.com/windows/ie/previous/gallery/default.mspx.
[2] Ajax: http://www.adaptivepath.com/publications/essays/archives/000385.php.
[3] Apple mail: http://www.apple.com/macosx/features/mail/.
[4] Y. Fukagaya, T. Ozono, T. Ito, and T. Shintani.: Mispider: A continuous agent on web pages. In Proceedings of the 14th International World Wide Web Conference, pages 1008-1009, May 2005.
[5] Gmail: http://mail.google.com/.
[6] T. Kapyla, I. Niemi, and A. Lehtola.: Towardds an accessible web by applying push technology, In the 4th ERCIM Workshop, User Interface for All number, 15 in Position Papers: Information Filtering and Presentation, 1998.
[7] L. McDowell, O. Etzioni, A. Halevy, and H. Levy: Semantic email. In Proceedings of the 13th International World Wide Conference, pages 244-254, 2004.
[8] Microsoft outlook: http://office.microsoft.com/ja-jp/fx01085 7931041. aspx.
[9] MSN hotmail: http://www.hotmail.co.jp/.
[10] C. Roecher, V. Bayon, M. Memisoglu, and N. Streitz., Context-dependent email notification using ambient displays and mobile devices., In Proceedings of the 2005 International Conference on Active Media Technology, pages 137-138, 2005.
[11] S. Whittaker and C. Sidner., Email overload: exploring personal information management of email., In Human factors in computing systems}, 1996.
[12] Yahoo! mail: http://mail.yahoo.com/.

Tadachika Ozono received his bachelor degree in engineering from Nagoya Institute of Technology, his master degree in engineering from Nagoya Institute of Technology, and his Ph.D in engineering from Nagoya Institute of Technology. He is a research associate of the Graduate School of Computer Science and Engineering at Nagoya Institute of Technology. His research topic is a web intelligence using multiagent and machine learning technologies.