MEMORY MANAGEMENT DEVICE, PROXY SERVER, BROWSER TERMINAL, MEMORY MANAGEMENT PROGRAM, SERVER PROGRAM, TERMINAL PROGRAM, MONITOR PROGRAM, AND MEMORY MANAGEMENT METHOD

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates to devices, servers, terminals, programs, and methods which mediate data acquisition between a network resource and another network resource. More particularly, it relates to a memory management device, proxy server, browser terminal, memory management program, server program, terminal program, monitor program, and memory management method suitable for solving the problem that a lack of memory resources can render a browser terminal inoperable if the browser terminal employs a static memory management system or a dynamic memory management scheme which may allow executable programs to run without releasing resources.

Description of the Related Art

With the dissemination of Web-based technologies, so-called client-type systems which use a Web browser as a user interface and run applications proper on a WWW (World Wide Web) server have come to be used. In such a system, information sent from the WWW server to the client is made up of HTML (HyperText Markup Language) data which contains a form and the client terminal can run applications without installing them in advance as long as the Web browser operates. Also, the client can generally operate on hardware with lower throughput than when the applications are run on the client.

Conventionally, low-end PCs were used as such client terminals. Recently, however, browser terminals have been developed for casual use at shops and the like. They consist of a touch panel mounted on a liquid crystal display and can be operated simply by using the touch panel. Web browsers can run on these browser terminals as well.

Unlike low-end PCs, such browser terminals are generally designed with low CPU throughput and limited memory resources to conserve power, costs, space, etc. Thus, when running JAVA (registered trademark) applets (hereinafter referred to simply as applets), Web browsers installed on browser terminals use a static memory management system characterized by reduced CPU loads and

great memory efficiency to manage memory areas allocated to the applets.

The static memory management system manages each Web page using a table and allocates unused memory areas to applets included in the Web page in sequence as they are read. The table is reset when an update command to reload the Web page is issued.

However, under the static memory management system, when displaying a Web page consisting of a plurality of frames, the applets included in the Web page may be read one after another without an update command depending on user actions. This may cause a lack of memory resources, rendering the browser terminal inoperable. For example, on a Web page consisting of a menu frame for listing menu items and a content frame for displaying content which corresponds to the menu items, when the user selects menu items to display their content, the content frame is updated each time a menu item is selected, but no update command to update the entire Web page is issued, and thus an applet included in the Web page is read and stored in memory each time a menu item is selected. With terminals intended for personal users such as low-end PCs, which do not display the same Web page repeatedly, it is not likely that applets will be read excessively, resulting in a lack of memory resources. On the other hand, with browser terminals installed at shops or the like, since they compose Web pages using applets and often provide services specific to the shops such as a print service, if a plurality of users display the same Web page repeatedly, it is highly likely that applets will be read excessively, resulting in a lack of memory resources.

Browser terminals use applets downloaded from a WWW server, but the applet download may fail halfway due to a communications failure. However, under the static memory management system, which allocates memory areas to applets regardless of whether a download succeeds or fails, unnecessary applets will occupy memory areas in case of a download failure. This is highly likely to cause a lack of memory resources. Furthermore, the applets left after the download failure, if executed, may cause unstable operation.

Also, before executing applets downloaded from a WWW server, browser terminals initialize them to make them available for use, but the applet initialization may fail due to user error in terminal operation or the like. However, under the static memory management system, which allocates memory areas to applets regardless of whether initialization succeeds or fails, unnecessary applets

will occupy memory areas in case of an initialization failure. This is highly likely to cause a lack of memory resources. Furthermore, the applets left after the initialization failure, if executed, may cause unstable operation.

Also, since browser terminals do not release memory areas unless an update request is issued, the memory areas remain occupied after a user ends a session until the next user begins a session. Such conditions, if accumulated, are highly likely to cause a lack of memory resources. Thus, it is desirable to release the memory areas after a user ends a session until the next user begins a session.

The present invention has been made in view of the above problems with conventional technologies. Its object is to provide a memory management device, proxy server, browser terminal, memory management program, server program, terminal program, monitor program, and memory management method suitable for solving the problem that a lack of memory resources can render a browser terminal inoperable if the browser terminal employs a static memory management system or a dynamic memory management scheme which may allow executable programs to run without releasing resources.

BRIEF SUMMARY OF THE INVENTION

[Invention 1]

To achieve the above object, invention 1 is a memory management device applied to a memory management system which comprises data acquisition means for acquiring data, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, the memory management device comprising:

acquisition failure detection means for detecting whether the data acquisition means has failed to acquire the data in the specific format; and update command output means for outputting the update command to the memory management system based on a detection result from the acquisition failure detection means.

In the memory management system with this configuration, when the data acquisition means acquires data, the memory area allocation means allocates memory areas to data in a specific format from among the

acquired data, according to a static memory management system.

On the memory management device, when the acquisition failure detection means detects that the data acquisition means has failed to acquire the data in the specific format, the update command output means outputs an update command to the memory management system based on the detection result.

When the memory management system receives the update command, the memory area releasing means releases, according to the update command, the memory areas allocated to the data in the specific format.

The memory management device here may be either integral with the memory management system or separate from the memory management system. In the former case, the memory management device can be configured, for example, as software incorporated into the memory management device. In the latter case, it can be configured, for example, as a proxy server used by a computer with the memory management system built in to access a network. This also applies to the memory management devices according to inventions 2 and 3.

The data in the specific format may be executable program data (e.g., a JAVA (registered trademark) applet), image data, voice data, other multimedia data, text data, and other document data. This also applies to the memory management devices according to inventions 2 and 3, proxy servers according to inventions 4, 5, and 7, memory management programs according to inventions 23 to 25, server programs according to inventions 26 to 28, and memory management methods according to inventions 33 to 38.

The acquisition failure detection means may have any configuration as long as it can detect that the data acquisition means has failed to acquire data in the specific format. More specifically, it can detect failures to acquire data in the specific format, by estimation or by actual measurement. For example, when the memory management device is integral with the memory management system, the failures can be detected by actual measurement and when the memory management device is separate from the memory management system, the failures can be detected by estimation because it is difficult to take actual measurements. This also applies to the memory management program according to invention 23.

[Invention 2]

Invention 2 is a memory management device applied to a memory management system which comprises data

acquisition means for acquiring data, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the data acquisition means, data initialization means for initializing the data in the specific format acquired by the data acquisition means to make it available for use, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, the memory management device comprising:

initialization failure detection means for detecting whether the data initialization means has failed to initialize the data in the specific format; and update command output means for outputting the update command to the memory management system based on a detection result from the initialization failure detection means.

In the memory management system with this configuration, when the data acquisition means acquires data, the memory area allocation means allocates memory areas to data in a specific format from among the acquired data, according to a static memory management system, and the data initialization means initializes the acquired data in the specific format to make it available for use.

On the memory management device, when the initialization failure detection means detects that the data initialization means has failed to initialize the data in the specific format, the update command output means outputs an update command to the memory management system based on the detection result.

When the memory management system receives the update command, the memory area releasing means releases, according to the update command, the memory areas allocated to the data in the specific format.

The initialization failure detection means may have any configuration as long as it can detect that the data initialization means has failed to initialize data in the specific format. More specifically, it can detect failures to initialize data in the specific format, by estimation or by actual measurement. For example, when the memory management device is integral with the memory management system, the failures can be detected by actual measurement and when the memory management device is separate from the memory management system, the failures can be detected by estimation because it is difficult to take actual measurements. This also applies to the memory management program according to invention 24.

[Invention 3]

Invention 3 is a memory management device applied to a memory management system which comprises data acquisition means for acquiring data, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, the memory management device comprising:

timeout detection means for detecting whether a service using the data in the specific format acquired by the data acquisition means has timed out; and update command output means for outputting the update command to the memory management system based on a detection result from the timeout detection means.

In the memory management system with this configuration, when the data acquisition means acquires data, the memory area allocation means allocates memory areas to data in a specific format from among the acquired data, according to a static memory management system.

On the memory management device, when the timeout detection means detects that a service using the data in the specific format acquired by the data acquisition means has timed out, the update command output means outputs an update command to the memory management system based on the detection result.

When the memory management system receives the update command, the memory area releasing means releases, according to the update command, the memory areas allocated to the data in the specific format.

The timeout detection means may have any configuration as long as it can detect that a service using the data in the specific format acquired by the data acquisition means has timed out. More specifically, it can detect service timeouts by estimation or by actual measurement. For example, when the memory management device is integral with the memory management system, the service timeouts can be detected by actual measurement and when the memory management device is separate from the memory management system, the service timeouts can be detected by estimation because it is difficult to take actual measurements. This also applies to the memory management program according to invention 25.

[Invention 4]

Invention 4 is a proxy server which communicably connects via a network to browser terminal and to network resource other than the browser terminal and which, upon receiving a data acquisition request from the browser terminal, acquires data specified by the received acquisition request from a target network resource and sends the acquired data to the browser terminal, the browser terminal comprising terminal-side data acquisition means for acquiring data from a network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the terminal-side data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, the proxy server comprising:

server-side data acquisition means for acquiring, according to the acquisition request, data from the network resource; data transmission means for sending the data acquired by the server-side data acquisition means to the browser terminal; acquisition failure detection means for detecting via communications with the browser terminal whether the terminal-side data acquisition means has failed to acquire the data in the specific format; and update command transmission means for sending the update command to the browser terminal based on a detection result from the acquisition failure detection means.

With this configuration, when the browser terminal intends to access a network resource, its terminal-side data acquisition means sends an acquisition request to the proxy server.

When the proxy server receives the acquisition request, the server-side data acquisition means acquires, according to the acquisition request, data from the network resource and the data transmission means sends the acquired data to the browser terminal.

On the browser terminal, when the terminal-side data acquisition means receives the data, the memory area allocation means allocates memory areas, according to a static memory management system, to data in a specific format from among the received data.

On the proxy server, when the acquisition failure detection means detects via communications with the browser terminal that the terminal-side data acquisition means has failed to acquire the data in the specific format, the update command output means outputs an update command to the browser terminal based on the detection

result. Incidentally, the acquisition failure can be detected, for example, when the browser terminal notifies the proxy server of the acquisition failure.

When the browser terminal receives the update command, the memory area releasing means releases, according to the update command, the memory areas allocated to the data in the specific format.

[Invention 5]

Invention 5 is a proxy server which communicably connects via a network to browser terminal and to network resource other than the browser terminal and which, upon receiving a data acquisition request from the browser terminal, acquires data specified by the received acquisition request from a target network resource and sends the acquired data to the browser terminal, the browser terminal comprising terminal-side data acquisition means for acquiring data from a network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the terminal-side data acquisition means, data initialization means for initializing the data in the specific format acquired by the terminal-side data acquisition means to make it available for use, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, the proxy server comprising:

server-side data acquisition means for acquiring, according to the acquisition request, data from the network resource; data transmission means for sending the data acquired by the server-side data acquisition means to the browser terminal; initialization failure detection means for detecting via communications with the browser terminal whether the data initialization means has failed to initialize the data in the specific format; and update command transmission means for sending the update command to the browser terminal based on a detection result from the initialization failure detection means.

With this configuration, when the browser terminal intends to access a network resource, its terminal-side data acquisition means sends an acquisition request to the proxy server.

When the proxy server receives the acquisition request, the server-side data acquisition means acquires, according to the acquisition request, data from the network resource and the data transmission means sends the acquired data to the browser terminal.

On the browser terminal, when the terminal-side data

acquisition means receives the data, the memory area allocation means allocates memory areas, according to a static memory management system, to data in the specific format from among the received data and the data initialization means initializes the acquired data in the specific format to make it available for use.

On the proxy server, when the initialization failure detection means detects via communications with the browser terminal that the data initialization means has failed to initialize the data in the specific format, the update command output means outputs an update command to the browser terminal based on the detection result. Incidentally, the initialization failure can be detected, for example, when the browser terminal notifies the proxy server of the initialization failure.

When the browser terminal receives the update command, the memory area releasing means releases, according to the update command, the memory areas allocated to the data in the specific format.

[Invention 6]

Invention 6 is the proxy server according to invention 5, further comprising:

acquisition failure detection means for detecting via communications with the browser terminal whether the terminal-side data acquisition means has failed to acquire the data in the specific format, wherein

the update command transmission means sends the update command to the browser terminal based on a detection result from the initialization failure detection means or the acquisition failure detection means.

On the proxy server with this configuration, when the acquisition failure detection means detects via communications with the browser terminal that the terminal-side data acquisition means has failed to acquire the data in the specific format, the update command output means outputs an update command to the browser terminal based on the detection result.

When the browser terminal receives the update command, the memory area releasing means releases, according to the update command, the memory areas allocated to the data in the specific format.

[Invention 7]

Invention 7 is a proxy server which communicably connects via a network to browser terminal and to network resource other than the browser terminal and which, upon receiving a data acquisition request from the browser terminal, acquires data specified by the received

acquisition request from a target network resource and sends the acquired data to the browser terminal, the browser terminal comprising terminal-side data acquisition means for acquiring data from a network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the terminal-side data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, the proxy server comprising:

server-side data acquisition means for acquiring, according to the acquisition request, data from the network resource; data transmission means for sending the data acquired by the server-side data acquisition means to the browser terminal; timeout detection means for detecting via communications with the browser terminal whether a service using the data in the specific format acquired by the terminal-side data acquisition means has timed out; and update command transmission means for sending the update command to the browser terminal based on a detection result from the timeout detection means.

With this configuration, when the browser terminal intends to access a network resource, its terminal-side data acquisition means sends an acquisition request to the proxy server.

When the proxy server receives the acquisition request, the server-side data acquisition means acquires, according to the acquisition request, data from the network resource and the data transmission means sends the acquired data to the browser terminal.

On the browser terminal, when the terminal-side data acquisition means receives the data, the memory area allocation means allocates memory areas, according to a static memory management system, to data in a specific format from among the received data.

On the proxy server, when the timeout detection means detects via communications with the browser terminal that a service using the data in the specific format acquired by the terminal-side data acquisition means has timed out, the update command output means outputs an update command to the browser terminal based on the detection result.

When the browser terminal receives the update command, the memory area releasing means releases, according to the update command, the memory areas allocated to the data in the specific format.

[Invention 8]

Invention 8 is the proxy server according to invention 7, further comprising:

acquisition failure detection means for detecting via communications with the browser terminal whether the terminal-side data acquisition means has failed to acquire the data in the specific format, wherein

the update command transmission means sends the update command to the browser terminal based on a detection result from the timeout detection means or the acquisition failure detection means.

On the proxy server with this configuration, when the acquisition failure detection means detects via communications with the browser terminal that the terminal-side data acquisition means has failed to acquire the data in the specific format, the update command output means outputs an update command to the browser terminal based on the detection result.

When the browser terminal receives the update command, the memory area releasing means releases, according to the update command, the memory areas allocated to the data in the specific format.

[Invention 9]

Invention 9 is the proxy server according to invention 7, further comprising:

initialization failure detection means for detecting via communications with the browser terminal whether the data initialization means has failed to initialize the data in the specific format, wherein

the update command transmission means sends the update command to the browser terminal based on a detection result from the timeout detection means or the initialization failure detection means.

On the proxy server with this configuration, when the initialization failure detection means detects via communications with the browser terminal that the data initialization means has failed to initialize the data in the specific format, the update command output means outputs an update command to the browser terminal based on the detection result.

When the browser terminal receives the update command, the memory area releasing means releases, according to the update command, the memory areas allocated to the data in the specific format.

[Invention 10]

Invention 10 is the proxy server according to invention 8, further comprising:

initialization failure detection means for detecting via communications with the browser terminal whether the data initialization means has failed to initialize the data in the specific format, wherein

the update command transmission means sends the update command to the browser terminal based on a detection result from the timeout detection means, the acquisition failure detection means, or the initialization failure detection means.

On the proxy server with this configuration, when the initialization failure detection means detects via communications with the browser terminal that the data initialization means has failed to initialize the data in the specific format, the update command output means outputs an update command to the browser terminal based on the detection result.

When the browser terminal receives the update command, the memory area releasing means releases, according to the update command, the memory areas allocated to the data in the specific format.

[Invention 11]

Invention 11 is the proxy server according to invention 10, further comprising:

acquisition count detection means for detecting whether an acquisition count of the data in the specific format acquired by the terminal-side data acquisition means has reached a predetermined number, wherein

the update command transmission means sends the update command to the browser terminal based on a detection result from the timeout detection means, the acquisition failure detection means, the initialization failure detection means, or the acquisition count detection means.

On the proxy server with this configuration, when the acquisition count detection means detects via communications with the browser terminal that an acquisition count of the data in the specific format acquired by the terminal-side data acquisition means has reached a predetermined number, the update command output means outputs an update command to the browser terminal based on the detection result.

When the browser terminal receives the update command, the memory area releasing means releases, according to the update command, the memory areas allocated to the data in the specific format.

The acquisition count detection means may have any configuration as long as it can detect that an acquisition count of the data in the specific format

acquired by the terminal-side data acquisition means has reached a predetermined number. More specifically, it can detect acquisition counts by estimation or by actual measurement. To detect an acquisition count by estimation, for example, the acquisition count detection means can count the number of times the data transmission means sends the data in the specific format and then detect that the result of counting has reached a predetermined number.

[Invention 12]

Invention 12 is the proxy server according to invention 10 or 11, wherein:

the data in the specific format is either executable service program data which is assigned to one of classification display areas obtained by dividing a display screen into multiple areas and composes a service program that provides a service in the given display area or executable monitor program data which is assigned to one of the classification display areas and composes a monitor program that monitors one or more service programs;

the monitor program executes processes implemented as the memory area releasing means and upon receiving a full-screen update command, updates all the classification display areas to which the service program data or the monitor program data are assigned from among the multiple classification display areas, thereby releasing the memory areas allocated to the program data; and

the update command transmission means sends the full-screen update command to the browser terminal if the acquisition failure detection means detects an acquisition failure.

With this configuration, on the browser terminal, when the terminal-side data acquisition means receives monitor program data, the memory area allocation means allocates memory areas, according to a static memory management system, to the received monitor program data and the monitor program is executed. Similarly, when the terminal-side data acquisition means receives service program data, the memory area allocation means allocates memory areas, according to a static memory management system, to the received service program data and the service program is executed.

On the proxy server, when the acquisition failure detection means detects an acquisition failure, the update command transmission means sends a full-screen update command to the browser terminal.

When the browser terminal receives the full-screen update command, the monitor program updates all the classification display areas to which the service program data or the monitor program data are assigned from among the multiple classification display areas, thereby releasing the memory areas allocated to the program data.

[Invention 13]

Invention 13 is the proxy server according to invention 10 or 11, wherein:

the data in the specific format is either executable service program data which is assigned to one of classification display areas obtained by dividing a display screen into multiple areas and composes a service program that provides a service in the given display area or executable monitor program data which is assigned to one of the classification display areas and composes a monitor program that monitors one or more service programs;

the monitor program executes processes implemented as the memory area releasing means and upon receiving a local-screen update command, updates the display area to which the monitor program data is assigned from among the multiple classification display areas, thereby releasing the memory area allocated to the monitor program data; and

the update command transmission means sends the local-screen update command to the browser terminal if the initialization failure detection means detects an initialization failure.

With this configuration, on the browser terminal, when the terminal-side data acquisition means receives monitor program data, the memory area allocation means allocates memory areas, according to a static memory management system, to the received monitor program data and the monitor program is executed. Similarly, when the terminal-side data acquisition means receives service program data, the memory area allocation means allocates memory areas, according to a static memory management system, to the received service program data and the service program is executed.

On the proxy server, when the initialization failure detection means detects an initialization failure, the update command transmission means sends a local-screen update command to the browser terminal.

When the browser terminal receives the local-screen update command, the monitor program updates the classification display areas to which the monitor program data are assigned from among the multiple classification

display areas, thereby releasing the memory areas allocated to the monitor program data.

[Invention 14]

Invention 14 is the proxy server according to invention 10 or 11, wherein:

the data in the specific format is either executable service program data which is assigned to one of classification display areas obtained by dividing a display screen into multiple areas and composes a service program that provides a service in the given display area or executable monitor program data which is assigned to one of the classification display areas and composes a monitor program that monitors one or more service programs;

the monitor program executes processes implemented as the memory area releasing means and upon receiving a local-screen update command, updates the display area to which the monitor program data is assigned from among the multiple classification display areas, thereby releasing the memory area allocated to the monitor program data; and

the update command transmission means sends the local-screen update command to the browser terminal if the timeout detection means detects a service timeout.

With this configuration, on the browser terminal, when the terminal-side data acquisition means receives monitor program data, the memory area allocation means allocates memory areas, according to a static memory management system, to the received monitor program data and the monitor program is executed. Similarly, when the terminal-side data acquisition means receives service program data, the memory area allocation means allocates memory areas, according to a static memory management system, to the received service program data and the service program is executed.

On the proxy server, when the timeout detection means detects a service timeout, the update command transmission means sends a local-screen update command to the browser terminal.

When the browser terminal receives the local-screen update command, the monitor program updates the classification display areas to which the monitor program data are assigned from among the multiple classification display areas, thereby releasing the memory areas allocated to the monitor program data.

[Invention 15]

Invention 15 is the proxy server according to invention 11, wherein:

the data in the specific format is either executable service program data which is assigned to one of classification display areas obtained by dividing a display screen into multiple areas and composes a service program that provides a service in the given display area or executable monitor program data which is assigned to one of the classification display areas and composes a monitor program that monitors one or more service programs;

the monitor program executes processes implemented as the memory area releasing means and upon receiving a local-screen update command, updates the display area to which the monitor program data is assigned from among the multiple classification display areas, thereby releasing the memory area allocated to the monitor program data; and

the update command transmission means sends the local-screen update command to the browser terminal if the acquisition count detection means detects that the predetermined number is reached.

With this configuration, on the browser terminal, when the terminal-side data acquisition means receives monitor program data, the memory area allocation means allocates memory areas, according to a static memory management system, to the received monitor program data and the monitor program is executed. Similarly, when the terminal-side data acquisition means receives service program data, the memory area allocation means allocates memory areas, according to a static memory management system, to the received service program data and the service program is executed.

On the proxy server, when the acquisition count detection means detects that the predetermined number is reached, the update command transmission means sends a local-screen update command to the browser terminal.

When the browser terminal receives the local-screen update command, the monitor program updates the classification display areas to which the monitor program data are assigned from among the multiple classification display areas, thereby releasing the memory areas allocated to the monitor program data.

[Invention 16]

Invention 16 is the proxy server according to any of inventions 4 to 15, further comprising:

specific acquisition request detection means for detecting whether the data acquired by the server-side data acquisition means contains a specific acquisition request to acquire the data in the specific format; and

data presence judging means for judging whether the data in the specific format requested by the specific acquisition request exists in a target network resource based on a detection result from the specific acquisition request detection means, wherein

if the data presence judging means judges that the data in the specific format requested by the specific acquisition request contained in the data acquired by the server-side data acquisition means does not exist, the data transmission means sends notice that the data in the specific format does not exist.

On the proxy server with this configuration, if the data acquired by the server-side data acquisition means contains a specific acquisition request, the specific acquisition request detection means detects the specific acquisition request and the data presence judging means judges whether the data in the specific format requested by the specific acquisition request exists in a target network resource, based on the detection result from the specific acquisition request detection means. If it is found that the data in the specific format does not exist, the data transmission means sends notice that the data in the specific format does not exist.

The addressee of the notice that the data in the specific format does not exist is not limited to the browser terminal, and it may be an administrative terminal which manages the browser terminal. Also, the notice that the data in the specific format does not exist may be provided as an event log.

[Invention 17]

Invention 17 is the proxy server according to invention 16, wherein:

if the data presence judging means judges that the data in the specific format requested by the specific acquisition request contained in the data acquired by the server-side data acquisition means does not exist, the data transmission means sends notification data to the effect that the data in the specific format does not exist to the browser terminal instead of sending the data in the specific format.

On the proxy server with this configuration, if it is judged that the data in the specific format does not exist, the data transmission means sends notification data to the effect that the data in the specific format does not exist to the browser terminal instead of sending the data in the specific format.

[Invention 18]

Invention 18 is the proxy server according to invention 16 or 17, wherein:

after accessing the target network resource and judging whether the data in the specific format requested by the specific acquisition request exists there, the data presence judging means registers data identification information for identifying the data in the specific format and an acquisition source address of the data in the specific format in storage means, by associating them with each other; and

if the specific acquisition request detection means detects the specific acquisition request, the data presence judging means acquires the data identification information and the acquisition source address from the data which contains the specific acquisition request, and then if the acquired data identification information and acquisition source address have not been registered in the storage means, the data presence judging means accesses the target network resource and judges whether or not the data in the specific format requested by the specific acquisition request exists there.

On the proxy server with this configuration, after the data presence judging means accesses the target network resource and judges whether or not the data in the specific format requested by the specific acquisition request exists, data identification information about the data in the specific format and an acquisition source address of the data in the specific format are stored in storage means, being associated with each other.

On the other hand, if the specific acquisition request is detected, the data presence judging means acquires the data identification information and the acquisition source address from the data which contains the specific acquisition request. Then, if the acquired data identification information and acquisition source address have not been registered in the storage means, the data presence judging means accesses the target network resource and judges whether or not the data in the specific format requested by the specific acquisition request exists.

[Invention 19]

Invention 19 is the proxy server according to invention 18, wherein:

the storage means stores an access registration table in which the acquisition source address is registered for each item of the data in the specific format which will be accessed by the browser terminal; and

before receiving acquisition request from the browser terminal, the data presence judging means accesses the target network resource based on each acquisition source address in the access registration table, judges whether or not the appropriate data in the specific format exists, and registers the data identification information for identifying the data in the specific format and acquisition source address of the data in the specific format in the storage means, by associating them with each other.

On the proxy server with this configuration, before receiving acquisition request from the browser terminal, the data presence judging means accesses the target network resource based on each acquisition source address in the access registration table, judges whether or not the appropriate data in the specific format exists, and registers the data identification information for identifying the data in the specific format and acquisition source address of the data in the specific format in the storage means, by associating them with each other.

[Invention 20]

Invention 20 is the proxy server according to any of inventions 16 to 19,

wherein upon receiving an acquisition request containing network resource identification information for identifying a network resource, the proxy server extracts the network resource identification information from the received acquisition request, and then if it is judged that the network resource identified by the extracted network resource identification information is the browser terminal, the specific acquisition request detection means performs detection and the data presence judging means makes judgment.

On the proxy server with this configuration, when an acquisition request containing network resource identification information is received, the network resource identification information is extracted from the received acquisition request. Then, if it is judged that the network resource identified by the extracted network resource identification information is the browser terminal, the specific acquisition request detection means performs detection and the data presence judging means makes judgment.

[Invention 21]

On the other hand, to achieve the above object, invention 21 is a browser terminal communicably connected

to the proxy server set forth in invention 12, the browser terminal comprising:

terminal-side data acquisition means for acquiring data from a network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the terminal-side data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, wherein:

the data in the specific format is either executable service program data which is assigned to one of classification display areas obtained by dividing a display screen into multiple areas and composes a service program that provides a service in the given display area or executable monitor program data which is assigned to one of the classification display areas and composes a monitor program that monitors one or more service programs; and

upon receiving a full-screen update command, the memory area releasing means updates all the classification display areas to which the service program data or the monitor program data are assigned from among the multiple classification display areas, thereby releasing the memory areas allocated to the program data.

The browser terminal with this configuration has an effect equivalent to that of the browser terminal set forth in invention 12.

[Invention 22]

Invention 22 is a browser terminal communicably connected to the proxy server set forth in any of inventions 13 to 15, the browser terminal comprising:

terminal-side data acquisition means for acquiring data from a network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the terminal-side data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, wherein:

the data in the specific format is either executable service program data which is assigned to one of classification display areas obtained by dividing a display screen into multiple areas and composes a service program that provides a service in the given display area or executable monitor program data which is assigned to

one of the classification display areas and composes a monitor program that monitors one or more service programs; and

upon receiving a local-screen update command, the memory area releasing means updates the display area to which the monitor program data is assigned from among the multiple classification display areas, thereby releasing the memory area allocated to the monitor program data.

The browser terminal with this configuration has an effect equivalent to that of the browser terminal set forth in any of inventions 13 to 15.

[Invention 23]

On the other hand, to achieve the above object, invention 23 is a memory management program applied to a memory management system which comprises data acquisition means for acquiring data, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, wherein the memory management program makes a computer execute processes implemented as:

acquisition failure detection means for detecting whether the data acquisition means has failed to acquire the data in the specific format; and update command output means for outputting the update command to the memory management system based on a detection result from the acquisition failure detection means.

With this configuration, the program read and executed by a computer has an effect equivalent to that of the memory management device set forth in invention 1.

[Invention 24]

Invention 24 is a memory management program applied to a memory management system which comprises data acquisition means for acquiring data, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the data acquisition means, data initialization means for initializing the data in the specific format acquired by the data acquisition means to make it available for use, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, wherein the memory management program makes a computer execute processes implemented as:

initialization failure detection means for detecting whether the data initialization means has failed to initialize the data in the specific format; and update command output means for outputting the update command to the memory management system based on a detection result from the initialization failure detection means.

With this configuration, the program read and executed by a computer has an effect equivalent to that of the memory management device set forth in invention 2.

[Invention 25]

Invention 25 is a memory management program applied to a memory management system which comprises data acquisition means for acquiring data, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, wherein the memory management program makes a computer execute processes implemented as:

timeout detection means for detecting whether a service using the data in the specific format acquired by the data acquisition means has timed out; and update command output means for outputting the update command to the memory management system based on a detection result from the timeout detection means.

With this configuration, the program read and executed by a computer has an effect equivalent to that of the memory management device set forth in invention 3.

[Invention 26]

On the other hand, to achieve the above object, invention 26 is a server program which communicably connects via a network to browser terminal and to network resource other than the browser terminal and which, upon receiving a data acquisition request from the browser terminal, acquires data specified by the received acquisition request from a target network resource and sends the acquired data to the browser terminal, the browser terminal comprising terminal-side data acquisition means for acquiring data from a network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the terminal-side data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data

in the specific format, wherein the server program makes a computer execute processes implemented as:

server-side data acquisition means for acquiring, according to the acquisition request, data from the network resource; data transmission means for sending the data acquired by the server-side data acquisition means to the browser terminal; acquisition failure detection means for detecting via communications with the browser terminal whether the terminal-side data acquisition means has failed to acquire the data in the specific format; and update command transmission means for sending the update command to the browser terminal based on a detection result from the acquisition failure detection means.

With this configuration, the program read and executed by a computer has an effect equivalent to that of the proxy server set forth in invention 4.

[Invention 27]

Invention 27 is a server program which communicably connects via a network to browser terminal and to network resource other than the browser terminal and which, upon receiving a data acquisition request from the browser terminal, acquires data specified by the received acquisition request from a target network resource and sends the acquired data to the browser terminal, the browser terminal comprising terminal-side data acquisition means for acquiring data from a network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the terminal-side data acquisition means, data initialization means for initializing the data in the specific format acquired by the terminal-side data acquisition means to make it available for use, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, wherein the server program makes a computer execute processes implemented as:

server-side data acquisition means for acquiring, according to the acquisition request, data from the network resource; data transmission means for sending the data acquired by the server-side data acquisition means to the browser terminal; initialization failure detection means for detecting via communications with the browser terminal whether the data initialization means has failed to initialize the data in the specific format; and update command transmission means for sending the update command

to the browser terminal based on a detection result from the initialization failure detection means.

With this configuration, the program read and executed by a computer has an effect equivalent to that of the proxy server set forth in invention 5.

[Invention 28]

Invention 28 is a server program which communicably connects via a network to browser terminal and to network resource other than the browser terminal and which, upon receiving a data acquisition request from the browser terminal, acquires data specified by the received acquisition request from a target network resource and sends the acquired data to the browser terminal, the browser terminal comprising terminal-side data acquisition means for acquiring data from a network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the terminal-side data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, wherein the server program makes a computer execute processes implemented as:

server-side data acquisition means for acquiring, according to the acquisition request, data from the first network resource; data transmission means for sending the data acquired by the server-side data acquisition means to the browser terminal; timeout detection means for detecting via communications with the browser terminal whether a service using the data in the specific format acquired by the terminal-side data acquisition means has timed out; and update command transmission means for sending the update command to the browser terminal based on a detection result from the timeout detection means.

With this configuration, the program read and executed by a computer has an effect equivalent to that of the proxy server set forth in invention 7.

[Invention 29]

On the other hand, to achieve the above object, invention 29 is a terminal program running on the browser terminal set forth in invention 21 consisting of a computer, wherein:

the terminal program executes processes implemented as terminal-side data acquisition means for acquiring data from a first network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the

terminal-side data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format;

the data in the specific format is either executable service program data which is assigned to one of classification display areas obtained by dividing a display screen into multiple areas and composes a service program that provides a service in the given display area or executable monitor program data which is assigned to one of the classification display areas and composes a monitor program that monitors one or more service programs; and

upon receiving a full-screen update command, the memory area releasing means updates all the classification display areas to which the service program data or the monitor program data are assigned from among the multiple classification display areas, thereby releasing the memory areas allocated to the program data.

With this configuration, the program read and executed by a computer has an effect equivalent to that of the browser terminal set forth in invention 21.

[Invention 30]

Invention 30 is a terminal program running on the browser terminal set forth in invention 22 consisting of a computer, wherein:

the terminal program executes processes implemented as terminal-side data acquisition means for acquiring data from a network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the terminal-side data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format;

the data in the specific format is either executable service program data which is assigned to one of classification display areas obtained by dividing a display screen into multiple areas and composes a service program that provides a service in the given display area or executable monitor program data which is assigned to one of the classification display areas and composes a monitor program that monitors one or more service programs; and

upon receiving a local-screen update command, the memory area releasing means updates the display area to which the monitor program data is assigned from among the

multiple classification display areas, thereby releasing the memory area allocated to the monitor program data.

With this configuration, the program read and executed by a computer has an effect equivalent to that of the browser terminal set forth in invention 22.

[Invention 31]

On the other hand, to achieve the above object, invention 31 is the monitor program set forth in invention 12, wherein upon receiving a full-screen update command, the monitor program makes a computer execute processes for updating all the classification display areas to which the service program data or the monitor program data are assigned from among the multiple classification display areas, thereby releasing the memory areas allocated to the program data.

With this configuration, the program read and executed by a computer has an effect equivalent to that of the monitor program set forth in invention 12.

[Invention 32]

Invention 32 is the monitor program set forth in any of inventions 13 to 15, wherein upon receiving a local-screen update command, the monitor program makes a computer execute processes for updating the classification display areas to which the monitor program data is assigned from among the multiple classification display areas, thereby releasing the memory areas allocated to the monitor program data.

With this configuration, the program read and executed by a computer has an effect equivalent to that of the monitor program set forth in any of inventions 13 to 15.

[Invention 33]

On the other hand, to achieve the above object, invention 33 is a memory management method applied to a memory management system which comprises data acquisition means for acquiring data, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, the memory management method comprising:

an acquisition failure detection step of detecting whether the data acquisition means has failed to acquire the data in the specific format; and an update command output step of outputting the update command to the

memory management system based on a detection result from the acquisition failure detection step.

The acquisition failure detection step may use any method as long as it can detect that the data acquisition means has failed to acquire data in the specific format. More specifically, it can detect failures to acquire data in the specific format, by estimation or by actual measurement.

[Invention 34]

Invention 34 is a memory management method applied to a memory management system which comprises data acquisition means for acquiring data, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the data acquisition means, data initialization means for initializing the data in the specific format acquired by the data acquisition means to make it available for use, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, the memory management method comprising:

an initialization failure detection step of detecting whether the data initialization means has failed to initialize the data in the specific format; and an update command output step of outputting the update command to the memory management system based on a detection result from the initialization failure detection step.

The initialization failure detection step may use any method as long as it can detect that the data initialization means has failed to initialize data in the specific format. More specifically, it can detect failures to initialize data in the specific format, by estimation or by actual measurement.

[Invention 35]

Invention 35 is a memory management method applied to a memory management system which comprises data acquisition means for acquiring data, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, the memory management method comprising:

a timeout detection step of detecting whether a service using the data in the specific format acquired by

the data acquisition means has timed out; and an update command output step of outputting the update command to the memory management system based on a detection result from the timeout detection step.

The timeout detection step may use any method as long as it can detect that a service using the data in the specific format acquired by the data acquisition means has timed out. More specifically, it can detect service timeouts by estimation or by actual measurement.

[Invention 36]

Invention 36 is a memory management method which communicably connects via a network to browser terminal and to network resource other than the browser terminal and which, upon receiving a data acquisition request from the browser terminal, acquires data specified by the received acquisition request from a target network resource and sends the acquired data to the browser terminal, the browser terminal comprising terminal-side data acquisition means for acquiring data from a network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the terminal-side data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, the memory management method comprising:

a server-side data acquisition step of acquiring, according to the acquisition request, data from the first network resource; a data transmission step of sending data acquired by the server-side data acquisition step to the browser terminal; an acquisition failure detection step of detecting via communications with the browser terminal whether the terminal-side data acquisition means has failed to acquire the data in the specific format; and an update command transmission step of sending the update command to the browser terminal based on a detection result from the acquisition failure detection step.

[Invention 37]

Invention 37 is a memory management method which communicably connects via a network to browser terminal and to network resource other than the browser terminal and which, upon receiving a data acquisition request from the browser terminal, acquires data specified by the received acquisition request from a target network resource and sends the acquired data to the browser terminal, the browser terminal comprising terminal-side

data acquisition means for acquiring data from a network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the terminal-side data acquisition means, data initialization means for initializing the data in the specific format acquired by the terminal-side data acquisition means to make it available for use, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, the memory management method comprising:

a server-side data acquisition step of acquiring, according to the acquisition request, data from the first network resource; a data transmission step of sending data acquired by the server-side data acquisition step to the browser terminal; an initialization failure detection step of detecting via communications with the browser terminal whether the data initialization means has failed to initialize the data in the specific format; and an update command transmission step of sending the update command to the browser terminal based on a detection result from the initialization failure detection step.

[Invention 38]

Invention 38 is a memory management method which communicably connects via a network to browser terminal and to network resource other than the browser terminal and which, upon receiving a data acquisition request from the browser terminal, acquires data specified by the received acquisition request from a target network resource and sends the acquired data to the browser terminal, the browser terminal comprising terminal-side data acquisition means for acquiring data from a network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the terminal-side data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, the memory management method comprising:

a server-side data acquisition step of acquiring, according to the acquisition request, data from the first network resource; a data transmission step of sending data acquired by the server-side data acquisition step to the browser terminal; a timeout detection step of detecting via communications with the browser terminal whether a service using the data in the specific format

acquired by the terminal-side data acquisition means has timed out; and an update command transmission step of sending the update command to the browser terminal based on a detection result from the timeout detection step.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram showing a configuration of a network system to which the present invention is applied;

FIG. 2 is a block diagram showing a configuration of a proxy server 200;

FIG. 3 is a diagram showing a data structure of an access registration table 400;

FIG. 4 is a diagram showing a data structure of a terminal management table 420;

FIG. 5 is a diagram showing a data structure of an applet management table 440;

FIG. 6 is a diagram showing a data structure of an applet presence management table 460;

FIG. 7 is a flowchart showing an applet presence registration process;

FIG. 8 is a flowchart showing an applet management process;

FIG. 9 is a flowchart showing a monitor applet transmission process;

FIG. 10 is a flowchart showing a data acquisition mediating process;

FIG. 11 is a flowchart showing a service applet transmission process;

FIG. 12 is a flowchart showing a service applet monitoring process;

FIG. 13 is a flowchart showing an update request transmission process;

FIG. 14 is a block diagram showing a configuration of a browser terminal 300;

FIG. 15 is a flowchart showing a monitor applet acquisition process;

FIG. 16 is a flowchart showing a service applet acquisition process;

FIG. 17 is a flowchart showing a screen update process;

FIG. 18 is a flowchart showing a reboot process;

FIG. 19A and FIG. 19B show display screens of the browser terminals 300; and

FIG. 20 is a diagram illustrating the operation of updating the display screen of the browser terminal 300.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

A preferred embodiment of the present invention will be described below with reference to the drawings. FIGS. 1 to 19 are diagrams showing embodiments of a memory management device, proxy server, browser terminal, memory management program, server program, terminal program, monitor program, and memory management method according to the present invention.

This embodiment applies the memory management device, proxy server, browser terminal, memory management program, server program, terminal program, monitor program, and memory management method according to the present invention to a proxy server 200 and browser terminals 300 installed between the Internet 198 and a private network 199 as shown in FIG. 1.

First, configuration of a network system to which the present invention is applied will be described with reference to FIG. 1.

FIG. 1 is a block diagram showing the configuration of the network system to which the present invention is applied.

In FIG. 1, the Internet 198 is connected with a plurality of WWW servers 100 which accumulate Web pages composed of HTML data while the private network 199 is connected with a plurality of special-purpose browser terminals 300 and a plurality of ordinary terminals 310 which are used for purposes other than a special purpose as well. Also, the Internet 198 and private network 199 is connected with a proxy server 200 which mediate data acquisition between the browser terminals 300 or ordinary terminals 310 and WWW servers 100 using the HTTP (HyperText Transfer Protocol) or other protocol.

Each browser terminal 300 consists of a touch panel mounted on a liquid crystal display (not shown). It starts up a Web browser which can be operated simply by using the touch panel. The Web browser which runs on the browser terminals 300 allocates and deallocates memory areas to/from applets, according to a static memory management system. Besides, to provide services through Web pages reliably, the Web browser is not equipped with a Back button.

The browser terminals 300 are classified into two types--"ND-2000" and "ND-3000"--according to the Web browsers installed on them. The "ND-2000" browser terminal 300 interprets and properly processes a tag contained in HTML data in a format (hereinafter referred to as ND-2000 format) which describes identification codes, versions, and names of applets, parameters given to the applets, etc. in a single tag, but ignores tags

contained in HTML data in a format (hereinafter referred to as ND-3000 format) which describes an identification code, version, and name of an applet, parameters given to the applet, etc. in separate tags. The "ND-3000" browser terminals 300 interprets and properly processes tags contained in HTML data in the ND-3000 format, but ignores a tag contained in HTML data in the ND-2000 format.

Applets are roughly divided into two types. When displaying a Web page consisting of multiple frames, the applet of one type (hereinafter referred to as a service applet) is assigned to one of the frames and provides a service (e.g., print service) in the assigned frame while the applet of the other type (hereinafter referred to as a monitor applet) is assigned to one of the frames and monitors one or more service applets. On the browser terminals 300, a static memory management system is used for both types of applet, but service applets can be executed only if a monitor applet exists in memory. Sometimes, service details of service applets are described in the monitor applet. Incidentally, if there is no need to distinguish between service applet and monitor applet, they will be referred to simply as applets

Next, configuration of the proxy server 200 will be described in detail with reference to FIG. 2.

FIG. 2 is a block diagram showing the configuration of the proxy server 200.

As shown in FIG. 2, the proxy server 200 consists of a CPU 30 which performs computations and controls the entire system based on a control program, a ROM 32 which prestores the CPU 30 control program and the like in predetermined areas, a RAM 34 which stores data read out of the ROM 32 and the like and computational results needed for the CPU 30 during computation, and an I/F 38 which mediates data input and output to/from external devices. These components are connected to a bus 39, which is a signal line for data transfer, in such a way that they can exchange data.

The I/F 38 is connected with external devices: an input device 40 which serves as a human interface and consists of a keyboard, mouse, etc. for data input; a storage device 42 which stores data, tables, etc. in the form of files; and a display device 44 which displays images according to image signals. Also, it is connected with a signal line for connecting to the Internet 198 and a signal line for connecting to the private network 199.

FIG. 3 is a diagram showing a data structure of an access registration table 400.

As shown in FIG. 3, the storage device 42 stores the access registration table 400 for registering URLs (Uniform Resource Locator) of Web pages which may be accessed by the terminals connecting to the private network 199.

As shown in FIG. 3, the access registration table 400 contains one record per URL. Each record contains a field 402 in which a URL is registered. Before receiving acquisition request from a terminal connecting to the private network 199, the proxy server 200 acquires the Web pages identified by the registered URLs with reference to the access registration table 400 and registers information as to whether or not service applets used by the acquired Web pages exist. This improves a response rate with respect to the terminal connecting to the private network 199.

FIG. 4 is a diagram showing a data structure of a terminal management table 420.

As shown in FIG. 4, the storage device 42 also stores the terminal management table 420 for registering an IP address, terminal identification information, the number of service applet transmissions, the presence/absence of a monitor applet, the presence/absence of service applets, and download status for each terminal which connects to the private network 199.

As shown in FIG. 4, the terminal management table 420 contains one record for each terminal which connects to the private network 199. Each record contains a field 422 for registering the IP address of the terminal, a field 424 for registering terminal identification information for identifying the terminal, a field 426 for registering the number of service applet transmissions to the terminal, a field 428 for registering the presence/absence of a monitor applet, a field 430 for registering the presence/absence of service applets, and a field 432 for registering the download status of applets.

In the example of FIG. 4, the record in the first row contains "192.168.100.10" as the IP address, "ordinary terminal" as the terminal identification information, "1" as the number of transmissions, "1" as the presence/absence of a monitor applet, "1" as the presence/absence of service applets, and "Download Complete" as the download status. This means that the terminal to which the IP address "192.168.100.10" is assigned is an ordinary terminal 310, that the number of times a service applet has been transmitted to the

ordinary terminal 310 is "1," that a monitor applet and service applet exist on the ordinary terminal 310, and that the downloading of the applets has been completed. Also, the record in the fourth row contains "192.168.101.10" as the IP address, "brouser terminal" as the terminal identification information, "0" as the number of transmissions, "0" as the presence/absence of a monitor applet, and "0" as the presence/absence of service applets. This means that the terminal to which the IP address "192.168.101.10" is assigned is a browser terminal 300, that the number of times a service applet has been transmitted to the browser terminal 300 is "0," and that no monitor applet or service applet exists on the browser terminal 300.

FIG. 5 is a diagram showing a data structure of an applet management table 440.

As shown in FIG. 5, the storage device 42 also stores the applet management table 440 which manages service applets for Web pages.

As shown in FIG. 5, one record is created in the applet management table 440 each time the proxy server 200 acquires, according to an acquisition request, a Web page. Each record contains a field 442 for registering the URL of the Web page, fields 444 and 450 for registering the number of service applets used by the Web page, fields 446 and 452 for registering the presence/absence of all the service applets used by the Web page, and fields 448 and 454 for registering a pointer to an applet presence management table (described later) which manages information as to whether the service applets used by the Web page exist. The fields 444 to 448 contain the number of service applets used by the Web page, presence/absence of the service applets, and table name for Web pages composed of HTML data in the ND-2000 format while the fields 450 to 454 contain the number of service applets used by the Web page, presence/absence of the service applets, and table name for Web pages composed of HTML data in the ND-3000 format.

In the example of FIG. 5, the record in the first row contains "http://192.168.200.1/service/print.htm" as the URL; "3," "0," and "table A1" as the number of service applets, presence/absence of the service applets, and pointer to the applet presence management table, respectively, for the HTML data in the ND-2000 format; and "4," "0," and "table A2" as the number of service applets, presence/absence of the service applets, and pointer to the applet presence management table,

respectively, for the HTML data in the ND-3000 format. This means that the Web page identified by "http://192.168.200.1/service/print.htm" contains seven specific acquisition requests for service applets and that the information as to whether or not the service applets are present is managed by table A1 in the case of the ND-2000 format, and by table A2 in the case of the ND-3000 format. Furthermore, the data in the record also means that part or all of the service applets do not exist in relation to the ND-2000 format and that part or all of the service applets do not exist in relation to the ND-3000 format. Incidentally, to detect specific acquisition requests reliably, each Web page contains applet identification codes together with the specific acquisition requests.

FIG. 6 is a diagram showing a data structure of an applet presence management table 460.

As shown in FIG. 6, the applet presence management table 460 contains one record for each service applet. Each record contains a field 462 for registering an applet name and a field 464 for registering information about the presence/absence of the service applet.

In the example of FIG. 6, the record in the first row contains "exe_001" as the applet name and "1" as the information about the presence/absence of the service applet. This means that the service applet identified by the applet name "exe_001" exists in a prescribed WWW server 100. Also, the record in the third row contains "exe_003" as the applet name and "0" as the information about the presence/absence of the service applet. This means that the service applet identified by the applet name "exe_003" does not exist in the prescribed WWW server 100.

The CPU 30 consists of a microprocessing unit (MPU) and the like. It starts a predetermined program stored in a predetermined area of the ROM 32 and performs an applet presence registration process and applet management process shown in flowcharts in FIGS. 7 and 8 on a time division basis according to the program.

To begin with, the applet presence registration process will be described in detail with reference to FIG. 7.

FIG. 7 is a flowchart showing the applet presence registration process.

The applet presence registration process involves acquiring the Web pages identified by registered URLs with reference to the access registration table 400 and registering information as to whether or not service

applets used by the acquired Web pages exist before receiving acquisition request from a terminal which connects to the private network 199. The process is performed by the CPU 30, beginning with Step S100 as shown in FIG. 7.

The CPU 30 reads the URL from the first record of the access registration table 400 in Step S100, accesses the WWW server 100 based on the URL in Step S102, acquires the Web page from the accessed WWW server 100 in Step S104, and then goes to Step S106.

In Step S106, the CPU 30 parses the acquired Web page. Then, in Step S108, the CPU 30 judges, based on the results of the parsing, whether an applet identification code has been detected in the acquired Web page. If it is judged that an applet identification code has been detected (Yes), the CPU 30 goes to Step S110.

In Step S110, the CPU 30 accesses the WWW server 100 relevant to the specific acquisition request associated with the detected applet identification code. Then, in Step S112, the CPU 30 judges whether the service applet requested by the specific acquisition request exists on the WWW server 100. If it is judged that the service applet exists (Yes), the CPU 30 goes to Step S114, where the CPU 30 registers information that the service applet which corresponds to the detected applet identification code exists on the WWW server 100, in the applet management table 440 and applet presence management table 460. Incidentally, the registrations are made separately according to the format of the HTML data composing the acquired Web page. This is also true of Steps S120, S420, and S432.

Next, the CPU 30 goes again to Step S108, where it judges, based on the results of the parsing, whether another applet identification code has been detected in the acquired Web page. Step S108 is repeated in case the Web page contains two or more applet identification codes.

On the other hand, if it is judged in Step S112 that the service applet does not exist on the WWW server 100 (No), the CPU 30 goes to Step S120, where the CPU 30 registers information that the service applet which corresponds to the detected applet identification code does not exist on the WWW server 100, in the applet management table 440 and applet presence management table 460. Then, the CPU 30 goes again to Step S108, where it judges, based on the results of the parsing, whether another applet identification code has been detected in the acquired Web page. Step S108 is repeated in case the

Web page contains two or more applet identification codes.

On the other hand, if it is judged in Step S108 that no other applet identification code has been detected (No), the CPU 30 goes to Step S116, where it judges whether all the URLs registered in the access registration table 400 have gone through Steps S102 to S114 and S120. If it is judged that all the URLs have been processed (Yes), the CPU 30 finishes the sequence of processes and returns to the original process.

On the other hand, if it is judged in Step S116 that all the URLs registered in the access registration table 400 have not gone through Steps S102 to S114 and S120 (No), the CPU 30 goes to Step S118, where it reads the URL from the next record in the access registration table 400, and then it goes to Step S102.

Next, the applet management process will be described in detail with reference to FIG. 8.

FIG. 8 is a flowchart showing the applet management process.

The applet management process is performed by the CPU 30, beginning with Step S200 as shown in FIG. 8.

In Step S200, the CPU 30 performs a data acquisition mediating process to mediate data acquisition between the browser terminal 300 or ordinary terminal 310 and the WWW server 100, and then it goes to Step S202.

In Step S202, the CPU 30 judges whether a monitor applet acquisition request, which is a request to acquire a monitor applet, has been received either from the browser terminal 300 or ordinary terminal 310. If it is judged that a monitor applet acquisition request has been received (Yes), the CPU 30 goes to Step S204.

In Step S204, the CPU 30 performs a monitor applet transmission process to transmit the monitor applet to the browser terminal 300 or ordinary terminal 310. Then, in Step S206, the CPU 30 starts a monitoring service using the monitor applet on the browser terminal 300 or ordinary terminal 310 whichever has received the monitor applet, and then the CPU 30 goes to Step S208.

In Step S208, the CPU 30 judges whether a service applet acquisition request, which is a request to acquire a service applet, has been received either from the browser terminal 300 or ordinary terminal 310. If it is judged that a service applet acquisition request has been received (Yes), the CPU 30 goes to Step S210.

In Step S210, the CPU 30 performs a service applet transmission process to transmit the service applet to the browser terminal 300 or ordinary terminal 310. Then,

in Step S212, the CPU 30 starts a service applet monitoring process to monitor the service applet on the browser terminal 300 or ordinary terminal 310 whichever has received the service applet. Then, in Step S214, the CPU 30 performs an update request transmission process to transmit an update request to the browser terminal 300, and then the CPU 30 goes to Step S200.

On the other hand, if it is judged in Step S208 that no service applet acquisition request has been received (No), the CPU 30 goes to Step S200.

On the other hand, if it is judged in Step S202 that no monitor applet acquisition request has been received (No), the CPU 30 goes to Step S208.

Next, the monitor applet transmission process in Step S204 will be described in detail with reference to FIG. 9.

FIG. 9 is a flowchart showing the monitor applet transmission process.

The monitor applet transmission process is performed by the CPU 30, beginning with Step S300 as shown in FIG. 9.

In Step S300, the CPU 30 judges whether the monitor applet exists on the browser terminal 300 or ordinary terminal 310, whichever is requesting the monitor applet (in the monitor applet transmission process, this terminal will be referred to as the requesting terminal), by referring to the terminal management table 420. If it is judged that the monitor applet does not exist on the browser terminal 300 or ordinary terminal 310 (No), the CPU 30 goes to Step S302, where it sets the download status in the record for the requesting terminal in the terminal management table 420 to "Start." Then, the CPU 30 goes to Step S304.

In Step S304, the CPU 30 starts a timer. Then, in Step S306, the CPU 30 starts downloading the monitor applet requested by the monitor applet acquisition request. In Step S308, the CPU 30 sets the download status in the record for the requesting terminal in the terminal management table 420 to "Downloading," and then it goes to Step S310.

In Step S310, the CPU 30 judges whether a timeout has occurred, meaning that a predetermined time has elapsed since the monitor applet was started to be downloaded. If it is judged that no timeout has occurred (No), the CPU 30 goes to Step S312, where it judges whether a download-complete notice has been received indicating that the downloading of the monitor applet has been completed. If it is judged that a download-complete

notice has been received (Yes), the CPU 30 goes to Step S314.

In Step S314, the CPU 30 registers the presence of the monitor applet, in the record for the requesting terminal in the terminal management table 420. Then, in Step S316, the CPU 30 stops the timer, and then it goes to Step S318.

In Step S318, the CPU 30 sends a notice about acquisition of service applet initialization status. Then, in Step S320, the CPU 30 judges whether an initialization failure notice has been received, indicating that initialization of the monitor applet has failed. If it is judged that no initialization failure notice has been received (No), the CPU 30 goes to Step S322, where it sets the download status in the record for the requesting terminal in the terminal management table 420 to "Download Successful." Then, the CPU 30 finishes the sequence of processes and returns to the original process.

On the other hand, if it is judged in Step S320 that an initialization failure notice has been received (Yes), the CPU 30 goes to Step S324, where it sets update mode of the requesting browser terminal 300 to "REFRESH." Then, the CPU 30 finishes the sequence of processes and returns to the original process.

On the other hand, if it is judged in Step S312 that no download-complete notice has been received (No), the CPU 30 goes to Step S310.

On the other hand, if it is judged in Step S310 that a timeout has occurred (Yes), the CPU 30 goes to Step S326, where it sets the update mode of the requesting browser terminal 300 to "STOP." Then, the CPU 30 finishes the sequence of processes and returns to the original process.

On the other hand, if it is judged in Step S300 that the monitor applet exists on the requesting browser terminal (Yes), the CPU 30 goes to Step S328, where it sets the update mode of the requesting browser terminal 300 to "REBOOT." Then, the CPU 30 finishes the sequence of processes and returns to the original process.

Next, the data acquisition mediating process in Step S200 will be described in detail with reference to FIG. 10.

FIG. 10 is a flowchart showing the data acquisition mediating process.

The data acquisition mediating process is performed by the CPU 30, beginning with Step S400 as shown in FIG. 10.

In Step S400, the CPU 30 judges whether a page acquisition request has been received via HTTP. If it is judged that a page acquisition request has been received (Yes), the CPU 30 goes to Step S401. Otherwise, it waits in Step S400 until a page acquisition request is received.

In Step S401, the CPU 30 judges whether the IP address contained in the acquisition request has been registered in the terminal management table 420. If it is judged that the IP address has not been registered in the terminal management table 420 (No), the CPU 30 goes to Step S402, where it parses the HTTP header of the acquisition request. Then, in Step S404, the CPU 30 judges whether the requesting terminal is a browser terminal 300. If it is judged that the requesting terminal is a browser terminal 300 (Yes), the CPU 30 goes to Step S405.

In Step S405, the CPU 30 registers the IP address contained in the acquisition request and terminal identification information which indicates that the requesting terminal is a browser terminal 300 in the terminal management table 420. Then, the CPU 30 goes to Step S406, where it accesses the WWW server 100 relevant to the acquisition request. In Step S408, the CPU 30 acquires the Web page requested by the acquisition request, from the WWW server 100, and then it goes to Step S410.

In Step S410, the CPU 30 parses the acquired Web page. Then, in Step S412, the CPU 30 judges, based on the results of the parsing, whether an applet identification code has been detected in the acquired Web page. If it is judged that an applet identification code has been detected (Yes), the CPU 30 goes to Step S414, where it judges whether a service applet which corresponds to the detected applet identification code has been registered in the applet management table 440 and applet presence management table 460. If it is judged that the service applet has not been registered in the applet management table 440 and applet presence management table 460 (No), the CPU 30 goes to Step S416.

In Step S416, the CPU 30 accesses the WWW server 100 relevant to the specific acquisition request associated with the detected applet identification code and goes to Step S418. Then, in Step S418, the CPU 30 judges whether the service applet requested by the specific acquisition request exists on the WWW server 100. If it is judged that the service applet exists (Yes), the CPU 30 goes to Step S420.

In Step S420, the CPU 30 registers information that the service applet which corresponds to the detected applet identification code exists on the WWW server 100, in the applet management table 440 and applet presence management table 460. Then, in Step S422, the CPU 30 increments the number of service applet transmissions in the record for the requesting browser terminal 300 in the terminal management table 420. Then, the CPU 30 goes to Step S412, where it judges, based on the results of the parsing, whether another applet identification code has been detected in the acquired Web page. Step S412 is repeated in case the Web page contains two or more applet identification codes.

On the other hand, if it is judged in Step S418 that the service applet does not exist on the WWW server 100 (No), the CPU 30 goes to Step S432, where the CPU 30 registers information that the service applet which corresponds to the detected applet identification code does not exist on the WWW server 100, in the applet management table 440 and applet presence management table 460. Then, in Step S434, the CPU 30 sends a substitute page instead of the acquired Web page to the browser terminal 300 to notify it that the service applet does not exist. Then, the CPU 30 finishes the sequence of processes and returns to the original process.

On the other hand, if it is judged in Step S414 that the service applet which corresponds to the detected applet identification code has been registered in the applet management table 440 and applet presence management table 460 (Yes), the CPU 30 goes to Step S436, where it judges whether the service applet requested by the specific acquisition request exists on the WWW server 100 by referring to the applet management table 440 and applet presence management table 460. If it is judged that the service applet does not exist (No), the CPU 30 goes to Step S434.

On the other hand, if it is judged in Step S436 that the service applet requested by the specific acquisition request exists on the WWW server 100 (Yes), the CPU 30 goes to Step S422.

On the other hand, if it is judged in Step S412 that no applet identification code has been detected in the acquired Web page (No), the CPU 30 goes to Step S430, where it sends the acquired Web page to the requesting terminal. Then, the CPU 30 finishes the sequence of processes and returns to the original process.

On the other hand, if it is judged in Step S404 that the requesting terminal is not a browser terminal 300

(No), the CPU 30 goes to Step S438, where it registers the IP address contained in the acquisition request and terminal identification information which indicates that the requesting terminal is an ordinary terminal 310 in the terminal management table 420. Then, the CPU 30 goes to Step S440, where it accesses the WWW server 100 relevant to the acquisition request. Then, in Step S442, the CPU 30 acquires the Web page requested by the acquisition request, from the WWW server 100, and then it goes to Step S430.

On the other hand, if it is judged in Step S401 that the IP address contained in the acquisition request has been registered in the terminal management table 420 (Yes), the CPU 30 goes to Step S444, where it judges whether the requesting terminal is a browser terminal 300 by referring to the terminal management table 420. If it is judged that the requesting terminal is a browser terminal 300 (Yes), the CPU 30 goes to Step S406.

On the other hand, if it is judged in Step S444 that the requesting terminal is not a browser terminal 300 (No), the CPU 30 goes to Step S440.

Next, the service applet transmission process in Step S210 will be described in detail with reference to FIG. 11.

FIG. 11 is a flowchart showing the service applet transmission process.

The service applet transmission process is performed by the CPU 30, beginning with Step S500 as shown in FIG. 11.

In Step S500, the CPU 30 judges whether the service applet exists on the browser terminal 300 or ordinary terminal 310, whichever is requesting the service applet (in the service applet transmission process, this terminal will be referred to as the requesting terminal), by referring to the terminal management table 420. If it is judged that the service applet does not exist on the browser terminal 300 or ordinary terminal 310 (No), the CPU 30 goes to Step S502, where it sets the download status in the record for the requesting terminal in the terminal management table 420 to "Start." Then, the CPU 30 goes to Step S504.

In Step S504, the CPU 30 starts a timer. Then, in Step S506, the CPU 30 starts downloading the service applet requested by the service applet acquisition request. In Step S508, the CPU 30 sets the download status in the record for the requesting terminal in the terminal management table 420 to "Downloading," and then it goes to Step S510.

In Step S510, the CPU 30 judges whether a timeout has occurred, meaning that a predetermined time has elapsed since the service applet was started to be downloaded. If it is judged that no timeout has occurred (No), the CPU 30 goes to Step S512, where it judges whether a download-complete notice has been received indicating that the downloading of the service applet has been completed. If it is judged that a download-complete notice has been received (Yes), the CPU 30 goes to Step S514.

In Step S514, the CPU 30 registers the presence of the service applet, in the record for the requesting terminal in the terminal management table 420. Then, in Step S516, the CPU 30 stops the timer, and then it goes to Step S518.

In Step S518, the CPU 30 sends a notice about acquisition of service applet initialization status and goes to Step S520. Then, in Step S520, the CPU 30 judges whether an initialization failure notice has been received, indicating that initialization of the service applet has failed. If it is judged that no initialization failure notice has been received (No), the CPU 30 goes to Step S522, where it sets the download status in the record for the requesting terminal in the terminal management table 420 to "Download Successful." Then, the CPU 30 finishes the sequence of processes and returns to the original process.

On the other hand, if it is judged in Step S520 that an initialization failure notice has been received (Yes), the CPU 30 goes to Step S524, where it sets the update mode of the requesting browser terminal 300 to "REFRESH." Then, the CPU 30 finishes the sequence of processes and returns to the original process.

On the other hand, if it is judged in Step S512 that no download-complete notice has been received (No), the CPU 30 goes to Step S510.

On the other hand, if it is judged in Step S510 that a timeout has occurred (Yes), the CPU 30 goes to Step S526, where it sets the update mode of the requesting browser terminal 300 to "STOP." Then, the CPU 30 finishes the sequence of processes and returns to the original process.

On the other hand, if it is judged in Step S500 that the service applet exists on the requesting browser terminal (Yes), the CPU 30 goes to Step S528, where it sets the update mode of the requesting browser terminal 300 to "STOP." Then, the CPU 30 finishes the sequence of processes and returns to the original process.

Next, the service applet monitoring process in Step S212 will be described in detail with reference to FIG. 12.

FIG. 12 is a flowchart showing the service applet monitoring process.

The service applet monitoring process is performed by the CPU 30, beginning with Step S600 as shown in FIG. 12.

In Step S600, via communications with the browser terminal 300 being monitored (in the service applet monitoring process, this terminal will be referred to as the monitored terminal), the CPU 30 judges whether an error occurred during the service applet's service. If it is judged that no error occurred during the service (No), the CPU 30 goes to Step S602, where it judges the current mode. If the current mode is forced update mode, the CPU 30 goes to Step S604.

In Step S604, the CPU 30 judges whether the number of service applet transmissions has reached a predetermined number (e.g., 100 times), by referring to the terminal management table 420. If it is judged that the number of service applet transmissions has reached the predetermined number (Yes), the CPU 30 goes to Step S608, where it sets update mode of the monitored terminal to "REFRESH." Then, the CPU 30 finishes the sequence of processes and returns to the original process.

On the other hand, if it is judged in Step S600 that an error occurred during the service (Yes), the CPU 30 goes to Step S608, where it sets the update mode of the monitored terminal to "REFRESH." Then, the CPU 30 finishes the sequence of processes and returns to the original process.

On the other hand, if it is judged in Step S604 that the number of service applet transmissions has not reached the predetermined number (No), the CPU 30 goes to Step S610, where it judges whether the service applet's service has timed out. If it is judged that the service has timed out (Yes), the CPU 30 goes to Step S608.

On the other hand, if it is judged in Step S610 that the service has not timed out (No), the CPU 30 finishes the sequence of processes and returns to the original process.

On the other hand, if it is judged in Step S602 that the current mode is transitional update mode, the CPU 30 goes to Step S612, where it judges whether an idle URL is requested by the monitored terminal. If it is judged that an idle URL is requested (Yes), the CPU 30 goes to Step S604.

On the other hand, if it is judged in Step S612 that no idle URL is requested (No), the CPU 30 goes to Step S610.

Next, the update request transmission process in Step S214 will be described in detail with reference to FIG. 13.

FIG. 13 is a flowchart showing the update request transmission process.

The update request transmission process is performed by the CPU 30, beginning with Step S700 as shown in FIG. 13.

In Step S700, the CPU 30 judges whether the update mode is "REBOOT." If it is judged that the update mode is "REBOOT" (Yes), the CPU 30 goes to Step S702, where it sends a reboot command to the browser terminal 300 (in the update request transmission process, this terminal will be referred to as the receiving terminal) via SNMP (Simple Network Management Protocol), instructing it to reboot itself. Then, the CPU 30 goes to Step S704.

In Step S704, the CPU 30 registers the absence of a monitor applet, in the record for the receiving terminal in the terminal management table 420. Then, in Step S706, the CPU 30 registers the absence of a service applet, in the record for the receiving terminal in the terminal management table 420, and then it goes to Step S708.

In Step S708, the CPU 30 initializes the download status in the record for the receiving terminal in the terminal management table 420 and goes to Step S710. Then, in Step S710, the CPU 30 sets the number of service applet transmissions in the record for the receiving terminal in the terminal management table 420 to "0." Then, the CPU 30 finishes the sequence of processes and returns to the original process.

On the other hand, if it is judged in Step S700 that the update mode is not "REBOOT" (No), the CPU 30 goes to Step S712, where it judges whether the update mode is "STOP." If it is judged that the update mode is "STOP" (Yes), the CPU 30 goes to Step S714, where it sends a full-screen update command to the receiving terminal, and then it goes to Step S704. The full-screen update command is issued, when no service applet exists in any of the frames composing a displayed Web page, to update all the frames to which the service applets or monitor applet are assigned from among the multiple frames, thereby releasing the memory areas allocated to the applets and initializing all the memory under management.

On the other hand, if it is judged in Step S712 that the update mode is not "STOP" (No), the CPU 30 goes to Step S716, where it judges whether the update mode is "REFRESH." If it is judged that the update mode is "REFRESH" (Yes), the CPU 30 goes to Step S718, where it sends a local-screen update command to the receiving terminal, and then it goes to Step S704. The local-screen update command is issued to update the frames to which the monitor applet is assigned from among the multiple frames composing a displayed Web page, thereby releasing the memory areas allocated to the monitor applet.

On the other hand, in Step S716, if it is judged that the update mode is not "REFRESH" (No), the CPU 30 goes to Step S706.

Next, configuration of the browser terminal 300 will be described in detail with reference to FIG. 14.

FIG. 14 is a block diagram showing the configuration of the browser terminal 300.

As shown in FIG. 14, the browser terminal 300 consists of a CPU 50 which performs computations and controls the entire system based on a control program, a ROM 52 which prestores the CPU 50 control program and the like in predetermined areas, a RAM 54 which stores data read out of the ROM 52 and the like and computational results needed for the CPU 50 during computation, and an I/F 58 which mediates data input and output to/from external devices. These components are connected to a bus 59, which is a signal line for data transfer, in such a way that they can exchange data.

The I/F 58 is connected with a touch panel 60 that data inputting and display is available which is an external device serving as a human interface and with a signal line for connecting to the private network 199.

The CPU 50 consists of a microprocessing unit (MPU) and the like. It starts a predetermined program stored in a predetermined area of the ROM 52, and performs a monitor applet acquisition process, service applet acquisition process, screen update process, and reboot process shown in flowcharts in FIGS. 15 to 18 on a time division basis according to the program.

To begin with, the monitor applet acquisition process will be described in detail with reference to FIG. 15.

FIG. 15 is a flowchart showing the monitor applet acquisition process.

The monitor applet acquisition process is performed by the CPU 50, beginning with Step S800 upon startup of the Web browser as shown in FIG. 15.

In Step S800, the CPU 50 sends a monitor applet acquisition request to the proxy server 200 and goes to Step S802. Then, in Step S802, the CPU 50 downloads the monitor applet, and then goes to Step S804.

In Step S804, the CPU 50 allocates a memory area to the downloaded monitor applet and goes to Step S806. Then, in Step S806, the CPU 50 initializes the downloaded monitor applet to make it available for use, and then it goes to Step S808.

In Step S808, when basic functions of the monitor applet such as communications services have been initialized, the CPU 50 determines that memory area allocation to the downloaded monitor applet has been completed successfully and sends a download-complete notice to the proxy server 200, and then it goes to Step S810.

In Step S810, the CPU 50 judges whether the initialization of the monitor applet has failed. If it is judged that the initialization of the monitor applet has failed (Yes), the CPU 50 goes to Step S812, where it sends an initialization failure notice to the proxy server 200. Then, the CPU 50 finishes the sequence of processes and returns to the original process.

On the other hand, if it is judged in Step S810 that the initialization of the monitor applet has not failed (No), the CPU 50 finishes the sequence of processes and returns to the original process.

On the other hand, if it is judged in Step S802 that the downloading of the monitor applet has not been completed, being interrupted by a communications failure or the like, this means that the memory area allocation in Step S804 and consequently the initialization in Step S806 are not complete. Thus, the CPU 50 waits without going to Step S808.

Next, the service applet acquisition process will be described in detail with reference to FIG. 16.

FIG. 16 is a flowchart showing the service applet acquisition process.

The service applet acquisition process, which involves detecting applet identification codes in an acquired Web page and acquiring service applets requested by a specific acquisition request associated with the detected applet identification codes is performed by the CPU 50, beginning with Step S900 as shown in FIG. 16.

In Step S900, the CPU 50 sends a service applet acquisition request to the proxy server 200. Then, in Step S902, the CPU 50 downloads the service applet, and then goes to Step S904.

In Step S904, the CPU 50 allocates a memory area to the downloaded service applet and goes to Step S906. Then, in Step S906, the CPU 50 initializes the downloaded service applet to make it available for use, and then it goes to Step S908.

In Step S908, when basic functions of the service applet such as communications services have been initialized, the CPU 50 determines that memory area allocation to the downloaded service applet has been completed successfully and sends a download-complete notice to the proxy server 200, and then it goes to Step S910.

In Step S910, the CPU 50 judges whether the initialization of the service itself has failed. If it is judged that the initialization of the service itself has failed (Yes), the CPU 50 goes to Step S912, where it sends an initialization failure notice to the proxy server 200. Then, the CPU 50 finishes the sequence of processes and returns to the original process.

On the other hand, if it is judged in Step S910 that the initialization of the service itself has not failed (No), the CPU 50 finishes the sequence of processes and returns to the original process.

On the other hand, if the downloading of the service applet has not been completed in Step S902, being interrupted by a communications failure or the like, this means that the memory area allocation in Step S904 and consequently the initialization in Step S906 are not complete. Thus, the CPU 50 waits without going to Step S908.

Next, the screen update process will be described in detail with reference to FIG. 17.

FIG. 17 is a flowchart showing the screen update process.

The screen update process, which is performed as part of monitor applet processes and involves updating a touch-panel screen 60 according to an update command from the proxy server 200, is performed by the CPU 50, beginning with Step S1000 as shown in FIG. 17.

In Step S1000, the CPU 50 judges whether a full-screen update command has been received. If it is judged that a full-screen update command has been received, the CPU 50 goes to Step S1002, where it updates all the frames to which the service applets or monitor applet are

assigned from among the multiple frames composing a displayed Web page. Thus, in Step S1002, as the frames are updated in this way, the static memory management system releases the memory areas allocated to the applets.

In Step S1004, the CPU 50 judges whether a local-screen update command has been received. If it is judged that a local-screen update command has been received (Yes), the CPU 50 goes to Step S1006, where it updates the frames to which the monitor applet is assigned from among the multiple frames composing a displayed Web page. Then, the CPU 50 finishes the sequence of processes and returns to the original process. Thus, in Step S1006, as the frames are updated in this way, the static memory management system releases the memory areas allocated to the monitor applet.

On the other hand, if it is judged in Step S1004 that no local-screen update command has been received (No) or if it is judged in Step S1000 that no full-screen update command has been received (No), the CPU 50 finishes the sequence of processes and returns to the original process.

Next, the reboot process will be described in detail with reference to FIG. 18.

FIG. 18 is a flowchart showing the reboot process.

The reboot process, which is the process of rebooting the browser terminal 300 at a reboot command from the proxy server 200, is performed by the CPU 50, beginning with Step S1100 as shown in FIG. 18.

In Step S1100, the CPU 50 judges whether a reboot command has been received via SNMP. If it is judged that a reboot command has been received (Yes), the CPU 50 goes to Step S1102, where it reboots the browser terminal 300. Then, the CPU 50 finishes the sequence of processes and returns to the original process.

On the other hand, if it is judged in Step S1100 that no reboot command has been received via SNMP(No), the CPU 50 finishes the sequence of processes and returns to the original process.

Next, operation of this embodiment will be described with reference to FIGS. 19 and 20.

FIG. 19A and FIG. 19B show display screens of the browser terminals 300.

FIG. 20 is a diagram illustrating the operation of updating the display screen of the browser terminal 300.

First, description will be given of the process of registering the presence or absence of service applets

used by Web pages which may be accessed by the terminals connecting to the private network 199.

The proxy server 200 performs an applet presence registration process upon power-up. When the applet presence registration process is started, the proxy server 200 goes through Steps S100 to S104 to read the URL from the first record of the access registration table 400, access the WWW server 100 based on the URL, and acquire the Web page from the accessed WWW server 100.

Then, the proxy server 200 goes through Steps S106 to S110 to parse the acquired Web page, extract any applet identification code found to be contained in the acquired Web page, and access the WWW server 100 relevant to the specific acquisition request associated with the detected applet identification code. If it is found that the service applet requested by the specific acquisition request exists on the WWW server 100, the proxy server 200 goes through Step S114 to register information that the service applet which corresponds to the detected applet identification code exists on the WWW server 100, in the applet management table 440 and applet presence management table 460. On the other hand, if the service applet requested by the specific acquisition request does not exist on the WWW server 100, the proxy server 200 goes through Step S120 to register information that the service applet which corresponds to the detected applet identification code does not exist on the WWW server 100, in the applet management table 440 and applet presence management table 460.

As the above processes are performed for all the URLs registered in the access registration table 400, the presence or absence of the service applets used by Web pages which may be accessed by the terminals connecting to the private network 199 are registered with the proxy server 200.

Next, description will be given of how a monitor applet is applied to the browser terminal 300.

The browser terminal 300 performs a monitor applet acquisition process upon startup of the Web browser. The browser terminal starts the monitor applet acquisition process by sending a monitor applet acquisition request to the proxy server 200 in Step S800.

Upon receiving the monitor applet acquisition request, since no monitor applet exists on the browser terminal 300 just after startup, the proxy server 200 goes through Steps S300 to S308 to download the monitor

applet requested by the monitor applet acquisition request to the browser terminal 300.

When the downloading of the monitor applet is completed, the browser terminal 300 goes through Step S808 to send a download-complete notice to the proxy server 200.

Upon receiving the download-complete notice, the proxy server 200 goes through Steps S314 and S316 to register the presence of the monitor applet, in the terminal management table 420.

When the downloading of the monitor applet is completed, the browser terminal 300 also goes through Steps S804 and S806 to allocate a memory area to the monitor applet and initialize the monitor applet to make it available for use. If the monitor applet is initialized successfully, it runs properly, allowing service applets to be executed.

Next, description will be given of how the proxy server 200 mediates data acquisition.

When the user operates the Web browser, the browser terminal 300 sends an acquisition request for a Web page the user intends to view to the proxy server 200. The acquisition request is sent as an HTTP packet.

When the acquisition request is received, if the IP address contained in the acquisition request has not been registered in the terminal management table 420, the proxy server 200 goes through Steps S401 to S404 to parse the HTTP header of the acquisition request and judge whether the requesting terminal is a browser terminal 300 based on the results of the parsing. Since the requesting terminal is the browser terminal 300 in this case, the proxy server 200 goes through Steps S405 to S408 to register the IP address contained in the acquisition request and terminal identification information which indicates the browser terminal 300 in the terminal management table 420, access the WWW server 100 relevant to the acquisition request, and acquire the Web page requested by the acquisition request, from the WWW server 100.

Then, the proxy server 200 goes through Steps S410 and S412 to parse the acquired Web page and extract any applet identification code found to be contained in the acquired Web page. If the service applet has not been registered in the applet management table 440, the proxy server 200 goes through Step S416 to access the WWW server 100 relevant to the specific acquisition request associated with the detected applet identification code. If it is found that the service applet requested by the

specific acquisition request exists on the WWW server 100, the proxy server 200 goes through Steps S420 and S422 to register information that the service applet which corresponds to the extracted applet identification code exists on the WWW server 100, in the applet management table 440 and applet presence management table 460 and increment the number of service applet transmissions for the requesting browser terminal 300 in the terminal management table 420.

On the other hand, if it is judged that the service applet requested by the specific acquisition request does not exist on the WWW server 100, the proxy server 200 goes through Steps S432 and S434 to register information that the service applet which corresponds to the detected applet identification code does not exist on the WWW server 100, in the applet management table 440 and applet presence management table 460 and send a substitute page instead of the acquired Web page to the browser terminal 300, notifying it that the service applet does not exist.

Upon receiving the substitute page, the browser terminal 300 displays it as shown in FIG. 19A or FIG. 19B. For example, if the browser terminal 300 is installed at a shop or the like for customer use, the customer who looks at the display knows something is wrong and can take appropriate measures such as calling a shop employee. Since measures can be taken before running the applet, it is possible to reduce the chance of rendering the browser terminal 300 inoperable.

Next, description will be given of how a service applet is applied to the browser terminal 300.

If an applet identification code is contained in a displayed Web page composed of multiple frames, the browser terminal 300 detects the applet identification code in the Web page as shown in FIG. 20, and then goes through Step S900 to send the proxy server 200 a service applet acquisition request for the service applet which corresponds to the detected applet identification code.

Upon receiving the service applet acquisition request, since the service applet does not exist on the browser terminal 300, the proxy server 200 goes through Steps S500 to S508 to download the service applet requested by the service applet acquisition request to the browser terminal 300.

When the downloading of the service applet is completed, the browser terminal 300 goes through Step S908 to send a download-complete notice to the proxy server 200.

Upon receiving the download-complete notice, the proxy server 200 goes through Steps S514 and S516 to register the presence of the service applet, in the terminal management table 420.

When the downloading of the service applet is completed, the browser terminal 300 also goes through Steps S904 and S906 to allocate a memory area to the service applet and initialize the service applet to make it available for use. If the service applet is initialized successfully, the service applet can be executed properly.

On the other hand, if the download fails halfway due to a communications failure or the like, the browser terminal 300 does not send a download-complete notice to the proxy server 200 before an expiration time.

If a download-complete notice is not received before the expiration time, a timeout occurs and the download is judged to have failed. Consequently, the proxy server 200 goes through Step S526 to set the update mode to "STOP," and then goes through Steps S712 and S714 to send a full-screen update command to the browser terminal 300.

The browser terminal 300, which runs the monitor applet, goes through Steps S1000 and S1002 to update all the frames to which the service applets or monitor applet are assigned from among the multiple frames. When the frames are updated, the static memory management system releases the memory areas allocated to the applets.

On the other hand, if the applet initialization fails due to user error in terminal operation or the like, the browser terminal 300 goes through Step S912 to send an initialization failure notice to the proxy server 200.

Upon receiving the initialization failure notice, the proxy server 200 sends the service applet a request to terminate itself, goes through Step S524 to set the update mode to "REFRESH," and goes through Steps S716 and S718 to send a local-screen update command to the browser terminal 300.

The browser terminal 300, which runs the monitor applet, goes through Steps S1004 and S1006 to update the frames to which the monitor applet is assigned from among the multiple frames. When the frames are updated, the static memory management system releases and initializes the memory areas allocated to the monitor applet as well as all the memory managed together with them.

On the other hand, when the current mode is forced update mode, if the number of service applet transmissions has reached a predetermined number, the

proxy server 200 goes through Steps S604 and S608 to set the update mode to "REFRESH" and send a local-screen update command to the browser terminal 300.

Upon receiving the local-screen update command, the browser terminal 300 similarly updates the frames to which the monitor applet is assigned from among the multiple frames, thereby releasing and initializing the memory areas allocated to the monitor applet as well as all the memory managed together with them.

On the other hand, when the current mode is transitional update mode, if an idle URL is requested by the browser terminal 300 and if the number of service applet transmissions has reached a predetermined number, the proxy server 200 goes through Steps S612, S604, and S608 to set the update mode to "REFRESH" and send a local-screen update command to the browser terminal 300.

Upon receiving the local-screen update command, the browser terminal 300 similarly updates the frames to which the monitor applet is assigned from among the multiple frames, thereby releasing and initializing the memory areas allocated to the monitor applet as well as all the memory managed together with them.

On the other hand, if the browser terminal 300 is left idle for some time after the user finishes using it, the proxy server 200 goes through Steps S610 and S608 to detect that the service applet's service has timed out, set the update mode to "REFRESH," and send a local-screen update command to the browser terminal 300.

Upon receiving the local-screen update command, the browser terminal 300 similarly updates the frames to which the monitor applet is assigned from among the multiple frames, thereby releasing and initializing the memory areas allocated to the monitor applet as well as all the memory managed together with them.

On the other hand, if it is judged that the service applet requested by the service applet acquisition request exists on the requesting browser terminal 300, the proxy server 200 goes through Steps S528, S712, S714 to set the update mode to "STOP" and send a full-screen update command to the browser terminal 300.

Upon receiving the full-screen update command, the browser terminal 300 updates all the frames to which the service applets or monitor applet are assigned from among the multiple frames, thereby releasing and initializing the memory areas allocated to the applets.

On the other hand, if it is judged that the monitor applet requested by the monitor applet acquisition request exists on the requesting browser terminal 300 or

if it is judged that a service applet exists even though no monitor applet exists, the proxy server 200 goes through Steps S328, S700, S702 to set the update mode to "REBOOT" and send a reboot command to the browser terminal 300 via SNMP.

Upon receiving the reboot command via SNMP, the browser terminal 300 goes through Steps S1100 and S1102 to reboot itself. After the reboot, the Web browser starts up and a monitor applet acquisition process is performed.

Subsequently, to view the same Web page on another browser terminal 300, when the user operates a Web browser, the browser terminal 300 sends an acquisition request for the Web page to the proxy server 200.

Upon receiving the acquisition request, the proxy server 200 parses the HTTP header of the acquisition request and judges whether the requesting terminal is a browser terminal 300 based on the results of the parsing. Since the requesting terminal is judged to be the browser terminal 300 in this case, the proxy server 200 accesses the WWW server 100 relevant to the acquisition request and acquires the Web page requested by the acquisition request, from the WWW server 100.

Then, the proxy server 200 parses the acquired Web page and detects any applet identification code contained in the acquired Web page. Since the service applet which corresponds to the detected applet identification code has been registered in the applet presence management table 460 as described above, the proxy server 200 goes through Steps S436 and S434, where it judges that the service applet requested by the specific acquisition request does not exist on the WWW server 100 and sends a substitute page instead of the acquired Web page to the browser terminal 300, notifying it that the service applet does not exist.

Incidentally, if the parsing of the HTTP header shows that the requesting terminal is an ordinary terminal 310 or if no applet identification code is contained in the acquired Web page, the proxy server 200 sends the acquired Web page as it is to the requesting terminal.

In this way, according to this embodiment, the proxy server 200 is communicably connected with the browser terminal 300 which allocates and deallocates memory areas to/from applets, according to a static memory management system, and with a plurality of WWW servers 100, acquires, according to an acquisition request, an Web page from a WWW server 100 from the browser terminal 300,

and sends the acquired Web page to the requesting browser terminal 300. On the other hand, if it detects that the browser terminal 300 has failed to acquire an applet, the proxy server 200 sends a full-screen update command to the browser terminal 300.

As described above, if the browser terminal 300 fails to acquire an applet, it updates all the frames to which the service applets or monitor applet are assigned from among the multiple frames, thereby releasing resources including the memory areas allocated to the applets. This makes it possible to reduce the chance of rendering the browser terminal 300 inoperable compared to conventional methods.

Furthermore, according to this embodiment, if the proxy server 200 detects that the browser terminal 300 has failed to initialize an applet, the proxy server 200 sends a full-screen update command to the browser terminal 300.

Consequently, if the browser terminal 300 fails to initialize an applet, it updates the frames to which the monitor applet is assigned from among the multiple frames in the absence of a service applet, thereby releasing and initializing the memory areas allocated to the monitor applet as well as all the memory managed together with them. This makes it possible to reduce the chance of rendering the browser terminal 300 inoperable due to a lack of memory resources.

Furthermore, according to this embodiment, upon detecting a service timeout of a service applet applied to the browser terminal 300, the proxy server 200 sends a local-screen update command to the browser terminal 300.

Consequently, if the service applet's service times out on the browser terminal 300, i.e., if the user finishes using the browser terminal 300 or leaves it idle for some time, the service applet is terminated automatically. Subsequently, the browser terminal 300 updates the frames to which the monitor applet is assigned from among the multiple frames, thereby releasing and initializing the memory areas allocated to the monitor applet as well as all the memory managed together with them. This makes it possible to further reduce the chance of rendering the browser terminal 300 inoperable due to a lack of memory resources.

Furthermore, according to this embodiment, the proxy server 200 counts the number of service applet transmissions to the browser terminal 300 and sends a local-screen update command to the browser terminal 300 based on the result of counting.

Consequently, memory areas are released and initialized according to the number of service applet transmissions, making it possible to further reduce the chance of rendering the browser terminal 300 which employs the static memory management system inoperable due to a lack of memory resources.

Furthermore, according to this embodiment, the proxy server 200 acquires, according to an acquisition request, a Web page, detects any specific acquisition request contained in the acquired Web page, and judges whether the service applet requested by the specific acquisition request exists on a WWW server 100. Then, if it is judged that the service applet requested by the specific acquisition request does not exist on the WWW server 100, the proxy server 200 sends a substitute page instead of the acquired Web page to the requesting browser terminal 300, notifying it that the service applet does not exist.

Consequently, if the service applet requested by the specific acquisition request does not exist on the WWW server 100, the proxy server 200 can notify the requesting terminal that the service applet does not exist before the requesting terminal sends the specific acquisition request. This makes it possible to further reduce the chance of rendering the browser terminal 300 inoperable. Besides, this capability can be implemented without using a Back button, and thus the reliability of service provision will not be impaired very much.

Furthermore, according to this embodiment, the proxy server 200 detects a specific acquisition request contained in an acquired Web page by detecting an applet identification code contained in the acquired Web page.

This makes it possible to detect a specific acquisition request with relatively high reliability.

Furthermore, according to this embodiment, when it is found that a specific acquisition request is contained in an acquired Web page, the proxy server 200 parses the Web page containing the specific acquisition request. Only if the results of the parsing indicates that the service applet has not been registered in the applet management table 440 and applet presence management table 460, the proxy server 200 accesses the WWW server 100 and judges whether the service applet requested by the specific acquisition request exists on the WWW server 100.

Consequently, the proxy server 200 accesses the WWW server 100 only in relation to those service applets whose presence has never been checked for. This makes it

possible to reduce the number of accesses to the WWW server 100.

Furthermore, according to this embodiment, upon receiving an acquisition request, the proxy server 200 judges whether the requesting terminal is a browser terminal 300 based on parsing results of the received acquisition request. If it is found that the requesting terminal is a browser terminal 300, the proxy server 200 detects a specific acquisition request and checks for the presence of service applets.

Consequently, the proxy server 200 detects a specific acquisition request and checks for the presence of service applets only in relation to the Web pages which should be sent to the browser terminal 300. This reduces processing loads.

Furthermore, according to this embodiment, before receiving acquisition request from the browser terminal 300, the proxy server 200 accesses the WWW server 100 based on the URLs registered in the access registration table 400 and judges whether relevant service applets exist, and then registers the presence/absence of the service applets, in the applet management table 440 and applet presence management table 460.

Consequently, since registrations in the applet management table 440 and applet presence management table 460 are made before any acquisition request is received, it is possible to improve the response rate with respect to the browser terminal 300.

Furthermore, according to this embodiment, upon receiving an acquisition request, the proxy server 200 parses the HTTP header of the acquisition request, judges whether the requesting terminal is a browser terminal 300 based on the results of the parsing, and registers the result of judgment in the terminal management table 420. On the other hand, if the IP address contained in the acquisition request has already been registered in the terminal management table 420, the proxy server 200 judges whether the requesting terminal is a browser terminal 300 by referring to the terminal management table 420.

In this way, once registration is made in the terminal management table 420, the proxy server 200 does not need to parse the HTTP header to distinguish the requesting terminal, which makes it possible to reduce the processing load on the proxy server 200 and improve the response rate with respect to the browser terminal 300.

In the above embodiment, the Internet 198 and private network 199 correspond to the network described in inventions 4, 5, 7, 26 to 28, and 36 to 38; and the WWW server 100 corresponds to the network resource described in inventions 4, 5, 7, 16, 18 to 22, 26 to 30, and 36 to 38. Also, the storage device 42 corresponds to the storage means described in invention 18 or 19; and Steps S310 and S510 correspond to the acquisition failure detection means described in inventions 1, 4, 6, 8, 10 to 12, 23 or 26 or the acquisition failure detection step described in invention 33 or 36.

Also, in the above embodiment, Steps S318 and S518 correspond to the initialization failure detection means described in inventions 2, 5, 6, 9 to 11, 13, 24 or 27 or the initialization failure detection step described in invention 34 or 37; and Steps S406 and S408 correspond to the server-side data acquisition means described in inventions 4, 5, 7, 16, 17 and 26 to 28 or the server-side data acquisition step described in inventions 36 to 38. Also, Steps S410 and S412 correspond to the specific acquisition request detection means described in invention 16, 18 or 20; Steps S416 and S418 correspond to the data presence judging means described in inventions 16 to 20; and Steps S422 and S604 correspond to the acquisition count detection means described in invention 11 or 15.

Also, in the above embodiment, Steps S430 and S434 correspond to the data transmission means described in inventions 4, 5, 7, 16 and 26 to 28 or the data transmission step described in invention 36 to 38; and Step S610 corresponds to the timeout detection means described in inventions 3, 7 to 11, 14, 25 or 28 or the timeout detection step described in invention 35 or 38. Also, Steps S714 and S718 correspond to the update command output means described in inventions 1 to 3 and 23 to 25, the update command transmission means described in inventions 4 to 15 and 26 to 28, the update command output step described in inventions 33 to 35, or the update command transmission step described in inventions 36 to 38.

Also, in the above embodiment, Steps S802 and S902 correspond to the data acquisition means described in inventions 1 to 3, 23 to 25 and 33 to 35 or the terminal-side data acquisition means described in inventions 4 to 8, 11, 21, 22, 26 to 30, and 36 to 38; and Steps S804 and S904 correspond to the memory area allocation means described in inventions 1 to 5, 7, 21 to 30 and 33 to 38. Also, Steps S806 and S906 correspond to the data

initialization means described in invention 2, 5, 9, 10, 24, 27, 34 or 37; and Steps S1002 and S1006 correspond to the memory area releasing means described in inventions 1 to 5, 7, 12 to 15, 21 to 30 and 33 to 38.

Also, in the above embodiment, the substitute page corresponds to the notification data described in invention 17; the full-screen update command and local-screen update command correspond to the update command described in inventions 1 to 15 and 21 to 38; and the applets correspond to the data in a specific format described in inventions 1 to 19, 21 to 30 and 33 to 38. Also, the monitor applet corresponds to the monitor program data described in inventions 12 to 15, 21, 22 and 29 to 32; the service applet corresponds to the service program data described in inventions 12 to 15, 21, 22 and 29 to 31; and the URL corresponds to the data identification information described in invention 18 or 19 or the acquisition source address described in invention 18 or 19.

Also, in the above embodiment, the IP address corresponds to the network resource identification information described in invention 20; and the frame corresponds to the display section in inventions 12 to 15, 21, 22 and 29 to 32.

Incidentally, although according to the above embodiment, the proxy server 200 always parses acquired Web pages in Steps S106 and S410, this is not restrictive. Alternatively, the proxy server 200 may judge whether the URL of an acquired Web page has been registered in the applet management table 440 and may omit parsing the Web page if it is judged that the URL has been registered in the applet management table 440.

In that case, once Web page is registered in the applet management table 440, the proxy server 200 does not need to parse the Web page, which makes it possible to reduce the processing load on the proxy server 200 and improve the response rate with respect to the requesting terminal.

Also, although according to the above embodiment, the applet presence management table 460 shown in FIG. 6 is provided for each Web page, this is not restrictive. Alternatively, the plurality of applet presence management tables 460 may be integrated into a single table, which may speed up search operations.

Also, although according to the above embodiment, a reboot command is sent from the proxy server 200 to the browser terminal 300 via SNMP, this is not restrictive. Alternatively, it is possible to use a protocol other

than SNMP or use the protocol used for transmission of the full-screen update command or local-screen update command.

Also, although according to the above embodiment, the requesting terminal is identified by the IP address contained in the acquisition request, this is not restrictive. Alternatively, it is possible to identify the requesting terminal by user information or the like exchanged by means of a Cookie.

Also, although according to the above embodiment, the processes shown in the flowcharts of FIGS 7 to 13 are performed according to the control program stored in the ROM 32, this is not restrictive. Alternatively, it is possible to load a program for the procedures from a storage medium into the RAM 34 for execution.

Also, although according to the above embodiment, the processes shown in the flowcharts of FIGS 15 to 18 are performed according to the control program stored in the ROM 52, this is not restrictive. Alternatively, it is possible to load a program for the procedures from a storage medium into the RAM 54 for execution.

The storage medium here may be of any type, provided it is computer-readable, regardless of whether it is electronic, magnetic, or optical: it may be a semiconductor storage medium such as a RAM or ROM; magnetic storage medium such as an FD or HD; optical storage medium such as a CD, CDV, LD, or DVD; or magneto-optical storage medium such as an MO.

Also, although according to the above embodiment, the memory management device, proxy server, browser terminal, memory management program, server program, terminal program, monitor program, and memory management method according to the present invention are applied to a network system constituted by the Internet 198, this is not restrictive. They can also be applied to intranets which conduct communications in the manner of the Internet 198. Of course, they can also be applied to ordinary networks in addition to the networks which conduct communications in the manner of the Internet 198.

Also, although according to the above embodiment, the memory management device, proxy server, browser terminal, memory management program, server program, terminal program, monitor program, and memory management method according to the present invention are applied to the proxy server 200 installed between the Internet 198 and private network 199 as well as to the browser terminals 300 as shown in FIG. 1, this is not restrictive. They can have other applications without

departing from the spirit of the present invention. For example, the present invention may be configured as software which can be used to incorporate the processes implemented by the proxy server 200 into the browser terminals 300.

As described above, with the memory management device set forth in claim 1 of the present invention or the proxy server set forth in claims 4, 6, 8, and 11 to 20 of the present invention, memory areas are released upon failure to acquire data in a specific format. This has the effect of reducing the chance of rendering a browser terminal inoperable due to a lack of memory resources when the browser terminal employs a static memory management system or a dynamic memory management scheme which may allow executable programs to run without releasing resources.

Furthermore, with the memory management device set forth in claim 2 of the present invention or the proxy server set forth in claims 5, and 9 to 20 of the present invention, memory areas are released upon failure to initialize data in a specific format. This has the effect of reducing the chance of rendering a browser terminal inoperable due to a lack of memory resources when the browser terminal employs a static memory management system or a dynamic memory management scheme which may allow executable programs to run without releasing resources.

Furthermore, with the memory management device set forth in claim 3 of the present invention or the proxy server set forth in claims 7 and 11 to 20 of the present invention, memory areas are released when a service using data in a specific format times out. That is, some time after the user finishes using a browser terminal, the service using the data in the specific format times out, releasing the memory areas. This has the effect of reducing the chance of rendering the browser terminal inoperable due to a lack of memory resources when the browser terminal employs a static memory management system or a dynamic memory management scheme which may allow executable programs to run without releasing resources.

Furthermore, with the proxy server set forth in claim 11 or 15 of the present invention, memory areas are released according to the acquisition count of data in a specific format. This has the effect of further reducing the chance of rendering a browser terminal inoperable due to a lack of memory resources when the browser terminal employs a static memory management system or a dynamic

memory management scheme which may allow executable programs to run without releasing resources.

Furthermore, in case of a failure to acquire data in a specific format, the proxy server set forth in claim 12 of the present invention updates all the classification display areas to which service program data or the monitor program data are assigned from among multiple classification display areas, thereby releasing the memory areas allocated to the program data. This has the effect of further reducing the chance of rendering a browser terminal inoperable.

Furthermore, in case of a failure to initialize data in a specific format, the proxy server set forth in claim 13 of the present invention updates the classification display areas to which monitor program data is assigned from among multiple classification display areas, thereby releasing the memory areas allocated to the monitor program data. This has the effect of further reducing the chance of rendering a browser terminal inoperable.

Furthermore, when a service using the data in a specific format times out on a browser terminal, the proxy server set forth in claim 14 of the present invention updates the classification display areas to which monitor program data is assigned from among multiple classification display areas, thereby releasing the memory areas allocated to the monitor program data. This has the effect of further reducing the chance of rendering a browser terminal inoperable.

Furthermore, when the acquisition count of data in a specific format reaches a predetermined number, the proxy server set forth in claim 15 of the present invention updates the classification display areas to which monitor program data is assigned from among multiple classification display areas, thereby releasing the memory areas allocated to the monitor program data. This has the effect of further reducing the chance of rendering a browser terminal inoperable.

Furthermore, if data in a specific format requested by a specific acquisition request does not exist in a target network resource, the proxy server set forth in claims 16 to 20 of the present invention can notify the browser terminal about absence of the data in the specific format. This has the effect of reducing the risk of impairing the reliability of service provision on the browser terminal as well as reducing the chance of rendering the browser terminal inoperable by waiting for the data in the specific format.

Furthermore, if data in a specific format requested by a specific acquisition request does not exist in a target network resource, the proxy server set forth in claim 17 of the present invention can notify the browser terminal about absence of the data in the specific format before the browser terminal sends the specific acquisition request. This has the effect of further reducing the chance of rendering the browser terminal inoperable by waiting for the data in the specific format.

Furthermore, the proxy server set forth in claim 18 or 19 of the present invention accesses a target network resource only in relation to data in a specific format whose presence has never been checked for. This has the effect of reducing the number of accesses to the target network resource.

Furthermore, with the proxy server set forth in claim 19 of the present invention, registration in the storage means is made before an acquisition request is received. This has the effect of improving the response rate with respect to the browser terminal.

Furthermore, with the proxy server set forth in claim 20 of the present invention, the specific acquisition request detection means performs detection and the data presence judging means makes judgment only in relation to the data which should be sent to the browser terminal. This has the effect of reducing processing loads.

On the other hand, the browser terminal set forth in claim 21 of the present invention has an effect equivalent to that of the proxy server set forth in claim 12.

Furthermore, the browser terminal set forth in claim 22 of the present invention has an effect equivalent to that of the proxy server set forth in claim 13 to 15.

Furthermore, the memory management program set forth in claim 23 of the present invention has an effect equivalent to that of the memory management device set forth in claim 1.

Furthermore, the memory management program set forth in claim 24 of the present invention has an effect equivalent to that of the memory management device set forth in claim 2.

Furthermore, the memory management program set forth in claim 25 of the present invention has an effect equivalent to that of the memory management device set forth in claim 3.

Furthermore, the server program set forth in claim 26 of the present invention has an effect equivalent to that of the proxy server set forth in claim 4.

Furthermore, the server program set forth in claim 27 of the present invention has an effect equivalent to that of the proxy server set forth in claim 5.

Furthermore, the server program set forth in claim 28 of the present invention has an effect equivalent to that of the proxy server set forth in claim 7.

On the other hand, the terminal program set forth in claim 29 of the present invention has an effect equivalent to that of the browser terminal set forth in claim 21.

Furthermore, the terminal program set forth in claim 30 of the present invention has an effect equivalent to that of the browser terminal set forth in claim 22.

On the other hand, the monitor program set forth in claim 31 of the present invention has an effect equivalent to that of the proxy server set forth in claim 12.

Furthermore, the monitor program set forth in claim 32 of the present invention has an effect equivalent to that of the proxy server set forth in claim 13 to 15.

On the other hand, the memory management method set forth in claim 33 of the present invention has an effect equivalent to that of the memory management device set forth in claim 1.

Furthermore, the memory management method set forth in claim 34 of the present invention has an effect equivalent to that of the memory management device set forth in claim 2.

Furthermore, the memory management method set forth in claim 35 of the present invention has an effect equivalent to that of the memory management device set forth in claim 3.

Furthermore, the memory management method set forth in claim 36 of the present invention has an effect equivalent to that of the proxy server set forth in claim 4.

Furthermore, the memory management method set forth in claim 37 of the present invention has an effect equivalent to that of the proxy server set forth in claim 5.

Furthermore, the memory management method set forth in claim 38 of the present invention has an effect equivalent to that of the proxy server set forth in claim 7.

What is claimed is:

1. A memory management device applied to a memory management system which comprises data acquisition means for acquiring data, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, the memory management device comprising:

acquisition failure detection means for detecting whether the data acquisition means has failed to acquire the data in the specific format; and update command output means for outputting the update command to the memory management system based on a detection result from the acquisition failure detection means.

2. A memory management device applied to a memory management system which comprises data acquisition means for acquiring data, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the data acquisition means, data initialization means for initializing the data in the specific format acquired by the data acquisition means to make it available for use, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, the memory management device comprising:

initialization failure detection means for detecting whether the data initialization means has failed to initialize the data in the specific format; and update command output means for outputting the update command to the memory management system based on a detection result from the initialization failure detection means.

3. A memory management device applied to a memory management system which comprises data acquisition means for acquiring data, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, the memory management device comprising:

timeout detection means for detecting whether a service using the data in the specific format acquired by the data acquisition means has timed out; and update command output means for outputting the update command to the memory management system based on a detection result from the timeout detection means.

4. A proxy server which communicably connects via a network to browser terminal and to network resource other than the browser terminal and which, upon receiving a data acquisition request from the browser terminal, acquires data specified by the received acquisition request from a target network resource and sends the acquired data to the browser terminal, the browser terminal comprising terminal-side data acquisition means for acquiring data from a network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the terminal-side data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, the proxy server comprising:

server-side data acquisition means for acquiring, according to the acquisition request, data from the network resource; data transmission means for sending the data acquired by the server-side data acquisition means to the browser terminal; acquisition failure detection means for detecting via communications with the browser terminal whether the terminal-side data acquisition means has failed to acquire the data in the specific format; and update command transmission means for sending the update command to the browser terminal based on a detection result from the acquisition failure detection means.

5. A proxy server which communicably connects via a network to browser terminal and to network resource other than the browser terminal and which, upon receiving a data acquisition request from the browser terminal, acquires data specified by the received acquisition request from a target network resource and sends the acquired data to the browser terminal, the browser terminal comprising terminal-side data acquisition means for acquiring data from a network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the

terminal-side data acquisition means, data initialization means for initializing the data in the specific format acquired by the terminal-side data acquisition means to make it available for use, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, the proxy server comprising:

server-side data acquisition means for acquiring, according to the acquisition request, data from the network resource; data transmission means for sending the data acquired by the server-side data acquisition means to the browser terminal; initialization failure detection means for detecting via communications with the browser terminal whether the data initialization means has failed to initialize the data in the specific format; and update command transmission means for sending the update command to the browser terminal based on a detection result from the initialization failure detection means.

6. The proxy server according to claim 5, further comprising:

acquisition failure detection means for detecting via communications with the browser terminal whether the terminal-side data acquisition means has failed to acquire the data in the specific format, wherein

the update command transmission means sends the update command to the browser terminal based on a detection result from the initialization failure detection means or the acquisition failure detection means.

7. A proxy server which communicably connects via a network to browser terminal and to network resource other than the browser terminal and which, upon receiving a data acquisition request from the browser terminal, acquires data specified by the received acquisition request from a target network resource and sends the acquired data to the browser terminal, the browser terminal comprising terminal-side data acquisition means for acquiring data from a network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the terminal-side data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, the proxy server comprising:

server-side data acquisition means for acquiring, according to the acquisition request, data from the network resource; data transmission means for sending the data acquired by the server-side data acquisition means to the browser terminal; timeout detection means for detecting via communications with the browser terminal whether a service using the data in the specific format acquired by the terminal-side data acquisition means has timed out; and update command transmission means for sending the update command to the browser terminal based on a detection result from the timeout detection means.

8. The proxy server according to claim 7, further comprising:

acquisition failure detection means for detecting via communications with the browser terminal whether the terminal-side data acquisition means has failed to acquire the data in the specific format, wherein

the update command transmission means sends the update command to the browser terminal based on a detection result from the timeout detection means or the acquisition failure detection means.

9. The proxy server according to claim 7, further comprising:

initialization failure detection means for detecting via communications with the browser terminal whether the data initialization means has failed to initialize the data in the specific format, wherein

the update command transmission means sends the update command to the browser terminal based on a detection result from the timeout detection means or the initialization failure detection means.

10. The proxy server according to claim 8, further comprising:

initialization failure detection means for detecting via communications with the browser terminal whether the data initialization means has failed to initialize the data in the specific format, wherein

the update command transmission means sends the update command to the browser terminal based on a detection result from the timeout detection means, the acquisition failure detection means, or the initialization failure detection means.

11. The proxy server according to claim 10, further comprising:

acquisition count detection means for detecting whether an acquisition count of the data in the specific format acquired by the terminal-side data acquisition means has reached a predetermined number, wherein

the update command transmission means sends the update command to the browser terminal based on a detection result from the timeout detection means, the acquisition failure detection means, the initialization failure detection means, or the acquisition count detection means.

12. The proxy server according to claim 10 or 11, wherein:

the data in the specific format is either executable service program data which is assigned to one of classification display areas obtained by dividing a display screen into multiple areas and composes a service program that provides a service in the given display area or executable monitor program data which is assigned to one of the classification display areas and composes a monitor program that monitors one or more service programs;

the monitor program executes processes implemented as the memory area releasing means and upon receiving a full-screen update command, updates all the classification display areas to which the service program data or the monitor program data are assigned from among the multiple classification display areas, thereby releasing the memory areas allocated to the program data; and

the update command transmission means sends the full-screen update command to the browser terminal if the acquisition failure detection means detects an acquisition failure.

13. The proxy server according to claim 10 or 11, wherein:

the data in the specific format is either executable service program data which is assigned to one of classification display areas obtained by dividing a display screen into multiple areas and composes a service program that provides a service in the given display area or executable monitor program data which is assigned to one of the classification display areas and composes a monitor program that monitors one or more service programs;

the monitor program executes processes implemented as the memory area releasing means and upon receiving a

local-screen update command, updates the display area to which the monitor program data is assigned from among the multiple classification display areas, thereby releasing the memory area allocated to the monitor program data; and

the update command transmission means sends the local-screen update command to the browser terminal if the initialization failure detection means detects an initialization failure.

14. The proxy server according to claim 10 or 11, wherein:

the data in the specific format is either executable service program data which is assigned to one of classification display areas obtained by dividing a display screen into multiple areas and composes a service program that provides a service in the given display area or executable monitor program data which is assigned to one of the classification display areas and composes a monitor program that monitors one or more service programs;

the monitor program executes processes implemented as the memory area releasing means and upon receiving a local-screen update command, updates the display area to which the monitor program data is assigned from among the multiple classification display areas, thereby releasing the memory area allocated to the monitor program data; and

the update command transmission means sends the local-screen update command to the browser terminal if the timeout detection means detects a service timeout.

15. The proxy server according to claim 11, wherein:

the data in the specific format is either executable service program data which is assigned to one of classification display areas obtained by dividing a display screen into multiple areas and composes a service program that provides a service in the given display area or executable monitor program data which is assigned to one of the classification display areas and composes a monitor program that monitors one or more service programs;

the monitor program executes processes implemented as the memory area releasing means and upon receiving a local-screen update command, updates the display area to which the monitor program data is assigned from among the multiple classification display areas, thereby releasing

the memory area allocated to the monitor program data; and

the update command transmission means sends the local-screen update command to the browser terminal if the acquisition count detection means detects that the predetermined number is reached.

16. The proxy server according to any of claims 4 to 15, further comprising:

specific acquisition request detection means for detecting whether the data acquired by the server-side data acquisition means contains a specific acquisition request to acquire the data in the specific format; and

data presence judging means for judging whether the data in the specific format requested by the specific acquisition request exists in a target network resource based on a detection result from the specific acquisition request detection means, wherein

if the data presence judging means judges that the data in the specific format requested by the specific acquisition request contained in the data acquired by the server-side data acquisition means does not exist, the data transmission means sends notice that the data in the specific format does not exist.

17. The proxy server according to claim 16, wherein:

if the data presence judging means judges that the data in the specific format requested by the specific acquisition request contained in the data acquired by the server-side data acquisition means does not exist, the data transmission means sends notification data to the effect that the data in the specific format does not exist to the browser terminal instead of sending the data in the specific format.

18. The proxy server according to claim 16 or 17, wherein:

after accessing the target network resource and judging whether the data in the specific format requested by the specific acquisition request exists there, the data presence judging means registers data identification information for identifying the data in the specific format and an acquisition source address of the data in the specific format in storage means, by associating them with each other; and

if the specific acquisition request detection means detects the specific acquisition request, the data presence judging means acquires the data identification

information and the acquisition source address from the data which contains the specific acquisition request, and then if the acquired data identification information and acquisition source address have not been registered in the storage means, the data presence judging means accesses the target network resource and judges whether or not the data in the specific format requested by the specific acquisition request exists there.

19. The proxy server according to claim 18, wherein:

the storage means stores an access registration table in which the acquisition source address is registered for each item of the data in the specific format which will be accessed by the browser terminal; and

before receiving acquisition request from the browser terminal, the data presence judging means accesses the target network resource based on each acquisition source address in the access registration table, judges whether or not the appropriate data in the specific format exists, and registers the data identification information for identifying the data in the specific format and acquisition source address of the data in the specific format in the storage means, by associating them with each other.

20. The proxy server according to any of claims 16 to 19, wherein upon receiving an acquisition request containing network resource identification information for identifying a network resource, the proxy server extracts the network resource identification information from the received acquisition request, and then if it is judged that the network resource identified by the extracted network resource identification information is the browser terminal, the specific acquisition request detection means performs detection and the data presence judging means makes judgment.

21. A browser terminal communicably connected to the proxy server set forth in claim 12, the browser terminal comprising:

terminal-side data acquisition means for acquiring data from a network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the terminal-side data acquisition means, and memory area releasing means for releasing, according to an update command, the memory

areas allocated to the data in the specific format, wherein:

the data in the specific format is either executable service program data which is assigned to one of classification display areas obtained by dividing a display screen into multiple areas and composes a service program that provides a service in the given display area or executable monitor program data which is assigned to one of the classification display areas and composes a monitor program that monitors one or more service programs; and

upon receiving a full-screen update command, the memory area releasing means updates all the classification display areas to which the service program data or the monitor program data are assigned from among the multiple classification display areas, thereby releasing the memory areas allocated to the program data.

22. A browser terminal communicably connected to the proxy server set forth in any of claims 13 to 15, the browser terminal comprising:

terminal-side data acquisition means for acquiring data from a network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the terminal-side data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, wherein:

the data in the specific format is either executable service program data which is assigned to one of classification display areas obtained by dividing a display screen into multiple areas and composes a service program that provides a service in the given display area or executable monitor program data which is assigned to one of the classification display areas and composes a monitor program that monitors one or more service programs; and

upon receiving a local-screen update command, the memory area releasing means updates the display area to which the monitor program data is assigned from among the multiple classification display areas, thereby releasing the memory area allocated to the monitor program data.

23. A memory management program applied to a memory management system which comprises data acquisition means for acquiring data, memory area allocation means for

allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, wherein the memory management program makes a computer execute processes implemented as:

acquisition failure detection means for detecting whether the data acquisition means has failed to acquire the data in the specific format; and update command output means for outputting the update command to the memory management system based on a detection result from the acquisition failure detection means.

24. A memory management program applied to a memory management system which comprises data acquisition means for acquiring data, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the data acquisition means, data initialization means for initializing the data in the specific format acquired by the data acquisition means to make it available for use, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, wherein the memory management program makes a computer execute processes implemented as:

initialization failure detection means for detecting whether the data initialization means has failed to initialize the data in the specific format; and update command output means for outputting the update command to the memory management system based on a detection result from the initialization failure detection means.

25. A memory management program applied to a memory management system which comprises data acquisition means for acquiring data, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, wherein the memory management program makes a computer execute processes implemented as:

timeout detection means for detecting whether a service using the data in the specific format acquired by

the data acquisition means has timed out; and update command output means for outputting the update command to the memory management system based on a detection result from the timeout detection means.

26. A server program which communicably connects via a network to browser terminal and to network resource other than the browser terminal and which, upon receiving a data acquisition request from the browser terminal, acquires data specified by the received acquisition request from a target network resource and sends the acquired data to the browser terminal, the browser terminal comprising terminal-side data acquisition means for acquiring data from a network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the terminal-side data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, wherein the server program makes a computer execute processes implemented as:

server-side data acquisition means for acquiring, according to the acquisition request, data from the network resource; data transmission means for sending the data acquired by the server-side data acquisition means to the browser terminal; acquisition failure detection means for detecting via communications with the browser terminal whether the terminal-side data acquisition means has failed to acquire the data in the specific format; and update command transmission means for sending the update command to the browser terminal based on a detection result from the acquisition failure detection means.

27. A server program which communicably connects via a network to browser terminal and to network resource other than the browser terminal and which, upon receiving a data acquisition request from the browser terminal, acquires data specified by the received acquisition request from a target network resource and sends the acquired data to the browser terminal, the browser terminal comprising terminal-side data acquisition means for acquiring data from a network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the terminal-side data acquisition means, data initialization

means for initializing the data in the specific format acquired by the terminal-side data acquisition means to make it available for use, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, wherein the server program makes a computer execute processes implemented as:

server-side data acquisition means for acquiring, according to the acquisition request, data from the network resource; data transmission means for sending the data acquired by the server-side data acquisition means to the browser terminal; initialization failure detection means for detecting via communications with the browser terminal whether the data initialization means has failed to initialize the data in the specific format; and update command transmission means for sending the update command to the browser terminal based on a detection result from the initialization failure detection means.

28. A server program which communicably connects via a network to browser terminal and to network resource other than the browser terminal and which, upon receiving a data acquisition request from the browser terminal, acquires data specified by the received acquisition request from a target network resource and sends the acquired data to the browser terminal, the browser terminal comprising terminal-side data acquisition means for acquiring data from a network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the terminal-side data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, wherein the server program makes a computer execute processes implemented as:

server-side data acquisition means for acquiring, according to the acquisition request, data from the network resource; data transmission means for sending the data acquired by the server-side data acquisition means to the browser terminal; timeout detection means for detecting via communications with the browser terminal whether a service using the data in the specific format acquired by the terminal-side data acquisition means has timed out; and update command transmission means for sending the update command to the browser terminal based on a detection result from the timeout detection means.

29. A terminal program running on the browser terminal set forth in claim 21 consisting of a computer, wherein:

the terminal program executes processes implemented as terminal-side data acquisition means for acquiring data from a first network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the terminal-side data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format;

the data in the specific format is either executable service program data which is assigned to one of classification display areas obtained by dividing a display screen into multiple areas and composes a service program that provides a service in the given display area or executable monitor program data which is assigned to one of the classification display areas and composes a monitor program that monitors one or more service programs; and

upon receiving a full-screen update command, the memory area releasing means updates all the classification display areas to which the service program data or the monitor program data are assigned from among the multiple classification display areas, thereby releasing the memory areas allocated to the program data.

30. A terminal program running on the browser terminal set forth in claim 22 consisting of a computer, wherein:

the terminal program executes processes implemented as terminal-side data acquisition means for acquiring data from a network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the terminal-side data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format;

the data in the specific format is either executable service program data which is assigned to one of classification display areas obtained by dividing a display screen into multiple areas and composes a service program that provides a service in the given display area or executable monitor program data which is assigned to one of the classification display areas and composes a monitor program that monitors one or more service programs; and

upon receiving a local-screen update command, the memory area releasing means updates the display area to which the monitor program data is assigned from among the multiple classification display areas, thereby releasing the memory area allocated to the monitor program data.

31. The monitor program set forth in claim 12, wherein upon receiving a full-screen update command, the monitor program makes a computer execute processes for updating all the classification display areas to which the service program data or the monitor program data are assigned from among the multiple classification display areas, thereby releasing the memory areas allocated to the program data.

32. The monitor program set forth in any of claims 13 to 15, wherein upon receiving a local-screen update command, the monitor program makes a computer execute processes for updating the classification display areas to which the monitor program data is assigned from among the multiple classification display areas, thereby releasing the memory areas allocated to the monitor program data.

33. A memory management method applied to a memory management system which comprises data acquisition means for acquiring data, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, the memory management method comprising:

an acquisition failure detection step of detecting whether the data acquisition means has failed to acquire the data in the specific format; and an update command output step of outputting the update command to the memory management system based on a detection result from the acquisition failure detection step.

34. A memory management method applied to a memory management system which comprises data acquisition means for acquiring data, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the data acquisition means, data initialization means for initializing the data in the specific format acquired by the data acquisition

means to make it available for use, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, the memory management method comprising:

an initialization failure detection step of detecting whether the data initialization means has failed to initialize the data in the specific format; and an update command output step of outputting the update command to the memory management system based on a detection result from the initialization failure detection step.

35. A memory management method applied to a memory management system which comprises data acquisition means for acquiring data, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, the memory management method comprising:

a timeout detection step of detecting whether a service using the data in the specific format acquired by the data acquisition means has timed out; and an update command output step of outputting the update command to the memory management system based on a detection result from the timeout detection step.

36. A memory management method which communicably connects via a network to browser terminal and to network resource other than the browser terminal and which, upon receiving a data acquisition request from the browser terminal, acquires data specified by the received acquisition request from a target network resource and sends the acquired data to the browser terminal, the browser terminal comprising terminal-side data acquisition means for acquiring data from a network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the terminal-side data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, the memory management method comprising:

a server-side data acquisition step of acquiring, according to the acquisition request, data from the first

network resource; a data transmission step of sending data acquired by the server-side data acquisition step to the browser terminal; an acquisition failure detection step of detecting via communications with the browser terminal whether the terminal-side data acquisition means has failed to acquire the data in the specific format; and an update command transmission step of sending the update command to the browser terminal based on a detection result from the acquisition failure detection step.

37. A memory management method which communicably connects via a network to browser terminal and to network resource other than the browser terminal and which, upon receiving a data acquisition request from the browser terminal, acquires data specified by the received acquisition request from a target network resource and sends the acquired data to the browser terminal, the browser terminal comprising terminal-side data acquisition means for acquiring data from a network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the terminal-side data acquisition means, data initialization means for initializing the data in the specific format acquired by the terminal-side data acquisition means to make it available for use, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, the memory management method comprising:

a server-side data acquisition means of acquiring, according to the acquisition request, data from the first network resource; a data transmission step of sending data acquired by the server-side data acquisition means to the browser terminal; an initialization failure detection step of detecting via communications with the browser terminal whether the data initialization means has failed to initialize the data in the specific format; and an update command transmission step of sending the update command to the browser terminal based on a detection result from the initialization failure detection step.

38. A memory management method which communicably connects via a network to browser terminal and to network resource other than the browser terminal and which, upon receiving a data acquisition request from the browser

terminal, acquires data specified by the received acquisition request from a target network resource and sends the acquired data to the browser terminal, the browser terminal comprising terminal-side data acquisition means for acquiring data from a network resource, memory area allocation means for allocating memory areas, according to a static memory management system, to data in a specific format from among the data acquired by the terminal-side data acquisition means, and memory area releasing means for releasing, according to an update command, the memory areas allocated to the data in the specific format, the memory management method comprising:

a server-side data acquisition step of acquiring, according to the acquisition request, data from the first network resource; a data transmission step of sending data acquired by the server-side data acquisition step to the browser terminal; a timeout detection step of detecting via communications with the browser terminal whether a service using the data in the specific format acquired by the terminal-side data acquisition means has timed out; and an update command transmission step of sending the update command to the browser terminal based on a detection result from the timeout detection step.