Journal of Information Processing Vol.21 No.3 458–467 (July 2013)

[DOI: 10.2197/ipsjjip.21.458]

Regular Paper

An Account Provision and Management Architecture for Messaging Services in Emergencies

Kazuyuki Tasaka[1,a)] Takashi Ozu[1] Akira Idoue[1]

Received: October 22, 2012, Accepted: March 1, 2013

Abstract: In this paper, we propose an account provision and management (APAM) architecture for messaging services such as web mail in an emergency such as a massive earthquake. The APAM architecture stably and continually provides people (users) who hope to confirm each other's safety in a stricken area with message services, even if mobile phone lines and fixed lines are unavailable. This is realized by automatically establishing an emergent line such as a satellite line and stably providing the services from a server (emergent server) in the evacuation area. The emergent server provides all users with an emergent account and authenticates the account in the evacuation area so as to avoid traffic congestion in the emergent line (low-bandwidth line). If some users already have their own account, the emergent server and the server on the Internet binds the emergent account with their account for receiving message data from their relatives as usual. Moreover, even if users move to other evacuation areas to seek their relatives, this APAM architecture allows users to continually use the services by updating the binding information of the account. We deployed a prototype system and conducted experiments to evaluate the APAM architecture. The experimental results show that the APAM architecture can stably and continually provide 1,000 users (typical capacity of the area) with an emergent account and messaging services simultaneously.

Keywords: disaster recovery, messaging service, architecture, account position, account management

1. Introduction

When disasters such as earthquakes and tsunamis occur, many people who live in the stricken area collect and share information about the damage situation and the safety of their relatives and friends. People (users) hope to stably and continually use messaging services such as web mail and web message boards to collect and share the information in emergencies. They confirm each other's safety by using account information such as a user name in the messaging services.

However, when a massive earthquake hit Japan on March 11th, 2011, the disaster prevented users from stably and continually collecting and sharing information. Users used devices (e.g., PCs) that had been set up in evacuation areas such as public facilities and schools. They used the messaging services via emergent lines such as satellite lines [1], [2], [3] due to disconnection of mobile phone lines and fixed lines.

The traffic for the services in emergent lines tends to cause congestion because the bandwidth of the lines is lower than that of normal lines such as fixed lines. The limitation on the number of devices restricts the number of users who can use the services simultaneously.

There is an effective solution for congestion avoidance. A service provider sets up a server (emergent server) in the evacuation area. The emergent server manages the services in the evacuation area and controls the traffic.

The architecture for the solution must distribute some features (e.g., account provision and management, web service) of a server (normal server) on the Internet to the emergent server and other normal servers. The architecture has already been studied [4], [5], [6], [7], [8], [9], [10]. The conventional architecture achieves high availability and low downtime.

Moreover, we consider functions to resolve certain issues in emergencies. The conventional architecture causes traffic congestion according to the location of users (Issue 1). In the conventional architecture, users must move to the area where the emergent server managing their own account is set up. If not, the users must connect to an emergent server and a normal server via an emergent line for authentication and message data.

The movement of users in Issue 1 gives rise to Issue 2. Some users move from one evacuation area to another evacuation area in the wake of information (e.g., location of separated relatives and friends) obtained through message data. This means that the location of the user and the location of the emergent server managing his/her account become mismatched due to user movement.

To resolve Issue 1 and Issue 2, we consider a way in which all emergent servers manage account information for all users. However, this is infeasible due to the increased installation costs and running costs increase.

In addition to two issues, there is an issue regarding a service available area (Issue 3). The users can use the services only in the limited area where there is an administrator for a satellite device and the emergent server. The users must be able to use the services in the evacuation area where there is no administrator.

We propose an account provision and management (APAM) architecture to resolve these three issues. To solve Issue 1, the

[1] KDDI R&D Laboratories Inc., Fujimino, Saitama 356–8502, Japan
[a)] ka-tasaka@kddilabs.jp

APAM architecture provides all users with an emergent account and message data from the emergent server for stable services. The emergent server authenticates the account in the evacuation area so as to avoid traffic congestion. If users already have their own account, the emergent server and a normal server bind the emergent account with their account to receive message data from devices on the Internet as usual. To solve Issue 2, the APAM architecture updates the binding information between each account even if users move to a new emergent area. To solve Issue 3, the APAM architecture automatically detects an emergency from the status of the communication lines and starts the services.

The rest of this paper is structured as follows. Section 2 describes related work on conventional architectures. Section 3 proposes the APAM architecture to resolve the issues in emergencies. Section 4 presents an implementation summary and performance evaluation. Finally, Section 5 presents conclusions.

2. Related Work

This section describes typical conventional architecture and some issues in stably and continually providing messaging service in emergencies.

2.1 Assumed Environment

In an emergency, it is difficult for users to use messaging services via normal lines such as mobile phone lines and fixed lines. Users escape to an evacuation area. Some users move from area to area according to their purposes (e.g., searching for their relatives and friends who have become separated by the disaster).

In the evacuation area, users use messaging services with their own account information or new account information for user identification. Users collect and share information about the damage situation and the safety of their relatives through the messaging services.

Users use a device such as a PC that is set up in advance and use their devices such as smartphones. The devices are supplied with electricity from dynamos and access a router that establishes a local area network in the evacuation area and connects to the emergent line.

2.2 Conventional Architecture

Typical conventional architecture provides messaging services and manages account information such as the account name and password in the normal server [4], [5] (**Fig. 1** (1)).

An administrator of a normal server registers an account name and provides the user with it, or the user registers account information with the normal server (Fig. 1 (1)-(i)). If an account name is already registered, the normal server requests the administrator and user to re-register his/her account information with another account name. After authentication of the registered account information (Fig. 1 (1)-(ii)), the user can use the messaging services (Fig. 1 (1)-(iii)).

Architecture for some services (e.g., Gmail) provides account information after authentication by calling the phone number of the user. At this time, the users must input the number notified by an automated voice.

However, conventional architecture assumes a normal situation. In an emergency, the services in the architecture are unavailable due to traffic congestion and data concentration to the servers.

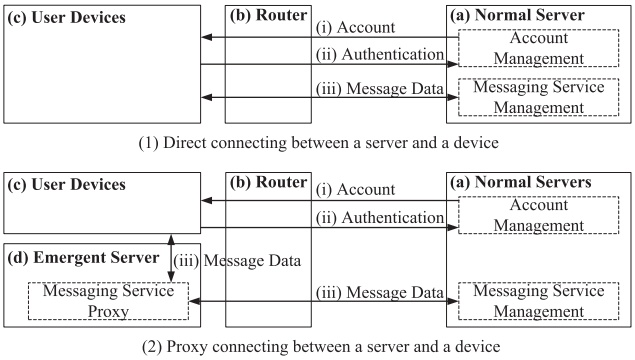

Fig. 1 Typical conventional architecture.

Architectures for resolving these issues have already been considered [6], [7], [8], [9], [10] (Fig. 1 (2)). The architecture in Refs. [6], [7], [8], [9] distributes the functions of the normal server to other servers (emergent server and normal server) for achieving high availability and low downtime. In Ref. [10], the user can easily send hand-written messages to the normal server by scanning it into a PC in an evacuation area even if the user has poor information technology literacy.

2.3 Issues of Conventional Architecture in the Assumed Environment

We show three issues of conventional architecture in the assumed environment.

The conventional architecture causes traffic congestion according to the location of users and leads to service downtime (Issue 1). We can consider a way whereby some functions (e.g., account management) of the normal server are distributed to an emergent server reducing traffic in the emergent line. The emergent server manages the account information and the messaging services in each evacuation area. However, in many cases, the emergent server must connect to another server in another evacuation area via the emergent line for authentication and messaging services. The conventional architecture is unsure of where users are going. This causes traffic congestion in the emergent line for authentication and services.

Moreover, the movement of users encourages Issue 1 (Issue 2). Some users move to a new evacuation area in the wake of information (e.g., location of separated relatives and friends) obtained through message data. If the location of the users and the location of the emergent server managing the user account is different after the user moves, the situation is the same as Issue 1. The emergent server must authenticate his/her normal account and send/receive message data to/from another emergent server via an emergent line. The result is increased traffic congestion and leads to service downtime.

For the above reason, we could consider a method that manages the normal account of all users in each emergent server. This way is independent of the user movement. However, in reality, this is impossible in terms of installation costs and running costs.

We must also consider auto management in the evacuation area

(Issue 3). In an emergency situation, there is no administrator for a satellite device and the emergent server in most evacuation areas. To solve this issue, we can also consider starting and managing the emergent server from a device on the Internet by remote control. However, the administrator cannot access the emergent server because normal lines are disconnected in an emergency. Therefore, the architecture must automatically establish an emergent line and provide the account and the messaging services in an evacuation area where there is no administrator.

3. An Account Provision and Management (APAM) Architecture for Messaging Services in Emergencies

3.1 Proposed Architecture

For stable and continual provision of messaging services in an emergency, we propose a novel architecture (APAM architecture) that resolves the issues described in Section 2. **Figure 2** shows the functions of the APAM architecture. The architecture design is summarized below.

3.1.1 Summary of the Architecture Design

To solve Issue 1, the APAM architecture authenticates the emergent account in an evacuation area and provides messaging services in the area. This is realized by providing all user devices (Fig. 2 (c)) with an emergent account from the emergent server (Fig. 2 (d)). The normal server (Fig. 2 (a)) allocates an emergent account to the emergent server in advance by using the emergent account provision function (Fig. 2 (1), Fig. 2 (i)). The emergent server provides the devices connecting the network (emergent network) with an emergent account in the emergency (Fig. 2 (ii)).

Moreover, the APAM architecture not only has this function but also has an account management function (Fig. 2 (2)) to allow users to use the normal account as usual. The mobile router (Fig. 2 (b)) guides access from the user device to the emergent server (Fig. 2 (iii)).

The emergent server receives a normal account from the user device (Fig. 2 (iv)) and binds the emergent account with the normal account by using the account management function (Fig. 2 (v)). The normal server changes the source address of the message data from the normal account to the emergent account when the normal server receives message data of the normal account from user devices on the Internet. The mobile router controls the sending/receiving time of control packets and message data to maintain stable communication (Fig. 2 (3)).

To solve Issue 2, when a user moves from one area to another area, the account management function updates the binding information to forward the message data (Fig. 2 (vi)). If the domain of the emergent account and the normal account is different, this function requests configuration for forwarding between the normal servers managing the domain to the normal server (Fig. 2 (vi)).

To solve Issue 3, the mobile router detects the emergency situation from the connection state of the normal lines. If the mobile router is continually unreachable by the normal server, it detects an emergency situation. The mobile router automatically switches the access network from the normal lines to the emergent lines and starts the emergent server and a network in an evacuation area. When mobile phone lines or fixed lines are recovered, the mobile router detects the normal situation from the connection state of the normal lines in emergencies. The mobile router automatically switches the access network from the emergent lines to the normal lines and disconnects the emergent line.

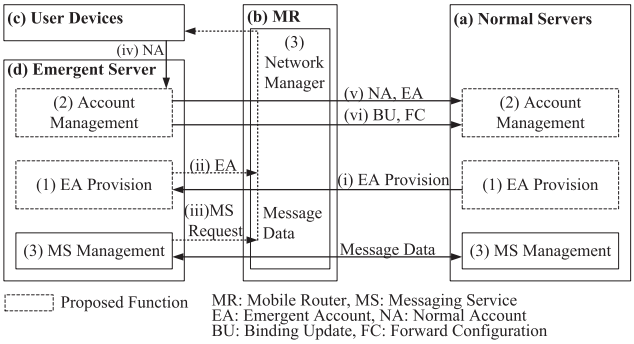

Fig. 2 Proposed architecture.

Table 1 Process in each server for using the normal account as usual in an emergency situation.

		Normal Account	
	Domain	Domain A	Domain B
Emergent Account	Domain A	(a) Bind Configuration	(b)Bind/Forward Configuration
	Movement from Domain A to A	(c) Binding Update	(e) Forward Update (Set)
	Movement from Domain A to B	(d) Forward Set	(f) Forward Update (Delete)

3.1.2 Proposed Function

Each server in the APAM architecture have the following functions.

Function 1: Emergent account provision function

This allocates a list of emergent accounts from the normal server to the emergent server in advance before the emergency situation. This function also manages the status of the allocation and provides users with an emergent account and the messaging services. Section 3.2 describes this function in detail.

Function 2: Emergent account management function

This function binds the emergent account with the normal account to allow users who already have an account to use the normal account. This function also manages the user account to continually provide the messaging services even if the user receives the account of a different domain between emergent servers or between the emergent server and a normal server. **Table 1** shows the process in each server for using the normal account as usual in an emergency situation. Section 3.3 describes this function in detail.

3.2 Function 1: Emergent Account Provision Function

The normal server allocates the emergent account to each emergent server in advance and manages the status of allocation of the account (Phase 1). The emergent server provides the user device with an emergent account in an evacuation area (Phase 2) and provides users with the messaging services (Phase 3). The details of each phase are as follows.

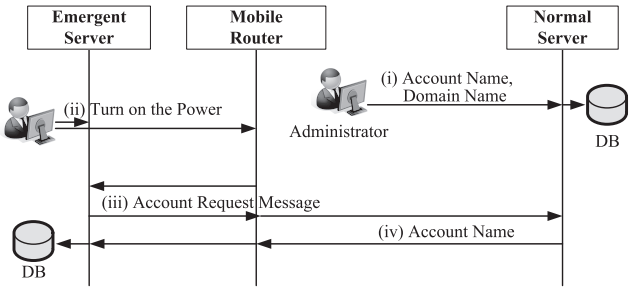

Fig. 3 Sequence for allocating an emergent account.

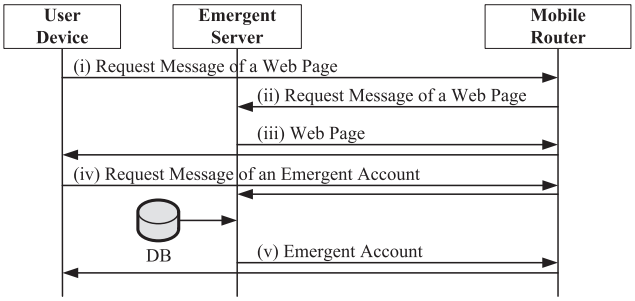

Fig. 4 Sequence for making an emergent account.

Table 2 Allocation example of an emergent account.

Account Name	Domain Name	Emergent Server ID	Area	Status
xxx-0000 - xxx-1000	abc.ddd.jp	EServ0001	School XX	Allocated - Allocated
yyy-0000 - yyy-0400	abc.ddd.jp	EServ0002	Public Space YY	Allocated - Allocated
zzz-0000 - zzz-0800	abc.ddd.jp	-	-	Ready - Ready

Phase 1: Allocation of the emergent account

An administrator of a normal server sets the emergent account that is allocated to each emergent server from the normal server in advance (before emergency situation). If the emergent server does not have an emergent account, it receives a list of emergent accounts from the normal server.

(1) Registration of an emergent account

The administrator of the normal server registers the account name and domain name (e.g., abc.ddd.jp) for allocation with each emergent server into the database (DB) of the normal server (**Fig. 3** (i) and **Table 2**). Table 2 shows an example of the DB table. The first status of each account in the table is "Ready" which means that the account can be allocated. In Table 2, the status of account name "zzz-0000 to zzz-0800" is "Ready." Attribution "Emergent server ID" and "Area" in Table 2 mean identification to identify each emergent server and the area in which the emergent server is installed, respectively.

(2) Request for allocation of an emergent account

An administrator (or a manager of an evacuation area) turns on the power of the mobile router and the emergent server (Fig. 3 (ii)). The emergent server boots the operating system and automatically sends a request message (account request message) for a list of emergent accounts to the normal server (Fig. 3 (iii)). The administrator (or a service provider) registers the address of the normal server on the emergent server in advance.

(3) Allocation of an emergent account based on the request

The normal server decides on the emergent account for allocation based on the account request message and replies to the emergent server with the list (Fig. 3 (iv)). After that, the normal server changes the attribution "Status" in the DB from "Ready" to "Allocated." The emergent server registers the emergent account with the DB and shuts down until an emergency situation arises.

Phase 2: Provision of an emergent account

The mobile router guides the web access of the user device to the emergent server, and the emergent server provides the user device with an emergent account.

(1) Request for an emergent account

A user uses the user device connecting the emergent network in the evacuation area and sends a request message of a web page using the user device (**Fig. 4** (i)). The mobile router forwards the request message to the emergent server (Fig. 4 (ii)).

(2) Provision of an emergent account based on the request

The emergent server replies with the web page that includes a link for the account creation page (Fig. 4 (iii)). When a user first selects the link on a web page, the user device sends a request message to the emergent server for the creation of a new account (Fig. 4 (iv)). When the emergent server receives the request message, it searches a new account name that can be allocated and replies with the account and the password (Fig. 4 (v)).

Phase 3: Provision of a messaging service

In an emergency situation, the emergent server provides users who have an emergent account with messaging services after the authentication in an evacuation area.

A user receives a web page for login to the services and enters the emergent account name and password. After authentication, the user device receives the web page for the messaging services. The user can send/receive the message data from the web page. The emergent server does not send the message data between users in the same evacuation area to the normal server. If the user uses the e-mail service as a messaging service, the authentication message of the emergent account and the receiving message are guided to the emergent server by the mobile router. The emergent server sends/receives the message data to/from the user device as well as web mail.

3.3 Function 2: Emergent Account Management Function

This binds the emergent account with the normal account and configures the forwarding table in the emergent server and the normal server so that the normal account is available in an emergency. The details are shown in Section 3.3(a).

When a user moves from an evacuation area to another area where the domain between each server is the same, this function updates the binding information. The route of the message data sent to the previous emergent server is changed to the new emergent server after the user moves. The details are shown in Section 3.3(b).

If the domain between each server is different, the normal server sets or updates the forwarding table for forwarding the message data. The details are shown in Section 3.3(c).

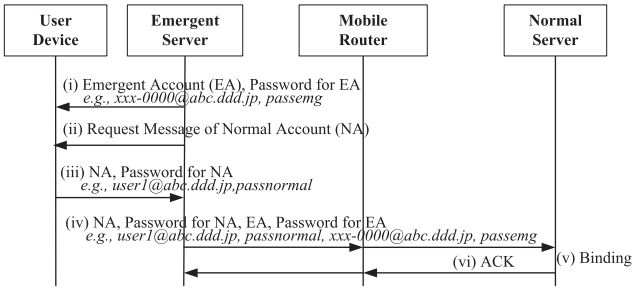

Fig. 5 Binding sequence of accounts.

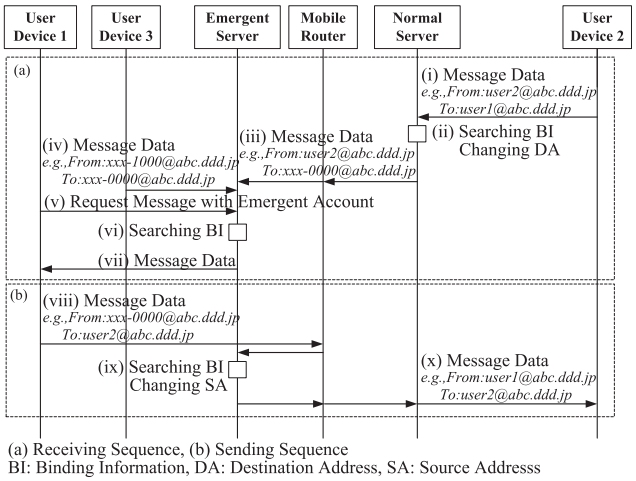

Fig. 6 Providing sequence of message data using a normal account.

(a) Binding each account (the emergent account and the normal account) and forward configuration in each server

(1) Binding sequence

The emergent server and the normal server bind the emergent account, with which the user device is provided in the evacuation area, with the normal account. The normal server forwards the message data for the normal account based on the binding information to the emergent server. **Figures 5** and **6** show the binding sequence of accounts and the providing sequence of message data using a normal account respectively.

The emergent server sends the emergent account (e.g., xxx-0000@abc.ddd.jp) and password (e.g., passemg) to the user device (Fig. 4 (v) and Fig. 5 (i)) and requests the normal account (Fig. 5 (ii)). If the user has a normal account (e.g., user1@abc.ddd.jp), the user device sends the normal account and the password to the emergent server (Fig. 5 (iii)). The emergent server sends each account information (account name and password) to the normal server (Fig. 5 (iv)). The normal server authenticates the user based on the account information. After the authentication, the normal server binds the emergent account with the normal account and registers the binding information of each account with the DB (Fig. 5 (v)). The normal server replies with an ACK message that indicates the success of the binding and makes the emergent server register the binding information (Fig. 5 (vi)). If the normal server receives the error message of the ACK message, the normal server deletes the binding information.

If the domain between the normal server and the normal account is different, the normal server requests a normal server that has the same domain as the normal account to forward message data. Each server sets a forward configuration for sending the message data to the normal account.

(2) Sequence receiving message data

After binding, the user can receive message data using not only the emergent account but also the normal account in the evacuation area (Fig. 6 (a)).

The user device (e.g., User Device 2) on the Internet sends the message data to the normal account (e.g., user1@abc.ddd.jp) (Fig. 6 (i)). The normal server searches the binding information from the DB (Fig. 6 (ii)). When the normal server discovers the emergent account (e.g., xxx-0000@abc.ddd.jp) binding with the normal account, it decides what is the emergent server that is the destination of the message data. The normal server changes the destination address from the normal account name to the emergent account name and sends the message data to the emergent server (Fig. 6 (iii)). On the other hand, when the normal server does not discover the emergent account binding with the normal account, the normal server does not change the destination address.

If the destination address is the emergent account (Fig. 6 (iv)), the account name does not need to be changed to another address (Fig. 6 (v)). The user requests the emergent server to send the new message data from the user device (e.g., User Device 1) using the emergent account on the digest authentication and authentication protocol such as Post Office Protocol (POP) and Intent Message Access Protocol (IMAP). Upon error, each server returns the message based on each protocol. The emergent server authenticates the user (the emergent account) and searches the normal account binding with the emergent account (Fig. 6 (vi)). When the emergent server discovers the normal account, the emergent server replies to the device with the message data for the emergent account and the normal account (Fig. 6 (vii)).

(3) Sequence sending message data

After binding, the user can send message data using the normal account from the evacuation area (Fig. 6 (b)).

The user uses the user device (e.g., User Device 1) with the emergent account (e.g., xxx-0000@abc.ddd.jp) and sends the message data to the other account (e.g., user2@abc.ddd.jp) of a device on the Internet (e.g., User Device 2) (Fig. 6 (viii)). The emergent server receives the message data via the mobile router. The emergent server changes the source address from the emergent account name to the normal account name (e.g., user1@abc.ddd.jp) based on binding information (Fig. 6 (ix)). The emergent server sends it to the destination address via the normal server that manages the account information for the destination address (Fig. 6 (x)). At this time, the mobile router decides the timing at which to send the message data to the normal server according to the state of the emergent line.

If the destination address is an emergent account managed in the same emergent server, the account name does not need to be changed to another address. On the other hand, if the domain between the normal server and the normal account is different, normal servers managing each account send the message data based on the forwarding table in each normal server.

(b) Binding update

The emergent server and the normal server update the binding

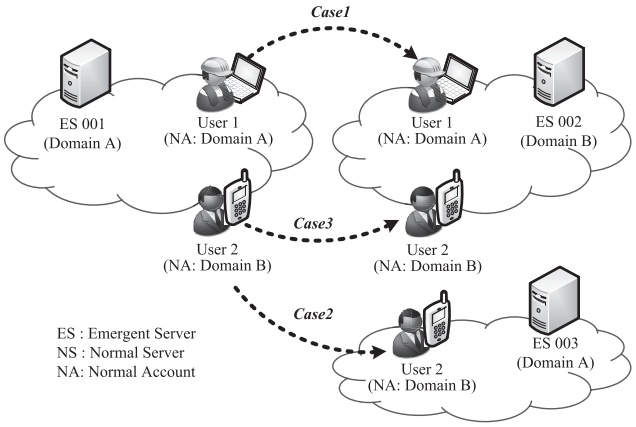

Fig. 7 Cases changing the forwarding table.

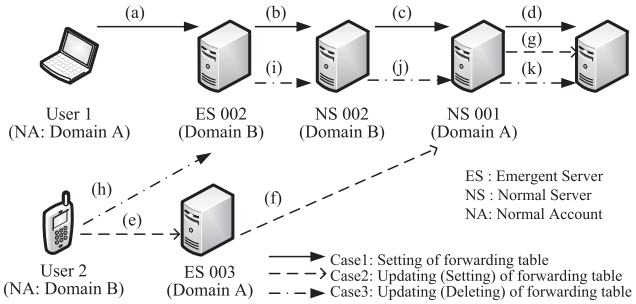

Fig. 8 Flow of the control packet in each case.

information (binding update) (Table 1 (c)) when users move from one area to another area. In the area after the user moves, there is an emergent server managing an account of the same domain as the emergent server before the user moves.

(1) Detecting the necessity for binding update

We show the case that a user who has an emergent account (e.g., xxx-0000@abc.ddd.jp) moves to another evacuation area. After the user moves, the user sends the emergent account and password to the new emergent server. The emergent server detects the necessity for binding update and sends the new account and password to the user device.

(2) Updating the binding information

The emergent server detects the domain name (e.g., abc.ddd.jp). If the domain is the same as the domain managing the new emergent server, the emergent server sends the old account (e.g., xxx-0000@abc.ddd.jp) and new emergent account (e.g., yyy-0300@abc.ddd.jp) to the normal server. The normal server searches the binding information and updates the emergent account from the old account to the new account.

The normal server sends a request message to the previous emergent server for update of the binding information. When the emergent server receives the request message, the emergent server releases the account allocated to the user and removes old message data.

(c) Setting and updating the forwarding table

If the domain between the emergent servers or between the emergent server and the normal server is different according to user movement, each server sets or updates the forwarding table (Table 1 (d)–(f), **Fig. 7** and **Fig. 8**).

Case 1 (Table 1 (d)): Forward Set

This is a case where User 1 with a normal account of Domain A moves from one area (ES001 of Domain A) to another area (ES002 of Domain B).

In this case, ES002 provides User 1 with an emergent account of Domain B and receives the normal account of Domain A and the previous emergent account of Domain A (Fig. 8 (a)). ES002 detects the different domain between the emergent servers and requests NS002 of Domain B to forward the message data for User 1 by sending the normal account and the emergent account of User 1 (Fig. 8 (b)). The request message includes flags indicating the old emergent account and new emergent account. NS002 identifies the old emergent account from the flag and requests NS001 of Domain A to forward the message data of User 1 (Fig. 8 (c)). After that, NS001 sets a route to NS002 for the forwarding table and requests ES001 to release the account information of User 1 (Fig. 8 (d)).

Case 2 (Table 1 (e)): Forward Updating (Setting)

This is the case where User 2 with a normal account of Domain B moves from one area (ES001) to another area (ES003 of Domain A).

In this case, ES003 provides User 2 with an emergent account of Domain A and receives the normal account of Domain B and the previous emergent account of Domain A from User 2 (Fig. 8 (e)). ES003 requests NS001 to update the forwarding table for User 2 by sending the normal account and the emergent account of User 2 (Fig. 8 (f)). NS001 updates the forwarding table from ES001 to ES003 and requests ES001 to release the account information of User 2 (Fig. 8 (g)).

Case 3 (Table 1 (f)): Forward Updating (Deleting)

This is the case where User 2 moves from one area (ES001) to another area (ES002).

In this case, ES002 provides User 2 with an emergent account of Domain B. ES002 receives the normal account of Domain B and the previous emergent account of Domain A from User 2 (Fig. 8 (h)). ES002 detects the same domain between the emergent account provided and the normal account. ES002 requests NS002 to delete a route for forwarding the message data to NS001 by sending the normal account and the emergent account of User 2 (Fig. 8 (i)). NS002 deletes the route for forwarding to NS001 and sets to forward to ES002. NS002 notifies NS001 of the normal account and the emergent account (Fig. 8 (j)). NS001 requests ES001 to release the account information of User 2 (Fig. 8 (k)).

4. Implementation Summary and Performance Evaluation of the Proposed Architecture

4.1 Implementation

We implemented a prototype system based on the proposed architecture (APAM architecture). **Figure 9** shows the module composition for the prototype system and **Table 3** shows software for each module.

(a) State Monitoring Module

The mobile router runs this module (Fig. 9 (a)), which monitors

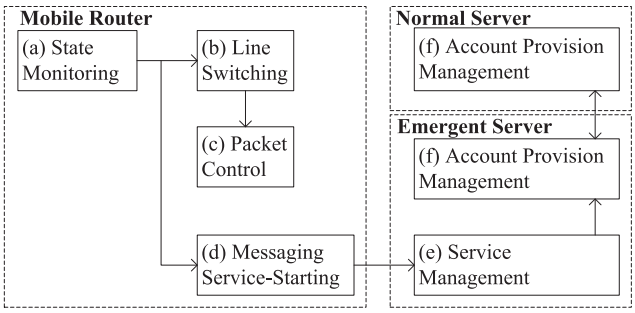

Fig. 9 Module composition.

Table 3 Equipment and software.

	OS	Software	Module
MR	Linux (CentOS 5)	Iptables	Fig. 8(b), (c)
		Squid	Fig. 8(d)
ES	Linux (RedHat Enterprise 5)	DNS	Fig. 8(e)
		Apache	Fig. 8(e)
		Squirrel mail	Fig. 8(e)
		Dovecot	Fig. 8(e)
		Sendmail	Fig. 8(e)
NS	Linux (RedHat Enterprise 5)	Apache	Fig. 8(f)
		DNS	Fig. 8(f)
		Sendmail	Fig. 8(f)

ES:Emergent Server, NS:Normal Server, MR:Mobile Router

the emergency situation based on connectivity of normal lines. This module detects the emergency situation and notifies the line switching module (Fig. 9 (b)) and the messaging service-starting module (Fig. 9 (d)) of it. If a service provider uses the APAM architecture for an earthquake, this module can corroborate with values of sensors such as seismic intensity meters. In this case, this module can detect an emergency situation from the status of normal lines and the values of each sensor.

(b) Line Switching Module

The mobile router runs this module (Fig. 9 (b)), which switches the communication line from normal lines to emergent lines in an emergency situation. On the other hand, this module switches the communication line from emergent lines to normal lines in a normal situation. This module uses iptables [11] to the change the network route according to change of the communication lines.

(c) Packet Control Module

The mobile router runs this module (Fig. 9 (c)), which runs or stops conventional functions such as traffic monitoring, bandwidth control, and priority control of the packet between the emergent server and the normal server. This module sends/receives message data according to the state (throughput) of the emergent lines. Moreover, this module uses squid [12] for proxy of messaging services based on the Web.

(d) Messaging Service-Starting Module

The mobile router runs this module (Fig. 9 (d)) which starts the service management module (Fig. 9 (e)) when this module receives a request message from the state monitoring module.

(e) Service management module

The emergent server runs this module, which starts messaging services by receiving the character indicating "Start" from the messaging service-starting module. This module runs a DNS (domain name system) [13], [14] for messaging services, Apache [15], SquirrelMail [16] for web mail, Dovecot [17] and Sendmail [18] for e-mail.

(f) Account provision management module

The emergent server and the normal server run this module (Fig. 9 (f)), which runs the emergency provision function and the emergent management function. This module starts the process for account provision and management by receiving a character indicating "Start" from the service management module. This module binds the emergent account with the normal account and updates the binding information.

4.2 Experimental Environment

Two normal servers (NS001 and NS002) have different domains connected to the fixed line (1GBase-TX). Three mobile routers have two interfaces: the fixed line (100Base-TX) for the normal line and the satellite line (Inmarsat BGAN, max throughput 492 kbps) for the emergent line respectively. Three emergent servers (ES001, ES002 and ES003) connect the fixed line (100Base-TX) with each mobile router. Two user devices (android smartphone) of two users can set a normal account of different domains to connect the wireless line (IEEE 802.11b) with the mobile router.

4.3 Experimental Items

We conducted experiments on three experimental items and evaluated the proposed architecture (APAM architecture) in an experimental environment. The experimental results show that APAM architecture can stably and continually provide account and messaging services compared with the conventional architecture that provides services from a normal server.

Item1: Number of users who can simultaneously obtain an account

This experimental item is the number of users who can simultaneously obtain an account in the APAM architecture and the conventional architecture. In this experiment, we increase the number of accounts (messages) that the emergent server or the normal server provides simultaneously from 10 to 1,000 (typical capacity of facilities in an evacuation area in Japan). From this result, we show that the emergent account provision function (described in Section 3.2) can solve Issue 1 and stably provide all users with an account.

Item2: Number of users who can simultaneously obtain message data

This experimental item is the number of users who can simultaneously authenticate and obtain message data in the APAM architecture and the conventional architecture. In this experiment, we increase the number of users as well as Item 1. The data size is 500 KBytes (max. data size per file in a smartphone (IS03: one type of smartphone)). From this result, we show that the emergent account provision function can solve Issue 1 and stably provide all users with the message data.

Item3: Time needed for starting the services with a normal account

This experimental item is the time from starting switching of an emergent line to binding between the emergent account and the normal account. From this result, we show that the emergent

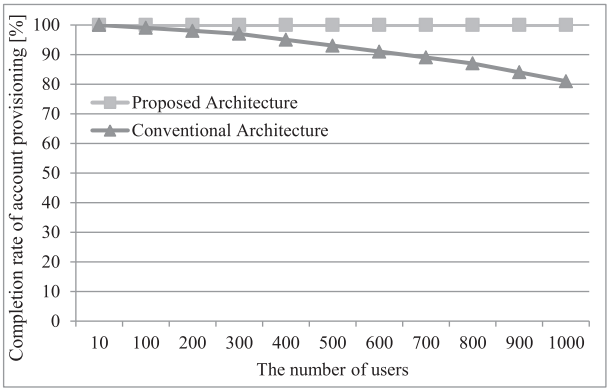

Fig. 10 Number of users provided with an emergent account.

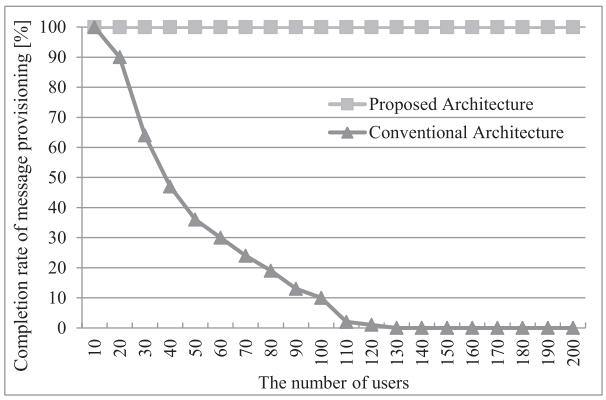

Fig. 11 Number of users provided with message data.

management function (described in Section 3.3) can solve Issue 2 and Issue 3 and continually provide all users with the messaging service in an evacuation area. In this experiment, we show that the time is within 180 s. This is the target time for the start of collecting and sharing information after the users have ensured their own safety. Some users can start the messaging services even if the users are in an evacuation area (e.g., school, public facility) from immediately after the disaster.

Item4: Number of sessions that can simultaneously handle setting or updating user's account binding

This experimental item is the number of sessions that can simultaneously handle setting or updating of user's account binding (Table 1 (c)–(f)) in the APAM architecture. In this experiment, we increase the number of user's account binding from 10 to 1,000 in each emergent server and each normal server. From this result, we show performance of the emergent management function (described in Section 3.3).

4.4 Performance Evaluation

(a) Performance evaluation regarding the number of users who can simultaneously obtain an account

Figure 10 indicates the experimental results regarding the number of users who simultaneously obtained an account by the APAM architecture and the conventional architecture.

The APAM architecture was able to provide all users with an account, even if the number of users increased from 10 to 1,000.

In the conventional architecture, when the number of users was over 100, packet loss arose and not all users could be provided with an account. When the number of users was 100, 90% of users could be provided with an account at the same time. When the number of users was 1,000, the completion rate decreased from 90% to 81% and the conventional architecture was unable to provide 19% (180) of users with an account.

The conventional architecture is dependent on throughput and delay on the emergent line. On the other hand, the APAM architecture can stably provide an emergent account until the number of users that the emergent server or the emergent network can process at the same time is reached. Even if the number of users increases to over 1,000 users, the APAM architecture can stably provide services to all users with an account because the APAM architecture provides an account in the evacuation.

(b) Performance evaluation regarding the number of users who can simultaneously obtain message data

Figure 11 indicates the experimental results regarding the number of users provided with message data by the APAM architecture and by the conventional architecture.

The APAM architecture could provide all users with message data from emergent server. On the other hand, in the conventional architecture, when the number of users was over 20, packet loss arose and not all users could be provided with an account. When the number of users reached 130, the users could not be provided with message data due to congestion on the emergent link and time-out of the request message.

The APAM architecture aggregates the request message for authentication and message data from user devices in the emergent server and achieves congestion avoidance. Even if the emergent server sends/receives message data to/from the normal server, the mobile router controls the timing to send/receive it according to the status of the line. Even if size of message data increases, the APAM architecture can stably provide message data until reaching limitation of the emergent server and emergent network. Service providers consider the specification of the emergent server and Wi-Fi access points according to the capacity of the evacuation area.

However, the total time from starting to receiving all message data to completing it was longer than in a normal situation due to traffic control. The total time for 100 users and 200 users was about 120 s and about 240 s respectively in the APAM architecture. The APAM architecture is effective regarding non-real-time services such as messaging services (e.g., mail and voice message). To improve the convenience of the proposed architecture, we can provide users with an estimation time at which the emergent server can provide the user with new message data after users request the message data. The emergent server sends a notification including the estimation time (e.g., Receiving Time is 13:02) to the users.

(c) Performance evaluation of the time needed for starting the services with a normal account

The experimental result was about 9.8 s. The switching time to the emergent line was about 6.0 s. The binding time between two accounts was about 3.8 s (The time from sending the emergent account to receiving the normal account in the emergent server was about 1.3 s. The time from receiving the normal account

information to receiving an ACK from the normal server was about 2.5 s.). The processing time in each server was about from 0.3 s to 0.5 s. Even if the domain between each account is different, the additional delay was about 1 s between normal servers. As the result, the total binding time is still about 10.8 s.

From these results, the time needed for starting the services with a normal account is within 180 s. Therefore, the APAM architecture can continually provide messaging services in emergencies. The proposed architecture needs additional processing such as doing search operation, the binding process of the account and changing the source/destination address in NS and ES compared with conventional architecture. However, the proposed architecture can run these additional processing on the millisecond time scale and can provide approximately the same efficiency as the conventional architecture. If this time increases due to a lack of memory size on each server, we should scale up servers and distribute the load on each server.

(d) Performance evaluation regarding the number of sessions that can simultaneously handle setting or updating of user's account binding

As the experimental results, the completion rate of binding was 100% in each case (Table 1 (c)–(f)). Even if the number of the user's account binding increased to over 1,000 users according to user's movements, the APAM architecture could stably complete it because the APAM architecture can handle the timing to send/receive the control packet for binding in each emergent line.

Moreover, by using the communication NS and ES in the APAM architecture, the users can skip the direct authentication process in the emergent line (including the binding of account between NS and ES). This leads to decreased authentication packets (efficiency in emergencies). The proposed architecture has advantages in terms of stability and scalability.

5. Conclusion

In this paper, we propose an account provision and management architecture for messaging services in emergencies. The proposed architecture can be used as a general solution for non-real-time web services such as messaging services and web message boards in emergencies. We also show a performance evaluation using a prototype system, which is implemented based on the proposed architecture. The results show that the proposed architecture provides all users with an account and service even if the number of users and the movement increase (from 10 to 1,000) and starts provision of message data within the target time (180 s). The results indicate that this architecture is suitable for messaging services to collect and share emergency information from immediately after a disaster in an evacuation area. As a future direction, we will consider countermeasures to help prevent security threats such as hijacking of an emergent account and/or normal account and spoofing in emergencies. As one countermeasure, there is a method for authenticating a user prior to providing access to the user's account, the user's account being accessible via a sign-in page upon verifying the user's credentials [19]. As another way, there is a contingency trust inference model that allows the information owner to infer the trustworthiness of requesters (users) [20]. Moreover, we will consider a method to easily manage the proposed modules for service providers because each service provider must manage not only the normal account but also the emergent account and distributed data. In addition to the above items, we will consider a method (e.g., delay-tolerant network etc.) to also provide users who are out of the evacuation area with an emergent account and messaging services. For this purpose, we can use the ad-hoc network [21] and delay-tolerant network (DTN) [22] and can expand the service available area in emergencies.

Acknowledgments We are indebted to Dr. Yasuyuki Nakajma, President and Chief Executive Officer of KDDI R&D Laboratories Inc. and Dr. Toru Hasegawa, Executive Director, for their continuous encouragement of this research.

References

[1] Kitaguchi, T. and Hamada, H.: Telecommunications Service Continuity and Disaster Recovery, *IEEE Communications Society Communications Quality and Reliability* (*CQR 2008*) *Workshop* (Apr. 2008).
[2] Fukumoto, N.: Business Continuity and Disaster Recovery in KDDI, *IEEE Communications Society Communications Quality and Reliability* (*CQR 2008*) *Workshop* (Apr. 2008).
[3] Lee, Y., Ku, B. and Ahn, D.: A Satellite Core Network System for Emergency Management and Disaster Recovery, *IEEE Information and Communication Technology Convergence* (*ICTC 2010*), pp.549–552 (Dec. 2010).
[4] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A. and Stewart, L.: HTTP Authentication: Basic and Digest Access Authentication, *IETF RFC2617* (June 1999).
[5] Klensin, J.: Simple Mail Transfer Protocol, *IETF RFC5321* (Oct. 2008).
[6] Pokharel, M., Lee, S. and Park, J.: Disaster Recovery for System Architecture using Cloud Computing, *Applications and the Internet* (*SAINT 2010*), pp.304–307 (July 2010).
[7] Ueno, Y., Miyano, N. and Suzuki, S.: Performance Evaluation of a Disaster Recovery System and Practical Network System Applications, *Systems and Networks Communications* (*ISWC2010*), pp.195–200 (Aug. 2010).
[8] Ping, Y., Bo, K., Jinping, L. and Mengxia, L.: Remote Disaster Recovery System Architecture Based on Database Replication Technology, *IEEE Computer and Communication Technologies in Agriculture Engineering* (*CCTAE 2010*), pp.254–257 (June 2010).
[9] Bianco, A., Girarudo, L. and Hay, D.: Optimal Resource Allocation for Disaster Recovery, *IEEE Global Telecommunications Conference* (*Globecom 2010*), pp.1–5 (Dec. 2010).
[10] Mase, K.: How to Deliver your Message from/to a Disaster Area, *IEEE Communications Magazine*, Vol.49, pp.52–57 (Jan. 2011).
[11] Osker, A.: Iptables Tutorial (2006), available from ⟨http://iptables-tutorial.frozentux.net/iptables-tutorial.html⟩.
[12] Squid-cache org: Using Squid 3.0 Configuration Manual (May 2007).
[13] Mockapetris, P.: Domain names - concepts and facilities, *IETF RFC1034* (Nov. 1987).
[14] Mockapetris, P.: Domain names - implementation and specification, *IETF RFC1035* (Nov. 1987).
[15] Apache HTTP Server Version 2.4 Documentation, available from ⟨http://httpd.apache.org/docs/2.4/en/⟩.
[16] SquirrelMail, available from ⟨http://squirrelmail.org/documentation/⟩.
[17] Dovecot, available from ⟨http://www.dovecot.org/documentation.html⟩.
[18] Sendmail, available from ⟨http://www.sendmail.com/sm/open_source/docs/⟩.
[19] Craddock, S. and Vitaldevara, C.: Account Hijacking Counter-Measures, US 2010/0199338 A1 (Feb. 2012).
[20] Yang, Q., Yao, D., Garnett, J. and Muller, K.: Using a Trust Inference Model for Flexible and Controlled Information Sharing During Crises, *Journal of Contingencies and Crisis Management*, pp.231–241 (Dec. 2010).
[21] Park, G., Shin, K., Park, K., Park, W., Yoon, H., Rho, W., Lee, W., Jang, W., Jung, S. and Lee, W.: Development of ad hoc network for emergency communication service in disaster areas, *The 9th WSEAS International Conference on Applications of Computer Engineering*, pp.337–341 (Mar. 2010).
[22] Huda, N., Yasmeen, F., Yamada, S. and Sonehara, N.: An Approach for Short Message Resilience in Disaster-Stricken Areas, *IEEE Information Networking* (*ICOIN 2012*), pp.120–125 (Feb. 2012).

Kazuyuki Tasaka received his B.E. degree from Niihama National College of Technology in 2002, and his M.E. and Ph.D. degrees from Nara Institute of Science and Technology in 2004 and 2010, respectively. Since joining KDDI in 2004, he has worked in the field of network architecture, communication protocols and mobile communications. He is currently a research engineer of the Smart Network Administration Lab. at KDDI R&D Laboratories, Inc. He received the Best Paper Award for Young Researcher of IPSJ National Convention in 2005, the FIT Young Researcher Award of IPSJ National Convention in 2009 and the Paper Award of Multimedia, Distributed Cooperative and Mobile Symposium in 2007 and 2011. He is a member of IPSJ and IEICE.

Takashi Ozu received his B.E. and M.E. degrees fom Kyoto University in 2009 and 2011, respectively. Since joining KDDI in 2011, he has worked in the field of network architecture, communication protocols and mobile communications. He is currently an associate research engineer of the Smart Network Administration Lab. at KDDI R&D Laboratories, Inc. He is a member of IPSJ and IEICE.

Akira Idoue received his B.E. and M.E. degrees from Kobe University in 1984 and 1986, and his Ph.D. degree from the University of Electro-Communications in 2007, respectively. Since joining KDD (now KDDI) in 1986, he has worked in the field of network architecture, communication protocols, protocol testing and mobile communications. He is currently a senior manager of the Smart Network Administration Lab. at KDDI R&D Laboratories, Inc. He received the IPSJ Convention Award in 1993 and the Best Paper Award of the IPSJ National Convention in 1998. He is a member of IPSJ and IEICE.