A Practical Design and Implementation of Active Information Resource based Network Management System

Kazuto Sasai†, Johan Sveholm†, Gen Kitagata†, Tetsuo Kinoshita†
†Research Institute of Electrical Communication, Tohoku University, Sendai, Japan.
kazuto@riec.tohoku.ac.jp

Abstract

An autonomic system which can deal with network management tasks is an important issue to reduce the burden for network administrators. However, to realize the autonomic feature of this system, various knowledge and information with respect to network management tasks have to be utilized and integrated in the system. In this paper, we propose the practical design method and implementation of the Active Information Resource based Network Management System (AIR-NMS) which consists of activated information resources in a distributed network environment. The effectiveness of an implemented prototypical AIR-NMS is evaluated by experiments conducted on an experimental network environment. The information resource oriented design is shown to reduce the burden for administrators by the support to utilize and manage various information and knowledge in an autonomic way.

Keywords: *AIR-NMS, active information resource, network management system, autonomic computing, multiagent system*

1 Introduction

When the amount of network nodes rapidly grows, network management tasks become complex, and the burden for administrators continuously increases. Network management facilities, such as Network Management System (NMS) is one promising solution to this problem where artificial intelligence technology, such as software agents [15], is applied. Distributed agent based NMSs [3, 6] which have been developed over ten years, are applied to the following tasks, network fault resolution, protection from threats and optimization of performance. Currently, the autonomic computing paradigm inspired from biological systems have been proposed and is applied to the NMS[8]. Although there have been many studies to improve the autonomic features of NMS [4, 17], it still remains a difficult problem of how to use various heuristics of administrators together with management tools, which support human tasks such as performance analysis, policy decision and equipment management. Thus, new techniques for building /developing an autonomic NMS should address this problem.

Hence, we have studied and proposed a new concept of distributed information resources called Active Information Resource (AIR) [12], and proposed the AIR-based NMS (AIR-NMS) [11] to overcome the above mentioned problem. The AIR-NMS consists of two types of AIRs, I-AIR and K-AIR, where the former manages status information of various network elements, and the latter manages network management heuristics of human administrators. To design an NMS with the AIR concept, we have realized autonomous monitoring functions with cooperative I-AIRs [10].

In this paper, we propose and discuss the practical design and implementation of an AIR-NMS focusing on the network fault resolution task, which is an important problem to overcome for human administrators. In Section 2, the essential concept of designing the AIR-NMS is presented. Section 3 introduces the practical design of the AIR-NMS focusing on the fault resolution tasks. The implementation of a prototype system and evaluation experiments are demonstrated in Section 4. Finally, Section 5 concludes this paper.

2 Design Concept of AIR-NMS

2.1 Problems of Network Management Tasks

Network management systems (NMS) are basically developed to reduce the burden of network administrators by automating and supporting management tasks. A concept of autonomic network management system (ANMS) is studied as a highly automated NMS architecture [8]. Several research projects of ANMS [18, 16, 5, 14] are evaluated and discussed in [17]. An ANMS exhibits the so-called self-CHOP features, which is short for self-configuration, -healing, -optimization and -protection. To realize these self-CHOP features, a MAPE loop which consists of functions for monitoring, analyzing, planning and execution of managed network has to be defined. In the MAPE loop, various information and knowledge are required to manage and control the network. However, effective tools for acquirement, representation and usage of information and knowledge have not been provided for administrators in the existing NMS. Since only a part of the information and knowledge is defined and utilized, limited support functions are provided to deal with the specific management tasks, and the self-CHOP features are served in a limited way. For instance, as knowledge handling methodologies with respect to management tasks, ontology-based methodologies have been proposed [20, 7], however, it is difficult to apply them to multiple networks because this kind of knowledge representation includes a lot of domain dependent knowledge. Hence, the techniques which effectively utilize and coordinate various information and knowledge of managed networks are still missing.

Therefore, we introduce the AIR concept into an ANMS to resolve this problem, and realize an AIR-based NMS (AIR-NMS), which operates as an ANMS with AIRs in an autonomic way.

2.2 Active Information Resource for Network Management Tasks

An Active Information Resource (AIR) [12] is an extended entity of a distributed electric information resources, based on the active feature supported by the Knowledge of Utilization Support (KUS) and the Function of Utilization Support (FUS). The KUS consists of meta-level knowledge, i.e., knowledge for handling information resources and cooperation knowledge with other AIRs. The active function of an AIR is supported by FUS, which consists of various functions to process its information resources and for communication between AIRs. By applying the AIR concept to distributed academic information materials, the realized AIRs provide intelligent retrieval and shared functions for academic users [13].

Generally, the administrators have to deal with many kinds of distributed information allocated in various distributed elements of the network. Using the AIR concept, a distributed information resource can be extended to an AIR by attaching KUS and FUS, which have been designed with respect to the distributed information of the network. The realized AIRs for network management tasks can support a part of the management tasks of human administrators, and as a result, the burden of the administrators can be reduced in a systematic way [11].

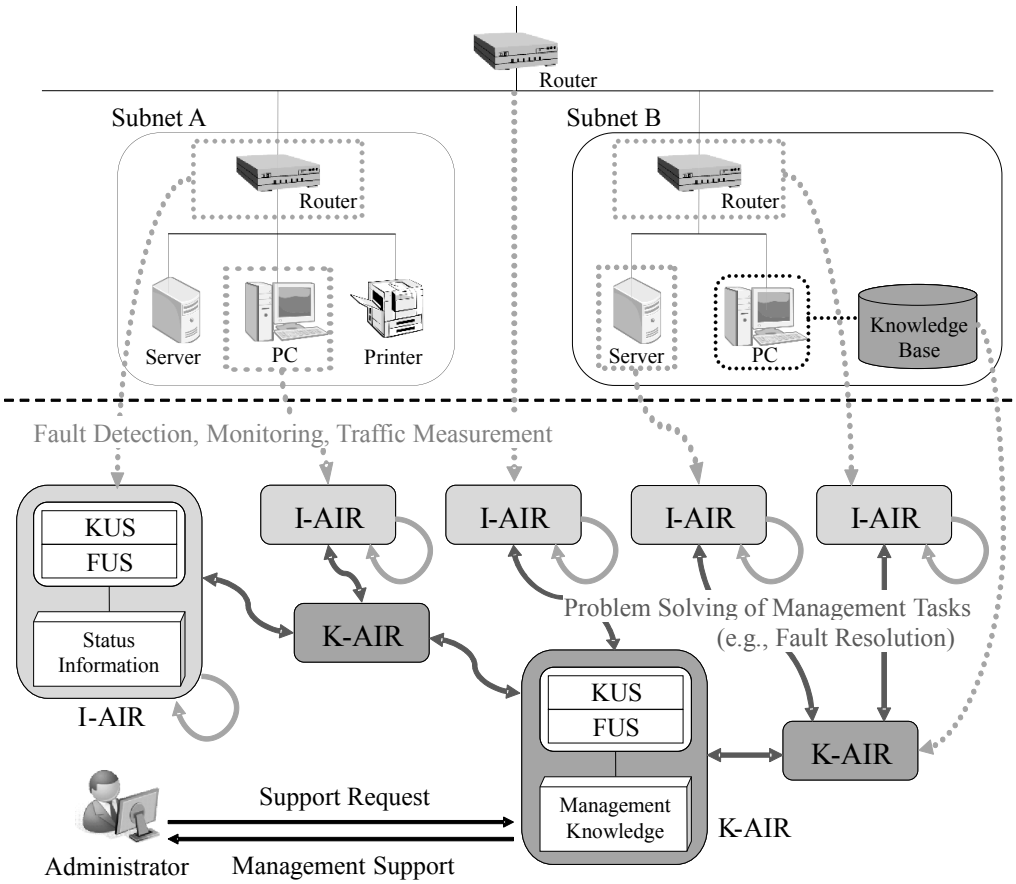


Figure 1. Schematic diagram of AIR-NMS.

2.3 Architecture of AIR-NMS

The essential functions of an AIR-NMS are as follows:

(1) Functions to accumulate, manage and utilize distributed status information of managed networks, such as fault detection, monitoring and traffic measurement.

(2) Functions to deal with various problem solving in network management tasks, such as network fault resolution and management operation.

To realize functions (1) and (2), we have designed and implemented a network status Information AIR (I-AIR) and a network management Knowledge AIR (K-AIR), respectively. Hence, an AIR-NMS is designed and implemented using I-AIRs and K-AIRs as shown in Figure 1.

An I-AIR manages the network status information that is acquired from network equipments as its information resource. The status information is generally classified into two types, static and dynamic information. The static status information indicates invariant information of the network such as domain name, host name, IP address, etc. The dynamic status information includes time varying information such as packet traffic, number of processes, application logs and so on. An I-AIR has the FUS to correct, retrieve and process the status information of the managed network, and the meta information of itself (KUS) to cooperate and respond to other AIRs. A design example of an I-AIR is presented in [10, 2], which improves the autonomic information acquisition function

of the experimental network system.

A K-AIR includes general network management knowledge (heuristics) of the human administrators as its information resource. The FUS of a K-AIR provides functions to retrieve and utilize the management knowledge, and the KUS provides meta knowledge of itself to cooperate with other AIRs.

These AIRs deal with the network management task together with an administrator in two ways. The administrator can send a support request to a K-AIR, and receive a response of support (Request-based driven mode), or a K-AIR can receive information of the detected event from an I-AIR, and then cooperate with other AIRs to solve the problem with respect to the event (Alarm-based driven mode). To realize effective cooperation among AIRs based on the above two operation modes, we have to provide a practical design method of the AIR-NMS for the administrators. In the next section, we describe the design and implementation of a prototype of an AIR-NMS for a network fault resolution task to demonstrate a practical design method of AIR-NMSs.

3 Building an AIR-NMS for Network Fault Resolution Task

3.1 Agent-based Design Method of AIR-NMS

The required functional characteristics of AIRs such as autonomic, flexible, situation-oriented, and knowledge-based problem solving capabilities, can easily be realized using software agents which can behave in an autonomous, reactive, cooperative, and proactive way, over the distributed environment. In the agent-oriented design of AIR, an AIR is generally designed and implemented as a single agent with respect to the contents of KUS and FUS of the AIR, which prescribe the problem solving capabilities of the AIR. Therefore, the design process of the AIR is divided into the following two phases with respect to the managed network, (1) design of knowledge for KUS and (2) design of functional behavior for FUS as well as a cooperation function. In the implementation of the AIR, we adopt the repository-based multiagent framework ADIPS/DASH and IDEA environment, that have been discussed in [9, 19], and where online tutorials are available in [1]. Since a rule-based agent architecture is supported in this framework, an agent with respect to the designed AIR is easily defined and realized by transforming and describing the results of the design of the AIR using the rule-based knowledge representation language provided by the ADIPS/DASH framework.

Thus, the design process of the AIR-NMS is defined as follows:

1. Definition of knowledge and functions of the required AIRs with respect to the management tasks.
2. Design of knowledge of AIRs.
3. Design of functions of AIRs.
4. Implementation of agents corresponding to AIRs.
5. Integration of AIRs to build an AIR-NMS.

According to the agent-oriented design of AIRs, a prototypical AIR-NMS is designed in this section, focusing on the network fault resolution task.

3.2 Design of K-AIRs for Network Fault Resolution Tasks

A network fault resolution task consists of the following subtasks:

1. Observation of *Symptoms*.
2. Assumption of conceivable *Causes*.
3. Diagnose of the exact *Causes* with *Diagnosis methods*.
4. Planning of the *Means* against the identified *Cause*.
5. Execution of the planned *Means*.

The management knowledge used in these subtasks are defined as the combination of the following knowledge description elements, *Symptom*, *Cause*, *Diagnosis method* and *Means*. Here, we design the following three types of K-AIRs for the fault resolution task,

K_{SC}-AIR: an AIR that assumes the conceivable *Cause* from the observed *Symptom*,

K_{CD}-AIR: an AIR that diagnoses the conceivable *Cause* using the empirical *Diagnosis method*,

K_{CM}-AIR: an AIR that plans the *Mean* for the detected *Cause*.

3.2.1 Description of management knowledge in K-AIRs

Figure 2 shows an example of the representation of the management knowledge using XML and XPath. The structure of assumption consists of the knowledge description of the mapping from a symptom to the expected conceivable causes, and it invokes the diagnosis request from the K_{SC}-AIR to the relevant K_{CD}-AIRs.

A K_{SC}-AIR deals with management knowledge K_{SC} to assume the conceivable causes from an observed symptom. A $\mathbf{K}_{SC}$ consists of a *Symptom* and the conceivable *Causes* assumed to be the origin of the symptom. Figure 2(a) shows an example description of $\mathbf{K}_{SC}$ for the case of the symptom "unable to send mail," which specifies the faulty situation that a user cannot send mail. A $\mathbf{K}_{SC}$ is structured by the tag `<sc>` which has an attribute “symptom" indicating a subject of a symptom, i.e., "unable to send mail" is set to “symptom." Assumed conceivable causes is designated by the tags `<cause>`, and in this example, three conceivable causes are specified for the symptom “unable to send mail", i.e., “unable to resolve name", which specifies the cause with respect to the error of DNS, “network connection failure," which specifies the cause with respect to a network connection error, and “over sendable size limit", which specifies the size of the sent mail exceeding the mail size limit.

A K_{CD}-AIR deals with management knowledge $\mathbf{K}_{CD}$ to diagnose and verify a cause assumed by a K_{SC}-AIR. A $\mathbf{K}_{CD}$ consists of three descriptions, a *Cause*, a *Diagnosis Method* to verify the cause and a *Diagnosis Report* to notify the result of the diagnosis to the administrator. Figure 2(b) shows an example description of a $\mathbf{K}_{CD}$ for the cause “over sendable size limit," which specifies the particular situation that the size of a mail sent by a user exceeds the configuration of mail size limit. A $\mathbf{K}_{CD}$ is structured by the tag `<cd>` which has an attribute “cause" indicating a subject of a cause, i.e., "over sendable size limit" is set to “cause." The tag `<cd>` has the child tags `<dm>` and `<dr>`, that specify a *Diagnosis Method* and a *Diagnosis Report*, respectively. The child tag `<p>` of the tag `<dm>` indicates the diagnosis processes to verify the condition of the cause. The processes are sequentially executed. In this example, the diagnosis process described as “`request #//sent_mail_size# from #source#`" is the syntactical representation, in which the term `#//sent_mail_size#` is a variable of the required information, and `#source#` is a variable of the destination. The process specifies an action to send a message which requests information `#//sent_mail_size#`, to a destination `#source#`. The last line of the tags `<p>`, “`true (#//sendable_size_limit# -lt #//sent_mail_size#)`" specifies an action to test that the variable `#//sendable_size_limit#` is less than

```
<sc symptom="unable to send mail">
  <cause>unable to resolve name</cause>
  <cause>network connection failure</cause>
  <cause>over sendable size limit</cause>
</sc>
```

(a). K_{SC}: Cause assumption.

```
<cd cause="over sendable size limit">
  <dm>
    <p>request #//sent_mail_size# from #source#</p>
    <p>request #//client/mta/servername# from #source#</p>
    <p>request #//sendable_size_limit# from #//client/mta/servername#</p>
    <p>true(#//sendable_size_limit# -lt #//sent_mail_size#)<p>
  </dm>
  <dr>
    Sent mail size, #//sent_mail_size#KB, is over sendable size limit,
    #//sendable_size_limit#KB, which is set to SMTP server
    #//server/mta/name#.
  </dr>
</cd>
```

(b). K_{CD}: Cause diagnosis.

```
<cm cause="over sendable size limit">
  <m>
    Setting at SMTP server #//client/mta/servername# can be changed,
    because its OS is
    #//host/os/name@//client/mta/servername#=(CentOS,Fedora)
    and its MTA is #//server/mta/name@//client/mta/servername#=(Postfix, Sendamail, Qmail)
        ,
    by following operations ...
  </m>
</cm>
```

(c). K_{CM}: Means planning.

Figure 2. Examples of described network management knowledge of the K-AIRs.

`#//sent_mail_size#`, and if the result is true, then the diagnosis of the cause is successfully finished. Moreover, `<dr>` designates a description of the *Diagnosis report* to issue a report to the administrators. It holds human readable text descriptions specifying variables such as `#//sent_mail_size#`, `#//sendable_size_limit#` and `#//server/mta/name#`.

Finally, a K_{CM}-AIR deals with management knowledge $\mathbf{K}_{CM}$ to recommend the *Means* for a detected cause to the administrators. A $\mathbf{K}_{CM}$ consists of a *Cause* and the *Means* for the cause. Figure 2(c) shows an example description of a $\mathbf{K}_{CM}$ for the cause “over sendable size limit." A $\mathbf{K}_{CM}$ is structured by the tag `<cm>` which has an attribute “cause" indicating a subject of a cause, i.e., "over sendable size limit" is set to “cause." The child tag `<m>` specifies the *Means* represented as a means template which includes the specific variables, in this examples,
`#//client/mta/servername#` which represents a hostname of MTA configured in the client, and `#/server/mta/name@//client/mta/servername#` which represents the information of a remote host indicated by the description followed by the symbol `@`, i.e., the name of the MTA server host. If the remote host is indicated in the network, the K_{CM}-AIR acquires its name from the description. A description such as `(CentOS,Fedora)` means the description is effective if the server OS is CentOS or Fedora.

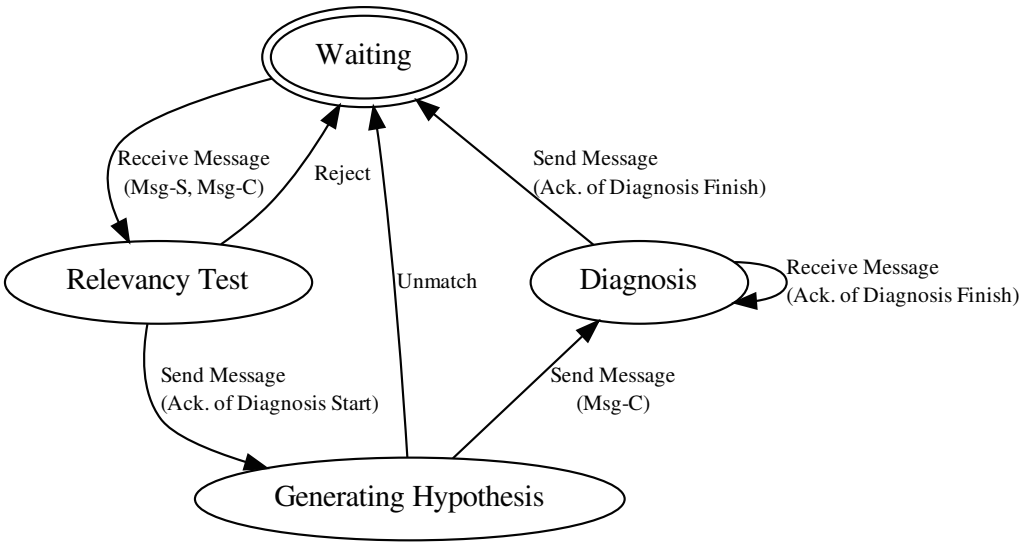


Figure 3. State transition diagram of K_{SC}-AIRs.

3.2.2 Behavior of K-AIRs

The K-AIRs with the described management knowledge, actively cooperate with each other to solve a problem of fault resolution. The behavior of the K_{SC}-AIR, the K_{CD}-AIR and the K_{CM}-AIR is designed in this section,

The K_{SC}-AIR deals with a fault resolution subtask to assume the conceivable causes for an observed symptom. The FUS of the K_{SC}-AIR provides the functions to send the messages to other K_{SC}-AIRs and K_{CD}-AIRs where the observed symptom can be handled, based on the state transition depicted in Figure 3. A K_{SC}-AIR has four states, “Waiting," “Relevancy Test," “Generating Hypothesis" and “Diagnosis." In the "Waiting" state, the K_{SC}-AIR becomes active if it receives the message Msg-S or Msg-C. The messages with the symbol “Msg-" are important to designate the operations of the cooperation among the AIRs. For instance, a Msg-S is used to announce an observed symptom to the AIRs, and a Msg-C is used to request a diagnosis task to detect assumed conceivable causes. The details of the messages are explained in Section 3.2. In the “Relevancy Test" state, the K_{SC}-AIR checks whether the received message is acceptable or not, and sends an Ack message to a message sender in order to start the diagnosis. In the “Generating Hypothesis" state, the K_{SC}-AIR assumes the conceivable causes for the symptom according to the management knowledge $\mathbf{K}_{SC}$ and sends a Msg-C to the K_{SC}-AIRs and/or K_{CD}-AIRs. In the “Diagnosis" state, the K_{SC}-AIR waits until the diagnosis is finished and thereafter returns to the “Waiting" state. In the “Generating Hypothesis" state, for an unmatched symptom, the AIR do nothing and return to the “Waiting" state. This behavior is usual for a software agent. In the “Diagnosis" state, the K_{SC}-AIR waits for the response from the K_{CD}-AIR. If the requested symptom is not a failure, the “Diagnosis" state timeouts and return to “Waiting" state. The exception-handling scheme in the state transition follows the basic software agent cooperation scheme [15].

The K_{CD}-AIR deals with a fault resolution subtask to diagnose an exact cause assumed by the K_{SC}-AIR. The FUS of the K_{CD}-AIR provides the functions to correct information from the I-AIRs. It sends messages to the K_{CM}-AIRs, and presents a diagnosis report to an administrator when assumed cause can be handled, based on the state transition depicted in Figure 4. The K_{CD}-AIR have six states, “Waiting," “Relevancy Test," “Diagnosis," “Correcting Information," “Generating

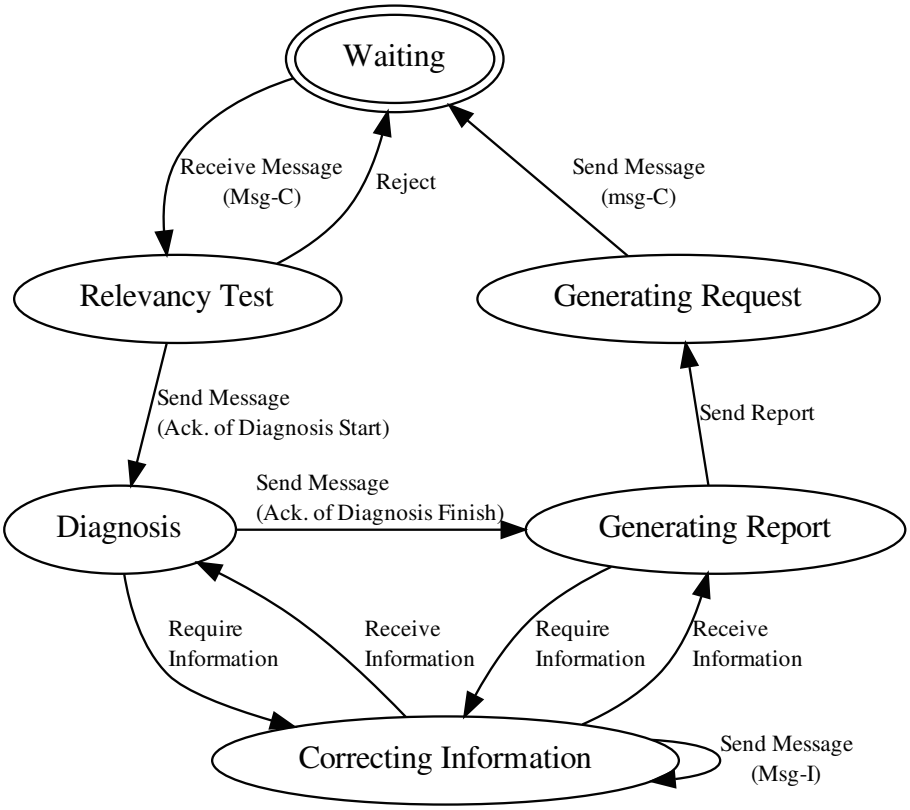


Figure 4. State transition diagram of K_{CD}-AIRs.

Report" and “Generating Request." In the “Waiting" state, a K_{CD}-AIR becomes active when it receives the message Msg-C, and transits to the “Relevancy Test" state. In the “Relevancy Test" state, a K_{CD}-AIR checks and sends an Ack message to the message sender in order to start the diagnosis if the received message is acceptable. In the “Diagnosis" state, a K_{CD}-AIR conducts the diagnosis method described in management knowledge $\mathbf{K}_{CD}$. If a variable corresponds to the required information is found in the management knowledge $\mathbf{K}_{CD}$, the K_{CD}-AIR transits to the “Correcting Information" state. In the “Correcting Information" state, a K_{CD}-AIR sends the message Msg-I to the I-AIRs in order to get information required for diagnosis and then returns to the “Diagnosis" state. When the diagnosis is successfully finished, the K_{CD}-AIR transits to the “Generating Report" state. In the “Generating Report" state, a K_{CD}-AIR generates a diagnosis report by filling in the variables of a template. If additional information to fill in the template is required, the K_{CD}-AIR transits to the “Correcting Information" state. In the “Generating Request" state, a K_{CD}-AIR generates a request Msg-C and sends it to the K_{CM}-AIRs.

The K_{CM}-AIR deals with the fault resolution subtask to plan the means for a detected cause. The FUS of the K_{CM}-AIR provides the functions to plan the means for a detected cause, and presents the means to an administrator when a detected cause can be handled, based on the state transition depicted in Figure 5. A K_{CM}-AIR has four states, “Waiting", “Relevancy Test", “Means Planning" and “Correcting Information." In the "Waiting" state, a K_{CD}-AIR becomes active if it receives a message Msg-C, and transits to the “Relevancy Test" state. In the “Relevancy Test" state, a K_{CM}-AIR checks whether the received message is acceptable or not, and sends an Ack message to a message sender in order to start the diagnosis. In the “Means Planning" state, a K_{CM}-AIR generates

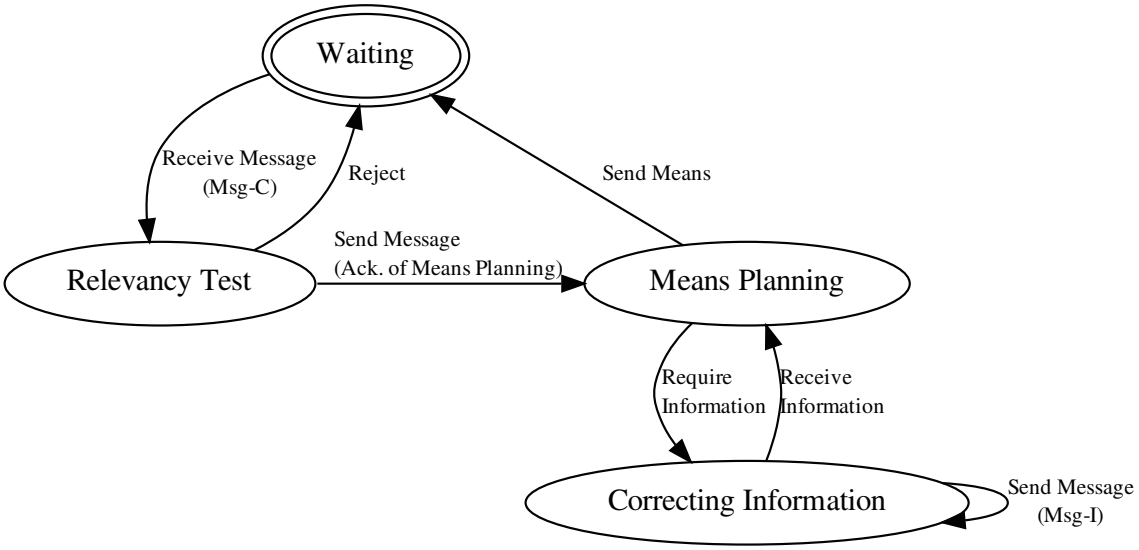


Figure 5. State transition diagram of K_{CM}-AIRs.

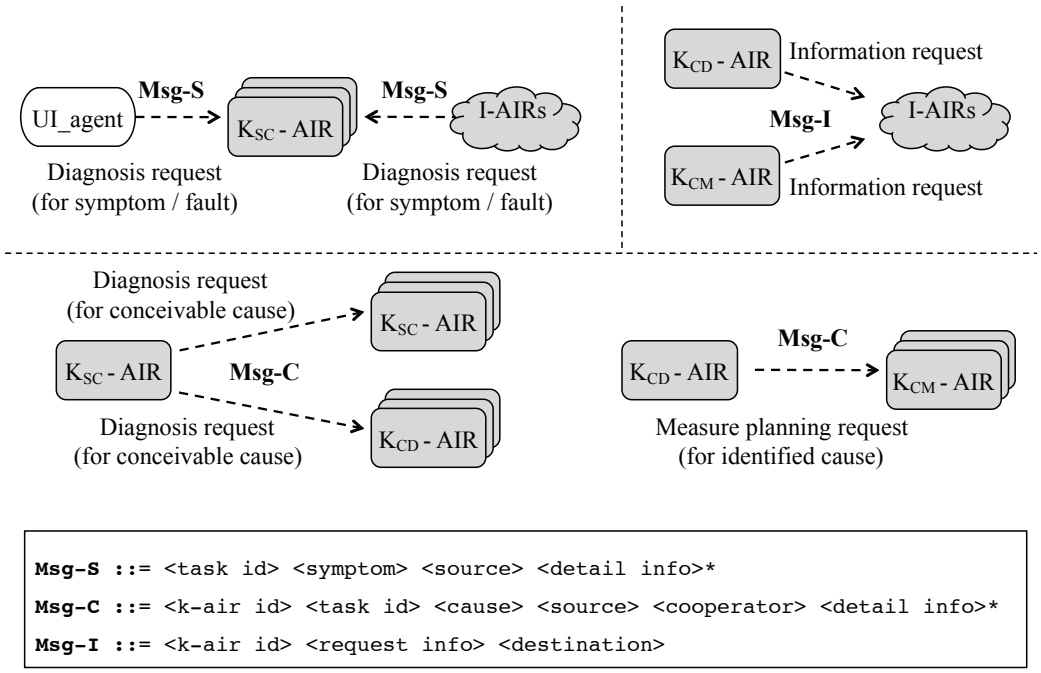
the means recommendation by filling in the variables of a template. If a variable corresponding to the required information found in the template, a K_{CM}-AIR transits to the “Correcting Information" state. In the “Correcting Information" state, a K_{CM}-AIR sends a message Msg-I to the I-AIRs in order to get the information required for diagnosis, and returns to the “Means Planning" state. When the means planning is finished, the K_{CM}-AIR sends means recommendation to an administrator, and returns to the “Waiting" state.

3.2.3 Cooperation of K-AIRs

Figure 6 shows a schematic diagram of both the messaging among K-AIRs and the messaging between K-AIRs and I-AIRs of a diagnosis process for a fault resolution support. The following three kinds of messages, Msg-S, Msg-C and Msg-I, are defined in this section.

The Msg-S is a support message of a fault resolution task that an administrator asks the AIR-NMS to get. The Msg-S is defined as the diagnosis request message sent from an UI-agent (administrator) and/or I-AIR. The Msg-S consists of the following four descriptions, `<task id>`, `<symptom>`, `<source>` and `<detail info>`$\star$. The `<task id>` is an identifier of a resolution support task, which is managed and generated by UI-agent, `<sympto>` is a text description of the observed symptom, `<source>` is a fault location corresponding to a host on which a fault occurs, and `<detail info>`$\star$ is additional specific information related to the symptom.

The Msg-C is used in the following two situations. The first is to represent a diagnosis request from a K_{SC}-AIR to the K_{SC}-AIRs and K_{CD}-AIRs, and the second is to represent a means planning request from a K_{CD}-AIR to the K_{CM}-AIRs. Note that Msg-C is sent not only to the K_{CD}-AIRs but also to the K_{SC}-AIRs. This is useful to elaborate the problem. For example, suppose that the K_{SC}-AIR which has the same knowledge as shown in Section 3.2.1, receives a diagnosis request of the fault symptom “unable to send mail." For the K_{SC}-AIR, three possible causes were described, however, since several deeper causes are conceivable for the cause “network connection failure," the K_{CD}-AIRs corresponding to the cause cannot detect the exact cause. In such a case, the K_{SC}-AIR sends a Msg-C to an other K_{SC}-AIR, which deals with the symptom “network connection failure" and the possible causes, e.g., “cable error," “misconfiguration of IP address," etc. The format




```
Msg-S ::= <task id> <symptom> <source> <detail info>*
Msg-C ::= <k-air id> <task id> <cause> <source> <cooperator> <detail info>*
Msg-I ::= <k-air id> <request info> <destination>
```

Figure 6. Messaging scheme in an AIR-NMS.

of Msg-C consists of <k-air id>, <task id>, <cause>, <source>, <cooperator> and <detail info>*. The <k-air id> is the unique AIR name of message sendar and <cooperator> is the information about the chain reaction activation of the AIRs. The AIR name is denoted as "<AIR-type>_<unique number>:<workplace name>:<hostname>". The other descriptions are inherited from Msg-S.

The Msg-I is used to send an information acquisition request from the K-AIRs to the I-AIRs. It consists of <k-air id>, <request info> and <destination>. <request info> is the attribute values of the requested information and <destination> is a destination address of the information requester.

3.3 Design of I-AIRs to Cooperate with K-AIRs

The fault diagnosis and resolution functions of the AIR-NMS are realized based on the flexible provision of the I-AIRs that handle various information of network equipment. To realize the flexible provision of the I-AIRs, we have designed an I-AIR, called I_H-AIR which can be instantiated onto various type of hosts and respond to various type of requests from the K-AIRs. In this section, we explain the design of KUS and FUS of the I_H-AIR to cooperate with the K-AIRs in a fault resolution task.

3.3.1 Description of I_H-AIRs

Here, we design a set of I_H-AIRs which deals with status information of hosts, servers and clients. Figure 7(a) shows an example of status information of a host, which is specified by attributes, such as hostname, subnet, OS, NIC, etc. In a network fault resolution tasks, we often need temporal status information such as an occurrence time of the fault. The KUS in an I_H-AIR should include knowledge to acquire, analyze and integrate temporal information. Figure 7(b) shows an example of KUS for a host which runs Windows OS. Since different kinds of hosts require different tools

```
<si>
  <host>
    <name>srvA</name>
    <subnet>subnetA</subnet>
    <domain>example.com</domain>
    <fqdn>srvA.subnetA.example.com</fqdn>
    <os>
      <name>CentOS</name>
      <version>5</version>
    </os>
    <nic name="eth0">
      <ipadress>172.16.0.2</ipaddress>
      <network>172.16.0.0</network>
      <netmask>255.255.255.0</netmask>
      <status>up</status>
      <driver>running</driver>
    </nic>
    <gateway>
      <ipaddress>172.16.0.1</ipaddress>
      <network>172.16.0.0</network>
      <netmask>255.255.255.0</netmask>
    </gateway>
    <link destination="172.16.0.1" ping_status="reachable"/>
  </host>
  ...
</si>
```

(a). Status information of I_H-AIR.

```
<host>
  <name>WMI</name>
  <subnet>Command_Windows</subnet>
       <Link>Command_Windows</Link>
  <fqdn>Command_Windows</fqdn>
  ...
</host>

<Command_Windows>
  <key>/link</key>
  <methodname>ping</methodname>
  <ok_checkString>0% loss</ok_checkString>
  <false_checkString>100% loss</false_checkString>
  <checkinfo>
    <check>exist</check>
    <ok_exist>reachable</ok_exist>
    <false_exist>unreachable</false_exist>
  </checkinfo>
  ...
</Command_Windows>
```

(b). Utilization support knowledge KUS of I_H-AIR.

Figure 7. Examples of the description of I_H-AIR.

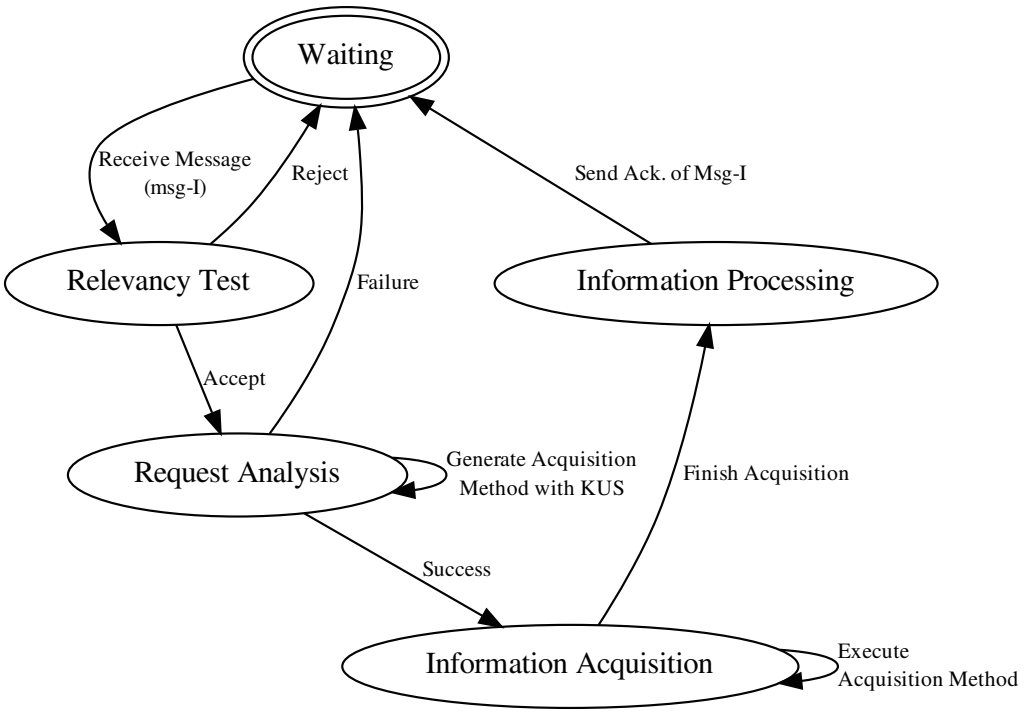


Figure 8. State transition diagram of I-AIR.

in order to acquire the same kind of information, the KUS should have knowledge of various types of acquisition methods. The descriptions annotated by the <host> tag specify the acquisition methods of the requested information. In this example, “WMI" indicated by the tag <name> specifies an acquisition method of the hostname. When an I_H-AIR receives a request of hostname from K-AIRs, it acquires the information by using the WMI (Windows Management Interface) tool. The <Command_Windows> tag specifies the procedures of acquisition methods for Windows command line tools. A description with the <key> tag represents a request key, and a description with <methodname> tag represents called command. If a method is activated, the result of the acquisition is analyzed by using command descriptions represented by the <*_checkString>, and information descriptions annotated by the <checkinfo> tag. When an I_H-AIR receives a request of information specified by the tag <Link>, the acquisition method described by the tag <Command_Windows> is referred, and the specific method “ping" is executed. If the result of the execution is “0 loss", it is evaluated by <ok_checkString>. Finally, the representation “reachable," indicated by the tag <ok_exist> is referred and registered to the status information. The I_H-AIR sends a reply message by using the updated status information.

3.3.2 Behavior of I-AIRs

Although the original behavior of an I-AIR discussed in [10] is more complex, in this paper, it is designed focusing on the functions related to cooperation with the K-AIRs. The FUS of the I_H-AIR consists of two kinds of functions, to communicate the K-AIRs and to acquire the status information required by the K-AIRs. These functions are realized by designing the behavior of the I_H-AIR, which is represented by state transition of an agent. The state transition diagram of I_H-AIR is shown in Figure 8. An I-AIR has three particular states, “Request Analysis", “Information Acquisition" and “Information Processing". In the “Request Analysis" state, the I-AIR checks

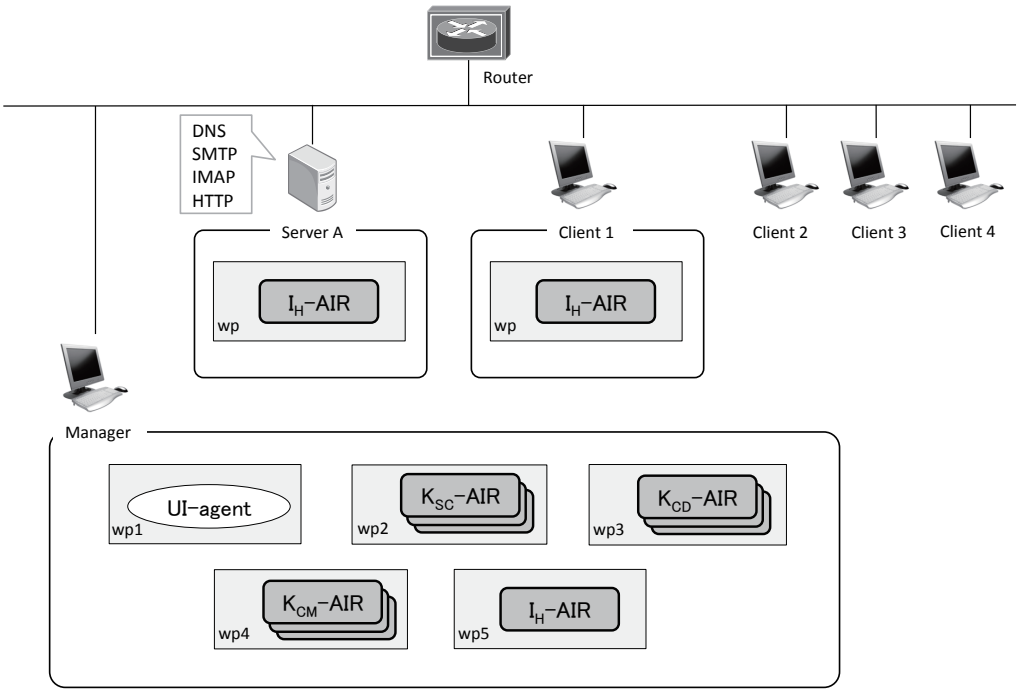


Figure 9. Configuration diagram of prototypical system.

if a requested information is processable. In the “Information Acquisition" state, an information acquisition method is executed. In the “Information Processing" state, the result of the acquisition is processed to requested format. The I-AIR sends the final result to the request sender, i.e., the K-AIRs.

The AIR-NMS is realized by allocating the designed AIRs to network environment. In the next section, we explain the implemented prototype system.

4 Experiments and Evaluation

4.1 Implementation of a Prototype System

We implement a prototype system of an AIR-NMS for network fault resolution using the repository-based multiagent framework ADIPS/DASH, as explained in Section 3. The AIRs of the AIR-NMS are designed as various agents stored in the agent repository. The ADIPS/DASH framework provides a runtime environment for agents, called the workplace, which operates on a distributed platform such as a PC allocated over the networked environment. The agents are instantiated from the agent repository onto the workplaces based on the requests of users, in order to execute the distributed problem solving tasks.

Figure 9 shows the configuration of the prototypical AIR-NMS over the experimental network environment, which consists of four client PCs, one server PC, and one administrator PC, that are depicted as Client1-4, ServerA and Manager, respectively. ServerA runs CentOS 5 with BIND 9.3.6 for DNS, Postfix 2.3.3 for SMTP, Dovecot 1.0.7 for IMAP, and Apache 2.2.3 for HTTP server. Client1-4 and Manager runs Windows XP OS.

At runtime of the AIR-NMS, the agents of both the I-AIRs and the K-AIRs are instantiated

Table 1. Implemented K-AIRs for prototype system.

Type	No.	Name	Tags(n.)
K_{SC}-AIR	1	unable to send mail	<sc>, <cause>(7)
	2	unable to receive mail	<sc>, <cause>(8)
	3	unable to open web page	<sc>, <cause>(9)
	4	name resolution error	<sc>, <cause>(7)
K_{CD}-AIR	1	restricted access	<cd>, <dm>, <p>(2), <dr>
	2	name server process down	<cd>, <dm>, <p>(2), <dr>
	3	http link not exist	<cd>, <dm>, <p>(2), <dr>
	4	http server process down	<cd>, <dm>, <p>(2), <dr>
	5	pop or imap server process down	<cd>, <dm>, <p>(4), <dr>
	6	connection port closed	<cd>, <dm>, <p>(2), <dr>
	7	smtp server process down	<cd>, <dm>, <p>(4), <dr>
K_{CM}-AIR	1	name server process down	<cm>, <m>
	2	http link not exist	<cm>, <m>
	3	connection port closed	<cm>, <m>
	4	http server process down	<cm>, <m>
	5	pop or imap server process down	<cm>, <m>
	6	smtp server process down	<cm>, <m>

Table 2. Implemented I-AIRs for prototype system.

Type	No.	Name	Host	Tags(n.)
I_H-AIR	1	Linux	ServerA	<si>, <host>, <agent>, <service>, etc.(50)
	2-5	Windows_XP	Client1-4	<si>, <host>, <agent>, <service>, etc.(38)
	6	Windows_XP	Manager	<si>, <host>, <agent>, <service>, etc.(38)

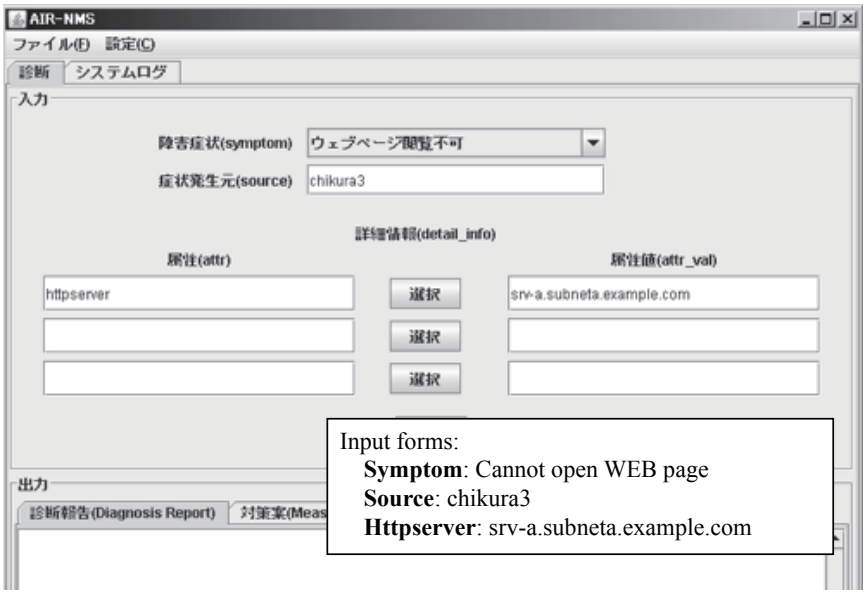
onto the distributed workplace of various PCs of the experimental network. Although the K-AIRs can operate on distributed workplaces in the prototype system, all K-AIRs are instantiated onto a workplace of the administrator PC. We show the list of implemented K-AIRs in Table 1. The tags and their numbers are shown together in the column Tags(n.) in order to represent the size of the AIRs. Four kinds of K_{SC}-AIRs, seven kinds of K_{CD}-AIRs and six kinds of K_{CM}-AIRs are implemented and allocated in the prototype system. The I-AIRs operate in the workplaces of the PCs, respectively. Table 2 shows a list of implemented I-AIRs. Two kinds of I_H-AIRs are designed with respect to the types of OS, and overall six I_H-AIRs are implemented and allocated in the prototype system.

4.2 Experiment on Fault Resolution Functions

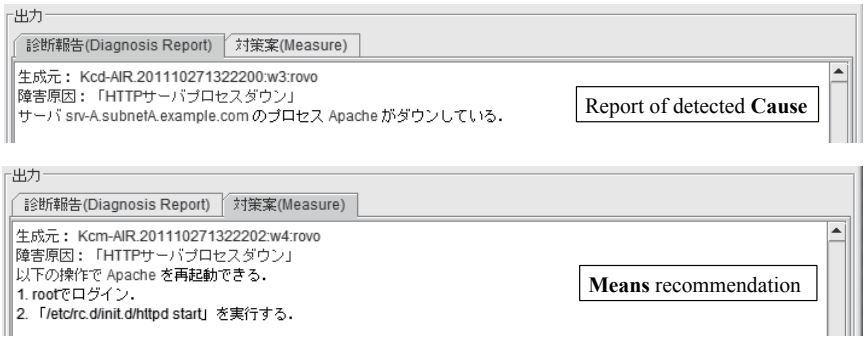
To validate the functions of fault resolution task of the prototypical AIR-NMS, we test the system's behavior in the given faulty situations.

For instance, we set a faulty situation that a HTTP server process on ServerA is down. Suppose that a user reports a failure of access to a Web page on ServerA from Client1. This situation can be considered that a failure occurs in an internal division business Web system, for instance.

Figure 10(a) shows a screenshot of the fault resolution interface of the AIR-NMS, which is provided by the UI-agent. Using the interface, first, an administrator selects the kind of *Symptom* from a drop-down menu on the top of the window and next, inputs an IP address or a hostname of the *Source* of the symptom using the text box below. Moreover, the administrator can input additional information in the form of pairs of attribute and attribute values using the text boxes.



(a) Interface of AIR-NMS.



(b) Output of AIR-NMS.

Figure 10. Screenshots of prototype system.

In this experiment, the following information are input:

Symptom: "`Cannot open Web page`"

Source: "`chikura3`" (Client1)

Detail information [*attribute*: *attribute value*]:

"`httpserver`":"`srv-a.subneta.example.com`" (Server's FQDN)

Then a fault resolution process can be started by clicking the button "diagnosis".

The response of the AIR-NMS with respect to the above input is given as shown in Figure 10(b). The AIR-NMS reports a diagnosis report and the means recommendation. In this experiment, the diagnosis report includes the following:

```
Source AIRid:  "Kcd-AIR_20110271322200:w3:manager"
Cause:  "HTTP server process down"
The process Apache of srvA.subnetA.example.com is down.
```

On the other hand, a means recommendation is given as follows,

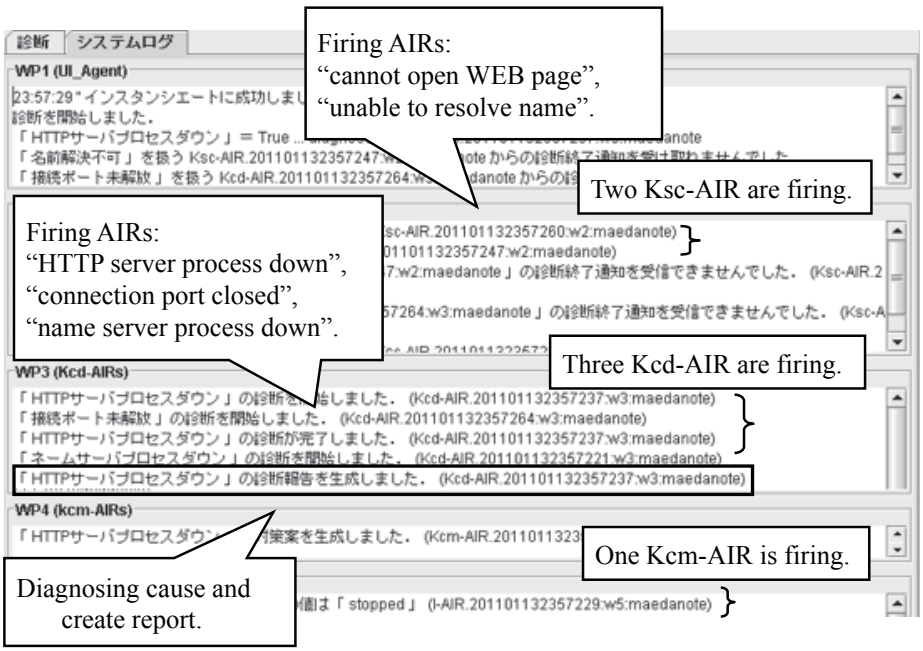


Figure 11. Screenshot of execution log.

```
Source AIRid:  "Kcm-AIR_201102071322202:w3:manager"
Cause:  "HTTP server process down"
You can restart the process Apache as follows:
 1. Login as root to the server.
 2. Execute the command "/etc/rc.d/init.d/httpd start".
```

By executing this procedure, we can exactly recover from the fault.

An execution log of the above recommendation process can be given as shown in Figure 11. In this case, the two K_{SC}-AIRs corresponding to the *symptoms* “cannot open web page" and “unable to resolve name," are activated. After that the three K_{CD}-AIRs corresponding to the *causes* “http process down," “connection port closed," and “name server process down" are activated. Finally, the K_{CM}-AIR corresponding to “http server process down" is activated, and the means recommendation is returned to the UI-agent. In the fault resolution task, the following information request to I_H-AIRs are conducted, #/si/service/server/dns/status#, #Permission denied@httpd# , #File does not exist@httpd# and #/si/service/server/web/status#. All requests are successfully received and processed by the I_H-AIR of ServerA.

Through this experiment, we can confirm that the prototypical AIR-NMS can deal with the given faulty situation and present a report on the cause of the fault autonomously based on the cooperation of K-AIRs and I-AIRs for fault resolution.

4.3 Experiment with Human Administrators

To test the capabilities of the prototypical AIR-NMS, we conduct a fault resolution experiment with actual human users. The experimental network is the same as the previous experiment, with user playing the role of an administrator to resolve the given faulty situation. Thereafter, we observe and compare the results of the operations by human users, called “manual-based operations," with

those supported by AIR-NMS, called “AIR-NMS-based operations."

We record the behavior of the user by two cameras and capture the screens of the PCs to movies. In this experiment, ServerA is implemented as a virtual machine on the Manager for convenience. The user playing the role as an administrator can operate any PC in the experimental environment.

The experimental procedure of the manual-based operation is described as follows:

Step1. An experimenter set up a faulty situation in the experimental network.

Step2. The experimenter asks the user to resolve the given faulty situation by telling a symptom and a site of fault occurence.

Step3. The user can operate all of the network’s management functions in the experimental environment, e.g., internet search, windows utilities, Linux/Unix utilities, etc.

Step4. The consumption time of the experiment is defined as the duration until the user recovers the given symptom.

In the AIR-NMS-based operation, the experimenter is allowed to teach the user how to use the AIR-NMS in Step2.

Five students of Graduate School of Information Science, Tohoku University, participate in the experiment as the users. All of the students have basic knowledge of network technologies, however, only two of them are familiar with network management tasks, and the others are not.

The following four cases are selected as the faulty situations:

Cause1. HTTP server process down.

Cause2. Link missing of target Web page.

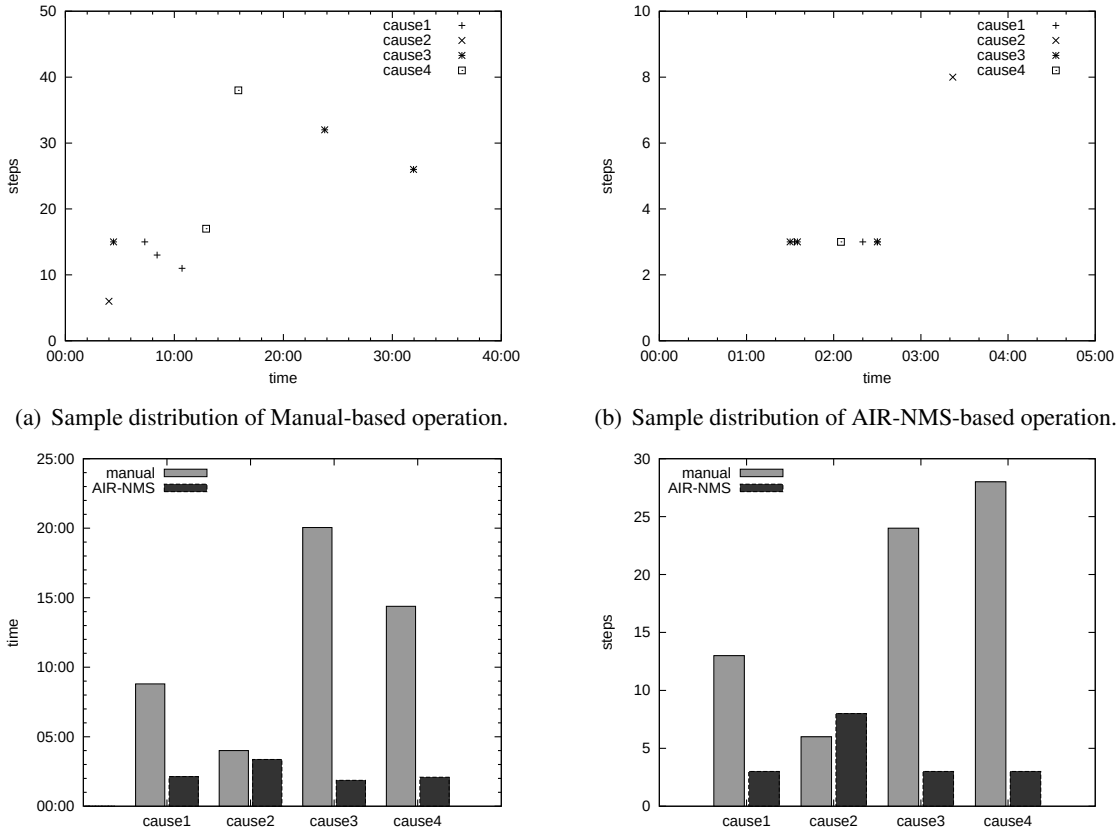
Cause3. DNS server process down.

Cause4. HTTP port is closed.

Cause1 implies a situation where a HTTP process of ServerA is down, Cause2 implies that the link of a Web page on the ServerA is missing because the target Web page is removed, Cause3 implies that a DNS server process on ServerA is down, and Cause4 implies that a HTTP port of ServerA is closed. We select several cases for respective users, and the users resolve the faults in two ways, manual-based and AIR-NMS-based operation.

We summarize the results of the experiment in Figure 12. Figure 12(a) and Figure 12(b) show the distributions of sample data of manual-based and AIR-NMS-based operation, respectively. The consumption time and the operation steps of the fault resolution tasks are depicted on horizontal and vertical axis, respectively. We can see that the time and steps depend on the types of conceivable causes, and the users can resolve the faults in pretty much the same time and steps by the AIR-NMS-based operation. It implies that an AIR-NMS can effectively support the users by guiding them in the fault resolution process. Note that the two figures are depicted with different scale. Figure 12(c) and Figure 12(d) show the comparison between manual and AIR-NMS-based operation with the time and steps, respectively. The AIR-NMS shows its effectiveness in most of the cases, but the result of Cause2 is almost the same because its conceivable cause “missing link" is comparatively easier than the other.

In this section, we conducted the two experiments with the experimental prototypical system to validate the effectiveness of the proposed practical design of the AIR-NMS. From the experiment, it is confirmed that the problem solving capabilities of the AIR-NMS for the fault resolution task can be realized by the cooperative problem solving behavior of the AIRs. The results of the time and steps for the fault resolution tasks imply that the AIR-NMS can efficiently guide the administrators to the means of the fault resolution. For unskilled users, the AIR-NMS dramatically reduce



(a) Sample distribution of Manual-based operation.

(b) Sample distribution of AIR-NMS-based operation.

(c) Comparison of consumption time.

(d) Comparison of operation steps.

Figure 12. Results of experiment with human administrators.

consumption time and workload. Actually, it shows an average 80% reduction of time and steps for the case of an unskilled user. On the other hand, for skilled users, an average 70% reduction of time and steps is realized with only one exception. It implies that the AIR-NMS effectively supports the fault resolution task not only for unskilled users but also for skilled users to greater and lesser degrees. As described in the previous sections, the effective support function is realized by autonomous cooperation between K-AIR and I-AIR that are agent-based designs of KUS and FUS for management knowledge (K-AIR) and status information (I-AIR). Hence, the proposed practical design of the AIR-NMS is useful and effective to realize the AIR-NMS, which can deal with a part of network management tasks to reduce the burden of administrators.

5 Conclusion

Focussing on the fault resolution task of network administrators, the practical design and implementation of an AIR-based NMS (AIR-NMS) is proposed and evaluated in this paper. Through the experiment, using a prototypical AIR-NMS, the AIR-NMS shows the capabilities of resolving faulty situations based on the cooperation of AIRs to support human administrators. It remains as future work to extend the capabilities of the AIR-NMS to deal with many kinds of faults and anomalies of network system by introducing various AIRs. The effective design support functions of AIRs have to be studied to support the designers of the AIR-NMS.

References

[1] Idea/dash tutrial [online]. Available: http://www.k.riec.tohoku.ac.jp/idea/index.html.

[2] Sameera Abar, Susumu Konno, and Tetsuo Kinoshita. Autonomous network monitoring system based on agent-mediated network information. *International Journal of Computer Science and Network Security*, 8(2):326–333, 2008.

[3] Andrezej Bieszczad, Bernard Pagurek, and Tony White. Mobile agents for netowrk management. *IEEE Communications Surveys*, 1(1):2–9, 1998.

[4] David M. Chess, Alla Segal, Ian Whalley, and Steve R. White. Unity: Experiences with a prototype autonomic computing system. In *ICAC'04*, pages 140–147, 2004.

[5] Xiangdong Dong, Salim Hariri, Lizhi Xue, Huoping Chen, Ming Zhang, and Soujanya Rao Sathija Pavuluri. Autonomia: An autonomic computing environment. pages 61–68, 2003.

[6] Ahmed Karmouch. Mobile software agents for telecommunications. *IEEE Communications Magazine*, pages 24–25, July 1998.

[7] John Keeney, David Lewis, and Declan O'Sullivan. Ontological semantics for distributing contextual knowledge in highly distributed autonomic systems. *Journal of Network and System Management, Special Issue on Autonomic Pervasive and Context-aware Systems*, 15(1):75–86, 2007.

[8] Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *IEEE Computer*, 36(1):41–50, 2003.

[9] Tetsuo Kinoshita and Kenji Sugawara. Adips framework for flexible distributed systems. In T. Ishide, editor, *Multiagent Platforms, LNAI 1599*. 1998.

[10] Susumu Konno, Sameer Abar, Yukio Iwaya, and Tetsuo Kinoshita. Effectiveness of autonomous network monitoring based on intelligent-agent-mediated status information. In H.G. Okuno and M. Ali, editors, *LNAI 4570*, pages 1078–1087. Springer-Verlag, Berlin Heidelberg, 2007.

[11] Susumu Konno, Yukio Iwaya, Toru Abe, and Tetsuo Kinoshita. Design of network management support system based on active information resource. In *AINA (1)'04*, pages 102–106, 2004.

[12] Baoning Li, Toru Abe, Kenji Sugawara, and Tetsuo Kinoshita. Active information resource: Design concept and example. In *AINA'03*, pages 274–277, 2003.

[13] Baoning Li and Tetsuo Kinoshita. Active support for using academic information resource in distributed environment. *International Journal of Computer Science and Network Security*, 7(6):69–73, 2007.

[14] Hua Liu and Manish Parashar. Accord: A programming framework for autonomic applications. *IEEE Transaction on Systems, Man and Cybernetics, Part C: Applications and Reviews*, 36(3):341–352, 2006.

[15] Hyacinth S. Nwana. Software agents: An overview. *Knowledge Enginnering Review*, 11(3):205–244, October/November 1996.

[16] Janak Parekh, Gail Kaiser, Phillip Gross, and Giuseppe Valetto. Retrofitting autonomic capabilities onto legacy systems. *Journal of Cluster Computing*, 9(2):141–159, 2006.

[17] Nancy Samaan and Ahmed Karmouch. Towards autonomic network management: an analysis of current and future research directions. *IEEE Communications Surveys & Tutorials*, 11(3):22–36, 2009.

[18] Gerald Tesauro, David M. Chess, William E. Walsh, Rajarshi Das, Alla Segal, Ian Whalley, Jeffrey O. Kephart, and Steve R. White. A multi-agent systems approach to autonomic computing. In *AAMAS'04*, pages 464–471, 2004.

[19] Takahiro Uchiya, Takehide Maemura, Li Xiaolu, and Tetsuo Kinoshita. Design and implementation of interactive design environment of agent system. In *IEA/AIE'07, LNAI 4570, AAAI/ACM*, pages 1088–1097, 2007.

[20] Alfred Ka Yiu Wong and Pradeep Ray. Ontology mapping for the interoperability problem in network management. *IEEE Journal on selected Areas in Communications*, 23(10):2058–2068, 2005.