Framework for Distributed Agents Reliability (FDAR)

Wesam Al Mobaideen **
Emad Qaddoura*
Eyas El-Qawasmeh***
Ahmad Omari*

* Authors are with the Faculty of Computer Science and Information Technology, Applied Science University, Amman, 11931 Jordan, {emadaq.a.omari}@asu.edu.jo, Ph: +962-6-560-9999
** Author is with King Abdullah II School for Information Technology, The University of Jordan, Amman 11942 Jordan, wesmoba@ju.edu.jo, Ph: +962-6-535-5000
***Author is with the Computer Science Department, Jordan University of Science and Technology, P.O.Box 3030, Irbid 21110 Jordan, eyas@just.edu.jo, Ph: +962-2-7201000

ABSTRACT

The wide evolving internet opened a tremendous possibility for the development of reliable distributed agents. This paper proposes a Framework for Distributed Agents Reliability (FDAR). The FDAR provides a reliable architecture that allows agents to operate as an autonomous system simplifying agent-to-agent, agent-to-client, and client-to-client interfaces. FDAR hides the reliability details that take place among agents from clients. By using the proposed approach, it is possible to get better performance comparable to other approaches.

Keywords. Reliability, Distributed Agents, Network.

1. Introduction

The network architecture is getting more complex due to the Internet evolution. Internet networks are affecting other types of networks, as networks integrate to provide expanded services. For example, the Global Mobile System (GSM) and Universal Mobile Telecommunication System (UMTS) has packet radio system such as GPRS Packet Radio System, (GPRS), and Enhanced Data rate for GSM Evolution (EDGE). Packet radio technologies add IP services to wireless systems and provided added value to subscribers [1]. The network is required to address reliability issues to provide a higher Quality of Services (QoS). The challenge is increased when adding Internet based components due to the heterogeneous technology mix. System components/agents provide services such as registration, authentication, authorization, routing, and roaming. Therefore, some of agents are state-full to keep status information about the clients such as system registration status. Other agents provide services that are stateless such as routing. Our focus in this paper is on state-full like agents [2].

The suggested FDAR’s operations provide for agents containment as an Autonomous System (AS) and is designed in a generic fashion where it can be applied to many distributed systems. The AS selects via configuration a Master Agent, MA, which manages the workload among the different agents within the AS. The work distribution uses a load balancing mechanism to distribute the workload among all agents in the AS. The load balancing may take on different characteristics such as CPU utilization, and the number of clients that an agent serves. The exchange of the management information that the MA uses to decide the work distribution among agents.
takes place via heartbeat messages between the agents and the MA.

The proposed FDAR provides a reliable architecture that allows agents to operate as an AS simplifying agent-to-agent, agent-to-client, and client-to-agent interfaces. FDAR hides the reliability details that take place among agents from clients. A client requests service from the AS. If a MA assigns an agent to serve a client goes out of service, the AS automatically switches the client’s service to another agent without any service interruption. The AS switches the service to another agent via a simple protocol interface.

The organization of this paper will be as follow. Section 2 is an introduction. Section 3 is the suggested system architecture and its framework. Section 5 handles failure conditions. Section 6 contains the conclusions.

2. Overview
Some network architectures require an agent or a server for clients to function properly. Clients demand services from the network via such agents. An example is Foreign Agents, (FA), and Home Agents, (HA), in an Internet based networks which support Mobile-IP (MIP) [3]. Another example is the Global Mobile System, (GSM), which has a Visiting Location Registrar, (VLR), and a Home Location Registrar, (HLR) [1]. These GSM and MIP agents serve clients, which are mobiles in the home and visiting networks and benefit from our distributed and reliable framework. The Dynamic Host Configuration Protocol (DHCP) [4], which allocates Internet Protocol, (IP), addresses to hosts can also benefit from our FDAR.

Figure-1 presents agents as $A_1$...$A_n$, and the clients as $C_1$...$C_n$, and the centralized Database, (DB). The agents act as an autonomous group providing a specified service for the clients. The agents provide a distributed reliable system in order to increase the service availability for the clients. Our framework provides a reliable distribution through a predefined interface for client-to-agent, agent-to-client, and agent-to-agent. A shared and centralized database provides a reliable storage that has the status of clients although a hierarchical distributed approach of the database can serve as well [5]. In the hierarchical approach, the MA should have access on the top level of the hierarchy where the most general view of the distributed agents' information is located. Other ordinary agents should have access to lower levels of the hierarchy according to their readiness and order of being MA as will be explain later on in this paper.

![Agents-Clients system view](image-url)
assigned agent acts to serve the client upon a request from the MA.

3. Related Work
Many framework and ongoing projects have been suggested. Among these is the project of DIAMOnDS - Distributed Agents for MObile & Dynamic Services. DIAMOnDS represents a mobile agent framework, which is based on service-oriented architecture that capable of hosting dynamic services. Registered and published services are actually software agents. Agent Stations scattered at remote location over a WAN provides reliable and secure mobility for the agents. On behalf of the user, these software agents try to offer dynamic services such as network monitoring, data manipulations and optimal route calculation [7].

A study of an effective schedule for the dissemination of mobile filter agents for distributed information filtering, in [8], shows that a significant reduction in communication cost needed for the examination of remote servers can be reached. Two scheduling algorithms have been compared, hierarchical clustering of mobile agent platform and a basic algorithm. The study does not assume that all the intended servers support mobile agent hosting and so these servers can be accessed from the nearest point the mobile agent can reach.

The authors in [9] have conducted another work related to distributed network management with dynamic rule-based managing agents. In their work, an expanded Managing Agents architecture has been proposed for Information Control (MAgIC). MagICs uses MIB to organize network management functionalities with a standardized interface to ease the workload of management applications as well as to reduce the traffic caused by management activities. In order to be able to deal with complex management functions a rule-based approach has been chosen in which these rules are sent to MagIC by the SNMP.

DECAF (Distributed Environment Centered Agent Framework). DECAF is a software toolkit for the rapid design, development, and execution of "intelligent" agents to achieve solutions in complex software systems. From a research community perspective, DECAF provides a modular platform for evaluating and disseminating results in agent architectures, including communication, planning, scheduling, and execution [10].

4. System Architecture and Framework
The following actions are used in the suggested framework:

4.1 Autonomous System Operation in Normal State
Clients start their registration process with the AS. Our framework does not address the authentication of the clients as it is outside the scope of our framework. However, our suggested framework is extensible to include the desired Authentication information. In addition, the database architecture is transparent to our framework.

4.2 Master Agent Identification
The MA is identified as one of the serving agents during system configuration. The MA must have backup MAs. The set of agents are configured in an orderly fashion to serve as a backup MA. For example if the AS has 4 agents, then one of the agents is configured to be an MA, another one will be selected as the first backup MA, another one will be selected as the second backup MA, and so on.

4.3 Clients AS Discovery
Clients discover the AS in two different mechanisms:
1. AS Identity Broadcast, (ASIB), message
The MA periodically sends out the ASIB message based on a configured value.

2. Clients AS Solicitation, (CASS), message:
   a. If a client does not detect an ASIB message, and it requires service by the AS, it can send a CASS message to discover an agent.
   b. The CASS message has a field, which indicates the type of service that is required by the client. Therefore, the MA must determine if it is the desired AS in the service provider’s network.
   c. The MA upon receiving a CASS messages sends out an ASIB message.

4.4 MA’s Assignment of Agents
The MA assigns the agents based on a pre-configured mechanism. The ones suggested in FDAR are based on equal and fair work distribution such as a load balancing scheme, which depends on the reported agent’s CPU utilization or a simple round robin scheme. When using the round robin scheme, the MA distributes clients to agents as they first contact the AS for service. The CPU utilization mechanism depends on information received by the MA via a Periodic Heartbeat, PH, message that is exchanged between all agents including the MA. Failure of receiving an agent’s PH message by the MA indicates that the agent is not alive and the MA performs a recovery action as indicated later.

4.5 Client’s AS Registration Process
   a. As a result of receiving the AS Identity Broadcast, (ASIB) message from the MA, the client starts its registration process.
   b. The ASIB message has the IP address of the MA as the source address.
   c. The clients send its AS Registration Request, (ASRRQ), message to the MA’s IP address.
   d. Upon receiving the client’s ASRRQ message, the MA redirects the request to one of the agents to continue the registration process with the client.
   e. The clients receive an AS Registration Reply, (ASRRP), from the assigned agent. As a result, the next interaction from the client will be to that agent’s IP address.
   f. The agent updates the DB with the proper state information.

4.6 Client’s AS De-Registration Process
The clients send an AS De-Registration, (ASDR), message once they are done using the services of the AS. As a result, the agent removes the client from its cache and updates the DB to reflect the new state of the client. The DB must be organized in a way that organizes registered clients per agent separately from clients that have completed their service and de-registered. If a service is requested by the client while it is not registered, an AS Registration Error, (ASRERR), message will be sent back to the client.

4.7 Client’s AS Re-Registration Process & Registration Timeout
Clients must register with the AS system using the AS Re-Registration, (ASRER), message. The frequency of the ASRER is determined in the ASRRP. If a client does not re-register within the specified period, its serving agent considers this as a registration timeout state and behaves in the same way as if the client has de-registered with the AS. However, the timeout has additional configurable grace period prior to avoid invoking the de-registration process unnecessarily. If the grace period passes without receiving the ASRER message, the client is assumed to be not alive, is de-registered, and the DB is updated with the new state for the client.
5. Handling of Failure Conditions
The subsections below discuss the different failure conditions that take place in the AS system.

5.1 Handling of an Agent Failure
Agents failure can be due to hardware or software. In addition, agents may be manually taken Out of Service, to perform maintenance. The MA takes the same actions recovery in both cases. However, if maintenance is to be performed, the MA starts the failure recovery process immediately without waiting for the Periodic Heartbeat message timeout. The steps taken by the MA are:

1. If the MA does not receive the periodic heartbeat message from an agent, or the agent is manually selected to OOS, the MA declares that agent to OOS.
2. The MA is responsible for redistributing the load of the OOS agent to other agents. This redistribution is achieved according to a selected method during system configuration. The system configuration specifies several options for the load redistribution that is done by the MA. Factors for redistribution may include CPU utilization, number of clients, and type of service.
3. Since agents keep the current state of its clients, and update the centralized DB with the same updated information, it will be an easy process for the MA to perform the transfer of responsibility of the clients, which are being served by the OOS to another agent. In addition, the DB is organized in a way that associates each agent with its clients. This enables quick retrieval of the clients for each agent.
4. The DB must be re-organized in accordance to the new assignment of clients to agents by the MA.
5. The MA sends a Client Redistribution Request, CRDR, to the in-service agents, and waits for confirmation via the Client Redistribution Accept, CRDA. This message includes the OOS agent’s ID and the range of database indices for which the clients will be transferred.
6. As a result, the agent will request the details of the clients’ records from the DB to be cached at the agent’s side. However, during this time, interactions between the client and the agent require the agent to interact with the DB for up to date status of the client. In addition, clients are honored requests even if the updated states are not obtained from the DB yet. Additional synchronization with the DB might be required if new state information is obtained.
7. While the MA is waiting for a CRDA message, it proxies for the OOS agent. Additionally, synchronization with the DB might be required if new state information is obtained and clients are not denied any service during this transitional period.
8. If a client obtains un-authorized service during the transitional period, the service will be interrupted when its new agent is assigned and fully synchronized.
9. Once an agent is assigned clients of an OOS agent, the agent will proxy on behalf of the OOS client in order to process requests/interactions from the clients. This takes place until agents inform clients of the new agent’s IP address by sending Agent Client Update, ACU, message. The client must send back a Client Agent Update, CAU, message.
10. Once the client has the new Agent’s IP address, then the interaction between the client and the agent proceeds normally as if the client was assigned the agent from the start of the interaction.

5.2 Handling of a Master Agent Failure
The MA might fail or taken manually OOS for maintenance reasons. The actions performed by the next MA in the backup list are the same in both cases. However, if the OOS reason is due to maintenance reason, the next MA in the backlist starts the failure recovery process immediately with waiting for the Periodic Heartbeat message timeout.

The process performed by the next back MA agent is:
1. The next backup MA will act as the primary backup MA, by responding to the initial client’s requests; perform failure recovery of agent’s failure.
2. The new MA re-distributes its load to other agents by applying the same process that is performed when a non-MA agent fails.

The order in which backup MAs are labeled can be changed anytime during system operation. Therefore, if an MA is to be manually switched to another agent, the agent which is to be an MA will be given the order, which indicates that it, will be the first backup MA. An MA, in addition to its management operation might be assigned clients as any other agent in accordance to specific system requirements.

5.3 Handling of a Database Failure
The database failure conditions are independent of the AS. Database systems provide their own reliability scheme; therefore, it is outside the scope of FDAR.

6. Conclusions and Future Work
We have presented a generic reliable system architectural framework, which might be used in many different distributed systems configuration called FDAR. The framework hides the details of recovery from the clients with one exception, when the new agent serving the clients sends Agent Client Update message. The framework allows clients to continue obtaining service during transactional periods while serving agents are being switched. This hides any impact of the offered service to the clients. The FDAR can be applied to new services in a fashion that is independent of the service details. In one network, several FDAR segments can be created based on the service offered without the impact or dependency among these independent and heterogeneous agents. An agent can be part of two FDAR segments without the impact to the service offering. This is allowed since the periodic heartbeat specifies the available agent’s bandwidth for the service. Therefore, an agent can allocate a predetermined percentage of its computing resources to a service of one segment while other percentages are allocated for other FDAR segments. This allows better sharing of resources while offering reliable system architecture. The separation of security and database architectures is strength of the FDAR approach as both of these components can be updated independent of the FDAR components.

An extended system might offer a standby agent for each agent in operation. The standby agent can have a mirror image of the state information of its primary agent. This allows an immediate switch to the new agent without full dependency on the DB for state-full information of clients.

References


