A STUDY ON ASR/TTS SERVER ARCHITECTURE FOR NETWORK ROBOT SYSTEM

In-Ho Choi
SAMSUNG Electronics Co., Ltd.
416 Maetan-3Dong, Yeongtong-Gu, Suwon-City, Gyeonggi-Do, Korea 443-742

Tae-Hoon Kim
SAMSUNG Electronics Co., Ltd.
416 Maetan-3Dong, Yeongtong-Gu, Suwon-City, Gyeonggi-Do, Korea 443-742

Keywords: URC, Network Robot System, ASR/TTS Server.

Abstract: "The URC (Ubiquitous Robotic Companion, Server computer-based networked robotic)" systems exploiting Internet-related technologies and Server computer require effective techniques for timely delivery of requested data to remote clients. In these systems, there is a need to process real-time data in server computer from/to robots and clients during system operation. In this paper, we describe and evaluate ASR, TTS server systems in the context of a real-time environment for the URC applications. Experimental results show that the server-based ASR, TTS support timely delivery of data to a potentially large number of robots during system operation.

1 INTRODUCTION

Distributed computing systems and Internet-related technologies have opened new application perspectives to robot tele-operation systems (Amoretti, 2003). Examples of novel applications, often broadly termed as "networked" or "on-line" robot systems, are tele-teaching/tele-leaming, virtual laboratories, remote and on-line equipment maintenance, and projects requiring collaboration among remote users, experts, and devices (Amoretti, 2003).

The main goal of URC Infra System project is to develop a high performance server system to process real-time requests and events from "networked" home robots connected to the server system via high speed network. In particular, the server system needs to provide highly-responsive ASR (Automatic Speech Recognition) and TTS (Text To Speech) functionality to the connected robots because the voice is the most appropriate communication method for human who wants to communicate with the robots.

This paper describes the architecture of ASR/TTS server system in the URC infrastructure and evaluates experimental results of the implementation based on the suggested architecture.

This paper is organized as follows: Section 1 presents an overview of the URC Infra system including its architecture and services; Section 2 outlines the architecture of ASR, TTS servers and URC Main servers; Section 3 evaluates a performance of the URC system in terms of responsiveness of the ASR/TTS server; and Section 4 presents concluding remarks.

1.1 Introduction to URC System

The URC is network robot system providing ubiquitous services to networked robots based on client-server architecture. The URC system consists of client (robot hardware) and server (application services) components distributed over networks.

The key idea is distributing robot's intelligence over remote server computers, which enables low cost hardware robots to access various application services running on the servers. It also allows existing hardware robots to extend their functionality by connecting to remote servers and making use of many application services, such as ASR and TTS, with minimal cost. Conceptually a URC robot is any terminal device, such as PDAs and cell phones, which can be accessible to the URC server system through networks.

1.2 A Structure and Services of URC Infra System

1.2.1 URC Infra System Structure

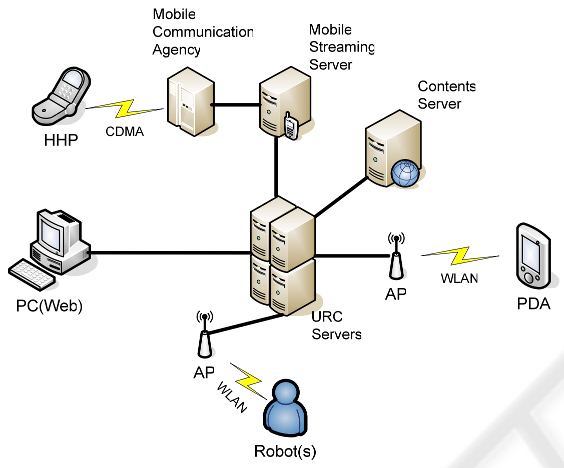

Figure 1: The Structure of URC System.

Figure 1 shows the structure of the URC system. The URC servers are clustered for high availability and fast responsiveness based on software-based clustering technique. A logical URC server cluster consists of multiple physical servers tied together with layer 4 load balancing method. The clustered architecture provides linear scalability of server performance in proportion to the number of servers in the cluster. The URC robot connects to the URC server cluster through wireless LAN and communicates with each other by following URC Protocols which includes remote control of the robot, voice recognition and TTS. The remote user is able to control and monitor the URC robot remotely by using remote PCs, PDAs, and cell phones.

1.2.2 URC Infra System Services

There are 3 types of URC services – basic services, common services, and robot-specific services in the URC system. The basic service, such as speech recognition and speech synthesis is provided by the URC servers to any type of URC robot with the URC protocol module. It serves as the basic building block for the various common services, for example, interactive speech recognition games, unmanned surveillance, remote monitoring/control of the URC robots, and so on. The robot-specific service is application services for particular types of robots such as cleaning services and robot dancing services.

2 STRUCTURE OF ASR, TTS AND URC MAIN SERVERS

This section outlines the URC server software structure with ASR, TTS engine to operate networked intelligent robot.
Because a standalone robot generally provides its all functions in 1 machine, there is a hardware limit to implement high performance functions. It is also difficult to expect high quality service as compared against cost. Application software such as speech recognition and synthesis which requires a lot of hardware resources is especially main factor to increase the cost of robot. As mentioned in section 1, the URC can realize to simplify robot functions about application software which provides a service that requires high performance as well as ASR, TTS

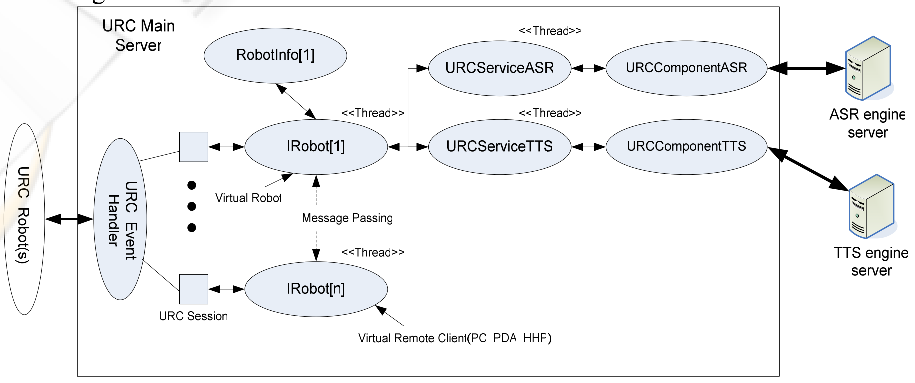

Figure 2: URC Software Architecture with ASR, TTS Servers.

as it uses server-based computing through network with high speed and bandwidth. It can likewise realize the low cost of robot by reduced computing power and to increase availability through providing unlimited services. Therefore, as ASR and TTS engine using many resources of robot execute in the URC system to serve user speech recognition and synthesis, it is possible to support unlimited lists of recognition words and speech synthesis for many languages.

Figure 2 shows the block diagram of URC Main server which is in charge of an interface between URC Robot and ASR/TTS engine servers. ASR/TTS engine can be constructed a separate server or not.

3 PERFORMANCE EVALUATION OF URC SYSTEM WITH ASR, TTS SERVER

This section describes a test environment, scenario and results for evaluating performance about server-based ASR/TTS system. The URC server system for ubiquitous robot satisfies requirements for real-time as follows:

(1) [Requirement 1] Average response time less than 1 second for request message of clients and/or robots.

(2) [Requirement 2] Providing sessions (clients + robots) more than minimum 100 per 1 server.

Above-mentioned conditions are minimum requirements to actually apply URC system to fields (home).

3.1 Experimental Environment and Scenario

An experimental scenario divides 2 cases according to the location of ASR and TTS engine servers.

(1) [Scenario 1] The ASR/TTS engine servers are located in local machine with the URC Main software.

(2) [Scenario 2] The ASR/TTS engine servers are separated to external server from machine with the URC Main software through network.

Figure 3 and 4 show the structure of the URC system model for [Scenario 1 and 2].

In the figure 3 and 4, URC Man server, ASR/TTS servers and virtual robots have IP addresses in the same subnet. We regarded 1 transaction time until receiving a corresponding response message after the virtual robot sends a request message to the URC Main server as RTT (Round-Trip Time). The [Requirement 1] means that it satisfies average RTT ≤ 1.

A hardware specification of URC Main server, ASR and TTS server, operating system and software specification for each [Scenario] is listed in Table 1.

Table 1: Specification of Servers (URC Main, ASR, TTS).

	URC Main Server	**ASR Server**	**TTS Server**
Hardware specification	**CPU** : Intel® Xeon™ Processor 3.2 GHz/1M, EM64T, 800MHz FSB * 2EA **Memory** : 4GB, DDR-2 400MHz ECC **HDD** : 146GB Ultra320 SCSI **LAN** : 100Mbps		
Operating System	Redhat Enterprise Linux AS(kernel 2.4.21-4.ELsmp)		
etc	✓ **ASR engine : HCILab[5] ASR software (Korean version, Independent Speaker, 10,000 words support)** ✓ **TTS engine : HCILab[5] TTS software (Korean version)**		

We constructed a test scenario to verify whether URC system model for [Scenario 1 and 2] satisfies [Requirement 1 and 2] or not. Figure 5 shows a sequential diagram for test scenarios. However, all case of [Scenario 1] except “URC Main server”, “ASR server” and “TTS server” are operated in 1 machine is identical.

1. Robots over 100 are connected to the URC Main server. The robot to be established the connection is completed an authentication step from the URC Main server.
2. Each robot receives a speech input, “What is the URC?” from user and transmits it to the URC Main server after the robot converts to WAV file.
3. The URC Main server receives the WAV file and transfers to an ASR server to request speech recognition.
4. The ASR server sends a text (string), symbol and score to the URC Main server as result after he processes speech recognition.
5. The URC Main server decides a response string to have to transmit with the recognized result.
6. The URC Main server transfers the response string to TTS server to convert speech output, which is WAV file, “I’m going to tell you about the URC.”
7. The URC Main server is obtained a response WAV file by TTS server and then transfers the robot.

Since we couldn't implement 100 physical robots for experiments, we constructed virtual robots corresponding to physical robot and execute them in 1 PC machine. We used "QALoad" Software which is an application load testing program of Compuware Corporation to achieve virtual robots. The test script of QALoad transmits a request WAV file, "What is the URC?" to the URC Main server after it generates more that 100 robots. It subsequently waits for a corresponding response WAV file during 60 seconds. If it can't be received the response within 60 seconds, it regards error. The test script of QALoad records whether an average RTT, that is a time for 1 transaction until the receipt of the response WAV file, completes less than 1 second or not. We executed the transaction of 20 times for average RTT to increase accuracy of test. Then, we estimated an average response time, maximum delay response time and standard deviation.
A timing diagram about 1 transaction of a robot in test environment is shown in Figure 6. After virtual robot has established the session to URC server and completed authentication step, transaction begins (B.T). A single transaction includes a request message (S.R), "What is the URC?" and a response message (R.R), "I'm going to tell you about the URC." After the transaction has started (B.T), there is a "Sleep time" which is a random time in the range of 0~20 seconds and then the request message (S.R) is sent to URC server. This is to apply random distribution to message generation model of the virtual robot. After the transaction has terminated (E.T), there is a "Pacing time" before a next transaction is started. The "Pacing time" is minimum interval time to prevent excessive ASR request messages to halt the ASR server system by overload. The next transaction will be followed after "Pacing time" is completed.
The test environment and related parameters as stated above are listed in Table 2.

3.2 The Results of Experimental Performance for Scenario 1

Figure 7 shows the results of average response time, maximum delay response time and standard deviation about [Scenario 1]. We estimated variation of response time while the numbers of virtual robot increases from 50 to 130. As shown in the result, [Scenario 1] only satisfies [Requirement 1], which is average response time within 1 second in case of about 70 virtual robots. However, maximum delay response in case of 50 virtual robots as well as the response time with standard deviation in case of 70 virtual robots exceeds 1 second.

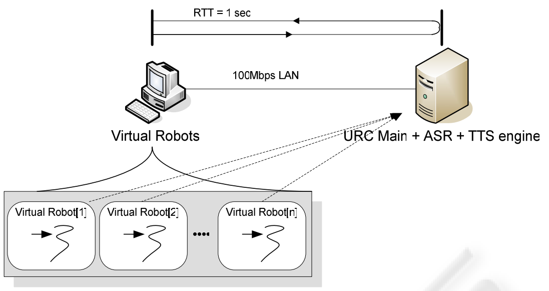

Figure 3: URC system model for [Scenario 1].

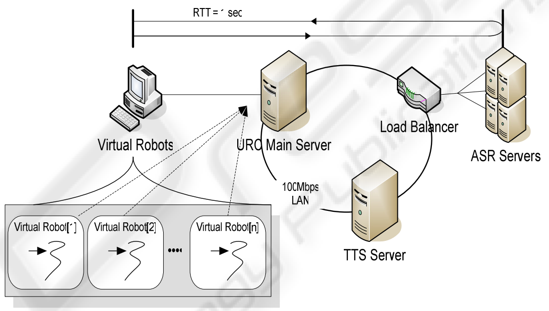

Figure 4: URC system model for [Scenario 2].

Table 2: Summary of Test Parameter.

Parameter	Value	Remark
Requirement #1	**RTT < 1sec**	**For 1 transaction**
Requirement #2	**[# of robots ≥ 100] / 1 server**	**-**
number of robots	From 100 to 200	Increase 10 robots
Total transaction	20 times	-
LAN	100Mbps	
Request message start time interval	random	Sleep Between 0 and 20 sec after begin-transaction start
Pacing time interval	20 sec	Sleep between transaction
Request message length	20404bytes	URC Protocol header with 20 bytes included
Response message length	46304bytes	URC Protocol header with 18 bytes included

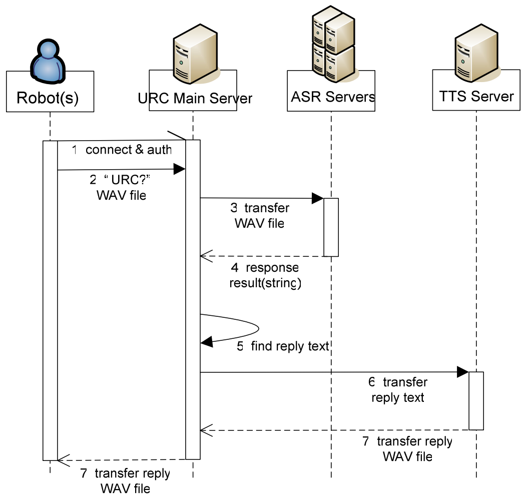

Figure 5: The Sequential Diagram of Test Scenario.

3.3 The Results of Experimental Performance for Scenario 2

The URC Main, ASR and TTS software at the result in section 3.2 make use of maximum 2%, 25% and 4% CPU resource respectively. An experimental result in Figure 7 shows that there is a bottleneck in the process of request for speech recognition in single ASR server. Therefore, we executed the URC Main and TTS on each machine such as Figure 4 and modified for ASR to be consisted of multiple server machines with clustering service through a load balancing algorithm. The load balancing algorithm used in this research is Least Connection Scheduling.

Figure 8~11 show the results of the average response time, maximum delay response time and standard deviation of [Scenario 2] according to changing clustered ASR server to 2~5 machines.

The results in Figure 8~11 describe that minimum 3 ASR servers are needed to stably meet [Requirement 1 and 2].

4 CONCLUSION

The ASR and TTS services play important roles in communication between human and intelligent service robots. Since these services require a lot of system resources, running ASR/TTS services in a networked remote server is able to provide high-

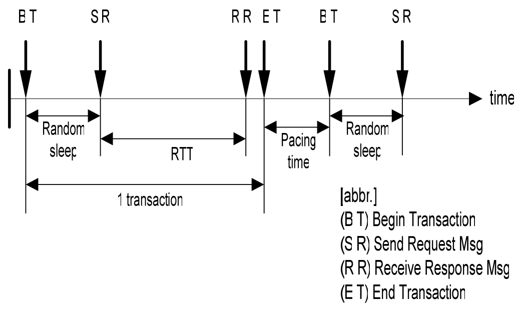

Figure 6: Timing Diagram for 1 Virtual Robot.

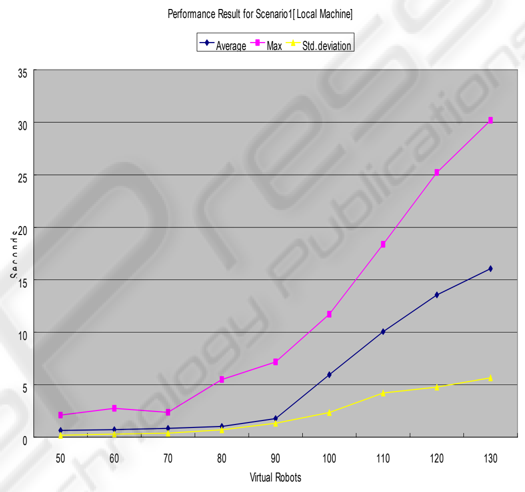

Figure 7: Experimental Performance Result for Scenario 1.

quality HRI (Human Robot Interaction) services to users with minimal cost.

In this paper, we present a practical way of distributing CPU-intensive tasks like ASR/TTS services over high performance network servers connected through high speed network. We identified that the ASR service is performance bottleneck and present appropriate server architecture for ASR/TTS services based on server clustering technology. The architecture presented in this paper includes load balancing algorithm that distributes incoming requests from remote robots over multiple servers efficiently, and we could validate the suggested architecture is able to ensure reasonable response time for 100 network robots by experimental performance tests.

Although the result presented in this paper is obtained in 100 Mbps LAN environments, it can be meaningful basis for performance model for WLAN environments with lower network bandwidth. The relevant ongoing research includes the implementation and analysis of experimental performance models in WLAN environments.

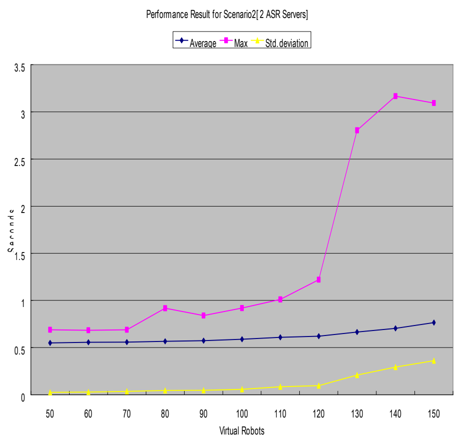

Figure 8: Experimental Performance Result for Scenario 2[2 ASR Servers].

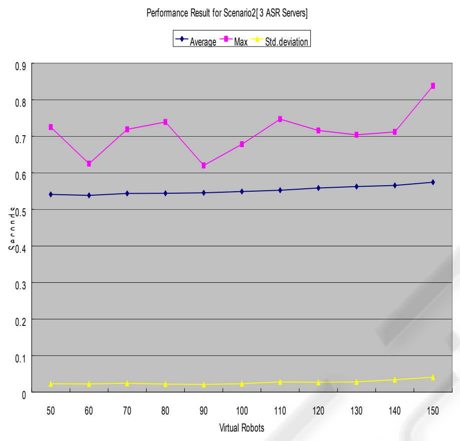

Figure 9: Experimental Performance Result for Scenario 2[3 ASR Servers].

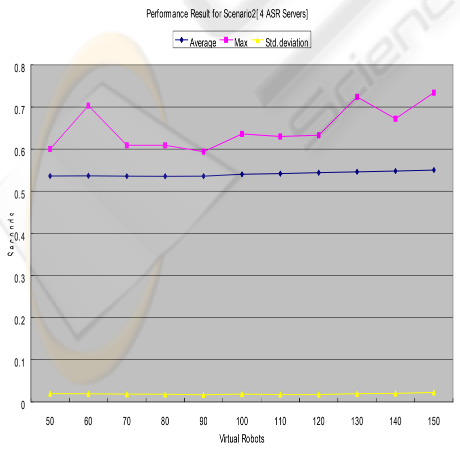

Figure 10: Experimental Performance Result for Scenario 2[4 ASR Servers].

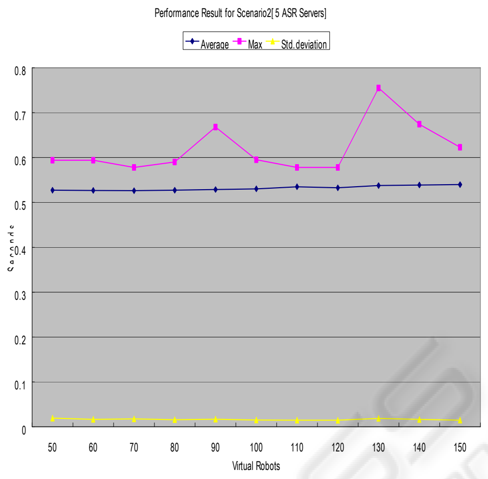

Figure 11: Experimental Performance Result for Scenario 2[5 ASR Servers].

REFERENCES

Michele Amoretti, Stefano Bottazzi, Monica Reggiani, Stefano Caselli, “Evaluation of Data Distribution Techniques in a CORBA-based Telerobotic System”, *Proceedings of the 2003 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, October 2003.pp.1100-1105*

Douglas C. Schmidt, Stephen D. Huston, 2002. *The book,* Addison-Wesley. C++ Network Programming: Mastering Complexity Using ACE and Patterns.

Douglas C. Schmidt, Stephen D. Huston, 2003. *The book,* Addison-Wesley. C++ Network Programming: Systematic Reuse with ACE and Frameworks,

Stephen D. Huston, James CE Johnson, Umar Syyid, 2004. *The book,* Addison-Wesley. The ACE Programmer's Guide: Practical Design Patterns for Network and Systems Programming.

The library of ASR/TTS engine. *HCI Lab company.* http://www.hcilab.co.kr/