Numerical Formula and Verification of Web Robot for Collection Speedup of Web Documents

김 원* 김 영 기** 진 용 옥***
Weon Kim Young-Ki Kim Yong-Ok Chin

Abstract

A web robot is a software that has abilities of tracking and collecting web documents on the Internet[1]. The performance scalability of recent web robots reached the limit as the number of web documents on the Internet has increased sharply as the rapid growth of the Internet continues. Accordingly, it is strongly demanded to study on the performance scalability in searching and collecting documents on the web.

"Design of web robot based on Multi-Agent to speed up documents collection" rather than "Sequentially executing Web Robot based on the existing Fork-Join method" and the results of analysis on its performance scalability is presented in the thesis.

For collection speedup, a Multi-Agent based web robot performs the independent process for inactive URL ("Dead-links" URL), which is caused by overloaded web documents, temporary network or web-server disturbance, after dividing them into each agent. The agents consist of four component; Loader, Extractor, Active URL Scanner and Inactive URL Scanner.

The thesis models a Multi-Agent based web robot based on "Amdahl's Law" to speed up documents collection, introduces a numerical formula for collection speedup, and verifies its performance improvement by comparing data from the formula with data from experiments based on the formula. Moreover, "Dynamic URL Partition algorithm" is introduced and realized to minimize the workload of the web server by maximizing a interval of the web server which can be a collection target.

☞ Keyword : web robot, Multi-Agent, Amdahl's Law, Nemerical Formula, Dynamic URL Partition algorithm, Performance

1. Introduction

In order to collect web documents effectively, cluster-based parallel and distributed processing is applied as a way of enhancing the performance of a web robot itself[2,3]. However, the thesis sought to design system structure of a web robot using Multi-Agent technique designed to enhance collection speed on a given system resource environment, and to analyze the performance.

For this, modeling of Multi-Agent technique to which Amdahl's law was applied and dynamic URL partition algorithm that can minimize web server's load were presented, and verification of numerical formula for performance enhancement suggested through performance analysis test of a web robot implemented was made.

* 정 회 원 : 한국인터넷진흥원 기술지원단장
wkim@nida.or.kr(제1저자)
** 정 회 원 : SK Telecom
gohappy@nate.com(공동저자)
*** 정 회 원 : 경희대학교 전파공학과 교수
chin3p@chollian.net(공동저자)

2. Related Studies

2.1 Sorting scheduler taking network load into consideration

One way of judging network load is that you measure time required for it to be returned after sending a certain amount of data to the server by using Ping.

Sorting scheduling[4] performs sorting with network load of the server at the current time range and write up the results into server list, and according to the order of the server list, the robot approaches the server to collect web documents. The algorithm of sorting scheduling is the followings.

Algorithm Sorting_Schedule(List, Time)

input : List, Time

output : Go_List

```
/* the class to store Server_List */
  WebServer List;
  /* Store Server_List to do at this
current time */
  Get_Server_List(List);
  /* Store Network Load per each
server at that time */
  Get_Ping(List, Time);
  /* Sort according to the order with
Load Ratio */
  Sort(List, Time);
  /* Make the order of table for ro-
bot to do */
  Go_List = Write_Table(List);
```

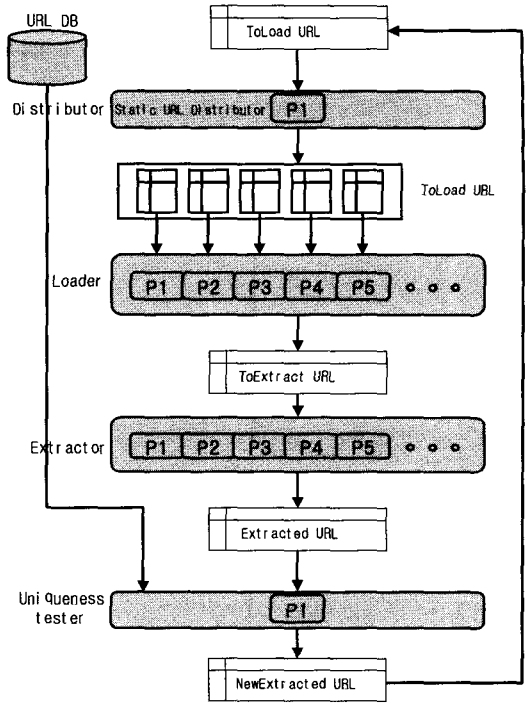
2.2 Web Robot using Fork-Join Technique

In order to simultaneously collect multiple URLs, the web robot, in general, is driven by multi-process to have the function of loading documents and the function of extracting new URLs from documents loaded[5,6].

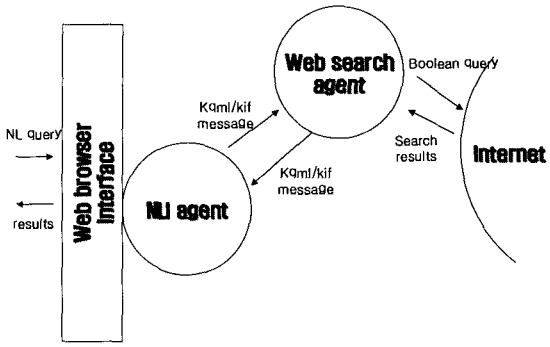
Fig. 1 shows the structure of web robot that performs a consecutive execution method using Fork-Join technique. Function module of the model is composed of webpage loader, URL extractor and uniqueness tester.

2.3 Multi-Agent web search model

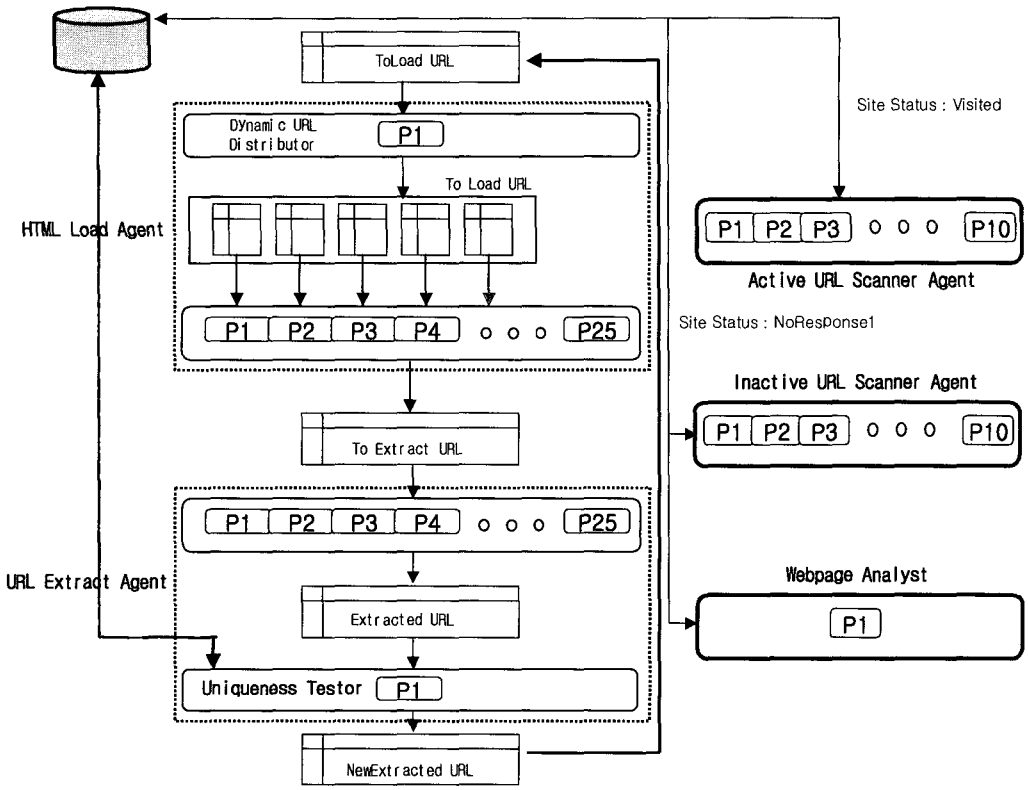
Fig. 2 shows the multi-agent web search model[7] where NLI(Natural Language Interface) agent and web search agent are communicating each other using kqml/kif messages. The NLI agent controls a web browser interface which receives natural language (NL) query from the users and displays the search results. The web search agent generates standard Boolean queries from the kif messages



〈Fig. 1〉 Web Robot using a Fork-Join Scheme



〈Fig. 2〉 The multi-agent web search model



〈Fig. 3〉 The Structure of Web Robot using a Multi-Agent Scheme

with the received kqml performatives and drives the existing web search engines.

3. Web Robot using Multi-Agent Technique

3.1 Multi-Agent applying Amdahl's law

A general web robot consists of the structure in which two steps of actions are repeated by series in terms of function. The first step is to connect designated web pages by socket communication and to receive and load the pages. The second step is to extract new URLs from web pages loaded and then execute the first step.

HTML to be collected at the first step can be, by nature, categorized into (1) normal web documents and (2) "dead-links" web documents. The existing web robots collect documents of classification (1) and (2) in order without dividing them. In this case, documents of classification (2) causes a delay in total collection time because there is a delay due to waiting while they are being collected. Suggested web robot using Multi-Agent technique classifies documents of (1) and (2) and assign them to different collection tasks to make a collection by parallel (Fig. 3). Web robot using Multi-Agent Scheme consists of four major functions as follows.

- HTML Load Agent : stores not only the new extracted URL using URL Extract Agent into database but also the result of success or fail-

ure into the files.

- URL Extract Agent : extracts the new URL to be gotten by parsing it's URL from web documents alreadycollected by HTML Load Agent.
- Active URL Scanner Agent : checks that the Active URL to be already extracted is whether active or not, and save the results to the files.
- Inactive URL Scanner Agent : checks that the Inactive URL to be already extracted is whether reactive or dead, and save the results to the files.

If we show the suggested method by applying this to the (1) of Amdahl's law[8], the part where loading of web documents is made can be expresses as the followings since that consists of HTML loading, active URL scanner and inactive URL scanner functions in parallel.

$$Speedup = \frac{1}{\frac{P}{N}+S} = \frac{1}{\frac{P_{load}}{N}+S_{extract}} \quad (1)$$

※ P : Parallel ratio of total codes, S : Serial ratio of total codes

N : The number of Process

P_{load}=P(HTML Load)+P(Active URL Scanner) +P(Inactive URL Scanner)

$S_{extract}$=URL Extract code

3.2 Numerical formulation of performance enhancement made by using Multi-Agent technique

If we make a formula out of the fact that loading time of web robot driven by multi-agent can make collection speed faster than when we use a Fork-Join based execution method, dividing the existing collection time by increased collection time makes the rate of speed increase.

Let's indicate access success and access failure probabilities at the time of web robot's collection by P_s and P_f respectively while indicating average time required for accessing a URL within the a given same period of time by T_s and T_f respectively.

As for Multi-Agent method, the followings are defined P_{s1} and P_{f1}, and T_{s1} and T_{f1}.

Symbol	Definition
P_{s1}	Probability of access success to a certain URL site with given period
P_{f1}	Probability of access failure to a certain URL site with given period
T_{s1}	Average measured response time to a certain URL site when the access succesed
T_{f1}	Timeout value when the access is decided to be failed

As for Fork-Join method, the followings are defined P_{s2} and P_{f2}, and T_{s2} and T_{f2}.

Symbol	Definition
P_{s2}	Probability of access success to a certain URL site with given period
P_{f2}	Probability of access failure to a certain URL site with given period
T_{s2}	Average measured response time to a certain URL site when the access succesed
T_{f2}	Timeout value when the access is decided to be failed

For the above P_{s1} and P_{f1}, and P_{s2} and P_{f2}, the following relationship is made.

$$P_{s1}+P_{f1}=1,\ P_{s2}+P_{f2}=1 \quad (2)$$

The formula for collection speed of Initial-State is inferred as follows.

$$S_{(Initial-State)} = \frac{T_{(Fork-Join)}}{T_{(Mlti-Agent)}} = \frac{P_{s2}T_{s2} + P_{f2}T_{f2}}{P_{s1}T_{s1} + P_{f1}T_{f1}} \approx 1 \quad (3)$$

If we examine Steady-State of web robot, the followings are inferred.

$$S_{(Steady-State)} = \frac{T_{(Fork-Join)}}{T_{(Mlti-Agent)}} \approx 1 + \frac{P_f}{1-P_{f1}} \cdot \alpha \approx 1 + \frac{P_f}{1-P_f} \cdot \alpha$$

$$where, \text{ for } variables,\ P_f = P_{f1} = P_{f2},\ \alpha = \frac{T_{f2}}{T_{s1}},\ \text{and } can\ be\ that\ P_f = 1 \quad (4)$$

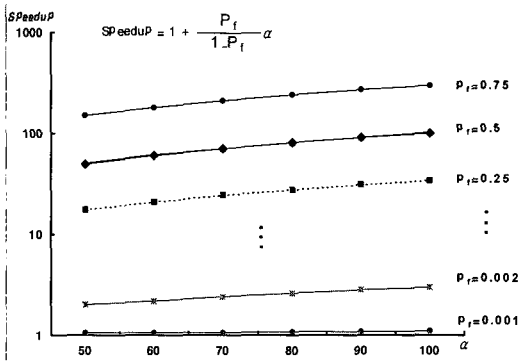
According to (4), performance increase in collection time for web robot using the suggested Multi-Agent technique compared to the existing Fork-Join based web robot can be summarized into the followings:

(1) As P_f, rate of access failure, increases, the value of performance increase becomes larger.

(2) As $\alpha = \frac{T_{f2}}{T_{s1}}$, rate of average time required in case of access failure based on Fork-Join method and average time required in case of access success based on Multi-Agent method increases, the value of collection speed becomes larger.

3.3 Dynamic URL Partition Algorithm

Suggested here is dynamic URL partition technique that enables minimizing web robot's impact on the server by maximizing making access interval for collection to distribution server.



〈Fig. 4〉 Collection Speed to the variable α in case of Steady-State

First of all, variables that indicate system environment can be defined as follows.

Variable	Definition (# : Number)
P	HTML Collection Process #
H	The number of Host
M	Throughput (the number of URL processed per second)
C_{max}	Maximum process that can be processed per second
U_i	The number of process for URL in one host
U_t	The total number of processes for URL in all hosts

In reality, there does not occur a case where U_i, number of URL by host, is the same as or less than C_{max}, the maximum number of processes allowed by host. Therefore, if we examine the case where $U_i > C_{max}$, there is a risk that, for U_i's host, the number of processes that access simultaneously becomes larger than C_{max}. In this case, we suggest the following dynamic URL partition algorithm that can maximize access interval for collection while maintaining load balancing.

```
Algorithm Partition(Ui, Ui, Cmax)
input : Ui /* Ui : The number of process for
        URL in one host */
        H /* The number of Host */
        Cmax /* Maximum process that can be
                processed per second */
output : PartitionNum /* The number of partition
         for URL in one host */
         PartitionSize[1....H] /* The size of parti-
         tion for host */
  for i = H to i = 1 /* find the reference host */
  begin
    if (Ui ≤ Cmax) continue
    else break
  end
  PartitionNum = Ui /Ui+1 ;
  for j = 1 to j = H /* find the par-
        tition size for each host */
  begin
    if (j < i)
      PartitionSize[j]= Uj /PartitionNum;
    else
      PartitionSize[j] = 1;
  end
}
```

4. Performance Analysis

4.1 Test Environment

Test environment used GNU C language on Linux platform, and 1 unit of DELL 4400 PC server. The server is loaded with 2 units of Pentium III 800MHz CPU and 2 Gbytes of RAM, and 120 Gbytes of Hard Disk Array. For OS, verson Linux RedHat 6.2 was used.

4.2. Performance Analysis

(1) URL collection speed

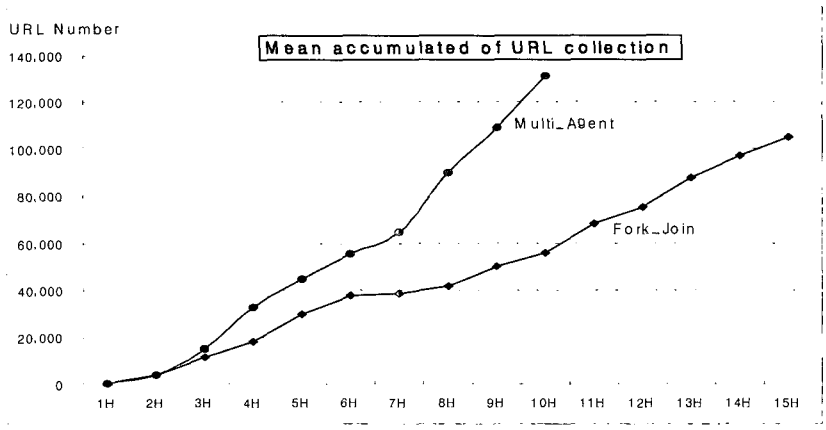
For performance evaluation, 10 domestic universities' web sites were randomly selected as initial URLs. In addition, in the early morning that guarantees lower load, under same network environment and with same conditions, a program was executed several times to measure an average amount of documents collected by time range. And also in comparing and analyzing performances of multi-process method that uses Fork-Join technique and of web robot based on Multi-Agent technique suggested in the thesis, we set as indicated in the Table 1 optimum numbers of HTML loading processes and URL extraction processes, and optimized timeout value when request for document to web server is made. Optimum settings of variables were made through an analysis test of the impact variable values have on performance.

〈Table 1〉 Optimized Value

Section(# : Number)	value	
	Multi-Agent	Fork-Join
HTML Load Process #	25ea	25ea
URL Extract Process #	25ea	25ea
Inactive URL Scanner Process #	10ea	None
Active URL Scanner Process #	10ea	None
Connection Timeout	15sec	15sec
Read Head Timeout	10sec	10sec
Read Body Timeout	20sec	20sec

〈Table 2〉 Analysis table to URL collection speed

Time	Multi-Agent			Fork-Join		
	1st	2nd	Average	1st	2nd	Average
1H	535	532	534	534	532	533
2H	4,139	4,121	4,130	4,131	4,120	4,126
3H	15,116	15,118	15,117	11,660	11,597	11,629
4H	37,730	27,479	32,605	20,546	15,499	18,023
5H	44,408	44,706	44,557	34,676	24,488	29,582
6H	65,731	44,710	55,221	40,740	34,757	37,824
7H	69,909	59,175	64,542	40,890	35,902	38,396
8H	78,439	101,549	89,994	41,732	41,380	41,060
9H	96,716	121,043	108,880	49,916	50,283	50,100
10H	118,112	144,446	131,279	55,731	56,087	55,909
11H	-	-	-	68,365	67,676	68,021
12H	-	-	-	79,614	71,437	75,526
13H	-	-	-	84,363	91,390	87,877
14H	-	-	-	93,840	100,635	97,238
15H	-	-	-	101,545	108,676	105,111



〈Fig. 5〉 Performance comparison of amount of URL collection

Fig. 5 shows how web robot using Multi-Agent technique suggested in the thesis increases the rate of performance enhancement as the amount of URLs collected increases: "active URLs" and "dead URLs" collected from loading agent are managed separately by "active URL test agent" and "dead URL test agent", which are respectively driven by 10 processes.

(2) Verification of numerical formula for collection speed enhancement

According to Fig. 4 and Table 2 that describes URL collection speed analysis table, web robot using Multi-Agent technique created 108,880 new URLs by going through processing of loading, extraction and uniqueness test for 9 hours (32,400 seconds).

Time required for extracting 1 new URL is calculated by dividing the total time by the number of URLs collected.

$$T_1 = \frac{T_{total}}{U_T} = \frac{Total\ time}{Nmber\ of\ URLs\ extracted} = \frac{32{,}400\sec}{108{,}880\,ea} \approx 290ms$$

The analysis result of web robot's execution time shows that network time accounts for 84% of total execution time, and that time required for connection within network time accounts for 30%. Therefore, average time required for succeeding in accessing a URL is the followings.

$$T_{sl} = T_l \cdot \text{Ratio of network time} = 260ms \cdot 0.84\,(84\%) \cdot 0.3(30\%) \approx 73.08ms$$

On the other hand, for Fork-Join based web robot, "dead-links" URL analysis table by unit hour shows that 304 "dead-links" URL occurred for 15 hours.

The following is the probability (P_f) that URL will appear as "dead-links" on average.

$$P_f = \frac{U_T(t)_{\in active}}{U_T(t)_{active}} = \frac{The\ total\ nmber\ of\ "\ dead\ links"\ \ URLs\ for\ a\ given\ time\,(15\ hours)}{The\ total\ nmber\ of\ URLs\ for\ a\ given\ time\,(15\ hours)}$$

$$= \frac{304}{105{,}111} = 0.00289 \approx 0.289\%$$

It can be also defined as $\alpha = \frac{T_{f2}}{T_{s1}} =$

$$\frac{\text{Average time required for succeeding in accessing a URL}}{\text{Average time required for failing in accessing a URL}}$$

Where, T_{s1} = 73.08ms, and T_{f2} is access delay time due to waiting (Timeout value)

If you increase timeout value, within the range of 1, 5, 10, 15, 30 and 60 seconds, increase rate of collection speed is shown as indicated in (Fig. 6) It was found that collection speed greatly increases when setting timeout value at numbers greater than 30 seconds.

Since the test was made by setting timeout value of Fork-Join method's loading process at optimum 15 seconds, it follows that

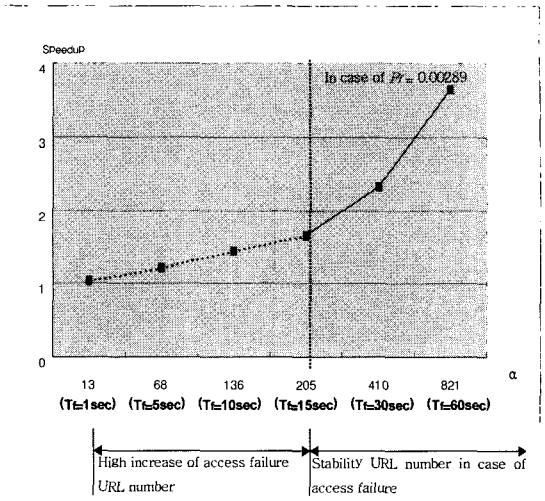
$$\alpha = \frac{T_{f2}}{T_{s1}} = \frac{15\sec}{73.08ms} \approx 205,$$

Since it is that P_f = 0.00289, by substituting the values of P_f and in the (formula 4), you can get the following result.

$$\begin{aligned} Speedup &= \frac{T_{(Fork-Join)}}{T_{(Mlti-Agent)}} \approx 1 + \frac{P_f}{1-P_{f1}} \cdot \alpha \\ &\approx 1 + \frac{0.00289}{1-0.00289} \cdot 205 \\ &\approx 1 + 0.59 \approx 1.59 \end{aligned}$$

In conclusion, according to the test, speedup (collection speed) showed an increase in collection speed by about 1.66 times, and by substituting optimum variables (timeout value, number of processes, etc) and "dead-links" URLs applied to the test in the (4), we were able to obtain an increase in collection speed by about 1.59 times.

Therefore, it can be said that the increase in collection time for web robot using Multi-Agent technique was verified both by test and by numerical formula.



〈Fig. 6〉 Performance graphics of collection speed

5. Concluding Remarks

It was proven by test that web robot based on Multi-Agent method shows higher performance by greater than 1.5 times than when using consecutive execution method. Furthermore, it was also found that the bigger the size of collection targets is, the higher performance increase you can get.

Against this backdrop, however, in order to take better advantage of the results of the study, follow-up studies are needed to be done. First, in selecting timeout values and number of processes, under a given system resources environment in which, during web robot's execution, response failure rate and web documents collection time can be observed and can be adjusted dynamically in order to fit the circumstances, a further research is required on techniques for selecting adaptive timeout value and number of processes[9,10]. Second, time for uniqueness test accounts for 60% of total time required for computing for web robot's execution thread. Therefore, when designing a intelligence-type[11,12] Multi-Agent web robot in the future, a research on

more enhanced algorithm for uniqueness test will need to be made. Finally, it is required to apply to the suggested web robot using Multi-Agent technique dynamic scheduling[13] that considers real-time load information and network load experienced when collecting documents against the web server.

References

[1] H. Kawano. Mondou, "Web search engine with textual data mining", In Proc. of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, 1997, pp. 402-405.

[2] J. R. Smith and S. F. Chang, "Visually searching the web for content", IEEE Multimedia , July 1997, pp. 12~20.

[3] Michael Baentsch, L. Baum, Georg Molter, S. Rothkugel, and P. Sturm. "World Wide Web caching: The application-level view of the internet", IEEE Communications Magazine, June 1997, pp. 170~178.

[4] C. Kim, "Design and Implementation of Region Administration- Oriented Search Engine Based on Robot Control Policy", Ph.D thesis, Kyungnam Univ., 1996. 6.

[5] M.F. Arlitt and C.L. Willianson, "Web Server Workload Characterizations: The Search for Invariants", Proceedings of SIGMETRICS96, 1996.

[6] Fielding, Roy. "maintaining Distributed Hypertext Infostructures: Welcome to MOM spider's Web". Proceedings of the First International World-Wide Web Conference, Geneva Switzerland, May 1994.

[7] G. Lee, J. Lee, H. Rho, Y. Park, J. Choi, and J. Seo, Interactive NLI Agent for Multi-Agent Web Search Model, International Workshop on Intelligent Agents on the Internet and Web, in 4th World Congress on Expert Systems, pp 67-74, Mexico City, 1998.

[8] Kai Hwang, F. Briggs, "Computer architecture and Parallel Processing", McGraw-Hill, pp. 533-535, 1993

[9] Sung Jin Kim and Sang Ho Lee, "Implementation of a Web Robot and Statistics on the Korean Web", Second International Conference on Human. Society@Internet, pp. 341-349, June. 2003.

[10] Nicholas Jennings, Katia Sycara and Michael Wooldridge, "Road map of Agent Research and Development : Autonomous Agents and Multi-Agent Systems", IEEE, vol. 1 pp. 7~38, 1998.

[11] Genesereth, M., and Ketchpel, P., Software Agents, Communications of the ACM, Vol.37, No.7. Jul. 1994.

[12] O. Etxioni and D. Weld, "Intelligent Agents on the Internet: Fact, Fiction, and Forecast", IEEE Expert, Vol. 10, No. 4, 1995, pp. 44~49.

[13] Riechen, Doug, "Intelligent Agents", Communications of the ACM Vol. 37 No. 7, July 1994.

◐ 저 자 소 개 ◑

김 원

1984년 한양대학교 전자공학과 졸업(학사)
1989년 한양대학교 대학원 전자공학과 졸업(석사)
2002년 경희대학교 대학원 전자공학과 졸업(공학박사)
1984～1986 국방과학연구소
1989～1992 (주)데이콤
1992～1999 한국전산원
1999～현재 한국인터넷진흥원 기술지원단장
관심분야 : 컴퓨터네트워킹, RFID, IPv6
E-mail : wkim@nida.or.kr

김 영 기

1987년 경희대학교 전자공학과 졸업(학사)
1989년 경희대학교 대학원 전자공학과 졸업(석사)
2000년 경희대학교 대학원 전자공학 박사수료
1992～현재 SK Telecom
관심분야 : 무선인터넷
E-mail : gohappy@nate.com

진 용 옥

1968년 연세대학교 전기공학졸업(학사)
1975년 연세대학교 대학원 전자공학과 졸업(석사)
1981년 연세대학교 대학원 전자공학과 졸업(공학박사)
1979년～현재 경희대학교 전파공학과 교수
관심분야 : 통신시스템, 한의정보공학, 국어정보공학
E-mail : chin3p@chollian.net