정확도 높은 검색 엔진을 위한 문서 수집 방법

하 은 용[†]·권 희 용[†]·황 호 영[††]

요 약

인터넷상의 정보 검색 엔진들은 웹 로봇을 이용해서 인터넷에 연결되어있는 수많은 웹 서버들을 주기적 또는 비주기적으로 방문하여 자체적인 인덱싱 방법에 따라 자료를 추출하고 분류해서 검색 엔진의 기초가 되는 데이터 베이스를 구축하고 변경하는 작업을 계속하고 있다. 이런 일련의 작업은 인터넷 상에 분산되어 있는 막대한 정보를 쉽고 정확하게 찾을 수 있는 게이트 사이트로서의 역할을 담당하기 위한 전략적인 목적으로 진행되고 있다. 수천만 이상의 웹 사이트들을 상대로 하는 정보 수집은 검색 엔진 사이트 중심으로 기존 데이터의 수정과 삭제 등과 같은 데이터 베이스 유지 관리와 신규 사이트들에 대한 자료 수집 작업이 이루어지고 있다. 이러한 작업은 웹 서버에 대한 사전 지식 없이 정보 추출을 위해 웹 로봇을 실행하므로 인터넷 상에 수많은 요구가 전송되고 이는 인터넷 트래픽을 증가 시키는 원인이 되고 있다. 따라서 웹 서버가 사전에 자신이 공개할 문서에 대한 변경 정보를 웹 로봇에게 통보하고 웹 로봇은 이 정보를 이용해서 웹 서버의 해당 문서에 대한 정보 수집 작업을 한다면 불필요한 인터넷 트래픽을 감소시킬 수 있을 뿐만 아니라 검색 엔진의 정보의 신뢰도도 높아지고 웹 서버의 시스템 부하와 검색 엔진의 시스템 부하를 줄일 수 있는 효과를 가질 수 있을 것이다. 본 논문에서는 웹 서버상의 웹 문서 파일의 변동 사항을 자동으로 검사하고 변동 사항들을 종합 정리해서 변경 문서에 대한 정보를 통보 받기 원하는 등록된 각 웹 로봇에게 전송하는 검사 통보 시스템을 설계 구현하였다. 웹 로봇을 운영하는 검색 엔진에서는 통보된 요약 정보를 이용해서 웹 서버로부터 해당 문서를 전송 받아 필요로 하는 인덱스 정보를 추출해서 데이터베이스를 구축하는 효율적인 웹 로봇을 설계 구현하였다.

A Document Collection Method for More Accurate Search Engine

Eun-Yong Ha[†]·Hee-Yong Kwon[†]·Ho-Young Hwang[††]

ABSTRACT

Internet information search engines using web robots visit web servers connected to the Internet periodically or non-periodically. They extract and classify data collected according to their own methods and construct their databases, which are the basis of web information search engines. These procedures are repeated very frequently on the Web. Many search engine sites operate this processing strategically to become popular internet portal sites which provide users ways how to find information on the web. Web search engine contacts to thousands of thousands web servers and maintains its existed databases and navigates to get data about newly connected web servers. But these jobs are decided and conducted by search engines. They run web robots to collect data from web servers without knowledge on the states of web servers. Each search engine issues lots of requests and receives responses from web servers. This is one cause to increase internet traffic on the web. If each web server notify web robots about summary on its public documents and then each web robot runs collecting operations using this summary to the corresponding documents on the web servers, the unnecessary internet traffic is eliminated and also the accuracy of data on search engines will become higher. And the processing overhead concerned with web related jobs on web servers and search engines will become lower. In this paper, a monitoring system on the web server is designed and implemented, which monitors states of documents on the web server and summarizes changes of modified documents and sends the summary information to web robots which want to get documents from the web server. And an efficient web robot on the web search engine is also designed and implemented, which uses the notified summary and gets corresponding documents from the web servers and extracts index and updates its databases.

키워드 : 웹 로봇(Web Robot), 문서 수집(Document Collection), 검색 엔진(Search Engine)

1. 서 론

인터넷에서 웹 사용자에게 손쉬운 정보 검색 서비스를 제공하는 대표적인 검색 엔진인 Google, Yahoo, Lycos, Northernlight, AltaVista, Excite, HotBot, MetaCrawler들은 자신이 제공하는 정보의 최신성과 신뢰도를 높이기 위해 수많은 웹 서버 사이트들이 연결되어 있는 인터넷이라는 정보의 바다를 항해하면서 원하는 양질의 자료를 수집한다. 이를 위해 독자적인 웹 로봇을 이용하고 있다. 이렇게 해서 얻어진 자료의 분량이 방대하기 때문에 정확한 자료로서 유지 관리하는데 드는 비용과 이로 인해 부가적으로 발생하는 인터넷 트래픽도 대단할 것이다. 인터넷상에서 자료를 수집하는 웹 로봇의 수가 알려진 것만 해도 수백 개에 이르고,

※ 이 논문은 1999년도 안양대학교 학술연구비의 지원을 받아 연구되었음.
† 종신회원 : 안양대학교 컴퓨터공학과 교수
†† 정 회 원 : 안양대학교 디지털미디어학부 교수
논문접수 : 2003년 7월 10일, 심사완료 : 2003년 10월 13일

알려지지 않은 웹 로봇을 헤아리면 그 수가 상당할 것이다[1]. 이런 웹 로봇들이 항해하는 횟수가 증가할수록 네트워크 트래픽은 증가하게 된다. 웹 로봇의 요구를 처리하는 웹 서버의 작업 부담은 자신의 로컬 작업을 처리하는데 있어서 효율 저하를 가져오는 원인이 되고 있다[2, 3]. 어쩌면 웹 사용자가 홈페이지에 접근하는 횟수보다 웹 로봇이 정보를 수집하기 위해 웹 서버에 접근하는 횟수가 더 많을 수도 있다[4].

이렇듯 인터넷 상에서 정보 검색이 대량화 되면서 정보 검색에 대한 최근 연구 동향은 첫째, 정보 검색 속도를 빠르게 하기 위해 정보의 근접성을 유지하는 Internet Caching 방법에 대한 연구가 진행되었다[5]. 둘째, 여러 검색 엔진에 분산되어 있는 정보를 종합해서 검색 서비스를 제공하는 메타 정보 검색 방법에 대한 연구도 진행되었다[6, 7]. 셋째, 다른 관점에서 정보 검색 결과의 신뢰도를 높이기 위한 방법에 대한 연구가 진행되고 있다. 다시 말하면 검색된 결과에 포함되어 있는 항목이 더 이상 존재하지 않는 죽은 링크(Dead Link)와 웹 서버상의 문서 변경 사항이 반영되지 않고 부정확한 인덱스 정보를 갖고 있는 등의 문제가 있기 때문에 신뢰도를 높이기 위한 방법이 필요하다.

정보의 생명은 정확성에 있다. 최신의 정보를 유지하기 위한 방법으로 검색 엔진 중심의 정보 수집 방법을 탈피해서 정보의 원천지인 웹 서버의 적극적인 협조를 통해 신뢰도 높은 최신의 정보를 유지할 수 있다. 정보 수집으로 인해 발생하는 인터넷 트래픽도 감소시키고 웹 서버의 부하도 줄일 수 있는 연구가 필요하다.

따라서 본 논문에서는 웹 서버상의 웹 문서의 변동 사항을 자동으로 검사하고 변경된 사항들을 종합 정리해서 변경 문서에 대한 정보를 수집하기를 원하는 등록된 각 웹 로봇에게 전송하는 웹 서버상에 검사 통보 시스템을 설계 구현하였다. 또한 웹 로봇을 운영하는 검색 엔진에서는 통보된 종합 정보를 이용해서 웹 서버로부터 해당 문서를 전송받아 필요로 하는 인덱스 정보를 추출해서 데이터베이스를 구축하는 효율적인 웹 로봇을 설계 구현하여 불필요한 인터넷 트래픽을 줄이고 신뢰도 높은 검색 엔진의 데이터베이스를 구축할 수 있는 프로토타입을 구현하였다.

1.2 논문의 구성

본 논문의 구성은 다음과 같다. 2장에서는 인터넷상의 정보수집 모델에 대해 일반적인 방법과 제안한 방법에 대한 설명을 한다. 3장에서는 변경 정보를 통보 받기 원하는 웹 로봇의 등록 프로토콜에 대해 설명한다. 4장에서는 웹 서버에서 웹 문서에 대한 감시로 생성된 변경 정보 파일과 처리 알고리즘에 대해 설명한다. 5장에서는 제안한 로봇이 정보 수집 방법과 이를 검증하는 검색 엔진의 사용자 인터페이스를 설명하였으며 결론에서는 연구의 결과를 요약한다.

2. 인터넷상의 정보수집 모델

2.1 일반적인 정보수집 방법의 문제점

일반적으로 웹 로봇은 일종의 웹 클라이언트로 초기의 URL(Universal Resource Locator)[8]로 지정된 인터넷 상의 웹서버를 접근하면서 HTML 문서를 찾아서 참조 안된 하이퍼링크를 자동으로 추적하여 원하는 정보를 수집하고 자신의 데이타베이스에 내용을 저장한다. 이때 중복되는 URL 방문을 방지하기 위해 모든 참조된 URL에 대한 상태를 유지한다.

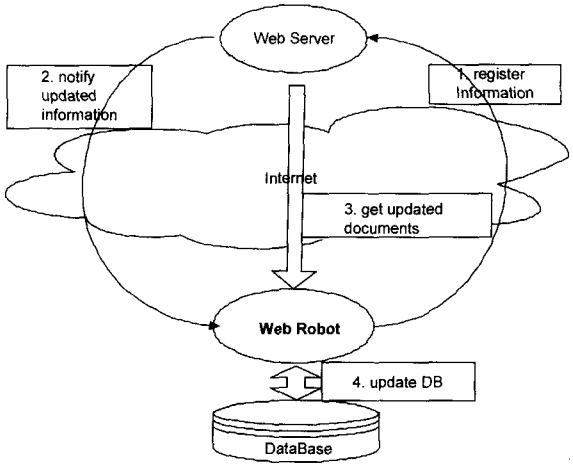
그러나 URL에 명시된 웹 서버가 현재 인터넷에 연결이 안된 상태이거나 다운된 경우에도 웹 로봇은 웹 서버 호스트에게 TCP 연결 요구를 보낸 후 서버의 무응답에 대한 처리를 하게 된다. 또한 URL에 명시된 문서가 존재하지 않는 경우에도 웹 서버에게 TCP 연결을 요구한다. 이로서 TCP 연결이 설정되고 이 연결을 이용해서 해당 문서에 대한 전송 요구를 보낸다. 이 전송 요구 역시 에러 메시지를 응답 받아 웹 로봇이 적절히 처리하게 된다. 이러한 현상은 결국 정보의 정확도가 낮아서 발생하는 것으로서 인터넷상의 불필요한 트래픽을 발생시키고 웹 서버 시스템과 웹 로봇이 실행되는 시스템의 부하를 가중시키는 결과를 초래한다.

2.2 웹 서버의 협조를 통한 정보수집 모델

앞에서 설명한 방법의 문제를 해결하기 위해 본 논문에서는 (그림 1)과 같은 웹 서버의 협조를 통한 효율적인 정보 수집 모델을 제시한다. 이 모델은 기본적으로 정보의 근원지인 웹 서버가 적극적으로 자신이 소유한 정보에 대해 전파 임무를 수행하고 웹 로봇은 전달받은 웹 서버 문서에 관한 요약 정보를 이용해서 실제 문서를 읽어와 자신의 데이터 베이스를 구축하게 된다. 즉 웹 서버가 중심이 되고 웹 로봇은 서버가 전달해 준 정보를 기초로 작업을 수행하는 관계에 있다.

웹 로봇은 웹 정보를 수집하기 전에 웹 서버로부터 정보에 대한 변경사항이 있음을 통보 받고 정보를 수집하므로 정보의 신뢰성을 보장받을 수 있을 뿐만 아니라 웹 로봇이 불필요한 인터넷 항해를 막을 수 있기 때문에 인터넷의 트래픽을 줄일 수 있는 장점을 갖는다.

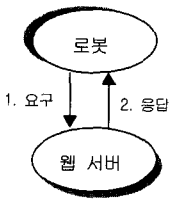
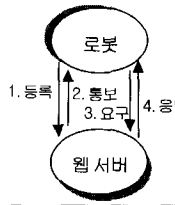
제시한 정보 수집 절차는 (그림 1)과 같다. ① 먼저 웹 로봇이 자신의 데이터베이스 변경주기(latency), E-Mail 주소 등과 같은 내용을 웹 서버에게 등록한다. ② 웹 서버는 웹 로봇 등록 정보를 유지하면서 자신의 공개 문서들의 변경 사항을 감시하고 그 결과를 종합해서 원하는 웹 로봇에게 통보한다. ③ 웹 로봇은 해당 웹 서버로부터 변경된 문서만을 읽어 검색 데이터 베이스를 변경하는 과정을 반복한다.



(그림 1) 새로운 정보 수집 모델

이상에서 설명한 일반적인 정보수집 방법과 제시한 정보 수집 방법에 특징을 비교하면 <표 1>과 같이 요약할 수 있다.

〈표 1〉 정보수집 방법 비교

	일반적인 수집 방법	새로운 수집 방법
정보 수집 대상 사이트	인터넷 상의 불특정 다수 또는 전체 웹 서버	자신이 관리하기를 원하는 등록 사이트
수집 실행 주체	웹 로봇이 주도적으로 수행	웹 서버의 적극적인 협조로 웹 로봇이 종속적으로 수행
네트워크 트래픽	불필요한 트래픽 발생이 잦음	거의 필요한 최소한의 트래픽 발생
수집 절차		
정보의 신뢰도	신뢰도가 낮음	신뢰도 높음

2.3 제안한 정보수집 모델의 시스템 구성도

제안한 정보 수집 모델의 전체적인 시스템 구성도는 (그림 2)와 같다. 웹 서버 시스템은 웹 문서 감시모듈, 요약 정보 생성모듈, 웹 로봇 등록모듈로 구성되며 검색엔진 시스템은 웹 로봇 등록 모듈, 요약정보를 이용한 문서 수집모듈, 인덱스 추출 및 DB 변경 모듈, 정보검색 모듈로 구성된다.

① 웹 로봇이 웹 서버에게 등록을 요구하면 웹 서버는 등록 에이전트 모듈을 구동시켜 적절한 등록 처리를 한다.

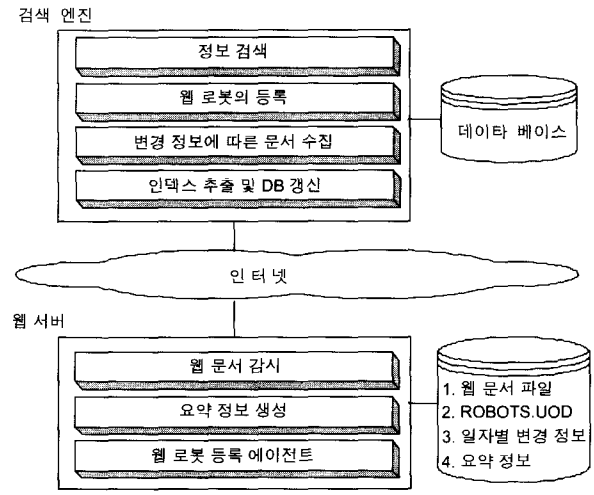
② 웹 서버는 자신의 웹 문서에 대한 변경과정을 감시한다(웹 문서 감시 모듈).

③ 웹 서버는 ①에서 등록된 정보에 따라 웹 로봇에게 변경 사항을 요약하여 메일로 보낸다(요약정보 생성).

④ 웹 로봇은 요약정보를 받아 웹 서버로부터 변경 웹 문서를 읽어 온다(문서수집).

⑤ 웹 로봇은 웹 서버에서 읽어온 문서를 분석하고 인덱싱하여 데이터 베이스를 갱신한다(인덱스 추출 및 DB 갱신).

⑥ 웹 사용자는 자신이 원하는 정보를 검색엔진을 통해 서비스 받는다.



(그림 2) 제안한 시스템 구성도

3. 웹 로봇 등록 프로토콜

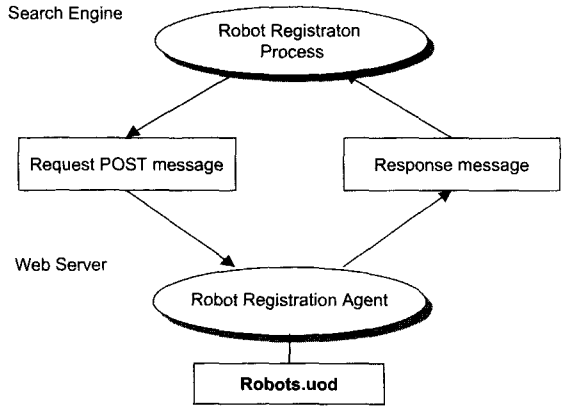
이 장에서는 웹 로봇이 주기적인 정보 수집을 위해 웹 서버에게 문서 변경 정보를 통보해 줄 것을 요구하는 등록 프로토콜에 대해서 설명하고, 이를 위한 웹 서버상의 모듈들과 필요한 자료구조에 대해서 구체적으로 설명한다.

3.1 웹 로봇 등록 에이전트

웹 로봇은 문서 변경 정보를 받기 원하는 웹 서버에게 자신에 대한 정보를 등록해야만 한다. 웹 서버가 이를 토대로 각 웹 로봇이 원하는 정보를 제공하게 된다. 이것은 웹 서버를 무차별하게 액세스하려는 웹 로봇으로부터 자신의 시스템을 보호하려는 로봇 배제 방법[3]과 웹 로봇에게 관심 있는 웹 서버를 선택할 수 있도록 하고 웹 서버가 변경 사항을 통보해 준다는 의미에서 엄격히 다르다.

웹 로봇의 등록은 웹 서버상의 에이전트를 실행해서 처리한다. 웹 서버는 자신의 디렉토리에 "robots.uod" 파일에 각 웹 로봇에 대한 정보를 기록한다. 이 파일에는 문서 변경 통보 주기, 웹 로봇의 E-Mail 주소, 일시 중단 여부 등과 같은 필드가 포함되어 있다. (그림 3)은 웹 로봇이 웹 서버에게 자신의 정보를 등록하는 절차를 보여주고 있다. 웹 로봇의 등록 프로그램은 등록을 위해 먼저 웹 서버와의 TCP 연결을 설정한다. 웹 서버상의 에이전트 호출을 위해 HTTP 프로토콜[9]의 POST 메시지를 구성해서 웹 서버로

전송한다. 이 POST메시지는 웹 서버상의 등록 에이전트의 URL과 웹 로봇에 대한 상세 정보가 포함되어 있다. 웹 서버에서 등록 에이전트가 구동 되면서 최초 등록일, 최종 전송일 등의 관리상 필요한 부가적 필드와 함께 "robots.uod" 파일에 기록한다. 이때 웹 로봇의 E-Mail주소가 기본 키로 사용되며, E-mail 주소가 "robots.uod" 파일에 있는 경우 정보가 수정되며 존재하지 않는 경우는 신규로 등록된다.



(그림 3) 웹 로봇 등록 과정

다음 (그림 4)는 웹 로봇이 자신을 등록하기 위해 HTTP/POST 메시지를 구성하고 웹 서버에게 메시지를 전송하고 웹 서버의 등록 에이전트를 호출하는 프로그램의 일부다.

```
send_robot_regist_POST_request() {
  uchar post_mesg [ HUGE_STRING ], mesg_body [ STRING ] ;
  uchar mesg_action [ STRING ], mesg_email [ STRING ],
                mesg_latency [ STRING ] ;
  int       n ;

  sprintf (mesg_action, "ACTION = %s", robot_action) ;
  sprintf (mesg_email, "ROBOT_EMAIL = %s", robot_email) ;
  sprintf (mesg_latency "LATENCY = %d", robot_latency) ;
  sprintf (mesg_body, "%s&%s&%s", mesg_action, mesg_email,
                mesg_latency) ;
  sprintf (post_mesg, "POST%s HTTP/1.1 : %sLength : %d%s",
              cgi_bin, SERVER_HOST, strlen (mesg_body),
              mesg_body) ;
  n = strlen (post_mesg) ;
  write (s, post_mesg, n) ;
}
```

(그림 4) 등록 메시지 생성

3.2 robots.uod 파일 구조

웹 서버가 여러 웹 로봇에 대한 정보를 유지 관리하는 "robots.uod" 파일의 각 엔트리 형식은 다음과 같다.

- Robot's E-mail : 웹 로봇의 E-mail 주소로 변경 통보 시각이 되면 웹 서버의 요약정보 생성 프로그램이 문서 변경 종합 정보를 작성하여 이 주소로 메일을 보낸다. 이는 웹 로봇을 구별하는 기본키 필드 역할을 한다.
- Notification Latency : 변경 통보 주기로 하루 단위로 기술된다.
- Registration Date : 웹 로봇이 등록한 날자.
- Last Notified Date : 웹 서버가 문서 변경 종합 정보를 웹 로봇에게 보낸 최종 날자.
- Status : 웹 로봇이 일시적으로 통보를 중단해 줄 것을 요구할 때 사용하는 필드로 ON은 정상 통보 OFF는 일시 중단을 의미한다. 웹 로봇은 등록 메시지에 이 필드 값을 지정해서 관리한다.

3.3 웹 로봇 등록 관련 메시지 구조

웹 로봇의 등록과 관련된 Request/Response 메시지들은 다음과 같다.

- Request 메시지

HTTP/POST 메시지에 캡슐화 되는 메시지는 [Action, E-mail, Latency] 값으로 구성된다. Action 값은 Register | Unregister | Change | Suspend | Resume 값 중에 하나를 갖는다. E-mail은 웹 로봇의 주소이고, Latency는 하루 단위의 통보 주기 값을 갖는다. 웹 로봇은 이 Request 메시지를 사용해서 웹 서버에 대해 등록/취소/변경/중지/재개 요구를 한다. Request 메시지의 형식은 (그림 5)과 같이 정의된다.

```
Request_messgage = Action CRLF
                   RobotEmail CRLF
                   Latency CRLF
Action = Register | Unregister | Change | Suspend | Resume
RobotEmail = Internet E-mail address format
Latency = Number
```

(그림 5) Request 메시지 형식

- Response 메시지

웹 로봇의 Request에 대한 응답인 Response 메시지는 요구에 대한 결과로서 결과 값과 수행 결과의 설명으로 구성된다. 결과 값이 100인 경우 성공을 의미하고 20X는 실패를 의미한다. Remark는 요구에 대한 실행 결과의 설명이다. 201은 등록에 대한 실패, 202는 등록 취소에 대한 실패, 203은 변경에 대한 실패, 204는 일시 중단에 대한 실패, 205는 재개에 대한 실패를 의미한다. Response 메시지 형식은 (그림 6)과 같이 정의된다.

```
Response_messgage = ResultValue CRLF
                    Remark CRLF CRLF
ResultValue = 100 | 201 | 202 | 203 | 204 | 205
Remark = Description of Request processing
```

(그림 6) Response 메시지 형식

3.4 알고리즘

웹 로봇이 웹 서버에게 등록을 요청하는 알고리즘은 다

음과 같다.

① 웹 서버에 대해 TCP 소켓을 생성하고, HTTP 등록 port로 연결을 설정한다.
② 원하는 작업에 대한 Request메시지를 생성한다.
③ 메시지를 소켓을 통해 웹 서버에게 보내고 결과를 기다린다.
④ 웹 서버가 응답한 결과를 읽어서 실패한 경우 적절한 처리를 한다.
⑤ TCP 소켓을 닫고 작업을 종료한다.

웹 서버가 웹 로봇의 요구를 처리하는 등록 에이전트 알고리즘은 다음과 같다.

① 전송 받은 POST메시지에서 각 필드들을 추출한다.
② Action 필드 값에 따라 요구한 작업을 수행한다
③ 처리 결과에 대한 Response 메시지를 작성해서 웹 로봇에게 전송한다.

(그림 7)는 웹 로봇의 E-mail주소로 "robots.uod" 파일을 먼저 검색하여 E-mail 주소에 해당하는 웹 로봇에 대한 레코드의 위치를 계산하며 Action 필드 값에 따라 등록 관련 작업을 수행하는 프로그램의 일부분이다.

```
/* Action에 따라 등록 처리를 한다 */
write_robot()
{
        ROBOT_INFO    *robot ;
        long          offset ;

        offset = search_robot() ;
        switch (action) {
                case 1 : /* register */
                        copy_to_record (&robot) ;
                        break ;
                case 2 : /* unregister */
                        copy_to_null (&robot) ;
                        break ;
                case 3 : /* change */
                        copy_to_record (&robot) ;
                        break ;
                case 4 : /* suspend */
                        robot.status = OFF ;
                        break ;
                case 5 : /* resume */
                        robot.status = ON ;
                        break ;
        }
        if (action == 1) lseek (fdrobot, 0L, SEEK_END) ;
        else lseek (fdrobot, offset, SEEK_SET) ;
        write (fdrobot, (unsigned char*) &robot,
                sizeof (ROBOT_INFO)) ;
}

long search_robot()
{
        ROBOT_INFO    robot ;
        long          pos ;
        lseek (fdrobot,0,SEEK_SET) ;
        while (read (fdrobot, (unsigned char *) &robot, sizeof
          (ROBOT_INFO) ₩== sizeof (ROBOT_INFO)) {
            if (strncmp (g_email, robot. email, strlen (robot.
              email)) == 0 ) {
                pos = lseek (fdrobot, sizeof (ROBOT_INFO) ×
                      long(-1), SEEK_CUR) ;
                      return(pos) ;
            }
        }
        return (0L) ;
}
```

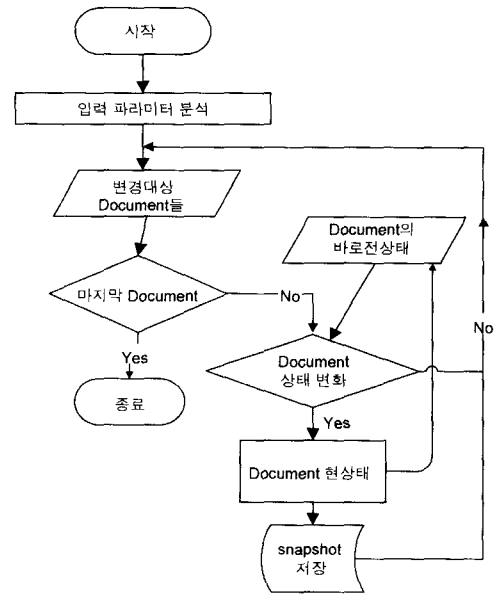
(그림 7) Action 필드 값에 따라 등록 처리

4. 웹 서버 문서 감시

이 장에서는 공개할 문서 디렉토리의 변화과정을 감시하여 변경정보 파일을 생성하고 변경정보를 통보 받기 원하는 날짜에 요약정보를 만들어 웹 로봇에게 전달하는 웹 서버상의 문서 감시에 대해 설명하겠다.

4.1 웹 서버 문서 감시 모듈

웹 서버는 공개하고자 하는 문서 전체를 감시하고 문서의 snapshot을 주기적으로 생성해서 파일에 기록해둔다. 이 파일은 일자별로 생성된다. 문서 감시 프로그램의 실행은 운영체제에서 주기적으로 프로세스의 실행을 스케줄링하는 기능을 이용했다. 그 이유는 문서를 감시하는 프로세스가 계속 실행 상태로 있는 경우 웹 서버의 시스템 부하를 가중시키는 요인이 될 수 있기 때문에 일정 시간마다 프로세스가 동작하는 방식을 채택했다. 문서 감시 프로그램의 처리 과정은 (그림 8)과 같다.



(그림 8) 문서 감시 처리 흐름도

웹 서버는 일자별로 문서에 대한 변경사항을 유지 관리하는데 일자별 변경파일명은 YYYYMMDD로 표기하고 확장자는 ".his"이다. 예로 20030701.his는 2003년 7월 1일의 문서들에 대한 변경 정보 파일이다. 일자별 변경 정보 파일의 각 엔트리는 다음 항목으로 구성된다.

- Document's Status : 문서의 변경 상태로 'A'는 새로운 문서 생성, 'U'는 기존문서의 변경, 'D'는 문서의 삭제를 나타낸다.
- Document's Name : 문서명을 나타낸다.
- Document's Size : 문서가 변경된 후의 문서의 크기를 나타낸다.
- Last Modified Date : 문서의 최종 변경시각을 기록한다.

하나의 웹 문서 파일에 대해 이전 상태와 현재 상태를 비교하여, 다른 경우 현재 상태를 변경 정보 파일에 기록한다. 이 과정을 모든 문서에 대해 반복 처리하여 문서 변경 정보를 일자 별로 생성한다. 다음은 웹 서버가 공개할 문서에 대한 감시 프로그램의 알고리즘이다.

① 감시 대상 파일 전체에 대한 이전 상태정보를 읽는다.
② 감시 대상 파일 중 하나의 파일에 대한 현재 상태를 얻는다.
③ 이전 상태와 비교해서 다르면 기존 문서의 변경 처리를 한다. 즉, 일자별 변경 정보 파일에 상태를 'U'라고 기록 한다.
④ 파일이 이전 상태는 있는데 현재 상태가 없는 경우는 기존 문서의 삭제 처리를 한다. 즉 일자별 변경 정보 파일에 상태를 'D'라고 기록 한다.
⑤ 현재 상태만 있는 경우는 새로운 문서의 생성 처리를 한다. 즉, 일자별 변경 정보 파일에 상태를 'A'라고 기록한다.
⑥ 감시 대상 파일이 더 이상 존재하지 않을 때까지 ②③④⑤과정을 반복 처리한다.

(그림 9)는 웹 서버가 공개할 문서에 대한 감시 처리를 하는 프로그램의 일부다.

```
monitor_documents(){
        int         i ;
        for (i = 0 ; i < fileCount ; i++)
                monifor_one_file (filename[i]) ;
        for (i = 0 ; i < MAX_FILES ; i++ ) {
                if (statinfo[i].last && ! statinfo[i].current)
                        write_infor("D", i) ;
                statinfo[i].last = statinfo[i].current ;
                statinfo[i].current = FALSE ;
        }
}
monitor_one_file (uchar *fname){
        struct      stat statbuf ;
        mode_t      mode ;
        int         retval ;
        retval = stat (fname,&statbuf) ;
        if (retval == FALSE) {
                printf ("Can't stat %s\n", fname) ;
                return ;
        }
        mode = statbuf.st_mode ;
        if (S_ISDIR (mode))
                access_directory (fname) ;
        else if (S_ISREG(mode) || S_ISCHR(mode) || S_ISBLK
                (mode)) update_file_status (fname, &statbuf) ;
}

update_file_status (uchar *fname, struct stat *statbuf){
        int     pos ;
        pos = find_record (fname) ;
        if (pos == NOTFOUND )
           pos = add_record (fname, &statbuf) ;
        else pos = update_record (pos, &statbuf) ;
        if (pos != NOTFOUND) statinfo[pos].current = TRUE ;
}

add_record(uchar *fname, struct stat *statbuf){
        int     i ;
        if ((i = next_free()) == NOMORE) return (NOMORE) ;
        strcpy (statinfo[i].filename, fname) ;
        statinfo[pos].status = *statbuf ;
        write_infor ("A", i) ;
        return (i) ;
}
update_record(int pos, struct stat *statbuf){
        if (statinfo[pos].status.st_time != statbuf → st_time) {
                statinfo[pos].status = *statbuf ;
                write_infor("U", i) ;
        }
}
```

(그림 9) 문서 감시 프로그램의 일부분

4.2 변경 요약 정보 생성

웹 로봇에게 변경 사항을 통보해야 할 시간이 되면 최종 통보일로부터 현재 일까지상태 정보를 처리하여 요약정보를 생성한다. 이 기간 동안 한 문서가 여러 번 변경되어도 처음 상태와 마지막 상태에 따라 요약 정보를 만들어 웹 로봇에게 E-mail로 전송한다.

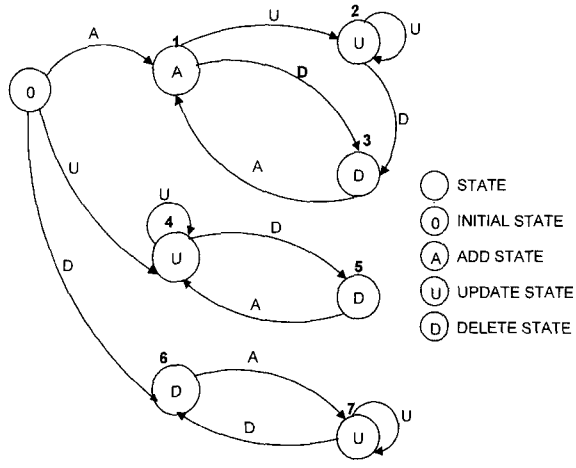
구체적인 요약 정보 생성 과정은 다음과 같다. 단, 'A'는 생성, 'U'는 변경, 'D'는 삭제를 각각 의미한다.

- 문서가 처음 생성된 상태에서 변하는 경우

① 문서파일이 처음으로 생성되면 상태는 'A'이다.
② 'A' 상태에서 문서가 수정되면 'U' 상태가 되고, 그 이후에 계속 수정이 발생해도 계속 'U' 상태를 유지한다.
③ (2)의 'U' 상태에서 문서가 삭제 되면 'D' 상태가 된다.
④ (3)의 'D' 상태에서 문서의 생성 외에는 다른 Event가 발생할 수 없으며 문서가 다시 생성되면 'A' 상태가 된다.

⑤ (1)의 'A' 상태에서 문서의 삭제가 발생하면 'D' 상태가 된다.

이외에 변경 및 삭제 상태에서 시작되는 경우도 같은 방식으로 처리해서 상태에 대한 요약 정보를 생성할 수 있다.



(그림 10) 요약정보 추출 상태 전이도

(그림 10)은 처음 상태에서 각 문서에 대한 상태 변화가 있을 때 전이되는 과정을 나타낸 것이다. 즉 현재상태에서 생성/변경/삭제 이벤트에 따라 다음 상태로 전이된다. 이 상태 전이도를 기반으로 상태 변환표를 만들면 <표 2>와 같다. 상태 변환 테이블에 따라 웹 문서에 대한 생성/변경/삭제 처리를 할 수 있다. 따라서 로봇이 변경 정보를 받기 원하는 기간 동안의 하나의 웹 문서의 변화 과정을 추적하면서 문서에 대한 요약 정보를 추출해 낼 수 있다.

〈표 2〉 상태 전이 변환표

Current State	Event	Next State	Current State	Event	Next State
0	A	1	3	A	1
0	U	4	4	U	4
0	D	6	4	D	5
1	U	2	5	A	4
1	D	3	6	A	7
2	U	2	7	U	7
2	D	3	7	D	6

〈표 3〉 상태 전이 결과표

시작 상태	마지막 상태	결 과	시작 상태	마지막 상태	결 과	시작 상태	마지막 상태	결 과
A	A	A	U	A	U	D	A	U
A	U	A	U	U	U	D	U	NO
A	D	NO	U	D	D	D	D	D

상태 전이도를 살펴보면 상태 번호가 1에서 2로 전이 되어도 문서의 상태는 문서가 처음 생성된 상태에서 전이되는 경우이므로 새로운 문서의 생성인 'A'가 된다. 이 의미는 하나의 웹 문서가 변화되는 과정 중에서 처음 시작 상태가 중요한 역할을 한다는 것을 의미 한다. <표 3>는 상태 전이 결과표를 보여준다.

4.3 요약 정보 명세

앞에서 생성된 요약 정보는 웹 로봇에게 E-Mail로 전달된다. 이때 Subject는 "Robot : Update of Index"라고 작성하고, 메일의 Body에 (그림 11)과 같은 형식으로 문서에 대한 변경 요약을 삽입하여 웹 로봇에게 전송한다. 웹 로봇에게 전달되는 메일 중에서 웹 서버가 문서의 변경 사항을 알리는 메일을 구별하기 위해 앞에서 설명한 Subject에는 "Robot : Update of Index"라는 태그와 Body의 각 행은 < RoBoT >이라는 태그를 사용했다.

웹 로봇은 요약 정보를 확인한 후 이에 따라 해당 웹 문서를 수집한다. 이때 웹 서버의 Hostname은 메일의 From 절에서 추출하여 어떤 웹 서버로부터 전달된 요약정보 인지를 확인하고 이 Hostname과 해당 웹 서버의 Document의 위치 정보를 이용하여 웹 문서를 수집한다.

```
< RoBoT > A:/web/html/index.html:156:20030720
< RoBoT > U:/web/html/test01.html:357:20030721
< RoBoT > D:/web/html/test02.html:420:20030722
< RoBoT > A:/web/html/test03.html:120:20030727
```

(그림 11) 요약정보 예

5. 새로운 로봇의 정보 수집

이 장에서는 웹 서버로부터 변경정보에 대해 통보를 받은 웹 로봇이 변경 문서들을 웹 서버로부터 수집하여 검색 데이터 베이스에 반영하는 과정에 대해 설명하고 검색 엔진의 사용자 인터페이스를 구현해서 실행되는 예를 설명하겠다.

5.1 웹 로봇의 변경 정보 수집

웹 로봇은 자신의 메일 박스를 읽어 메일의 내용 중에 Subject가 "RoBoT : Update Of Index"인 메일을 처리한다. 이 메일의 Body 부분은 앞에서 설명한 웹 서버의 문서 감시 프로그램에서 작성된 형식으로 레코드의 시작이 <RoBoT>이라는 태그로 시작된다. 따라서 웹 로봇은 웹 서버로부터 통보 받은 문서에 대한 변경 정보임을 확인하고 각 레코드를 분석하여 변경 문서를 웹 서버로부터 읽어 온다. 이 웹 문서를 분석해서 색인을 추출하고 검색 데이터 베이스에 저장한다.

변경된 웹 문서를 가져오는 방법은 두 가지가 있다. <표 4>에서 보듯이, 첫째, 웹 문서를 하나씩 읽어와서 데이터

베이스를 갱신하는 작업을 반복하는 온라인 처리 방법이 있다. 이 방법은 웹 로봇과 웹 서버와의 TCP 연결을 계속 유지하면서 데이터 베이스를 갱신하게 된다. 이는 검색 엔진에서 데이터 베이스 관리 시스템과 관련된 프로세스를 계속 유지해야 하는 부하와 인덱스가 처리되는 동안 TCP 연결이 유휴 상태로 있는 단점이 있다. 둘째는, TCP 연결을 설정해서 해당 문서들을 모두 수집한 후 TCP 연결을 닫고 해당 문서들에 대해 일괄적으로 데이터 베이스를 갱신하는 일괄 처리 방법이 있다. 본 논문에서는 두 번째 방법을 사용하였다.

〈표 4〉 변경 문서 처리 방법 비교

방법 1) 온라인 처리	방법 2) 일괄 처리
① 웹 서버에 대해 TCP 소켓을 생성하고, HTTP port로 연결을 설정한다. ② 변경문서를 가져오는 Request 메시지를 생성한다. ③ 메시지를 소켓을 통해 웹 서버에게 보내고, 결과를 기다린다. ④ 웹 서버가 응답한 결과를 읽어서 데이타 베이스에 반영한다. ⑤ (2), (3), (4) 과정을 변경 정보가 없을 때가지 반복 처리한다. ⑥ TCP 소켓을 닫고 작업을 종료한다	① 웹 서버에 대해 TCP 소켓을 생성하고, HTTP port로 연결을 설정한다. ② 변경문서를 가져오는 Request 메시지를 생성한다. ③ 메시지를 소켓을 통해 웹 서버에게 보내고, 결과를 기다린다. ④ 웹 서버가 응답한 결과를 읽어서 임시 파일에 기록한다. ⑤ (2), (3), (4) 과정을 변경 정보가 없을때가지 반복 처리한다. ⑥ TCP 소켓을 닫고 작업을 종료한다 ⑦ 데이터 베이스에 일괄 처리한다

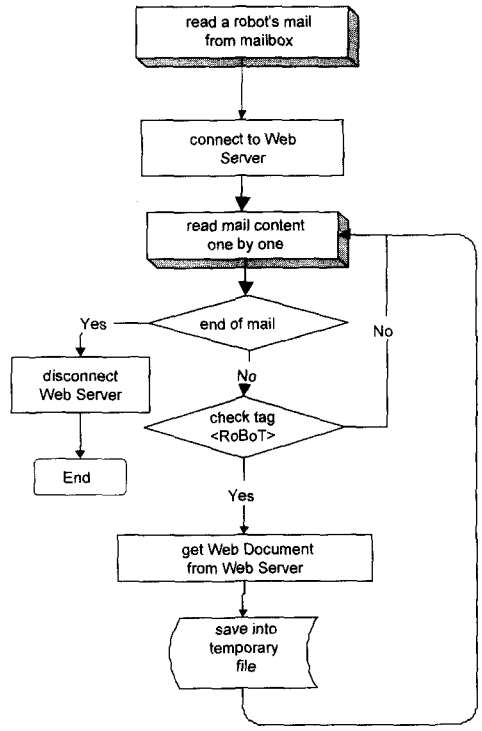
데이터의 최신성을 보장 받기 위해서 웹 서버에서 수집된 문서의 앞 부분에 Hostname, URL, Action, Size, Last-Modified Date를 포함시켰다. 이는 웹 로봇이 인덱스를 추출하는 과정에서 데이터 베이스의 내용과 비교하는데 사용된다.

다음 (그림 12)은 웹 서버에서 전달된 문서들이 데이터 베이스에 반영되기 전에 생성된 임시 일괄처리 파일의 예이다.

```
#HOSTNAME : 웹 서버명

#URL : url
#Action : 문서 상태
#SIZE : 문서 크기
#LASTMODIFIED : 최종 갱신일자

        HTML 본문

#URL : url
#Action : 문서 상태
#SIZE : 문서 크기
#LASTMODIFIED : 최종 갱신일자

        HTML 본문
```

(그림 12) 임시 일괄 처리 파일 예

웹 로봇이 정보를 수집하여 임시 일괄 처리 파일에 기록하는 과정은 (그림 13)과 같다. (그림 13)은 웹 로봇에게 전달된 메일 중에서 변경 정보를 알리는 하나의 메일을 처리하는 알고리즘이다. 메일은 전자우편 형식[7]에 따라 From, Subject, To, Body를 추출하고 Body의 내용을 읽으면서 웹 문서를 수집하여 임시 일괄 처리 파일에 기록한다. 또한 임시 파일 중에 HOSTNAME은 웹 서버를 구분하기 위해 사용된다. (그림 14)는 메일 박스에서 내용을 추출하여 임시 파일을 생성하는 모듈의 부분이다.



(그림 13) 임시 일괄 처리 파일 생성 과정

```
/* 로봇 메일박스를 검사하여 변경정보에 대한 메일를 추출하고
   웹 서버에서 문서를 읽어와 임시 일괄 처리 파일에 저장하는
   프로그램 */

/* 메일박스를 읽어서 하나의 메일 단위로 구분 */
int parse_mailbox(uchar *path){
        for (i = 0 ; i < fc ; i++ ) {
                rc = get_mail_content
                    (fp, ctx[i].mail_offset) ;
                if (rc == ROBOT_MAIL)
                    ctx[i].mail_status = 'D' ;
        }
}
/* 로봇 메일 중에서 변경정보 여부 조사 */
int get_mail_content (FILE  *fp, long offset){
        connect_webserver (server_host) ;
        tfp = fopen(tempfn, "a+") ;
        fprintf (tfp, "#HOSTNAME : %s\n", server_host) ;
        while (fgets (buf, HUGE_STRING, fp) != NULL ) {
                if ( strncmp(buf, "<RoBoT>", 7) == 0 ) {
                        extract_mesg_infor (buf) ;
                        get_html_from_server (html_url) ;
                        msgcount++ ;
                }
```

```
            if (is_from (buf) == 0) break ;
        }
    disconnect_webserver() ;
}
/* 하나의 웹 문서를 읽어서 임시화일에 저장하는 프로그램 */
get_html_from_server (uchar *html_url){
        frpintf (tfp, "# URL : %s\n# ACTION : %s\n# SIZE : %s\n
            # LASTMODIFIED : %s\n",\
                        html_url, html_status, html_size,
                        html_lastmodified) ;
        send_GET_request (html_url) ;
        send_GET_response() ;
}
```

(그림 14) 일괄 처리 파일 생성 모듈 일부분

5.2 인덱스 정보 추출

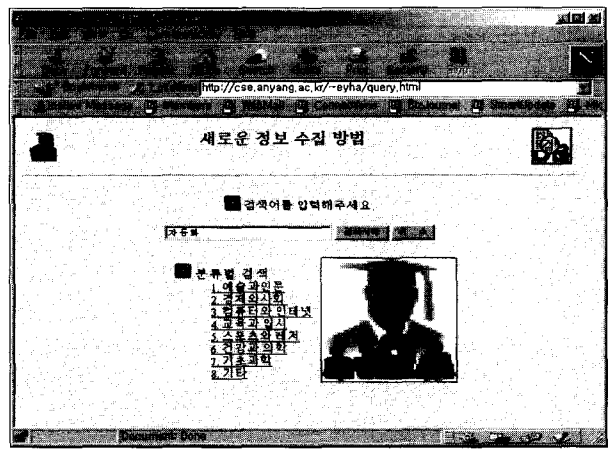
정보 검색 시스템의 성능은 사용자가 원하는 정보를 얼마나 빠르고 정확하게 찾느냐에 달려있다. 웹 문서의 본문을 포함하고 있는 임시 일괄 처리 파일을 읽어서 인덱스 정보를 추출하고 추출된 정보를 데이터 베이스에 반영 시킨다. 또한 인덱스를 추출 하기 전에 임시 파일에 들어 있는 문서의 Size, Last Modified Date 필드를 DB에 이미 기록된 내용과 비교한다. 만일 갱신이 필요하다고 결정되면 웹 문서의 본문을 파싱하여 키워드를 추출한다. 키워드와 제목 URL등의 정보를 데이터 베이스에 반영시켜 검색 시 사용한다. (그림 15)는 웹 문서에서 정보를 추출하는 프로그램의 일부다.

```
/* 웹 문서를 읽어서 미리 정의된 구분자에 의해 단어를
   추출한다. */
        ptr = strtok (buf, DELIMETER) ;
        while (ptr != NULL) {
                strcpy (keyword, ptr) ;
                proc_keyword (keyword) ;
                ptr = strtok (NULL, DELIMETER) ;
        }
/* 추출된 단어에 TAG 정보를 제외한 키워드를 만들어 데이터
   베이스에 반영시킨다.*/
proc_keyword (uchar *word){
        int         i ;
        uchar *t ;
        if (strncmp (word, "<", 1) == 0) {
                for (i = 0 ;  i < TABLESIZE ; i++)
                        if (strcasecmp (word, tag_tab[i].tag)
                           == 0 ) {
                                push_stack(i) ; return(1) ;
                        }
        }
        if (strncmp (word, "</", 2) == 0) {
                for (i = 0 ;  i < TABLESIZE ; i++)
                        if (strcasecmp (word+2, tag_tab[i].tag+1)
                           == 0) {
                                pop_stack(i) ; return(2) ;
                        }
        }
        if (top >= 0 && tag_tab[stack[top].flag == 1 ] {
                t = check_special_char (word) ;
                strcpy (word, t) ;
                query_kwd_processing (word) ;
                                /* 데이터 베이스 Update */
        }
}
```

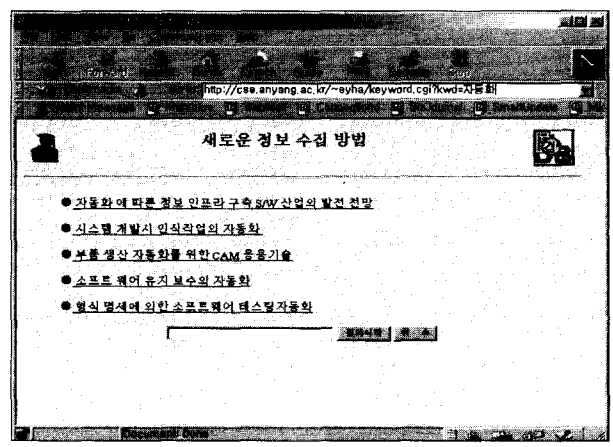
(그림 15) 웹 문서에서 정보를 추출하는 프로그램의 일부

5.3 데이터 베이스 검색 실행 예

지금까지 설계 구현된 시스템의 실행 예를 보기 위해서 (그림 16)과 같이 키워드 검색과 분야별 검색이 가능하도록 웹 페이지를 구성했다. (그림 16)에서 키워드를 입력하고 검색 버튼을 선택하거나, 분야별 검색 메뉴에서 하나의 메뉴 엔트리를 선택하면 질의가 생성되고 이 질의에 대한 검색 결과로 검색 목록이 (그림 17)과 같이 나타난다.



(그림 16) 데이터 베이스 검색 페이지



(그림 17) 검색 결과 페이지

6. 결론 및 향후 연구

인터넷상에서 정보 검색 엔진 사이트의 검색 서비스를 통해 검색한 내용에는 더 이상 존재하지 않는 링크들이 있

다. 이런 링크를 통해 사용자가 정보를 액세스할 때 발생하는 불필요한 네트워크 트래픽을 감소시키고 또한 검색 엔진 운영자가 정보 재수집을 할 때 부정확한 링크를 액세스할 때 발생하는 네트워크 트래픽도 줄이기 위해 연구가 시작되었다.

본 논문은 정보의 생명은 정확성에 있는 만큼 최신의 정보를 유지하기 위한 방법으로 검색 엔진 중심의 정보 수집 방법을 탈피해서 정보의 원천지인 웹 서버의 적극적인 협조를 통해 신뢰도 높은 최신의 정보를 유지하고, 정보 수집으로 인한 인터넷 트래픽도 감소시키고 웹 서버의 부하 또한 줄일 수 있는 방안을 제시했다.

웹 서버상에서는 웹 문서의 변동 사항을 자동으로 검사하고 변경된 사항들을 종합 정리해서 변경 문서에 대한 정보를 수집하기를 원하는 등록된 각 웹 로봇에게 전송하는 웹 서버상에 검사 통보 시스템을 설계 구현했다. 웹 로봇을 운영하는 검색 엔진에서는 통보된 종합 정보를 이용해서 웹 서버로부터 해당 문서를 전송 받아 필요로 하는 인덱스를 추출해서 데이터베이스를 구축하는 효율적인 웹 로봇의 프로토 타입을 설계 구현하였다. 본 논문에서 제시한 모델에 따라 웹 검색 엔진과 웹 서버들이 상호 협조하면 인터넷상의 불필요한 트래픽을 줄일 수 있을 뿐만 아니라 검색 엔진과 웹 서버 시스템의 부하를 줄일 수 있을 것이다.

향후 제시한 모델에 대한 성능 평가를 위한 연구가 진행되어야 할 것이다. 먼저 이 모델을 도입하는 웹 서버에서의 문서 감시를 위해 부가되는 시스템 부하에 대한 측정을 토대로 보다 나은 감시 시스템을 개발해야 할 것이고, 검색 엔진의 데이타 베이스에 저장되어 있는 전체 자료량, 부정확한 자료량, 자료 갱신 주기의 변화에 따라 발생하는 네트워크 트래픽에 대해 분석을 해서 제시한 모델의 성능을 평가해야 할 것이다.

참 고 문 헌

[1] "The Web Robots FAQ," http://www.robotstxt.org/wc/robots.html.

[2] Martijn Koster, "Robots in the Web : threat or treat?," April, 1995.

[3] Koster, M., "A Standard for Robot Exclusion," http://info.webcrawler.com/mak/projects/robots/exclusion.html.

[4] Search Engine Robots that visit your web site, http://www.jafsoft.com/searchengines/webbots.html.

[5] D.Wessels, K. Claffy, "Internet Cache Protocol (ICP), version 2," RFC 2186

[6] Metacrawler : search the search engines, http://www.metacrawler.com/info.metac/dog/index.htm.

[7] Debrifing Meta Search Engine : fast and accurate, http://www.debriefing.com/.

[8] Berners-Lee,T., Masinter,L. and M. McCahill, "Uniform Resource Locators (URL)," RFC 1738.

[9] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol -HTTP/1.1," RFC 2068.

하 은 용

e-mail : eyha@aycc.anyang.ac.kr

1986년 서울대학교 공과대학 전자계산기공학과(공학사)

1988년 서울대학교 대학원 컴퓨터공학과(공학석사)

1997년 서울대학교 대학원 컴퓨터공학과(공학박사)

1997년~현재 안양대학교 컴퓨터공학과 부교수

관심분야 : 무선 이동 통신, 인터넷 프로토콜 및 응용

권 희 용

e-mail : hykwon@aycc.anyang.ac.kr

1983년 서울대학교 공과대학 전자계산기공학과(공학사)

1985년 서울대학교 대학원 전자계산기공학과(공학석사)

1993년 서울대학교 대학원 컴퓨터공학과(공학박사)

1986년~1995년 한국통신 연구개발원

1995년~현재 안양대학교 컴퓨터공학과 부교수

관심분야 : 신경망이론 및 응용, 패턴인식

황 호 영

e-mail : hyhwang@aycc.anyang.ac.kr

1993년 서울대학교 공과대학 컴퓨터공학과(학사)

1995년 서울대학교 대학원 컴퓨터공학과(석사)

2003년 서울대학교 대학원 전기·컴퓨터공학부(박사)

2003년~현재 안양대학교 디지털미디어학부 전임강사

관심분야 : 인터넷프로토콜, 광통신망, 무선 통신, 센서 네트워크