웹 기반 실시간 모니터링 시스템의 구조

Architecture of Web-Based Real-Time Monitoring System

박 홍 성, 정 명 순, 김 봉 선
(Hong Seong Park, Myoung Soon Jung, and Bong Sun Kim)

Abstract : This paper proposes an improved architecture of web-based monitoring system for monitor of processes in plants from the soft real-time point of view. The suggested model is designed to be able to guarantee the temporal and spatial consistency and transmit the monitoring data periodically via the intranet and the Internet. The model generates one thread for monitoring management, one DB thread, one common memory, and corresponding monitoring threads to clients. The monitoring thread is executed during the smaller time than the execution time of the process used in the conventional methods such as CGI and servlet method. The Java API for the server API, VRML, EAI(External Authoring Interface) and Java Applets for efficient dimensional WEB monitoring are used. The proposed model is implemented and tested for a FMS plant. Some examples show that the proposed model is useful one.

Keywords : Servlet, WEB monitoring, temporal and spatial consistency

I. 서론

모니터링 시스템은 공장의 공정 중에서 발생하는 데이터를 수집하고 분석하여 네트웍 내의 원격 사용자에게 모니터링 대상의 상태정보를 알려줌으로써 공정감시와 공장의 효율적인 관리를 할 수 있게 한다. 일반적으로 모니터링 시스템은 자동차 생산라인, 반도체 제조공정, 정수 처리공정 등과 같은 자동화 설비에 포함되어 구축되며 원격 사용자는 모니터링이 필요한 각 공정을 GUI(Graphic User Interface) 타입의 2차원 그래픽으로 관찰할 수 있었다. 하지만 현재까지 이러한 모니터링 프로그램들은 제조업자가 제공하는 프로그램을 사용하였기 때문에 제조업자에 따라 다른 형태로 제공되고, 대부분 PC 플랫폼 기반으로 현재와 같은 다양화된 플랫폼들을 수용하는 FA 망에는 적합하지 않은 단점이 있다[1][2]. 또한 제조업자가 제공하는 모니터링 시스템은 초기 구현 비용이 거의 들지 않지만, 오랜 기간 사용할 경우 유지·보수 비용이 많이 들고 모니터링에 사용되는 클라이언트/서버가 모두 비대해지는 단점이 있다.

한편 개방형 네트웍인 TCP/IP 기반으로 발전하고 있는 인터넷은 다양한 기술을 이용하여 이미 문자 형태의 서비스뿐 아니라 비쥬얼한 형태로 상품 및 입점 업체를 보여주는 가상 쇼핑몰 등으로 멀티미디어화 되고 있다. 또한 제조업자나 사업자들은 인터넷을 이용하여 상품거래와 전시, 가상자원과 마케팅 시장 공유, 사용자 권한부여, 고객 서비스 등을 통하여 제품을 광고하고, 적극적인 자세로 고객과의 가상적인 만남을 이루려고 노력하고 있다. 이러한 경향들은 네트웍을 이용한 자원공유와 가상공간을 기반으로 한 광역 경쟁체제가 되게 하였다. 따라서, 공장 자동화 설비의 모니터링 시스템은 더 이상 제조업자만을 위한 것이 아닌 상품 판매와 홍보를 위한 마케팅 전략의 한 형태가 되고 있다. 모니터링 시스템을 이용하면 제조업자는 실수요자에게 인터넷을 통하여 상품의 구체적인 정보와 생산라인을 관찰할 수 있게 하여 상품판매와 홍보에 대한 비용절감 효과를 얻을 수 있을 것이고 고객은 그에 따른 효과로 고품질, 저가격의 제품을 얻을 수 있을 것이다. 그러나, 이러한 장점에도 불구하고 웹을 모니터링에 사용하기 위해서 3차원으로 표현하는 데는 다음과 같은 문제점들이 있을 수 있다: 첫째, 웹으로 모니터링을 하기 위해서 여러 클라이언트가 서버에 접속할 경우 많은 쓰레드가 생성되어 서버에 부하를 주게 된다. 둘째, 같은 웹 페이지를 여러 클라이언트가 동시에 접속하여 데이터를 모니터링하고자 할 때 데이터를 엑세스하는 쓰레드가 각각 달라 같은 웹 모니터링 페이지인 경우에도 다른 값이 나올 수 있다. 즉, 데이터의 시간 및 공간적 일치성(temporal and spatial consistency of data)이 보장할 수 없는 것이다. 시간적 일치성은 어떤 한 스테이션이 여러 데이터를 사용하는 경우에 그 데이터들은 동일 시간대에 얻어져야 하는 조건이며, 공간적 일치성은 여러 개의 스테이션이 여러 데이터를 사용할 때 같은 시점에 개별 스테이션에서 사용하는 데이터는 동일한 시간대에 얻어진 데이터이어야 한다는 조건이다. 셋째, 3차원 데이터를 로딩할 때 시간이 많이 걸린다. 첫 번째와 두 번째 문제는 일반적인 웹 모니터링에서도 생기는 문제이다. 두 번째 문제는 쓰레드가 데이터베이스를 엑세스하는 시점이 달라 생기는 문제이다. 예를 들면, 클라이언트 A, B, C가 같은 웹 모니터링 페이지를 각각 T_1, T_2, T_3($T_1 \neq T_2 \neq T_3$)시점에서 엑세스하고 모니터링 주기가 P_1이라 하고, 그 웹 페이지에 사용되는 데이터의 갱신(update) 주기가 P_1보다 1/4 정도 작다고 가정하자. 그러면 쓰레드 A, B, C는 각각 T_1+kP_1, T_2+kP_2, T_3+kP_3 시점에서 서버에 접속하게 된다. 이 때 $kP_1 < T_1 < kP_1 + P_1/3 < T_2 < kP_1 + 2P_1/3 < T_3 < kP_1 + P_1$ 이 성립하면, 클라이언트 A, B, C의 웹에서는 서로 다른 데이터를 보이게 되는 것이다. 즉, 데이터 갱신 주기가 모

접수일자 : 2000. 12. 4., 수정완료 : 2001. 4. 12.
박홍성, 정명순, 김봉선 : 강원대학교 전기전자정보통신공학부
(hspark@kangwon.ac.kr/jms@control.kangwon.ac.kr/bskim@control.kangwon.ac.kr)

니터링 주기보다 길면 문제가 되지 않겠지만 짧은 경우 문제가 생길 수 있다.

본 논문은 [3]의 논문을 기초로 하여 성능 측정 등을 추가한 것이다. 본 논문에서는 인터넷 기술과 VRML 기술을 이용하지만 위에서 언급한 문제점을 해결하기 위한 새로운 구조를 사용한 3차원 웹 모니터링 방안을 제안한다. 제안된 새로운 구조는 서버의 부하를 줄일 수 있도록 쓰레드의 생성을 제한하고 이를 이용하여 데이터의 시간적 일치성을 유지할 수 있는 구조이다. 또한 제안된 구조가 적용되는 모니터링 시스템은 서버에서 클라이언트로 주기적인 데이터 전송이 이루어진다. 본 논문에서는 제안된 구조를 웹 모니터링에 적용하여 다른 방법과 성능을 비교하여 유효성을 확인하였다.

논문의 구성은 2절에 3차원 웹 모니터링에 사용되는 기술을 소개하고 3차원 웹 모니터링의 구조와 최적화 방법에 대하여 3절에 기술하였다. 4절과 5절에서는 제안 시스템의 구현 및 테스트 방안을 소개하고 마지막으로 결론을 맺었다.

II. 웹 기술 개요

1. 서버 API

모니터링 시스템은 모니터링 서버의 부하를 줄이고 좀 더 연성실시간 적으로 모니터링 데이터를 갱신하는 것이 중요하다. 이를 위하여 웹서버 본래의 기능 외에 부가적 기능이 필요한데 이러한 기능을 확장하기 위한 방법으로 많은 응용 프로그램이 개발되어 왔다. 이에 대표적인 것으로 CGI(Common Gateway Interface)[4]는 웹서버와 응용 프로그램 사이의 연결에 사용하는 규약으로 정의되었고 현재 웹서버의 기능을 확장하기 위한 방법으로 널리 사용되고 있다. 하지만 CGI를 적용한 응용 프로그램의 수행은 개별적인 클라이언트의 CGI 실행 요구에 대하여 별도의 프로세스를 생성하기 때문에 서버의 부담을 가중시켜 속도가 느려지는 단점이 있다. 이러한 단점을 해결하기 위하여 서버 API들이 제안되어 개발되었다. 특히, 서버 API는 웹서버가 할 수 없는 작업들을 가능하게 한다. 본 논문의 구현 예제에서 사용한 서버 API는 Sun의 JavaAPI인 서블릿(Servlet)이다[5].

일반적으로 공장설비에서 발생하는 공정 데이터는 주기적으로 데이터베이스에 갱신된다. 서버 API는 데이터베이스에 접속하여 일반적인 웹 브라우저가 설치되어 있는 클라이언트의 요구에 응답하여 공장의 갱신되는 데이터를 서버밀기(SeverPush) 동작으로 계속해서 전송한다.

본 논문에서 제안한 모니터링 시스템은 서버 API를 지원하는 윈도우 기반 웹 서버와 호환성을 유지하고 윈도우 운영체제의 장점을 그대로 수용하여 설치, 관리, 유지 등이 용이하다는 장점을 가지고 있다.

2. 서버밀기(ServerPush)

웹 상에서 서버와 클라이언트의 관계는 단순하다. 일반적인 웹 서비스에서는 클라이언트가 브라우저를 이용하여 해당 URL을 요구하면 서버로 접속이 열리고 서버는 클라이언트가 요구한 데이터를 모두 보낸 후에 해당 접속을 닫는다. 하지만 서버밀기 동작은 한번 접속이 열리면 그 접속이 계속 유지된다. 즉, 클라이언트가 접속을 원하지 않아도 서버가 보내는 데이터를 계속 받게 되어 있다[6]. 접속이 이루어지고 끊어지는 동작은 초기 오버헤드를 많이 발생시켜 지연시간을 가져오지만 한 번 접속이 이루어지면 연결 동작 등에 필요한 오버헤드가 없기 때문에 좀 더 빠르게 데이터를 서버에서 클라이언트로 전송할 수 있다.

본 논문에서는 서버밀기 방식을 이용하여 서버의 쓰레드가 웹 페이지에 사용되는 데이터를 주어진 주기에 맞게 클라이언트에게 자동적으로 전송하여 줌으로써 가능한 주기성을 맞추도록 하였다. 물론 많은 웹 페이지를 서비스하고자 할 때는 쓰레드들이 최소 주기에 들어가느냐를 검증해야 하지만, 이에 대한 연구는 다른 분야에서 많이 진행되고 있고 본 논문의 범위를 벗어나 생략한다.

3. VRML(Virtual Reality Modeling Language)

VRML은 가상현실 모델링에 사용하는 언어이다[7][8]. VRML의 출현은 인터넷에서 가상현실을 구현할 수 있는 기반을 제공하였고 스크립트 노드를 통하여 객체와 행위를 연결하기 때문에 동적인 화면 조작이 가능하다. 즉, VRML은 외부와의 상호작용을 통하여 화면 그림을 동적으로 구동할 수 있는 스크립트 노드를 제공한다. 스크립트 노드는 이벤트 흐름에서 논리 역할을 담당하는 노드로 주로 자바 스크립트로 이루어진다. 이들 노드는 라우트 문을 통하여 이벤트를 주고받으며 DEF 문으로 노드와 필드를 정의할 수 있다. DEF으로 정의된 노드만이 EAI 규약을 통하여 외부의 자바 애플릿과 데이터를 주고받을 수 있다.

본 논문에서 사용하는 테스트 예제는 5축 로봇 팔과 컨베이어 벨트로서 VRML에서 주어지는 노드와 필드를 사용한 모니터링 시스템이다. 여기서 로봇 팔의 동적인 부분인 1축에서 5축 부분은 인터폴레이터 노드를 사용하여 DEF 문으로 정의하였고 자바 스크립트를 사용하여 외부의 자바 애플릿과 통신하도록 구현하였다.

3차원을 사용한 이유는 실제 동작이 어떻게 일어나고 있는지를 2차원보다는 3차원이 더 효과적으로 알 수 있다는 장점 때문이다. 그러나, 3차원으로 웹으로 표현하는 것은 매우 시간이 소요되는 단점이 있다. 본 논문에서는 이러한 단점을 극복하기 위한 방법으로 현재 클론을 사용하여 시간을 줄여보았다.

4. EAI 규약

EAI(External Authoring Interface) 규약은 VRML와 외부환경과의 인터페이스를 정의한 것이다. VRML의 외부에서 자바를 통하여 VRML 화면을 조작할 수 있다[9]. 본 논문에서 제안된 구현 예제의 인터프리터로는 자바와 자바 스크립트를 사용하였다. VRML에 접근하기 전에 자바 애플릿은 사용할 브라우저 클래스 객체 참조(reference)를 얻어야 한다. 여기서의 브라우저 클래스는 자바 애플릿에 VRML를 투영하여 가상 세계에 접근하는데 사용될 수 있다. VRML에 접근은 getbrowser() method로 얻는다. VRML 파일에서 'DEF'로 정의한 특정 노드와의 통신은 getNode() 메소드를 사용하여 노드의 참조를 얻는다. getNode() 메소드는 루트 VRML 화면의 한 노드에 대한 이름을 스트링 값으로 전달받는다. 일단 한 노

드의 참조를 얻으면, 각각의 이벤트는 VRML의 evenIns로 전달되고 eventOuts의 현재 값을 얻을 수 있다. 자바 애플릿은 서버와의 통신을 통하여 물체의 동작 좌표 데이터를 받는다. 이렇게 얻어진 동작 좌표로 동적 위치의 이벤트 값들을 eventIns를 통하여 VRML 화면에 반영하고, 이러한 반영 동작을 반복하여 구현된 VRML 화면상의 컨베이어 벨트 위의 물체를 서버에서 전송한 위치 및 속도 값에 따라 동적으로 움직이게 할 수 있다.

Ⅲ. 웹 기반 실시간 모니터링 시스템의 구조

이 장에서는 본 논문에서 제안하는 웹 모니터링 시스템의 효율성을 보장하기 위한 서버의 최적 구조에 대하여 기술한다. 이러한 최적 서버의 구조는 데이터의 시간 및 공간적 일치성과 속도를 높이기 위하여 쓰레드의 생성을 제한할 수 있도록 한 구조이다.

일반적인 웹 서비스의 형태로 CGI 및 자바 서블릿를 이용한 모니터링의 예를 그림 1에 보였다. 그림 1에서 보듯이 일반적인 CGI 및 자바 서블릿를 이용한 모니터링에서는, 각각 다른 모니터링 페이지를 요구하는 3개의 클라이언트가 문서를 요청하게 되면 웹 서버는 프로그램에 필요한 인수를 전달하며 각각의 클라이언트 요청에 대한 모니터링 서비스 프로세스와 DB 관리자 프로세스를 생성 및 실행하여 DB로부터 데이터를 가져와 클라이언트에게 전송한다. 하지만 그림 1과 같이 모니터링을 요구하는 클라이언트의 접속이 늘어날 경우 모니터링 데이터는 빠르고 동시적으로 클라이언트에게로 전송될 수가 없다. 또한 서버의 메모리와 같은 자원을 많이 소모할 수 있고, 데이터 일치성이 깨어질 수 있기 때문에 여러 클라이언트가 동시에 같은 페이지를 보더라도 다른 데이터가 표시되기 때문에 모니터링의 정확성을 보장할 수 없게 된다.

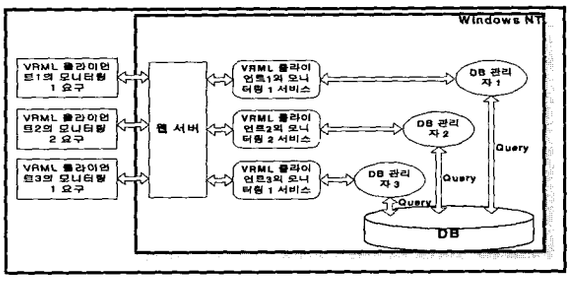

그림 1. CGI/서블릿을 사용한 모니터링 시스템 구조.

Fig. 1. the diagram of CGI/Servlet-based monitoring system.

본 논문에서는 이러한 기존의 CGI 혹은 서블릿 방식을 이용한 모니터링 방법에서의 프로세스 중복 수행을 막고 메모리 등의 자원을 효율적으로 관리하는 구조를 그림 2와 그림 3에 나타내었다. 실제의 모니터링 환경은 필요한 모니터링의 종류가 다양할 수 있고, 각기 다른 모니터링을 원하는 여러 개의 클라이언트 요구에 빠르게 응답하여야 한다. 이를 위하여 본 논문에서 제안한 모니터링 시스템에서는 쓰레드가 공통으로 참조하는 자원을 서블릿이 미리 인지하여 공통으로 필요한 자원이 메모리로 로딩 되어 있을 경우에는 자원을 다시 할당하지 않고 공통 메모리(common memory)로 접근한 후 데이터를 읽어서 웹 서버를 통하여 클라이언트로 전송하는 방법을 사용한다. 물론 하나의 클라이언트가 새로운 자원을 엑세스할 때는 그에 대한 자원을 새롭게 할당한다.

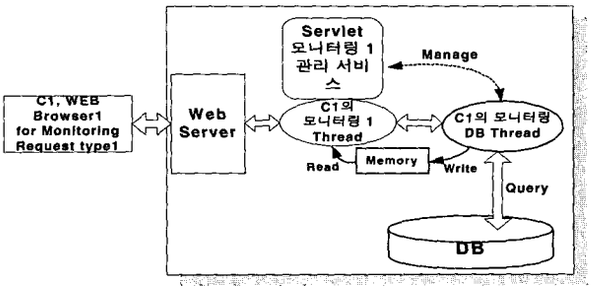

그림 2. 제안된 모델에서의 단독 클라이언트 접속시 구성.

Fig. 2. The configuration on access of a single client in the proposed model.

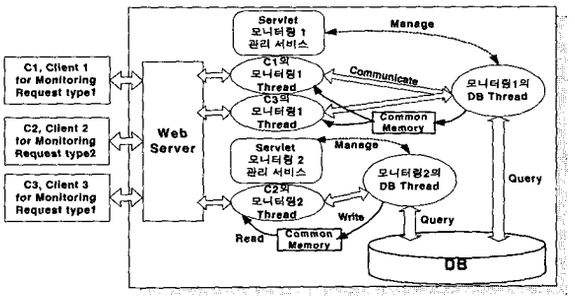

그림 3. 제안된 모델에서의 멀티 클라이언트의 접속시 구성.

Fig. 3. The configuration on access of multi clients in the proposed model.

여러 개의 각기 다른 모니터링을 원하는 클라이언트의 요구에 빠르게 응답하려면 쓰레드들이 공통으로 참조하는 자원을 서블릿이 미리 인지하여 공통자원이 메모리로 로딩되어 있을 경우 다시 그러한 자원을 할당할 필요 없이 공통 메모리로 접근한 후 데이터를 읽어 웹서버를 통해 클라이언트로 보내주기만 하면 된다. 물론 없을 경우에는 새롭게 할당하여야 한다. 그림 2와 그림 3은 각각 단독 클라이언트 및 다중 클라이언트가 서버에 접속할 때 서버에서 서블릿 및 각 쓰레드가 어떻게 구성되는지를 설명한 것이다. 그림 3에서 먼저, 클라이언트 1의 모니터링 요구 1을 웹 서버가 수신하면, 웹 서버는 모니터링 요구를 서비스하기 위하여 서블릿 모니터링 1 (관리) 서비스를 구동한다. 서블릿 모니터링 서비스는 하나의 DB 쓰레드와 클라이언트의 각 요구에 따른 개별 쓰레드를 생성하여 각 클라이언트에 데이터를 주기적으로 전송하게 한다. 또한 DB 쓰레드는 공통 메모리를 생성하고, 이 공통 메모리의 주소를 생성된

C1(클라이언트1) 쓰레드가 참조할 수 있게끔 등록한다. DB 쓰레드는 주기적으로 모니터링 1 요구에 해당되는 데이터를 DB로부터 주기적으로 읽어 공통 메모리에 저장한다. 그러면, 생성된 C1 쓰레드에서 주기적으로 할당된 공통 메모리로부터 저장된 데이터를 읽어 클라이언트 1로 전송한다. 물론 모니터링 요구 1에 대한 주기는 미리 설정해 놓은 것이다. 두 번째 클라이언트 2의 모니터링 요구 2가 들어오면 서블릿 모니터링 2 서비스를 구동하고 DB 쓰레드와 모니터링 2 쓰레드를 구동하고, DB 쓰레드는 공통 메모리를 생성한다. 클라이언트로 데이터를 전송하기 위한 C2(클라이언트2)의 쓰레드가 생성되고, 모니터링 2의 DB 쓰레드가 주기적으로 공통 메모리에 지정된 데이터를 저장하고, C2의 쓰레드가 주기적으로 C2에 해당되는 공통메모리로부터 읽어 클라이언트 2로 읽은 데이터를 보낸다. 세 번째로 들어오는 C3(Client3)의 요구는 C1과 동일한 모니터링의 종류이기 때문에 모니터링 1 서비스가 이를 인지하여 C3의 쓰레드만을 생성한다. C3에 데이터 전송은 이미 할당된 공통 메모리의 데이터를 읽어서 이루어진다. 모니터링 서비스들은 클라이언트로부터 들어오는 모니터링 요구를 조사하여 이미 수행되고 있는 모니터링 작업인 경우 DB 접근 쓰레드의 생성을 억제하고 공통 메모리의 데이터를 이용하여 수행되도록 하는 것이다. 이와 같은 방법은 네트웍에 분산되어 실시되는 모니터링 작업에서 발생하는 서버로의 과중한 접속으로 인한 부하의 증가를 막을 수 있다. 또한 공통 메모리를 통한 데이터의 접근은 동일한 시스템을 모니터링 하고자 하는 여러 클라이언트에게 시간적 일치성을 가진 데이터를 제공할 뿐 아니라 자원의 공유를 통한 시스템의 효율적 이용도 가능한 장점이 있다. 즉, 하나의 DB 쓰레드만을 이용하여 주기적으로 DB로부터 데이터를 읽어와 공통 메모리에 저장하고, 각 클라이언트 모니터링 쓰레드는 공통 메모리에서만 데이터를 읽어와 클라이언트로 분배하기 때문에 주어진 순간에는 모든 클라이언트의 데이터가 같게 된다. 어떤 클라이언트라도 처음으로 해당 모니터링 웹 페이지를 요구할 때만 서블릿 모니터링 관리 서비스와 DB 쓰레드가 생성된다. 서블릿 모니터링 관리 서비스는 클라이언트 쓰레드가 모두 없어졌을 때만 DB 쓰레드 및 자신을 삭제하기 위하여 사용되기 때문에 보통의 경우는 거의 실행에 시간이 소모되지 않는다. 따라서, 그림 2와 그림 3을 그림 1과 비교하면 구조상으로는 좀 더 복잡하게 보일지 모르지만, 서블릿 모니터링 관리 서비스는 매우 단순한 동작을 한다.

클라이언트 모니터링 쓰레드는 주기적으로 데이터를 각 클라이언트로 서버 밀기(push) 기술을 사용하여 전송하기 때문에 어느 정도 주기성을 가지고 데이터가 갱신하게 된다. 제안한 모니터링 시스템의 수행 과정을 흐름도로 그림 4에 표시하였다. 여기서는 하나의 클라이언트에서 모니터링 요구가 발생되고 이를 서비스하기 위하여 모니터링 서비스의 구동 알고리즘과 이에 대응하는 쓰레드의 생성에서 종료까지의 과정을 나타낸 것이다. 그림 4에서 20개의 클라이언트는 조정 가능한 값으로, 이 예에서는 20개의 클라이언트를 지원한다는 것이다. 지원하는 클라이언트 개수는 수행 속도에 영향을 주기 때문에 서버의 속도, 참조하고자 하는 페이지 수 및 클라이언트의 수를 참조하여 알맞게 결정해야 한다. 즉, 쓰레드의 수행 시간이 짧은 경우에도 쓰레드의 수가 증가하면 하나의 쓰레드를 수행한 후 그 쓰레드가 다시 수행될 때까지는 긴 시간동안 기다려야 한다. 따라서 쓰레드 증가에 따른 쓰레드의 대기 시간의 증가는 응답 시간에 큰 영향을 주게 되므로 모니터링 시스템과 같이 실시간 서비스 보장이 필요한 응용에서는 생성되는 쓰레드 수를 제한하는 것이다.

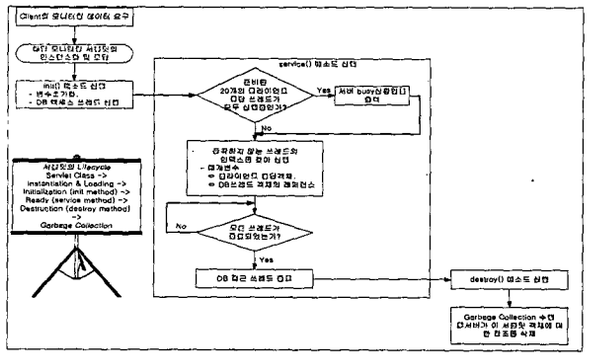

그림 4. 모니터링 흐름도.
Fig. 4. Flowchart of monitoring.

일반적인 모니터링 시스템의 구현에서는 한 화면에 표시 가능한 데이터들을 하나의 집합으로 취급하여 처리하고 있다. 예를 들어, 클라이언트 C1은 데이터 집합 M1, 클라이언트 C2는 데이터 집합 M2, 클라이언트 C3는 데이터 집합 M3를 사용하는 모니터링 화면과 관련되고 클라이언트 C4와 C5는 각각 M1, M2 및 M2, M3를 사용한다고 가정하자. 그러면, 클라이언트 C4는 M1과 M2의 합집합을 사용한다고 볼 수 있다. 이러한 합집합의 처리 방법에는 합집합을 하나의 다른 개별 집합으로 간주하여 구현하는 방법과 합집합을 구성하는 개별 집합을 사용하여 구현하는 방법이 있다. 예를 들면, 전자의 방법은 화면에 표시하는 집합을 M4(= $M1 \cup M2$)로 간주하며, 후자의 방법에서는 M1과 M2를 개별적으로 가져오는 것이다. 전자의 경우에는 서버에 다섯 개의 서블릿이 만들어지고 후자는 세 개의 서블릿이 만들어진다. 전자의 방식에서는 최악의 경우 즉, 모든 클라이언트가 서로 다른 데이터 집합을 요구하는 경우에 서버에서 생성되는 타스크(task)의 개수가 CGI 방식과 비슷해질 수 있다. 그러나, 타스크의 수행시간을 최적화하였기 때문에 CGI 방식보다는 수행 시간이 짧아질 수 있다. 후자의 방식에서는 서블릿이 각 클라이언트의 IP 주소를 관리해야 하고 모든 등록된 클라이언트에 해당 데이터를 보내야 한다. 또한 클라이언트에서도 여러 개의 서블릿이 보내는 데이터를 받아야 한다. 즉, 후자의 방식이 전자에 비하여 구현이 복잡하고 애플릿이 데이터 집합에 따라 다른 채널을 열기 때문에 유지 보수 측면에서 불리하다고 할 수 있다. 하지만 두 가지 방식 모두 동일한 데이터 집합을 요구하는 클라이언트 수의 증가에 따라 서블릿이 증가하는

것은 아니다. 이러한 점에서 기존의 CGI 방식과는 차이를 가진다.

이상에서 서버의 구조를 설명하였다. 다음에서는 이러한 서버에 대응하는 클라이언트 응용을 설명한다. 여기서 클라이언트는 화면을 3차원으로 보여주는 VRML을 이용하여 구현한 것이다. 클라이언트가 서버로부터 모니터링 데이터를 수신하여 3차원으로 표현하기 위해서는 먼저 3차원으로 표현된 모델링 데이터와 애니메이션 동작에 사용되는 움직임(motion) 데이터와 모니터링 데이터가 필요하다. 움직임 데이터는 클라이언트가 웹 서버에 처음 접속할 때 필요하며, 모니터링 데이터는 클라이언트의 동작 중에 서버가 계속해서 전송해준다. 본 논문의 모니터링 시스템의 클라이언트 응용 프로그램은 VRML로 구현한 것으로 움직임 데이터를 반영하여 물체의 움직임이 표현되도록 하고 자바 애플릿을 사용하여 VRML의 VRMLScript와 자바 스크립트로 해당 데이터를 전송하여 움직임을 처리하였다. 전체적인 처리 과정은 그림 5와 같다. 클라이언트에서 VRML과 자바 애플릿을 서버에서 로딩한 후에 자바 애플릿과 VRML 클라이언트 사이의 구조를 그림 6에 나타내었다. 여기서 Scene 그래프 노드와 스크립트 노드는 정적인 노드를 말하며 Active Scene Graph의 정의된 노드와 정의된 Active VRMLScript 노드는 인터페이스를 통하여 외부로부터 모니터링 데이터를 수신하여 동작하는 노드이다. 애플릿과 VRMLScript 사이에 직접적인 통신은 없고 "URL" 구문으로 명시된 클래스를 로딩하여 getNode 메소드를 사용하여 공유 데이터의 참조(reference)를 얻어 서로의 정보를 교환하게 하였다[10]. 전자는 애플릿으로부터 데이터를 받기만 하지만, 후자는 애플릿과 서로 데이터를 교환하여 VRML에서 생기는 이벤트 등을 보내어 애플릿에서 처리할 수 있다.

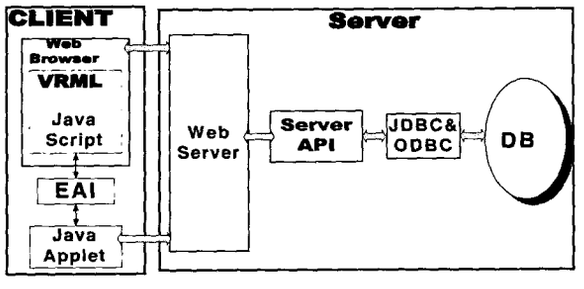

그림 5. 웹 모니터링의 전체 구성도.
Fig. 5. The diagram of web monitoring system.

Ⅳ. 평가

1. 테스트베드(test-bed)

본 논문에서 제안한 방법이 유효한가를 확인하기 위해 그림 6과 같은 VRML 모델을 여러 모델을 구현하여 시험하였다. VRML 브라우저를 웹 브라우저에 내장(embedding)하여 컨베이어 벨트 위에서 물체가 이동하는 모습으로 표현하였다. 물체의 이동은 데이터베이스로부터 전송되는 움직임 데이터에 따라 이동거리를 계산하여 반영하였다. 문자로 표시되는 숫자는 이동한 데이터를 나타내며 계속해서 변경되는 것이다. 움직임 데이터는 서버로부터 계속해서 서버 밀기를 사용하여 클라이언트로 전송되고, 이를 바탕으로 물체는 연속적으로 이동한다. 성능지표를 구하기 위해 서버에서 데이터는 주기적으로 보내지 않고 연속적으로 보내는 방법을 사용하였다.

그림 6. 모니터링 시스템의 구현 예제.
Fig. 6. Example of monitoring system.

본 논문에서 제안한 웹 모니터링 시스템을 평가하기 위한 테스트 시스템의 구성도가 그림 7에 있으며, 그림 6과 같은 VRML 모델을 화면에 표시하기 위하여 여러 개의 클라이언트가 사용되었다. 여기서 임의의 클라이언트(A, B, C,)는 웹 브라우저를 이용하여 VRML 모델 1을 모니터링하고, 또 다른 클라이언트(클라이언트 Z)는 기기 2를 모니터링하고 있다. 서버에서는 VRML 모델 1의 모니터링에 사용되는 서블릿과, VRML 모델 2의 모니터링에 사용되는 서블릿을 생성하여 개별 클라이언트의 요구에 대응하는 서비스를 제공한다.

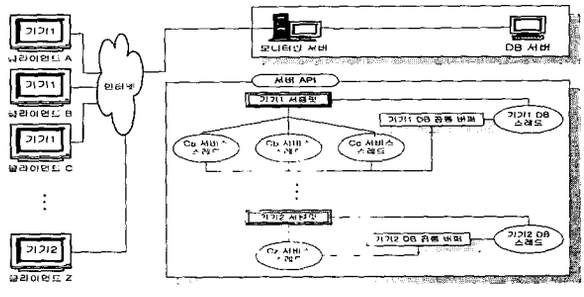

그림 7. 모의 테스트 네트웍 구조.
Fig. 7. The network architecture of test-bed.

제안된 모니터링 시스템에서 사용한 테스트는 공통적으로 클라이언트의 개수를 달리하면서 다음과 같은 두 가지 경우에서 수행하였다. 모든 클라이언트가 동일한 VRML 모델을 모니터링하는 경우와 클라이언트들이 서로 다른 VRML 모델을 동시에 모니터링하는 경우이다. 테스트에서는 사용되는 데이터는 각 VRML 모델이 모니터링 과정 중에 사용할 float형의 데이터로 DB에 저장하였다고 가정하였다. 클라이언트가 특정 VRML 모델의 모니터링을 요구하면 모니터링 서버는 DB로 질의하여 해당 VRML 모델에 대응하는 데이터를 요구 클라이언트로 전송한다. 이러한 서비스를 10000번 반복하여 수행하고, 클라이언트 쪽의 애플릿에서 매번 서버가 전송하는 모니터링

데이터의 도착시간을 기록하여 그 평균시간을 구하였다. 여기서, 모니터링 데이터의 도착 평균시간이 의미는 다음과 같다. 설명의 편의를 위하여 두 가지의 시간만을 고려한다. 첫 번째는 초기에 임의의 클라이언트가 모니터링 요구를 하는데 걸리는 시간, t_{req}이고 두 번째는 모니터링 서버가 모니터링 데이터를 DB 서버로 질의하여 구하고 이를 클라이언트에게 전송할 때까지의 시간, t_{res} 혹은 응답시간이다. 테스트에서 측정한 시간은 t_{res}로 하였다. 이에 대한 내용이 그림 8에 있다.

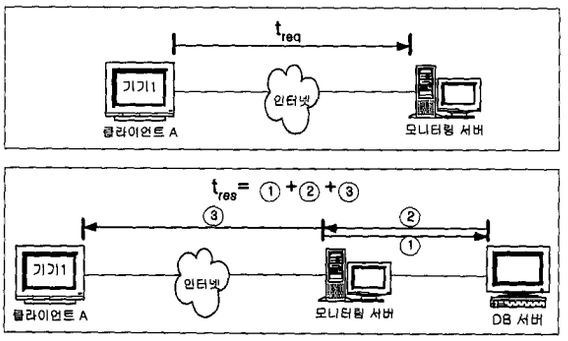

그림 8. 요구시간, t_{res}와 응답시간, t_{res}.
Fig. 8. t_{req}(request time) and t_{res}(response time).

2. 테스트 결과

테스트로부터 얻어진 응답시간 및 응답 시간의 편차를 그림 9에 나타내었다. 여기서 표시된 값은 5개의 클라이언트가 서로 다른 다섯 종류의 VRML 모델을 모니터링하는 경우와 각각 6, 7, 8, 9개의 클라이언트가 각각 다섯 종류의 VRML 모델을 모니터링하는 경우(이 경우에 중복해서 모니터링 되는 VRML 모델은 한 종류로 하였다.)로 나누어 측정한 것이다. 측정 결과에서 모니터링 응답시간이 클라이언트의 수와 무관하게 고른 분포를 보였다. 이러한 결과는 모니터링 응답시간은 서버측에서 새로운 클라이언트에 대한 쓰레드나 서블릿 개체(instance)를 생성하는 시간보다는 테스트하는 네트워크 환경의 트래픽이 더 큰 영향을 주는 것으로 생각할 수 있다. 이러한 결론은 응답 시간의 편차가 클라이언트의 개수가 9개일 때 가장 큰 것에서도 알 수 있다.

본 논문에서 제안하는 구조와 일반적으로 서블릿이 지원하는 방식을 비교한 결과를 표 1과 표 2에 보였다. 응답시간을 비교하기 위하여 사용한 테스트 환경은 대학 내의 실험실 컴퓨터 8대(서버 포함)와 대학 내의 다른 지역에 있는 2대의 컴퓨터를 이용하여 수행하였다. 실험실 내의

표 1. 테스트 서버 및 허브 사양.
Table 1. Test server/Hub specification.

구분	사양
CPU	Intel Pentium II-266MHz
RAM	SDRAM 128M(32M*4 or 64*2)
HUB	3COM 8Port or 12Port

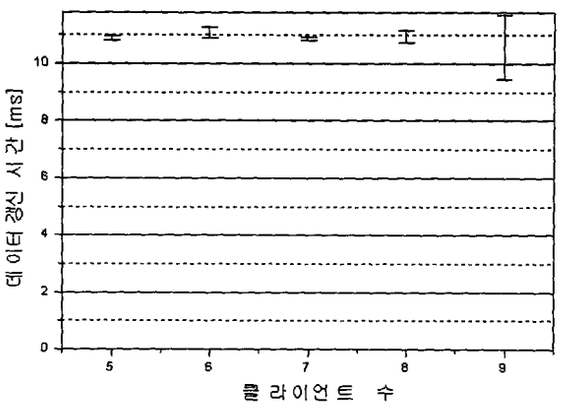

그림 9. 테스트 결과.
Fig. 9. Test result.

컴퓨터는 더미 허브를 통하여 10Mbps의 이더넷 망에 접속된 것이고 다른 지역은 ATM 스위치를 사용하여 접속한 네트워크 형태이다. 테스트는 밤 시간에 이루어졌다. 테스트에 사용한 모니터링 서버와 더미 허브의 사양은 표 1에 보였다.

모니터링을 요구할 수 있는 클라이언트가 9개이고 이들 클라이언트가 서버에 모니터링을 요구하여 얻어지는 결과를 보여주는 독립된 모니터링 화면의 수는 최고 다섯 개까지로 하였다. 표 2와 3의 값을 비교하면 제안된 구조에서의 결과 값이 일반적인 서블릿 구조에 비하여 약 1.5배정도 수행 속도가 빠른 것을 볼 수 있다. 표 2와 3을 그림 10에 그래프로 나타내었다. 일반적인 서블릿 방식의 응답 시간은 클라이언트의 수가 증가하면서 서버의 응답 시간도 비례하여 증가하는 모습을 보인다. 하지만 제안된 방식의 결과는 클라이언트 개수 증가에 따른 응답 시간의 증가가 거의 없는 것으로 나타난다. 즉, 일반적인 서블릿 방식의 경우는 클라이언트마다 쓰레드를 생성하고 이에 대한 데이터를 전송해야 하기 때문에 클라이언트의 수에 응답 시간이 비례하여 증가하지만, 본 논문에서 제안한 방식에서는 요구가 처음 발생하여 이에 대한 응답으로 쓰레드가 생성되는데 시간이 걸리지만 그 후에 요구에 대한 쓰레드 생성이 억제되기 때문에 클라이언트 증가에 따른 응답 시간의 증가는 거의 없는 것을 볼 수 있다. 따라서 제안된 방식이 좀 더 빠르고 안정되게 데이터를 전송 할 수 있다는 것을 알 수 있다.

이러한 결과는 본 논문에서 제안하는 구조의 특성 때문에 가능한 것이다. 즉, CGI 및 서블릿을 이용한 기존의 방식은 DB를 엑세스하는 것이 포함되어 있기 때문에 프로세스의 수행 시간이 본 논문에서 제안하는 구조에서 사용되는 쓰레드 보다 좀 더 길어진다. 예를 들어, 기존의 방식에서는 증가된 수행 시간에 쓰레드의 수가 곱해지기 때문에 쓰레드 수가 증가할수록 응답 시간이 길어지는 것이고 본 논문에서 제안하는 방법은 하나의 DB 쓰레드가 공통 메모리를 갱신하면 클라이언트에 대응되는 모니터링 쓰레드가 메모리에서 읽어가기 때문에 응답 시간에 영향이 적다. 이러한 결과는 표 3과 그림 10에 보였다.

표 2. 제안된 구조의 평균 응답 시간.
Table 2. The mean response time of proposed multi-thread model.

클라이언트 수	모니터링할 VRML 모델 수				
	1	2	3	4	5
1	10.83	-	-	-	-
2	11.01	10.96	-	-	-
3	11.00	10.97	10.92	-	-
4	10.98	10.94	10.89	10.89	-
5	10.96	10.95	10.89	10.88	10.9
6	10.97	10.93	10.85	10.89	10.09
7	10.94	10.84	10.85	10.87	10.86
8	10.95	10.90	10.9	10.92	10.97
9	10.93	10.92	10.84	10.89	10.60

표 3. CGI 기반 구조의 평균응답 시간.
Table 3. The mean response time of CGI- based model.

클라이언트 수	모니터링할 VRML 모델 수				
	1	2	3	4	5
1	14.11	-	-	-	-
2	16.23	15.87	-	-	-
3	16.65	16.52	15.21	-	-
4	15.38	16.29	16.09	17.38	-
5	17.23	16.47	17.21	16.29	17.03
6	16.22	16.62	17.12	17.56	17.21
7	17.76	18.17	17.78	17.22	16.39
8	19.08	18.21	15.13	16.73	18.01
9	18.94	17.33	16.86	19.01	16.19

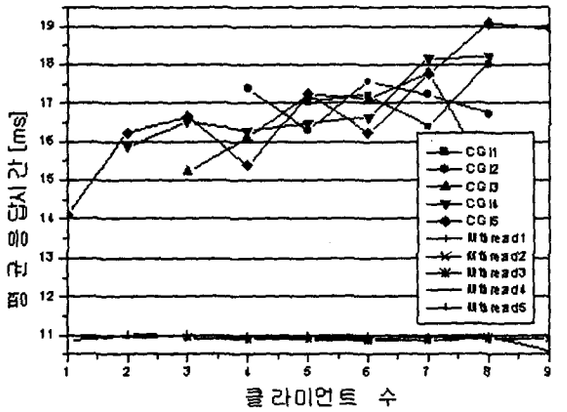

그림 10. 서블릿 방식과 제안된 방식의 평균응답시간.
Fig. 10. The mean response time of proposed multi thread model and Servlet-based model.

제안한 모니터링 시스템 구조와 VRML 기반 3차원 형상을 표시하도록 구성된 웹 브라우저를 이용하여 구현한 시스템이 그림 11에 있다. 그림 11은 DB로부터 각 포머, 롤링 머신과 같은 각 종 기계 등의 동작 및 상태 데이터를 받아 동작 혹은 상태를 표시하도록 구현된 3차원 모니터링 시스템의 예이다. 실제로 공장 시뮬레이션 프로세스가 DB에 데이터를 저장하는 경우에 적용을 하여 시험을 한 결과, 3차원으로 잘 동작함을 알 수 있었다.

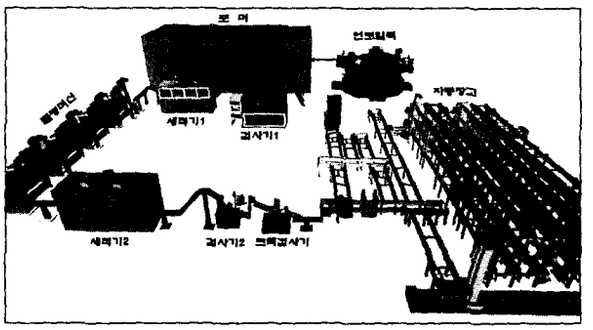

그림 11. 3차원 모니터링 시스템의 예.
Fig. 11. Three dimensional monitoring system.

V. 결론

본 논문에서는 기존의 2차원 모니터링 방법에 비하여 사용자 친화적인 환경(인터페이스, 비쥬얼한 화면 등)을 제공할 수 있는 VRML 기반 3차원 형상으로 표시할 수 있도록 하며, 기존의 CGI 및 서블릿 방식보다 빠르고 안정된 응답 시간을 제공하는 실시간 모니터링 시스템의 향상된 구조를 제안하였다. 또한 제안한 웹 모니터링 모델을 구현하여 테스트하고 응답 시간을 측정하여 성능을 검증하였다. 제안한 모니터링 시스템 모델은 기존의 방식보다 빠르고 안정화된 응답 속도를 보였고, 모니터링을 요구하는 클라이언트 수의 증가에 관계없이 안정된 응답 시간을 보였다. 따라서 본 논문에서 제안한 모델을 사용하면 모니터링 시스템에 필요한 연성 실시간성의 보장이 어느 정도 가능하다. 또한 제안된 모니터링 모델로 모니터링 각 화면에 표시되는 데이터들의 시간적 일관성 보장이 가능함을 알 수 있었다. 이러한 것이 가능한 이유는 제안된 모델이 서버 측면에서는 클라이언트의 수에 비례하여 쓰레드를 생성하지 않고 한 종류의 모니터링 페이지를 참조하는 클라이언트들로 전송되는 데이터는 하나의 공통메모리로부터 가져오게 하며 주기를 제어하는 것이 하나의 관리 쓰레드가 하도록 하였기 때문이다.

앞으로 좀 더 연구가 이루어져야 하는 부분은 인터넷 네트워크에서의 실시간 데이터 송수신을 위한 환경의 구성과 사용자에 대한 권한을 제한하는 보안문제 등에 연구가 이루어져야 할 것이다.

참고문헌

[1] *Intellution Web Site "http://www.intellution. com"*, 1998.
[2] Wonderware Web Site, "http://www.wonderware. com".
[3] H. S. Park, H. D. Kim, H. S. Kim, K. H. Na, and

S. Choi, "Three dimensional WEB monitoring systems using java", *IECON'99*, pp. 1233-1239, 1999.
[4] Genusa 저/정우경 역, "Using ISAPI", pp. 10-89, 1997.
[5] Laura Lemay & Rogers Cadenhead 공저, "JAVA 1.2", 1998.
[6] Netscape Web Site "An exploration of dynamic documents" http://home.netscape.com/assist/netsites · /pushpull.html, 1998.
[7] Chris Marrin & Bruce Campbell 저, 이상영 역, "초보자를 위한 VRML2 21일 완성", 1997.
[8] Stephen N. Matsuba&Bernie Roehl 저, 황태연, 장은지역, "Using VRML", 1997.
[9] "External authoring interface spec" http://www.web3d.org/WorkingGroups/vrml-eai/, Jan., 1999.
[10] Robert Orfali & Dan Harkey, "Client/Server programming with JAVA and CORBA", pp. 255-280, 1998.

박 홍 성

1983년 서울대 제어계측공학과 (학사). 1986년 동대학원 (석사). 1992 동대학원 (박사). 1983~1990 삼성전자 근무. 1992~현재 강원대 전기전자정보통신공학부 부교수. 관심분야 실시간 네트워크, 개방형 구조제어시스템, 이산현상시스템, 블루투스, 무선 데이터 통신 해석, 프로토콜 설계 및 분석.

김 봉 선

1999년 2월 삼척산업대 제어계측공학과 (학사). 2001년 2월 강원대 제어계측공학과 (석사). 2001년 6월 현재 성지인터넷(주) 근무.

정 명 순

1989년 강원대학교 전자공학과 (학사). 1992년 강원대학교 전기공학과 (석사). 1999년 강원대학교 전자공학과 (박사). 1999~현재 강원대학교 시간강사. 관심분야 실시간 멀티미디어 통신 서비스, Bluetooth.