論文 98-23-9-32

웹 기반 원격 계측 제어를 위한 실시간 웹 서버 구조

정회원 문 재 철*, 이 명 진*, 강 순 주*

A Real-Time Web-Server Architecture for Web-based Telemetric & Teleoperation

Jae-Chul Moon*, Myung-Jin Lee*, Soon-Ju Kang* *Regular Members*

*본 연구는 정보통신연구관리단의 1998년도 대학기초 연구지원사업의 연구비 지원에 의한 결과입니다.

요 약

인터넷에서 웹(World Wide Web) 기술을 적용한 원격 계측 제어 시스템 구축은 복잡한 응용환경임에도 불구하고 사용자의 거리제한을 쉽게 극복할 수 있다는 특징 때문에 매우 주목받고 있는 기술이다. 그러나 단순 정보 검색 서비스를 위해서 설계된 기존의 웹 기술로써는 계측 제어 응용환경에 필수요건인 실시간 처리 기능 구현이 불가능하다. 본 논문에서는 웹 기반 원격 계측 제어 시스템들의 일반적인 특징을 기반으로 하여 태스크 동기화 및 우선순위 처리와 같은 실시간 처리기능이 첨가된 새로운 개념의 내장형 실시간 웹 서버의 소프트웨어 구조를 제안하고, 프로토타입을 구현하여 그 성능을 검증하여 보았다.

ABSTRACT

Web-based telemetrics & teleoperations have attracted great interests due to its easiness and cost-benefit to development and operation of system. These telemetric & teleoperation systems require a specially designed web-server that supports real-time processing abilities such as task synchronization and priority scheduling of tasks. However most of current web-servers lack these abilities, only allowing remote information browsing. In this paper, we propose a web-server that provides the required abilities of real-time processing for supporting web-based telemetrics and teleoperations. This has been carried out by analyzing the general needs for the application domain and designing a domain-specific software architecture for the real-time web-server.

I. 서 론

원격 계측 제어 시스템은 사용자가 작업지에 직접 가지 않고 원격에서 작업을 가능하게 하는 매우 유용한 개념이나 일반적인 계측 제어 시스템에서 필요하지 않던 네트웍에서의 지연이나 네트웍과의 인터페이스 등의 고려가 필요하다. 이로 인해 특수한 장비들이 추가적으로 필요하여 시스템의 구현상에 기술적 어려움이 따르고, 개발비용이 높아질 뿐만 아니라 사용자의 이동성에 제한이 많아 실용화에 걸림돌이 되었다.

그러나, 최근에 인터넷 혹은 인트라넷의 대중화에 따라 인터넷 설비를 이용하는 연구가 시도되고 있

* 경북대학교 전자전기공학부
論文番號:98031-0119
接受日字:1997年 4月 29日

다. 특히 월드 와이드 웹(World Wide Web, 축약하여 웹이라고함)의 대중화로 인해서 원격 계측 제어 서비스가 인터넷에 연결만 되면 어디에서나 웹 브라우저(browser)만을 이용하여 원격 작업이 가능하고, 또한 이미 데이터 베이스등의 관련 자원들의 웹 연동 기술이 보편화되어 복잡한 기능의 원격 계측 제어 시스템의 설계 및 구현도 가능하다. 이런 장점을 이용한 원격 계측 제어 시스템의 연구가 근래에 활발히 시도되고 있다[1,2,3,4,5,6].

그러나 지금까지 시도된 시스템들은 일반적인 웹 서비스를 위해 고안된 기존의 웹 서버를 그대로 사용한다. 이로 인해 태스크간의 우선순위 관리, 통신 그리고 동기 등의 실시간 요건들을 만족시키기가 쉽지 않으며 기존의 웹 서버가 요구응답의 동기식 처리 방식을 이용함으로 비동기식 처리가 중심이 되는 원격 계측 제어 시스템에 합당하지 않는 단점들을 가지고 있다. 뿐만 아니라 계측 제어와 웹 서비스가 서로 분리되어 있어서 웹 서버와 계측 제어 시스템과의 즉각적인 연동 능력이 부족하다.

이러한 문제점을 해결하기 위해서 본 논문에서는 인터넷 혹은 인트라넷에서 원격 계측 제어 서비스 구현에 적합한 새로운 웹 서버의 소프트웨어 구조를 제안한다. 제안된 구조는 원격 계측 제어 시스템의 실시간 동작 특성을 보장하기 위해 태스크간의 우선순위를 결정할 수 있어 즉각적인 반응이 필요한 태스크를 우선적으로 처리할 수 있으며, 계측 제어 시스템에 합당하도록 비동기식 처리를 기반으로 하였고, 웹 서비스 태스크와 계측 제어 태스크를 하나의 구조로 통합하여 즉각적인 웹 연동이 가능하게 하였다.

본 논문에서는 1장에서 서론을 논하였고, 2장에서는 원격 계측 제어 시스템을 정의하여 원격 계측 제어용 웹 서버 설계에 필요한 요건들을 도출하고 기존의 웹 서버 구조와 문제점을 설명한다. 3장에서 제안된 새로운 웹 서버의 소프트웨어 구조를 제안하고 자세한 설명을 하겠다. 제안된 구조로 구현된 프로토타입의 성능 평가를 4장에서 설명하고 5장에서 결론을 맺는다.

II. 웹 기반 원격 계측 제어 시스템 분석

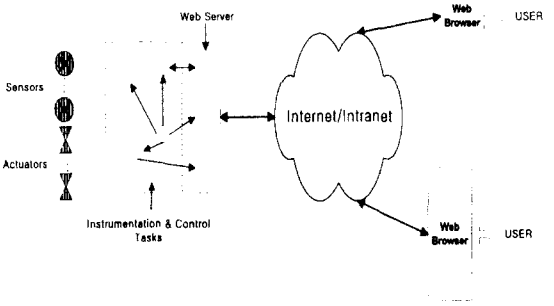

그림 1. 웹 기반 원격 계측 제어 시스템의 개요
Fig.1. Overview of web-based telemetric & teleoperation system.

웹을 통한 원격 계측 제어 작업에서 사용자는 웹 브라우저를 이용하여 그림 1에서와 같이 원격에 있는 센서(Sensor)와 작동기(Actuator)를 조작한다. 서론에서 밝혔듯이 사용자는 웹 브라우저만을 이용하여 작업이 가능함으로 인해 사용자의 이동성이 확대된다. 이렇게 웹 브라우저만을 이용하여 원격 계측 제어 작업을 가능하게 하는 시스템을 웹 기반 원격 계측 제어 시스템이라고 한다. 웹 기반 원격 계측 제어 시스템은 계측 제어 태스크(Instrumentation & Control Task)뿐만 아니라 그림 1에서 보듯이 웹 연동을 위한 웹 서버(Web Server)를 가지고 있어야 한다. 여기서 계측 및 제어 태스크는 사용자의 명령에 따라 작동기를 제어하거나 센서로부터 정보를 입수하여 처리하는 일을 담당한다. 그리고 웹 서버는 계측 제어가 웹을 통해서 일어나도록 계측 및 제어 정보를 HTML(Hyper-Text Markup Language)문서나 다른 웹 데이터 형태로 변환하여 웹 브라우저로 전달하는 기능을 담당한다.

1. 원격 계측 제어용 웹 서버의 필수요건

웹 기반 원격 계측 제어 시스템에서는 HTTP(Hyper-Text Transfer Protocol)을 이용한 서버와 사용자간에 통신뿐만 아니라 서버 내에 존재하는 각종 계측 제어 태스크와 빈번한 통신이 서버 시스템 내부에서 이루어진다. 따라서 원격 계측 제어용 웹 서버는 일반적인 상용 웹 서버와는 매우 다른 기능들이 필요하다.

요구되는 기능들은 우선 실시간 특성에 관계된 것과 원격에서의 시스템 조작을 지원하기 위한 분산처리에 관계된 것으로, 다음과 같이 나누어 설명할 수 있다.

- **이벤트 기반 처리**(event-driven processing) : 원격 계측 제어를 위한 웹 서버는 기본적으로 외부에서 비동기적으로 발생된 정보 즉 이벤트를 처리하는 구조가 필수적이다. 왜냐하면, 계측 제어 시스템은 센서, 작동기 및 타이머등의 하드웨어에서 발생하는 인터럽트의 처리를 통해서 외부의 상황을 인지하고 외부에 변화를 가한다. 따라서 이러한 비동기적 이벤트 처리를 초점에 둔 소프트웨어 구조가 요구된다.

- **동시 태스크**(concurrent task) : 원격 계측 제어 시스템 내에는 여러 개의 센서와 작동기로 구성되어 있으며 각각의 센서나 작동기는 시스템의 목적달성을 위해서 동시에 관리 또는 제어된다. 따라서 동시에 여러 개의 작업이 가능한 동시 태스크를 지원하는 구조가 필수적이다.

- **태스크간 통신과 동기화**(inter-task communication and synchronization) : 원격 계측 제어 시스템에서 원하는 작업 결과를 얻기 위해서는 동시에 수행되는 작업간의 통신 및 동기화가 절대적이다.

- **우선순위 기반 태스크 선취**(priority based task preemption) : 여러 개의 태스크가 수행되는 상태에서 각각의 태스크는 시간 제한 조건을 맞추어야 한다. 이때 소프트웨어는 우선 순위 기반의 선취를 지원하여 급한 태스크가 먼저 수행될 수 있도록 해야 한다. 이외에 네트웍을 통하여 이루어지는 작업임으로 인해 네트웍에서의 지연의 영향을 최소화하기 위한 요건으로 이벤트의 과다한 발생을 사전에 방지하기 위한 이벤트 필터링과 서로 연관된 이벤트끼리 통합시켜주는 이벤트 상관기능 등이 이벤트 전처리 과정에서 요구된다.

2. 관련 연구

기존에 시도되었던 웹 기반 계측 제어 시스템은 초기 시도이기 때문에 앞 절에서 언급한 기본 요건들을 대부분 반영하지 못했으며, 웹 인터페이스의 형태에 따라 CGI(Common Gateway Interface)[7]를 이용한 것, 자바 애플릿(Java applet)를 이용하는 것으로 구분할

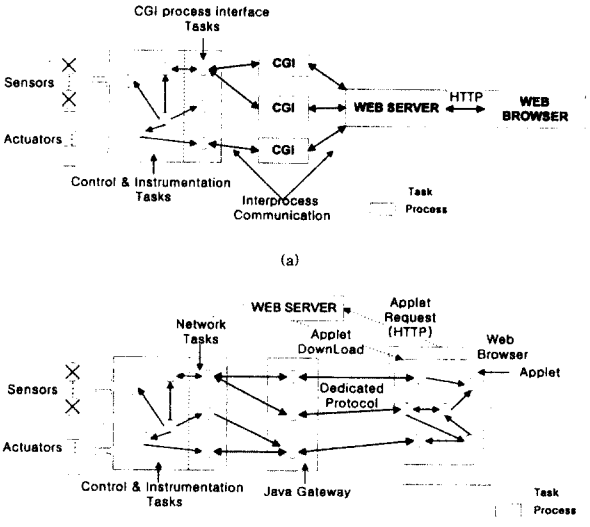

그림 2. 기존의 웹기반 원격 계측 제어 시스템 구조
(a) CGI 기반 시스템 (b) 자바 기반 시스템

Fig. 2. Architecture of web based telemetric & teleoperation system.
(a) CGI based system. (b) Java based system.

수 있다. CGI을 이용하는 시스템[1,2,3,4]은 그림 1의 경우처럼 웹 인터페이스 태스크가 CGI와 웹 서버인 경우이다. CGI를 이용한 시스템은 그림 2(a)에서 보듯이 웹 서비스를 위해 두 번의 프로세스간 통신(interprocess communication)이 필요하여 효율적인 서비스를 하지 못한다. 그리고 CGI 프로세스가 사용자 요구가 있을 경우에만 생성 작동하여 계측 제어 태스크에서 특정 CGI 프로세스로의 이벤트 발생이 불가능하다. 또한 CGI가 이벤트 처리 구조를 가지고 있지 않아서 이벤트 처리 기반으로 통일성 있게 전체 시스템을 설계할 수 없으며 CGI 프로세스의 우선 순위 관리가 쉽지 않다. 특히 CGI의 프로세스간 통신의 효율 문제로 인해서 대부분의 웹 서버는 CGI 이외의 웹 서버 확장 인터페이스를 제공한다. 웹 서버에서 제공하는 확장 인터페이스는 프로세스간의 통신, 프로세스 생성의 효율 문제를 해결하기 위해서 쓰레드로 수행되므로 CGI보다 빠른 반응이 가능하다. 그러나 그림 2(a)에서 보듯이 프로세스기반의 CGI를 쓰레드 기반의 확장 웹 인터페이스로 바꾼 것 이외에 구조적으로 다른 것이 없다. 따라서 CGI와 같이 이벤트

처리 기반 설계 및 우선 순위 관리에 문제를 가지고 있다.

Java[8]를 이용한 시스템[5,6]은 CGI 기반 시스템과는 다르게 그림 2(b)에서 보듯이 웹 서버는 단지 자바 애플릿을 전달하는 역할만을 한다. Java을 이용한 시스템의 경우 자바 애플릿이 웹 브라우저에 내장되는 특징을 이용하면 전체 시스템을 HTTP나 웹 서버의 기능에 제한 받지 않고 설계 구현이 가능하다. 그러나 자바 애플릿과의 통신을 위해 따로 프로토콜을 구현하거나 CORBA(Common Object Request Broker Architecture)[9]나 RMI(Remote Method Invocation)[10]와 같은 분산 객체 기술을 사용해야 한다. 이로 인해 시스템의 규모가 커지고 사용자 인터페이스를 위해서 HTML이나 HTTP를 이용하여 쉽게 해결 가능한 부분도 따로 설계해야 하는 단점을 가지고 있다. 뿐만 아니라 자바 애플릿의 보안에 관한 제한으로 웹 서버 호스트와만 통신 가능하다. 이로 인해 부가적으로 웹 서버 호스트 내에 계측 제어 태스크와 자바 애플릿하고 통신을 위한 게이트웨이(gateway)가 필요하다. 이와 같이 기존의 시스템은 앞 절에서 도출한 소프트웨어 요건을 만족시키지 못하거나, 이를 위해서 시스템 규모가 커지고 복잡해지는 단점을 가지고 있다.

III. 제안된 실시간 웹 서버의 구조

1. 시스템 개요

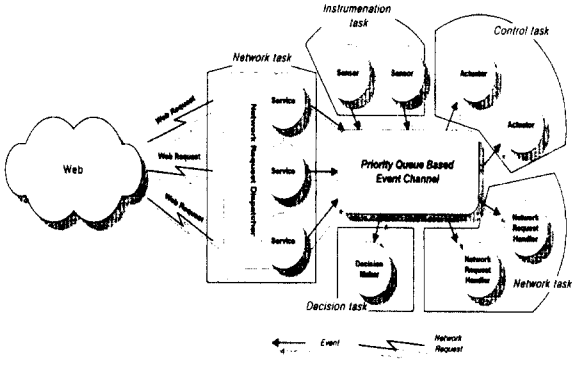

그림 3. 실시간 웹 서버 소프트웨어 전체 구조
Fig. 3. Overall architecture of proposed web-server software.

기존 웹 서버 기반의 원격 계측 제어 시스템의 문제를 해결하기 위해서 본 논문에서는 그림 3과 같은 구조를 제안한다. 제안된 구조에서는 그림 3에서 보듯이 계측 제어를 위한 태스크가 서비스 객체(service object)와 네트웍 요구 처리자(network request handler) 같은 웹 서버 태스크와 서버내에서 직접 연동이 가능하다. 이로 인해 웹 인터페이스를 위한 부수 계층이 필요없도록 설계하였다. 또한 통신을 위해서 CORBA의 이벤트 서비스 모델[11,12]을 응용한 우선순위 큐 기반의 이벤트 채널을 적용하여 실시간 이벤트 처리가 중심이 되는 응용 환경에 적합하도록 설계하였다.

네트웍으로 전달된 사용자 요구는 그림 3과 같이 서비스 객체와 네트웍 요구 처리자 두 부분으로 나누어 처리된다. 서비스 객체의 기능은 HTTP 명령을 해석하여 네트웍 요구 처리자로 이벤트를 발생시키고 네트웍 요구 처리자는 발생된 이벤트를 받아 웹 서비스를 한다. 네트웍 요구 처리자는 기존 시스템의 CGI와 웹 서버 확장 인터페이스에 해당하고, 그림 3에서 네트웍 요구 전달부(Network Request Dispatcher)는 HTTP요구를 서비스에게 전달하는 역할을 한다.

다음절에서는 우선 순위 큐 기반의 이벤트 채널 구조를 자세히 설명하고, 이어서 네트웍 요구 전달부와 제안된 구조에서 웹 서비스하는 방법에 대해서 자세히 설명하겠다.

2. 우선 순위 큐 기반의 이벤트 채널 구조

이벤트 전달 모델의 한 예로 CORBA의 객체 서비스 중 이벤트 서비스 모델이 있으며 제안된 웹 서버는 CORBA의 이벤트 서비스 모델에 정의되어 있는 이벤트 모델중 이벤트 채널 모델을 중심으로 설계되어 있다[11,12]. 그러나 CORBA에서 정의한 이벤트 채널 모델은 계측 제어 시스템과 같은 실시간 처리 특성을 고려하지 않았다. 이를 보안하여 웹 환경에서 실시간 처리에 적합하게 이벤트 필터(event filter), 이벤트 상관기(event correlator), 이벤트 스케줄(event schedule)기능을 첨가한 이벤트 채널 모델을 사용하였다.

그림 4는 제안한 이벤트 채널의 구조를 나타내며

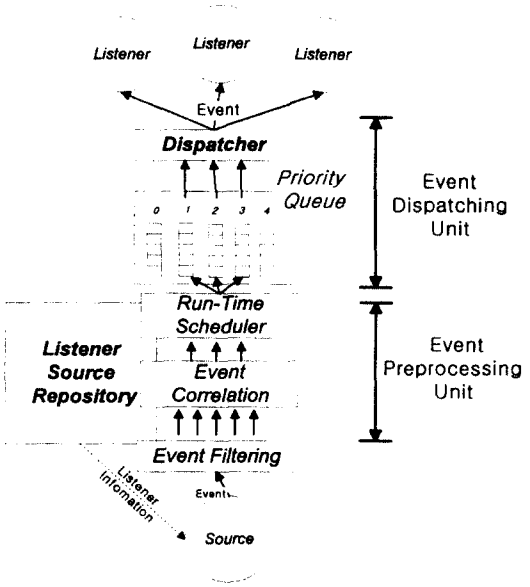

그림 4. 우선순위 큐 기반 이벤트 채널 구조
Fig. 4. Priority queue based event channel structure.

각 구성요소에 대한 간략한 설명은 다음과 같다.

- **이벤트**: 이벤트는 소프트웨어에서 비동기적으로 발생하는 정보를 말한다. 원격 계측 제어 시스템에서 비동기적인 정보로는 센서 정보, 타이머 등이 있다. 제안된 소프트웨어에서는 이벤트를 다음과 같은 클래스로 정의한다.

```
class Event {
  int sourceid;
  int groupid;
  int event_type;
}
```

Event::sourceid는 이벤트를 발생시킨 생성자의 번호를 나타내고 Event::groupid는 이벤트를 발생시킨 생성자가 속하는 생성자들의 그룹을 나타낸다. 그리고 Event::event_type은 발생한 이벤트를 실시간 처리할 것인가 아니면 실시간 처리를 하지 않을 것인가를 나타낸다. 실시간 처리를 할 경우는 우선 순위에 따른 선취가 가능하고 실시간 처리를 하지 않을 경우는 제일 낮은 우선순위를 가져 선취가 불가능하다.

- **생성자**(Source) : 센서나 타이머의 인터럽트에 따라 센서정보를 읽거나 다른 태스크의 이벤트에 반응하여 이벤트를 발생시키는 태스크이다. 이벤트를 발생시키는 태스크는 Event_Source::run()함수에 프로그램되고 이벤트 채널에 의해서 관리된다.

```
Class Event_Source{
  virtual void run();
}
```

- **청취자**(Listener) : 생성자에서 발생된 이벤트를 처리하는 태스크이다. 이벤트가 청취자에게 전달될 때 이벤트 채널은 청취자의 Event_Listener::processEvent (Event evt) 함수를 호출한다. Event_Listener::processEvent (Event evt)함수에는 청취자가 이벤트 처리를 위한 프로그램 루틴이 들어 있다.

```
class Event_Listener{
  virtual void processEvent(Event evt);
}
```

- **이벤트 필터**(Event Filter)**와 이벤트 상관기** (Event Correlator) : 이벤트 필터는 청취자에게 전달될 이벤트 중에서 필요 없는 이벤트를 제거하는 기능을 하고 이벤트 상관기는 생성된 이벤트 간의 상관 관계에 따라 이벤트를 전달할 것인지 아닌지 결정한다.

- **실행시간 스케줄러**(Run-time Scheduler) : 실행시간 스케줄러는 이벤트 필터와 이벤트 상관기를 거친 이벤트의 우선순위를 결정하고 결정된 우선 순위에 맞추어 우선 순위 큐로 이벤트를 전달하는 기능을 담당한다.

- **청취자, 생성자 정보 저장부**(Source, Listener Repository) : 청취자와 생성자의 이벤트에 관한 필터 정보, 객체 참조, 상관 정보를 포함한다.

- **우선순위 큐**(Priority Queue) : 이벤트 전달부는 각 이벤트의 우선 순위에 대응하는 이벤트 큐를 가지고 있다. 각 큐에 전달된 이벤트는 우선 순위에 따라 선취권(preemption)을 가진다.

- **전달부**(Dispatcher) : 큐에 전달된 이벤트를 우선순위에 따라 청취자에게 전달하고 청취자의 이벤트 처리 함수인 Event_Listener::processEvent(Event evt)호출하여 이벤트 처리 태스크를 수행시키는 기능을 한다.

그림 4에서 보는 바와 같이 제안된 이벤트 채널은 크게 이벤트 전처리부와 이벤트 전달부로 나뉜다. 이들 구성 및 역할에 대한 자세한 설명은 다음과 같다.

2.1 이벤트 전처리부

이벤트 전처리부의 기능은 생성자에서 발생된 이벤트의 최종 목적지와 이벤트의 우선 순위를 결정하는 것이다. 이러한 기능을 위해서 이벤트 전처리부는 이벤트 필터, 이벤트 상관기, 청취자 생성자 정보 저장부 그리고 실행시간 스케줄러로 구성되어 있다. 여기서 청취자의 결정과 관련이 되는 부분은 이벤트 필터와 이벤트 상관기이고 이벤트의 우선 순위를 결정하는 것은 실행시간 스케줄러이다. 청취자 생성자 정보는 이벤트의 목적지와 우선순위를 결정할 때 필요한 정보를 저장하는 기능을 담당한다. 이벤트 전처리부에서는 이벤트의 멀티캐스팅(multicasting)을 지원하여 여러 개의 청취자에게 이벤트가 전달되고 청취자는 여러 개의 생성자 또는 생성자 그룹에서 이벤트를 받을 수 있다. 이 기능을 담당하는 것이 이벤트 필터이다. 다음은 filter_info 클래스로 청취자가 이벤트 채널에 등록 할 때 넘겨주는 정보이다. filter_info는 청취자가 받기 원하는 이벤트를 저장한다.

```
class filter_info{
  Event_Listener listener;
  Linked_List Source;
  Linked_List Group;
}
```

filter_info::listener는 청취자 자신의 참조이고 filter_info:: Source는 청취자가 받기 원하는 이벤트 생성자의 Event::sourceid를 저장한다. 청취자는 여러 개의 생성자로부터 이벤트를 받을 수 있도록 링크드 리스트 자료 구조를 가지고 있다. filter_info::Group은 받고 싶은 생성자 그룹의 Event::groupid를 저장한다. 이 또한 여러 개의 생성자 그룹에서 이벤트를 받을 수 있도록 링크드 리스트 자료 구조를 가지고 있다. 이벤트 채널로 전달된 이벤트는 생성자를 나타내는 Event::sourceid와 생성자의 그룹을 나타내는 Event::groupid만을 가지고 있다. 따라서 이벤트 채널에서 이벤트 목적지 정보를 가지는 데이터 구조는 Event::sourceid와 Event::groupid를 기준으로 구성되어야 한다. 이를 위해 filter_info는 등록 될 때 다음의 자료 구조로 바뀌게 된다.

```
class Source_Destination{
    int sourceid;
    Linked_List dest_listener_ref;
}
class Group_Destination{
    int groupid;
    Linked_List dest_listener_ref;
}
class Destination_Table{
    Linked_List SourceDestination;
    Linked_List GroupDestination;
}
```

클래스 Source_Destination과 Group_Destination은 생성자가 발생시킨 이벤트가 전달될 모든 청취자의 객체 참조를 링크드 리스트 자료 구조로 저장하고 있다. 클래스 Destination_Table은 클래스 Source_Destination과 Group_Destination을 링크드 리스트로 저장하여 각 이벤트마다의 전달될 청취자를 저장한다.

이벤트 상관기에서는 이벤트 필터에서 결정된 청취자의 이벤트 발생 조건을 검사하여 이벤트를 전달할 것인지 아닌지를 결정한다. 이 기능은 발생된 이벤트 중에서 어떠한 조건을 만족하는 것만 청취자로 전달한다. 따라서 청취자의 처리 부담을 줄일 수 있으며 이벤트가 네트웍을 통해 전달해야 하는 경우 필요 없는 이벤트의 전달을 줄일 수 있다. 따라서 원격 계측 제어 시스템에서 지연의 영향을 줄일 수 있다. 다음은 이벤트 상관기를 위해서 청취자가 등록하는 정보이다.

```
class correlation_info{
    virtual Event correlation_event( Event evt);
}
```

상관기 정보는 청취자마다 다른 형태의 발생 조건 검사가 가능하도록 정의되어 있다. 기본적으로 이벤트간의 AND와 OR 조건을 지원하지만 보다 복잡한 조건이 필요한 경우에는 correlation_info을 상속받아 가상함수 correlation_event()를 재정의하면 된다. 가상함수 correlation_event()는 인자로 발생될 이벤트가 들어가고 결과로 이벤트가 발생조건을 만족하면 새롭게 발생된 이벤트를 준다. 만약 이벤트 발생 조건이 되지 않으면 결과로 널(null)을 준다.

이벤트 필터와 이벤트 상관기를 거친 이벤트는 우

선 순위 큐에 들어 갈 때 우선 순위를 결정해야 한다. 이 기능을 담당하는 부분이 실행 시간 스케줄러이다. 우선 순위의 결정은 생성자와 청취자가 이벤트 채널에 등록될 때 인자로 들어가는 클래스 schedule_info에 의해서 결정난다. 다음은 클래스 schedule_info이다.

```
class schedule_info{
    int event_type;
    long period;
    long ex_time;
}
```

주기적인 수행은 앞에서 말한 바와 같이 원격 계측 제어 시스템의 주요 특징이다. 제안된 이벤트 채널에서 주기적으로 수행되는 태스크는 비주기적 태스크보다 높은 우선 순위를 가진다. schedule_info::event_type은 생성자나 청취자가 주기적인 태스크인지 비주기적인 태스크인지를 나타낸다. schedule_info::period는 주기적인 태스크의 주기가 얼마인지를 나타내고 schedule_info::ex_time은 생성자나 청취자의 태스크 수행 시간이 얼마인지 나타낸다. 제안한 소프트웨어 구조에서는 주기적인 태스크의 이벤트 스케줄을 위하여 rate monotonic 스케줄 방식을 사용하고 있으며 비주기적인 태스크는 같은 최하위 우선 순위를 가지고 태스크의 실행 주기가 짧은 것이 높은 우선 순위를 가진다. 이 방식은 주기적 이벤트의 제한 조건을 신뢰성 있게 보장한다[13]. 제안된 소프트웨어에서 이벤트 우선 순위는 청취자나 생성자가 등록할 때 정해진다. 이를 통해 이벤트가 발생했을 때마다 우선 순위를 계산하는 것을 방지할 수 있다. 그리고 청취자가 등록할 때 주기적인 이벤트의 경우 시간 제한 조건을 만족할 수 있을 것인지 아닌지를 결정하여 QoS(Quality of Service)를 보장한다. 이때 제안된 소프트웨어 구조에서는 비주기적인 이벤트 전달을 위해서 CPU 시간(CPU time)을 남겨 둔다. 이는 최하위 우선순위의 비주기적인 이벤트의 전달을 빠르게 할 뿐만 아니라 비주기적 이벤트로 처리되는 웹 서비스의 반응이 빨라지는 장점을 가지고 있다. 이때 실시간 QoS보장에 관한 식은 다음과 같다.

$$U = \sum_{i=1}^{m} \frac{C_i}{T_i} \quad (1)$$

C_i는 schedule_info::ex_time을 나타내고 T_i는 schedule_info::period를 나타낸다. 식(1)에서 사용된 CPU 시간 U가 주기적 이벤트 전달에 남겨진 시간을 넘어 설 경우 QoS를 보장하지 못한다. 이때 이벤트 채널을 예외 상황(exception)을 발생시킨다.

2.2 이벤트 전달 부

이벤트 전달부는 우선 순위 큐에 도착된 이벤트를 청취자에게 전달한다. 이벤트 전달부는 그림 5에서 보듯이 우선 순위 큐와 이벤트 전달 쓰레드로 구성이 되어 있다. 그림 5과 같이 하나의 큐는 자신의 이벤트를 처리할 쓰레드 하나를 가진다. 각 큐는 우선 순위를 가지고 있어서 높은 우선 순위 큐에 이벤트가 전달되면 하위 우선 순위 큐의 이벤트는 선취된다. 이벤트의 우선 순위에 따라 청취자에게 전달하는 방식은 높은 우선 순위를 가지는 큐에 높은 우선 순위의 스레드를 할당하는 것이다. 이 방식은 OS의 태스크 선취 기능을 그대로 사용하여 새롭게 쓰레드 스케줄러를 구현할 필요가 없으며 선취 능력이 좋다.

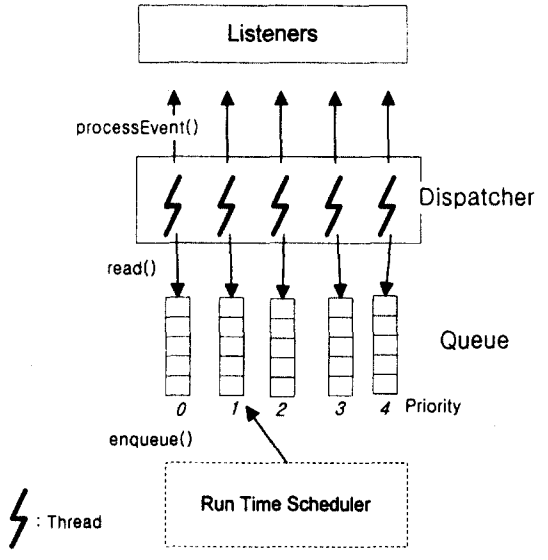

그림 5. 이벤트 전달부 구조
Fig. 5. Event dispatcher structure.

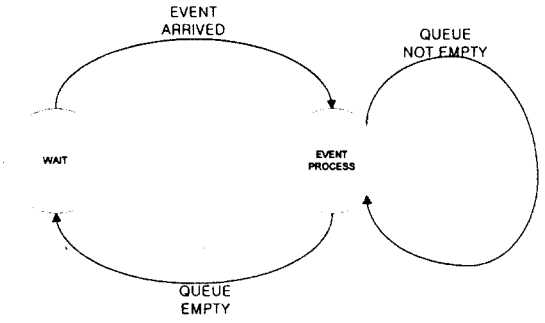

그림 6. 전달 스레드 상태도
Fig. 6. State diagram of event dispatcher thread.

그림 6은 이벤트를 전달하는 쓰레드의 상태도를 나타내고 있다. 그림 6에서 WAIT는 쓰레드는 큐에 이벤트가 들어오길 기다리는 상태이다. EVENT PROCESS는 청취자의 Event_Listener::processEvent()를 호출하여 이벤트 처리를 하는 상태이다. WAIT 상태에서 EVENT PROCESS 상태로 전환될 때 우선순위에 따른 쓰레드의 선취가 일어난다. 만약 우선 순위가 높은 이벤트가 처리되고 있으면 EVENT PROCESS 상태로 전환되지 못하고 우선 순위가 낮은 이벤트가 처리되고 있으면 선취되어 EVENT PROCESS 상태로 변환된다.

2.3 이벤트 처리 과정

그림 7은 앞에서 설명한 이벤트 채널에서 어떻게 이벤트가 전달되는지를 나타내고 있다. 이벤트가 전달되는 과정은 이벤트 발생(Event Generation), 이벤트 전달(Event Dispatching) 그리고 이벤트 처리(Event Handling) 세가지 과정으로 구분이 된다.

이벤트의 생성에서는 생성자가 하드웨어나 소프트웨어에서 생성된 인터럽트를 처리한다. 인터럽

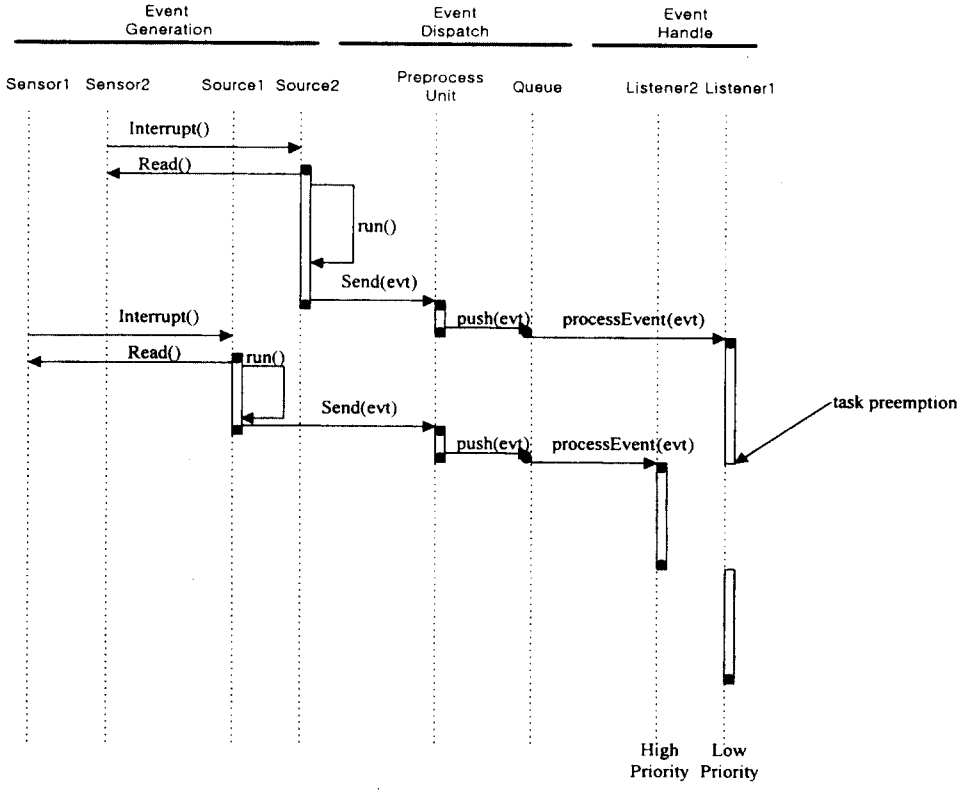

그림 7. 이벤트 생성, 전달 및 처리 과정
Fig. 7. Process of event generation, dispatching and handling.

트가 발생되면 생성자는 인터럽트를 발생시킨 소프트웨어 모듈이나 센서에서 정보를 읽고, 처리하여 이벤트를 생성시킨다. 생성자가 전달한 이벤트는 먼저 이벤트 채널의 전처리부에서 이벤트 필터링과 상관을 거친다. 그 다음 실행 시간 스케줄러에서 우선순위를 결정하여 이벤트 큐로 전달한다. 큐에 전달된 이벤트는 이벤트 전달 스레드에서 청취자의 EventListener::processEvent() 함수를 호출하여 이벤트를 처리하는 태스크를 실행시킨다. 태스크의 수행 중 높은 우선 순위의 이벤트가 큐에 전달되면 태스크는 선취되어서 그림 7에서 보는 것과 같이 낮은 우선 순위의 태스크는 높은 우선순위 태스크가 끝날 때 까지 중단된다.

3. 네트웍 요구 처리부 구조

네트웍을 통해서 전달된 HTTP 명령은 앞의 그림 3의 서비스 객체에서 해석된다. 이때 제안된 구조에서는 동시에 전달되는 HTTP 명령을 효율적으로 처리하기 위해서 네트웍 요구 처리부를 두었다. 네트웍 요구 처리부의 기능은 여러 개의 HTTP 명령을 서비스 태스크를 이용하여 다중 처리하는 것이다. 다중 처리를 위해서 네트웍 요구 처리부는 두 가지 구조를 가지고 있다. 첫째 HTTP명령 해석을 하는 서비스 객체에게 요구를 전달하는 구조이고, 둘째 HTTP명령 해석을 다중 처리 하기 위해서 서비스 객체를 관리하는 구조이다. 제안된 서버에서는 HTTP명령을 서비스 태스크에게 전달하는 구조로 네트웍 디바이스 드라이버에서 자주 사용되는 Half-sync/Half-async 구조[14]를 이용하였다. 그림 8은 그 구조를 나타낸다.

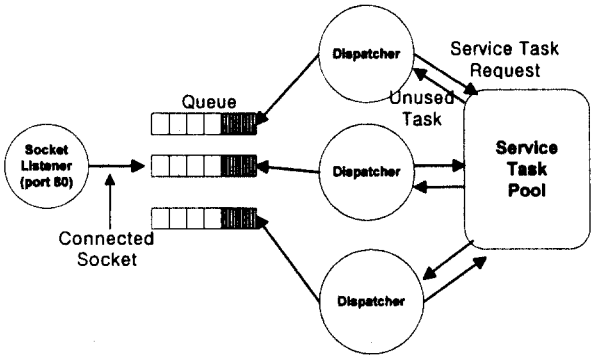

그림 8. 네트웍 요구 전달부 구조
Fig. 8. Network request dispatcher structure.

웹 브라우저는 HTTP 명령을 전달하기 전에 웹 서버와 소켓 연결을 한다. 그림 8에서 소켓 연결자(Socket Listener)는 웹 브라우저의 소켓 연결 요구를 기다리고 있다가 연결 요구가 도착하면 소켓을 연결하고 연결된 소켓을 큐로 전달한다. 이 부분이 Half-async로 비동기적으로 일어나는 소켓 연결 요구를 처리한다. 다음은 소켓 연결자의 과정을 나타내고 있다.

```
Socket_Listener::run(){
    while(true){
        //소켓 연결 확립
        Socket new_socket = accept();
        //queue 연결된 소켓 전달
        request_queue.enqueue(new_socket);
    }
}
```

accept() 함수는 소켓 연결 요구가 올 때까지 소켓 연결자의 수행을 중지시킨다.

그림 8에서 전달부(Dispatcher)는 Half-sync 부분이다. 전달부에서는 큐에 전달된 소켓을 서비스 태스크에 전달하고 태스크를 실행시키는 기능을 가지고 있다. 그리고 제안된 구조에서는 전달부가 여러 개가 있고 각 전달부는 쓰레드로 동작하여 동시에 여러 개의 요구를 처리할 수 있다.

제안된 서버에서는 다음 절에 설명할 서비스 태스크를 변경할 수 있어 HTTP 이외의 프로토콜을 지원할 수 있다. 이 경우 전달부에서는 HTTP 서비스 태스크 이외에 다른 프로토콜을 지원하는 서비스 태스크도 실행되어야 하는데 이를 위해서 서비스 태스크 풀(Service Task Pool)을 두었다. 서비스 태스크 풀에서는 여러 개의 서비스 태스크를 먼저 생성시켜 저장하고 있다가 전달부의 요구가 있을 경우 태스크 수행 중이 아닌 객체 참조를 전달한다. 이때 전달부는 전달받은 객체의 프로토콜 명령 해석 태스크를 실행시킨다.

4. 제안된 소프트웨어 구조에서의 웹 서비스

지금까지는 제안된 웹 서버의 소프트웨어 구조에

대해서 알아보았다. 본 절에서는 이 구조를 이용하여 어떻게 웹 서비스가 가능한지를 설명하겠다. 앞의 그림 3에서 제안된 서버의 개괄 구조에서 설명한 것과 같이 웹 서비스는 서비스 객체와 네트웍 요구 처리자가 담당한다. 그림 3에서 서비스 객체는 HTTP 명령을 해석하여 처리자에게 이벤트를 발생시킨다. 그러면 이벤트를 받은 처리자는 웹 서비스를 한다. 이때 웹 서비스를 하는 네트웍 요구 처리자는 이벤트 채널에 등록할 때 웹 서비스를 위한 가상 생성자를 생성시킨다. 그리고 서비스 객체는 가상 생성자의 생성자 번호를 가지는 이벤트를 발생시킨다. 이렇게 함으로써 서비스 객체는 이벤트가 전달되는 하나의 네트웍 요구 처리자에게만 이벤트를 발생시킬 수 있다.

서비스 객체가 발생시키는 이벤트에는 처리자가 웹 서비스에 필요한 객체가 포함된다. 다음은 서비스 객체가 발생시키는 웹 서비스 요구 이벤트이다.

```
class web_srv_req extends Event{
    Stream in;
    Stream out;
}
```

web_srv_req::in은 웹 브라우저에서 받은 데이터를 바이트 스트림(byte stream)의 형태로 처리자에게 전달한다. 그리고 웹 서비스 처리자는 web_srv_req::out에 웹 브라우저에게 전달한 데이터를 적는다. 그러면 소켓(Socket)을 통해서 데이터가 웹 브라우저로 전달된다. 그림 9은 웹 서비스가 일어나는 과정을 나타내고 있다.

서비스는 명령 해석 및 URL 분리, URL 형태 구분, 그리고 이벤트 생성 및 발생 세가지 단계로 구분할 수 있다. 명령 해석 및 URL 분리에서는 HTTP 명령을 해석하여 URL을 분리하고 브라우저에서 전달받은 데이터를 바이트 스트림형태인 web_srv_req::in로 변환시킨다. URL 형태 구분에서는 URL 형태에 알맞은 네트웍 요구 처리자를 알아낸다. 제안된 서버에서는 크게 두 가지로 URL형태가 구분된다. 하나는 파일 형태의 URL이고 다른 하나는 객체 형태 URL이다. 파일 형태의 URL은 웹 서버내의 파일을 가르친다. 제안된 웹 서버에서는 파일 형태의 URL을 처리하기 위해서 파일 처리 청취자가 이벤트 채널에 등록되어

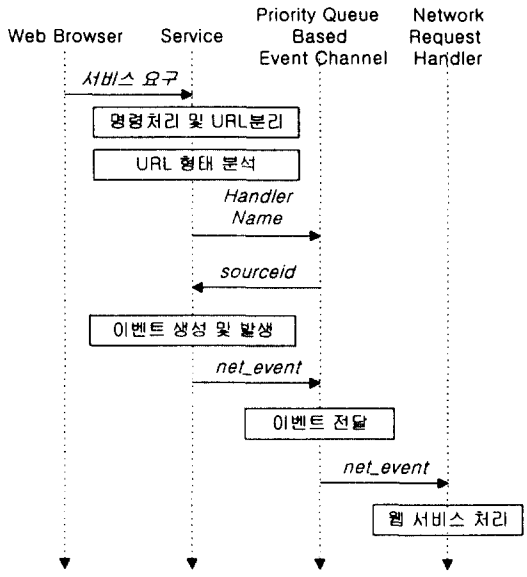

그림 9. 제안된 웹 서버에서의 웹 서비스 과정
Fig. 9. Process of web service on proposed web-server.

있다. 객체 형태에서 URL은 이벤트 채널에 등록되어 있는 웹 서비스를 할 청취자 객체를 가르친다. 객체 형태의 URL은 다시 두 가지로 구분이 된다. 하나는 이벤트 채널에 같은 클래스의 객체가 여러 개 등록되어 있고 같은 URL로 참조되는 경우이다. 이 경우에는 여러 개의 웹 서비스 요구가 각각 객체에 균등 분배되어 처리된다. 다른 하나는 하나의 객체가 이벤트 채널에 등록되는 경우이다. 이 경우는 여러 개의 요구가 동시에 요청되더라도 하나의 객체에서 모든 것을 처리하는 경우이다. 제안된 서버에서는 전자의 경우 *.obj 확장자를 가지고 후자의 경우 *.sobj라는 확장자를 가진다. *.sobj 형태의 객체의 경우 이벤트 채널에 등록될 때의 이름이 바로 URL이 되고, *.obj의 경우는 'URL::순서' 가 된다. URL과 URL형태가 정해지면 서비스는 이벤트 채널로부터 네트웍 요구 처리자가 웹 요구 이벤트를 받기 위해서 남겨둔 가상 생성자의 생성자 번호를 넘겨받는다. 그림 9에서 서비스가 Handler Name을 전달하고 sourceid를 받는 부분이 바로 이 과정을 나타낸다. 이렇게 전달된 이벤트를 처리하는 네트웍 요구 처리자의 이벤트 처리 소스 형태는 다음과 같다.

```
Listener::processEvent(Event e){
        switch(evt.sourceid) {
            case WEBREQUEST:
                do something;
            case SENSOR1:
                do something;
            case SENSOR2:
                do something;
        ...
        }
}
```

위의 예제에서 보는 바와 같이 제안된 웹 서버는 계측 및 제어 데이터 처리 뿐만 아니라 웹 서비스 처리 또한 이벤트 처리 구조를 가지고 있으므로 전체 시스템의 설계 구현을 이벤트 기반으로 통일할 수 있다.

IV. 구현 및 평가

본 논문에서 JDK1.1[15]을 이용하여 자바로 프로토타입을 구현하였다. 프로토타입에 자바를 이용한 이유는 객체 지향 언어로 C/C++보다 간단하게 제안된 구조를 구현할 수 있기 때문이다. 뿐만 아니라 자바의 특징들을 이용하면 서버의 확장이 쉽고 호스트의 종류에 상관없이 사용할 수 있어서 프로토타입 개발에 유리했기 때문이다. 프로토타입을 이용하여 CPU 사용량을 측정하는 시스템을 예로써 구현하였다. 예에서는 CPU 사용량에 관한 정보가 1초마다 측정된다. CPU 사용량을 측정하기 위해서 순수하게 자바로만 구현하기가 불가능하였다. 그래서 JNI(Java Native Interface)를 이용하여 구현하였다. 이벤트 생성자는 매 주기마다 측정된 정보를 이벤트로 발생시킨다. 이벤트 청취자로는 두 개의 모듈을 사용하였다. 일정기간 측정된 데이터를 저장하는 모듈과 일정기간 주어진 주기동안의 CPU 사용량을 평균하여 저장하는 모듈이 있다. 각각의 모듈에서 저장된 정보는 HTTP를 통해서 전달받을 수 있다. 그림 10은 Netscape를 이용하여 웹 서버의 CPU 사용량을 모니터하는 화면이다. 그림 10에서 A의 경우는 CPU 사용량을 처리 없이 저장하는 모듈에서 전달받은 데이터

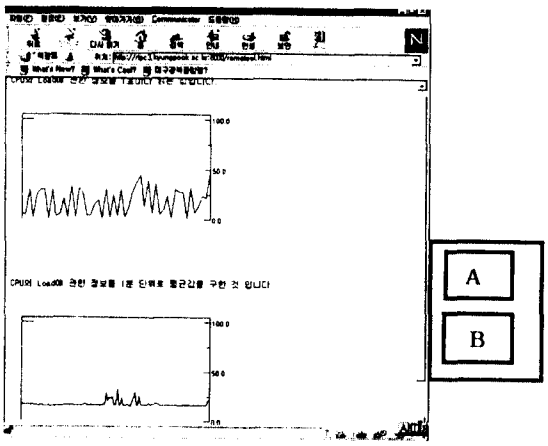

그림 10. 예제 시스템의 웹 인터페이스
Fig. 10. Web interface of example system.

를 그림 10에서 B의 경우는 CPU 사용량을 일정기간 동안 평균한 값을 보여주고 있다. 여기에는 기존 시스템에서 웹 서버와의 연동에 필요했던 부수 계층의 설계가 필요없고, 웹 서버 내부의 실시간 처리 특성을 이용함으로 따로 실시간 스케줄러등의 구현이 필요하지 않았다. 따라서 구현한 예제를 통해서 제안한 구조가 확장이 쉽고 웹 기반 계측 제어 시스템 개발에 유리함을 검증할 수 있었다.

제안한 웹 서버의 웹 서비스 성능의 확인을 위하여 상용화된 기존 웹 서버와 수행성능을 비교하기 위한 실험도 하였다. 성능 척도로는 확장 서비스의 응답시간을 이용하였으며 비교 대상은 현재 가장 많이 사용되고 있으며 우수한 확장 인터페이스인 ISAPI(Internet

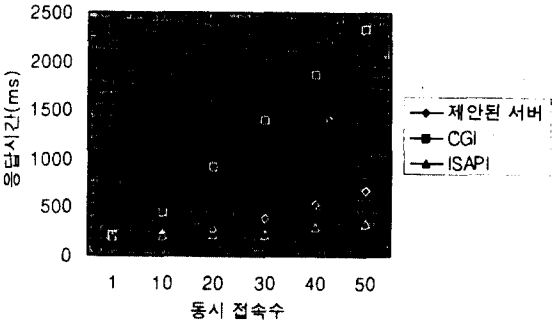

그림 11. 동시 접속수에 따른 웹 서비스 응답시간
Fig. 11. Web service response time with change of concurrent request number.

Service API)[16]를 가진 Microsoft IIS(Internet Information Server)을 이용하였다. 확장 서비스는 CGI와 ISAPI를 이용하였고, 사용한 호스트는 펜티움 100Mhz에 64M 바이트의 메모리를 사용하였으며 사용 OS는 Windows NT 서버 4.0이다. 자바 실행 환경으로는 Windows용 JIT 컴파일러를 이용하였다. 결과는 그림 11과 같다. 프로토타입은 CGI보다 20이상의 동시 접속수의 경우 400%정도 빠르고 ISAPI보다는 다소 느린 것으로 나타났다. 그러나 ISAPI의 경우 20미만의 경우 거의 같은 결과를 나타내고 있다. 결과적으로 프로토타입의 성능이 접속자수가 많을 경우에 ISAPI보다 느리다. 그러나 계측 제어 응용에 적용할 경우 사용자가 제한된다는 것을 감안하며 거의 성능에 차이가 없음을 알 수 있다. 또한 프로토타입이 자바로 구현되어 있어 자바 특성에 기인하는 성능 저하가 주요인이며 제안된 개념을 C/C++을 이용하여 구현했을 경우 성능이 더욱 향상될 것이다. 이를 입증하기 위하여 각 처리 단계마다 소요되는 시간을 측정하여 분석하여 보았다. 측정 단계는 A, B, C 그리고 D 네 단계로 구분하였다. 첫째 A단계는 소켓 연결에서부터 HTTP 요구 해석에 들어가기 전단계이고 B단계는 소켓으로부터 HTTP 요구를 읽어서 해석하는 부분이다. C단계는 이벤트 채널에서 걸리는 시간을 나타내고 D단계는 요구에 반응하여 소켓에 데이터를 적는데 걸리는 시간이다. 결과는 표 1과 같다. 표 1에서 응답 시간과 웹 서버에서의 총 처리 시간과의 차이는 전달 지연과 소켓 계층에서 소요되는 시간을 의미한다. 표 1에서 동시 접속수가 증가함에 따라 소켓 계층에서 소요되는 시간이 늘어나는 것을 볼 수 있다. 또한 소켓에서 데이터를 읽는 B구간에서의 시간을 뺀다면 웹 서버에서 처리에 들어가는 시간은 반응 시간에 비해 아주 작다. 따라서 동시 접속수가 증가함에 따라 웹 서버의 응답 시간은 소켓 계층에 의존적이다. 위의 결과는 소켓 계층에 따라 웹 서버의 성능이 변한다는 것을 보여준다. 따라서 단순한 반응속도 측면에서 제안된 프로토타입의 성능이 Microsoft IIS의 ISAPI방식보다 다소 느린 것은 제안한 서버의 소프트웨어 구조보다 자바 자체의 소켓 계층에서의 성능에 기인함을 확인하였다.

표 1. 동시 접속수에 따른 단계별 웹 서버 처리 시간.
Table 1. Processing time in each step with concurrent request number.

동시 접속 횟수	응답 시간 (ms)	구간별 처리 시간 (ms)				전체 처리 시간 (ms)
		A	B	C	D	
1	212.24	1미만	203.00	1미만	3.28	206.28
10	216.41	1미만	181.06	12.72	5.73	199.52
20	269.47	1미만	142.82	34.44	10.49	187.79
30	393.51	1미만	36.46	5.47	3.82	45.70
40	532.54	1미만	48.89	8.28	4.09	61.29
50	674.53	1미만	40.39	9.21	3.88	53.49

V. 결 론

본 논문에서 인터넷 상에서 WWW(World Wide Web) 서비스 기능을 이용하여 원격 계측 및 제어 서비스에 적합한 새로운 실시간 웹 서버를 제안하였다. 제안한 시스템에서는 웹 서버 내에서 실시간 태스크와 네트웍 태스크가 통신이 가능하도록 설계가 되어 있어 일체형 웹 기반 원격 제어 시스템의 설계 구현이 가능하였다. 또한 비동기적 이벤트 처리에 적합하도록 이벤트 채널기반의 태스크 통신을 적용하여 효과적인 계측 제어 태스크의 설계가 가능하였고. 이벤트 채널에서 우선순위 큐를 이용하여 실시간 태스크의 시간 제한 조건에 대응할 수 있었다. 제안된 개념으로 프로토타입을 구현하여 서버의 CPU 사용량을 감시할 수 있는 예를 통하여 그 성능을 입증할 수 있었다. 또한 다른 웹 서버와의 웹 서비스 성능 비교를 통해서 사용자 수가 제한되는 원격 계측 제어와 같은 환경에서 내부 실시간 처리 기능을 보장하면서도 사용자수가 어느 정도 이하에서는 접속 성능에 큰 차이가 없음을 확인할 수 있었다.

제안된 웹 서버 구조는 원격 검침, 원격 조작등 원격 계측 제어용 혹은 양방향 인터넷 TV, 홈 네트웍을 위한 set-top-box에 내장된 형태로 동작하는 기본 시스템 소프트웨어 구조로 활용될 수 있을 것이다.

현재 개발된 프로토타입은 자바를 이용하여 구현됨으로 인해 엄격한 실시간 요건을 보장해야 하는 내장형(embedded) 실시간 계측 제어 시스템에서는 처리속도에 한계를 나타낼 수 있으나 최근에 이러한 용도로 개발되고 있는 새로운 자바 개념들(Java chip,

JIT Compiler등)을 적용하면 해소될 것으로 보이며, 좀더 빠른 반응이나 복잡한 처리가 필요한 시스템에 적용하기 위해서 제안된 구조를 C/C++을 이용하여 VxWorks[17]등의 실시간 OS에서 구현도 고려되고 있다.

참고 문헌

1. M. Cox, and J. Baruch, "Robotic telescopes: An Interactive Exhibit on The World-wide Web," Proc. 2nd Int. Conf. World Wide Web, Chicago, Oct. 1994.
2. K. Goldberg, M. Mascha, S. Gentner, N. Rothenberg, C. Sutter, and J. Wiegley, "Desktop Teleoperation via the World Wide Web," Proc. IEEE Int. Conf. Robotics and Automation, May 1995.
3. K. Taylor, and J. Trevelyan, "Australia's Telerobotic On The Web," Proc. 25th Int. Industrial Robots Symp., Oct. 1995.
4. E. Paulos, and J. Canny, "Delivering Real Reality to the World Wide Web via Telerobotics," Proc. IEEE Int. Conf. Robotics and Automation, pp. 1694-1699, Apr. 1996.
5. A. Rifkin, "Reengineering The Hubble Space Telescope Control Center System," IEEE Internet Computing, vol. 1, no. 3, pp. 28-35, May, June 1997.
6. 문재철, 강순주, "분산 객체 기술을 이용한 원격 로봇 제어 시스템 설계," 한국통신학회 97하계학술발표 논문집, pp. 1051-1054, July 1997.
7. D. R. T. Robinson, "The WWW Common Gateway Interface Version 1.1," Internet Draft, IETF, Feb. 1996.
8. J. Gosling, "The Java Language Environment," Sun Microsystems, white paper, 1995.
9. OMG, "Common Object Request Broker Architecture," OMG, July, 1995.
10. Sun Microsystems, "Java Remote Method Invocation Specification beta draft," Dec. 1996.
11. OMG, "CORBA Services: Common Object Services Specification," OMG 95-3-31, 1995.
12. T. H. Harrison, D. L. Levine, and D. C. Schmidt, "The Design and Performance of a Real-Time CORBA Object Event Service," Proc. OOPSLA 97, Oct. 1997.
13. C. L. Liu, and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," J. ACM, vol. 20, pp. 46-61, Jan. 1973.
14. D. C. Schmidt, and C. D. Cranor, "Half-Sync/Half-Async: An Architectural Pattern for Efficient and Well-Structured Concurrent I/O," Pattern Languages of Program Design 2, Addison-Wesley, Reading, MA, pp. 437-459, 1995.
15. Sun Microsystems, "JDK 1.1 Documentation," 1997.
16. Microsoft, Microsoft Foundation Class Reference, 1997.
17. WindRiver Systems, VxWorks Programmer's Guide, Mar. 1997.

문 재 철(Jae-Chul Moon) 정회원

1973년 8월 6일생

1996년 2월:경북대학교 전자공학과 졸업(공학사)

1998년 2월:경북대학교 대학원 전자공학과 졸업(공학석사)

1998년 3월~현재:경북대학교 대학원 전자공학과 박사과정

※주요관심분야:실시간시스템설계, 내장형 자바 플랫폼, 객체 지향 소프트웨어설계, 인공 생명.

e-mail:vgate@palgong. kyungpook. ac. kr

이 명 진(Myung-jin Lee)정회원

1974년 12월 14일생

1997년 2월:경북대학교 전자공학과 졸업(공학사)

1997년 3월~현재:경북대학교 대학원 전자공학과 석사과정

※주요관심분야:실시간시스템 설계, 자율이동로봇, 내장형 소프트웨어 설계.

e-mail:mjlee@palgong. kyungpook. ac. kr

강 순 주(Soon-Ju Kang)정회원

1960년 3월 14일생

1983년:경북대학교 전자공학과 졸업(공학사)

1985년 2월:한국과학기술원 전산학과 졸업(공학석사)

1995년 2월:한국과학기술원 전산학과 졸업(공학박사)

1985~1996년 8월:한국원자력연구소 핵인공지능연구실 선임연구원, 전산정보실 실장 역임.

1996년 9월~현재:경북대학교 전자전기공학부 조교수, IEEE, KISS, KIEE, KICS 회원.

※주요관심분야:실시간시스템 설계, 지식처리형 소프트웨어 구조및 소프트웨어공학.

e-mail:sjkang@ee.kyungpook. ac. kr