분산된 서버 관리를 위한 실시간 모니터링 설계 및 구현

Design and Implementation of Real-time monitoring for Distributed Server Management

이 종 대* 구 용 완**
Jong-Dae Lee Yong-Wan Koo

요 약

금융, 증권, 화재 보험사, 뱅킹시스템 등의 대고객 응대 서비스는 대부분, 인터넷, DM, SMS, IVR, E-Mail, FAX 등을 통해서 서비스 하고 있다. 고객은 다양한 접점을 통해 서비스를 제공받을 수 있다. 기업의 입장에서는 호스트 트랜잭션, 혹은 DB 프로세싱에 대하여, 투명성 있게 서비스를 제공해야한다. 본 논문에서는 화재보험, 금융, 증권, 캐피탈, 은행권에서 이용될 수 있는 서버들에 대해 논하고, 고객에게 투명성을 제공하기위한 서버의 실시간 모니터링 시스템을 제안하고 설계 구현하였다. 제안된 모델은 대고객 응대 시스템에서 효율적으로 이용될 수 있다.

Abstract

Finance, Securities, banking system provides information through Internet, DM, SMS, IVR, E-mail and FAX for customer. Customers can use multi channel. Enterprise requires Host transaction and transparency for DB processing. This paper provides two contributions to the study. First, we discuss Server of Fire insurance (finance, securities and capital, bank system) in Enterprise. Second, we present Real time monitoring system for customer transparency. Finally, we design and implement the proposed system. These techniques can be efficiently supported in CRM(Customer Relationship Management).

☞ keyword : Real time monitoring system, CRM

1. 서 론

기업의 대 고객 응대 서비스는 시간 및 공간 제약성에 관계없이 신규 고객 및 기존 고객의 방대한 요구를 수용해야만 한다[1][2]. 특히, 고객으로부터 수익성 모델을 향상시키기 이전에 서비스의 안정화는 무엇보다도 중요한 문제이다. 기업의 고객서비스에서 이용할 수 있는 서비스의 유형은 방대하다. 또한 많은 지점 및 중간 노드 센터가 있는 환경을 분산 시스템으로 구축할 경우 시스템을 관리하는 부분에 있어서 일관성 유지가 매우 중요하다. 본 논문에서는 이기종간 분산된 시스템들의 시스템들을 관리하고 QoS(Quality of Service)를 보장하기 위한 실시간 모니터링 시스템을 설계 구현하였다. 그리고 제안된 모니터링 시스템에서는 시스템에 문제가 발생하거나 또는 오류 발생이 확인 되면 해당 서비스를 우회하기 위하여 부하 균등화(Load balancing) 기능을 포함하였다.

* 종신회원 : 국제대학 컴퓨터정보과 부교수
leejd3@hanmail.net
** 종신회원 : 수원대학교 IT대학 학장, 컴퓨터학과 교수
ywkoo@suwon.ac.kr
[2007/08/20 투고 - 2007/08/31 심사 - 2007/09/14 심사완료]

2. 관련 연구

2.1 모니터링과 부하 균등화의 개념 및 특성

기업의 어플리케이션 개발 시 상호 운용을 고려하여 구축되어야한다. 이는 시스템 간의 데이터 교환 및 기존 시스템 통합에 과다한 비용이 요구되는 문제점이 있기 때문이다. 본 논문에서는 패

킷을 정형화하여 모니터링과 부하 균등화의 상호 운용을 위한 시스템을 설계 개발 하였다.

• 패킷 필터링

일반적인 패킷 필터의 구조는 다음과 같다.

패킷 필터링이란 네트워크 인터페이스를 통과하는 패킷들을 지정한 조건에 의하여 검사하고 조건에 해당하는 패킷을 검출하는 것을 말한다. 사용자 계층에서 응용 프로그램이 소켓을 열고 데이터를 송신하거나 수신 받을 때 패킷 필터링 기능을 활성화 하면 링크 계층 드라이버는 패킷 필터에게 자신을 통과하는 패킷들을 전송한다. 이러한 구조를 통하여 네트워크 응용 프로그램 간의 데이터 송수신에 영향을 주지 않고 네트워크 인터페이스를 통과하는 패킷을 엿볼 수 있게 한다. 패킷 필터는 일반적으로 커널 계층 패킷 필터와 사용자 계층 패킷 필터로 구분 할 수 있으며 커널 계층 패킷 필터는 사용자 계층으로부터 패킷 필터링 규칙을 지정 받아서 규칙에 부합되는 패킷들을 사용자 계층으로 전달한다. 이렇게 사용자 계층으로 전달된 패킷은 다시 사용자 계층 패킷 필터에 의하여 필터링될 수 있다[3].

2.2 부하 균등화의 개념

부하 균등화는 분산 시스템 내 각 노드간의 부하 차이를 줄이고, 프로세서들의 이용도를 높여 전체 시스템의 성능을 향상시키기 위한 일련의 작업을 의미한다[4].

부하 균등화에 대한 일반적인 정의는 <표 1>과 같다.

〈표 1〉 부하균등화의 정의

부하균등화의 정의	학자
노드 간 부하의 불균형을 피하기 위해 컴퓨터 네트워크에 주어진 작업 부하를 재분배하는 작업	Zhou
부하 시스템 전체의 처리능력을 이용하여 각 노드 부하의 과체중을 작업 부하의 이동을 통해 완화시킴으로서 시스템 전체의 성능을 향상 시키는 작업	Eager
분산 시스템 내 각 노드들이 거의 같은 작업 부하를 갖도록 하기 위해 노드를, 혹은 프로세서들에게 프로세스가 할당하는 정책	Krueger
분산 시스템의 자원의 할당을 제어하는 분산된 결정 과정(Distribution Decision Process)	Livny

2.3 부하표현 방법

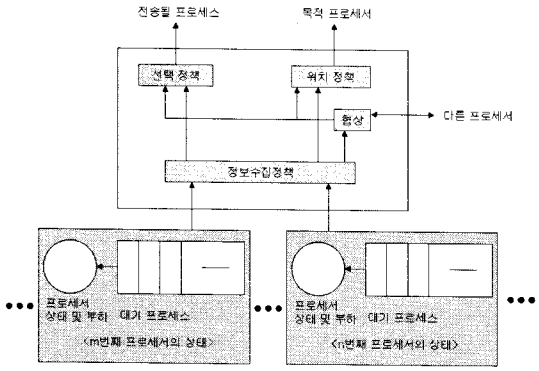
분산 시스템 내 각 노드들의 부하 표현 방법은 다음과 같다[4].

- CPU 대기 프로세스 수
- 일정기간 평균 CPU 대기 프로세스 수
- 가용메모리 크기
- 문맥 교환율
- 시스템 호출율
- CPU 이용률

분산 시스템에서 부하균등 알고리즘은 일반적으로 4개의 구성 요소를 구성되어 있으며 이들의 역할 및 상호 작용은 그림 1과 같다[4].

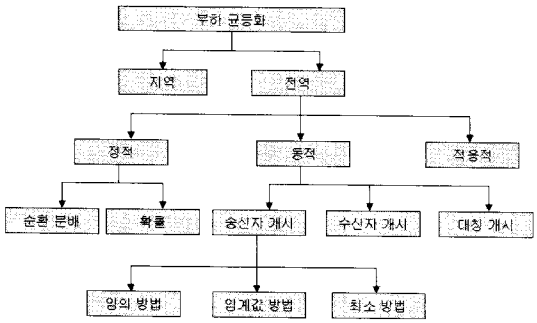
각각의 정책에 대한 내용은 다음과 같다.

- 전송 정책 : 프로세스의 전송에 있어서 어떤 노드가 송신자(Sender) 또는 수신자(Receiver) 인가를 결정하는 것이다.
- 선택 정책 : 송신자로 결정된 프로세서가 이전을 위한 태스크를 결정하는 역할을 수행한다.
- 위치 정책 : 선택정책에서 선정된 태스크를 어느 노드로 보낼 것인가를 결정하는 정책으로, 이전정책에서 결정된 송신자나 수신자에 대응하는 적절한 이전 파트너를 결정하는 기능을 수행한다.
- 정보 정책 : 시스템 내 다른 노드들의 상태 정보를 언제, 어떤 노드로 부터, 어떤 방식으로 수집할 것인지를 결정하는 것이 바로 정보 정책이다.



(그림 1) 부하 균등화 알고리즘의 구성 요소와 역할

다음 그림 2는 부하 균등화 알고리즘에 대한 분류를 나타낸다[4].



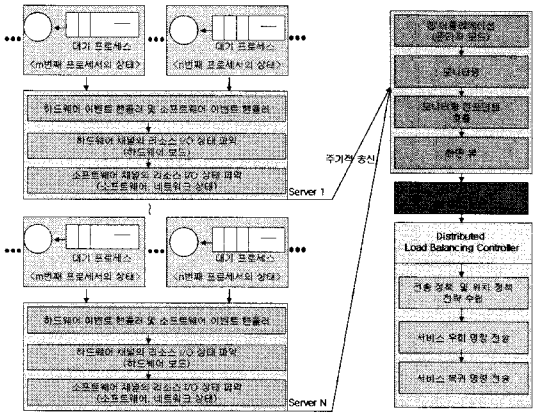
(그림 2) 부하 균등화 알고리즘의 분류

3. 분산된 서버 관리를 위한 실시간 모니터링 설계 및 구현

본 논문에서는 두 가지의 제안사항을 고려하였다. 첫째, 실시간 모니터링을 제안하였다. 둘째, 실시간 모니터링 시스템으로부터 시스템 오류 및 부하를 체크하여 문제가 되는 서버의 서비스를 다른 장비로 우회하는 부하 균등화 시스템을 제안한다.

3.1 실시간 모니터링 및 부하 균등화 시스템 제안

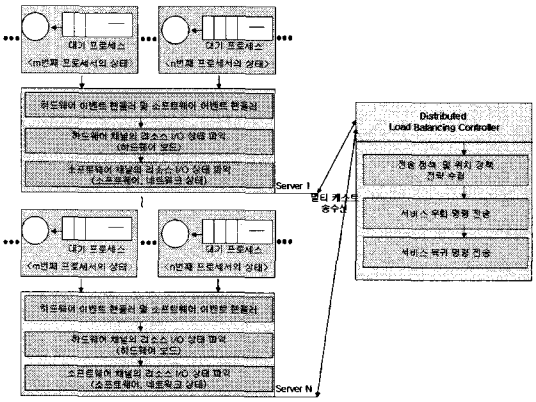
실시간 모니터링 시스템은 각 서버들의 하드웨어 채널 상태, 네트워크 정상여부, 데이터베이스 정상여부, 호스트 통신 정상여부, 시스템 정상 작동여부, 디스크용량, 프로세서의 사용률 등을 주기적으로 실시간 모니터링 서버에게 전송해준다. 단, 주기는 10초로 설정하였다. 실시간 모니터링 서버는 관리자가 인터넷을 통하여 모니터링 할 수 있도록 웹기반으로 설정하였다. 실시간 모니터링 후 관리자는 시스템 오류나 시스템 부하를 판단 한다. 만약, 각 서버들의 상황을 파악 후, 오류 또는 문제가 발생하거나, 부하균등화가 필요할 경우에는 부하 균등화 프로세스를 호출 한다. 부하 균등화 프로세스는 전송정책 및 위치 정책의 전략을 수립한다. 또한 서비스 우회 및 서비스 복귀 명령을 전송 할 수 있도록 설계하였다.



(그림 3) 모니터링 시스템과 부하 균등화 모듈간의 상호 연계도

그림 4는 부하 균등화 프로세스와 서버 간의 연계도를 나타낸다. 부하 균등화 컨트롤러는 시스템 부하 및 오류, 문제가 발생하였을 경우 정상적인 서버 및 부하율이 낮은 서버로 우회 할 수 있도록 설계하였다. 서비스 우회 명령 전송은 전송정책 및 위치 정책의 전략을 전송한다. 서비스 복귀 명령 전송은 원래의 정상적인 서비스로의 복구가 필요한 경우를 위해 설계되었다. 또한 부하 균등화 프로세스는 로거(Logger) 기능을 추가로 갖고 있다. 로거기능은 관리자가 한명 이상일 경우에 어떠한 관리자가 어떠한 명령을 내렸는지

분석할 수 있도록 하였다. 추가적으로 서비스 우회 및 서비스 복귀 명령은 관리자만이 처리 가능하여야 하기 때문에 인증 루틴을 추가하였다.

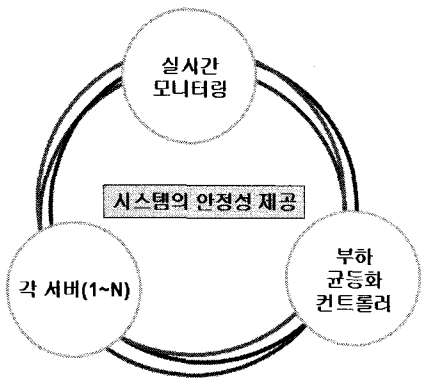


(그림 4) 부하 균등화 프로세스와 서버간의 연계도

부하 균등화 프로세스의 컨트롤러의 기능은 다음과 같다.

- 전송 전책 및 위치 정책 전략 수립기능
- 서비스 우회 기능
- 서비스 복귀 기능
- 로거기능
- 인증기능
- 부하 균등화 기능

그림 5는 각 서버와 실시간 모니터링 서버, 부하 균등화 컨트롤러간의 관계를 나타낸다.



(그림 5) 각 서버와 실시간 모니터링 서버, 부하 균등화 컨트롤러간의 관계

3.2. 실시간 모니터링 및 부하 균등화 시스템 설계

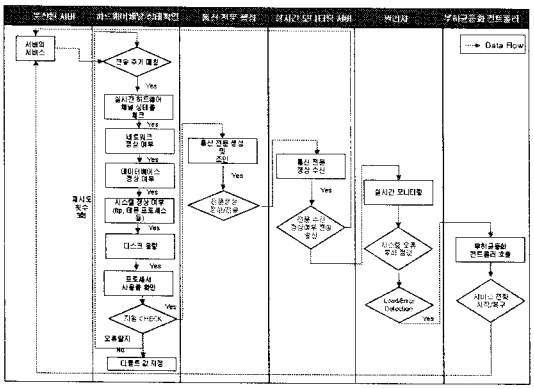
3.2.1 실시간 모니터링 알고리즘

실시간 모니터링 알고리즘은 다음 표2와 같다.

〈표 2〉 실시간 모니터링 알고리즘

1. 각 서버의 장비에서 실시간 서버로 하드웨어 채널 및 상태를 실시간 모니터링 서버로 전송한다.
2. 실시간 모니터링 서버는 패킷을 분석하여 각 서버 장비의 상태를 파악한다.
3. 실시간 패킷을 정상 적으로 받았을 경우 정상적으로 수신하였다는 메시지를 재 전송해준다.
4. 전송 메시지를 로그로 남긴다.
5. 시스템다운, 디스크 용량 등의 자원을 화면에 출력한다.
6. 관리자는 시스템오류 및 결함 상태를 파악한다.
7. 만약 시스템 오류가 발생하면, 부하 균등화 프로세스를 호출하여 서비스 우회 전략을 가동한다.
8. 서비스 우회 전략 등이 가동 되었을 경우 서비스 복귀 명령을 내려 주어야 한다.

비즈니스 업무 프로세스는 그림6과 같다. 분산된 서버에는 서비스가 구동 중에 있다. 이때, 하드웨어 채널 및 상태를 주기적으로 확인한다. 또한 전송 주기에 따라서 네트워크상태, 데이터베이스 상태, 시스템 정상여부, 디스크 용량, 프로세서 사용률을 확인 후, 통신 전문을 생성 한다. 전문생성 후 실시간 모니터링 서버로 전송을 한다. 이때, 통신 신뢰성을 위해서, 반환 신호를 받도록 설계하였다. 관리자는 실시간 모니터링을 통하여, 시스템 오류 및 부하를 점검 후, 부하 균등화 컨트롤러를 호출 하여, 특정 문제가 발생한 서버에 대해 부하 및 작업 우회를 하도록 설정할 수 있다.



(그림 6) 비즈니스 프로세스 구성도

3.2.2 실시간 모니터링 테이블 정의

실시간 모니터링의 대상 서버들은 관리 및 유지가 되어야하기 때문에 데이터베이스에 관리되어야 한다. 실시간 모니터링 테이블은 <표 3>와 같다.

〈표 3〉 실시간 모니터링 테이블

순서	컬럼명	형식	정의	비고
1	SEQ	int		not null
2	SERVER_POS	varchar	100	분산서버위치
3	SYSTEM_NO	varchar	30	시스템번호
4	GROUP_CODE	int	6	그룹코드
5	SYSTEM_CODE	varchar	6	시스템코드
6	IP_ADD	varchar	20	IP 어드레스

3.3 실시간 모니터링 통신 전문 정의

실시간 모니터링에서 각 서버로 정상 메시지 수신하였음을 전송을 위한 통신 패킷은 다음과 같다.

1~3 자리, 형식은 MON으로 채운다. 4~9 자리, 현재시간으로 채운다. HHMMDD형식으로 채운다.

각 서버에서 실시간 모니터링 서버로 전송하는 통신 정의는 다음과 같다.

① 채널 상태 정의는 다음과 같다.
1 : 정상
2 : 오류
3 : 연결
4 : 서비스 안함

각 서버에서 모니터링 서버로 전송하는 통신 전문에 대한 내용은 <표 4>과 같다.

〈표 4〉 각 서버에서 모니터링 서버로 전송하는 통신 전문

순서	항목명	구분	시작위치	길이
1	구분자	공통	0	3
2	현재 시간	공통	3	6
3	회선 수	공통	9	2
4	채널 번호	반복 1	11	3
5	채널 상태	반복 1	14	1
7	디스크 수	공통	15	1
8	하드 디스크 플래그	반복 2	16	1
9	하드 디스크 용량	반복 2	17	6
10	메모리 사용량	공통	23	3
11	네트워크 정상 여부,	공통	26	1
12	데이터베이스 정상 여부	공통	27	1
13	호스트 통신 정상 여부	공통	28	1
14	시스템 정상 작동 여부	공통	29	1
15	프로세서 사용률	공통	30	3

② 하드디스크 사용량 정의는 다음과 같다.
• 최소 단위 : 메가

③ 메모리 사용량은 다음과 같다.
• 100분율 3자리로 정의한다.

④ 디스크 플래그는 다음과 같이 정의한다.
1 : C 드라이브
2 : D 드라이브
3 : E 드라이브
4 : DVD 드라이브
5 : 네트워크 드라이브 1
6 : 네트워크 드라이브 2
7 : 네트워크 드라이브 3

⑤ 네트워크 정상여부 정의는 다음과 같다.
1 : 정상
2 : 오류
⑥ 데이터베이스 정상여부 정의는 다음과 같다.
1 : 정상
2 : 오류
⑦ 호스트 통신 정상 여부 정의는 다음과 같다.
1 : 정상
2 : 오류
⑧ 시스템 정상 작동 여부 정의는 다음과 같이 정의하였다.
1 : 정상
2 : 오류
⑨ 프로세서 사용률은 백분율 3자리로 정의한다.

3.4 부하 균등화 컨트롤러 데몬과 각 서버 서비스 간의 통신 전문 정의

부하 균등화 컨트롤러는 각 서버의 서비스 중지 및 서비스 복귀 명령을 내릴 수 있도록 설계하였다. 또한 서비스를 중지하지 않을 경우 특정 서버의 특성에 따라 추천 서버를 전송해줌으로써, 서버의 부하를 우회 하도록 하였다. 표 5는 부하 균등화 컨트롤러와 각 서버 서비스들 간의 통신을 위한 전문 정의이다.

〈표 5〉 부하 균등화 컨트롤러와 각 서버 간의 통신 전문 정의

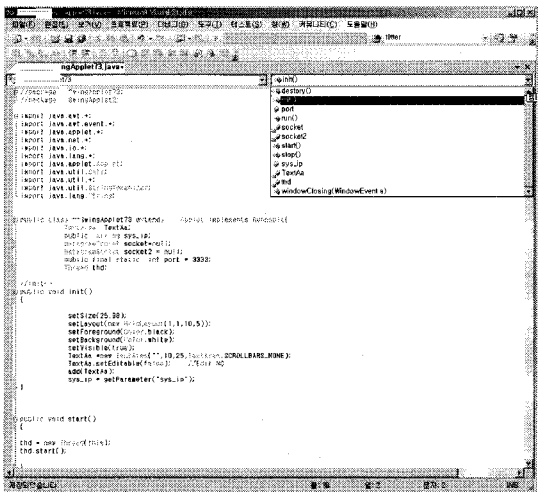
순서	항목명	영문명	시작위치	길이	I/O	형식
1	기본테그	AGTMS	0	5	I	C
2	데몬의 현재날짜	YYYYMMDD	5	8	I	C
3	데몬의 현재 시간	HHMMSS	13	6	I	C
4	채널 탭의 메시지 전송 번튼 누른 시간	HHMMSS	19	6	I	C
5	Y(시작:Y / 종료:N)	Y	25	1	I	D
6	IP	IP	26	20	I	C
7	응답코드	"00"	46	2	O	C
8	추천 서버 1	IP ADDRESS	48	20	O	C
9	추천 서버 2	IP ADDRESS	68	20	O	C
10	추천 서버 3	IP ADDRESS	88	20	O	C

4. 실시간 모니터링 및 부하 균등화 시스템 구현

4.1 시스템 환경

실시간 모니터링 서버와 부하 균등화 컨트롤러 서버는 CPU Board는 인텔 Xeon, CPU 모델명은 인텔 Xeon 2.8GZ를 사용하였다. NIC(Network Interface Card)는 3Com 905B-TX를 사용하였으며, 운영체제는 윈도우즈 2000 서버를 사용하였다. 각 서버의 구성은 CPU 모델명은 인텔 P-3 1GZ를 사용하였으며, 디스크 용량은 HDD 80GB에서 구현하였다. 개발환경은 웹서버는 IIS를 사용하였으며, Java 컴파일러, VC2005를 사용하였다.

다음 그림 7은 모니터링 구현을 위한 개발 화면이다.



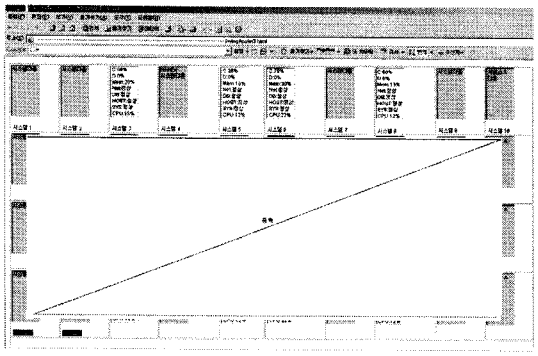
(그림 7) 실시간 모니터링 개발 환면

4.2 구현된 실시간 모니터링 화면(관리자 모드)

다음 그림 8은 실시간 모니터링의 관리자 표준 모드를 나타냈다.

관리자는 관리자 모드를 이용하여 시스템 오류 서버는 시스템 체크를 확인 후, 서비스를 개시 하

여야 한다. 또한 시스템 부하로 판정된 서버는 서비스를 이주시켜야 하는데 부하 균등화 컨트롤러 프로세스를 호출하여 부하를 이주시킨다.



(그림 8) 관리자 표준모드

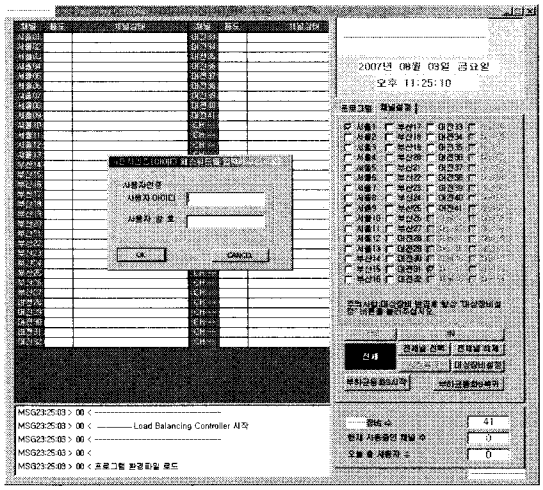
4.3 구현된 부하 균등화 컨트롤러

구현된 부하 균등화 컨트롤러는 그림 9에서 그림 14까지와 같다. <표 6> 부하 균등화 컨트롤러의 환경 구성을 나타낸다.

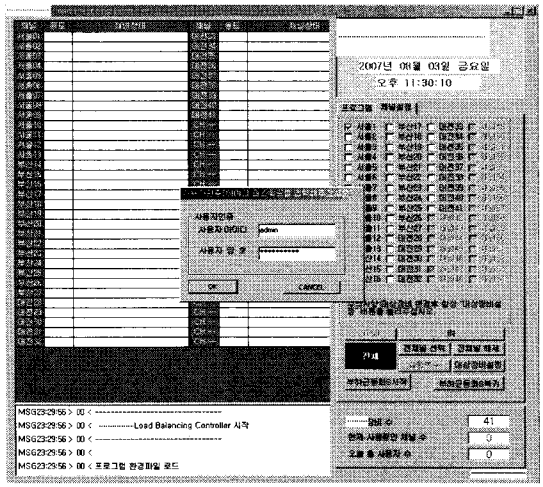
〈표 6〉 부하 균등화 환경 파일 구성

```
// Distributed LoadBalancing Controller
[SERVICE]
// 자동시작 On:1, Off:0
AutoRun=1

[OPTION]
// 서비스이름
AreaTitle=DLCon

// 서버위치(중앙집중 서버별 1)
ServerPosition=1

// 사용할 채널의 시작 채널번호
StartChannel=1

// 사용할 총 채널 수 설정
ChannelCount=41

// 0:발송리스 아님, 1:서비스 지원 시스템
[CHANNEL]
CH01=1
CH02=0
.. 중략
CH41=0

[CHANNELIP]
CH01=211.221.XXX.14X
.. 중략...
CH41=198.245.17.202
```

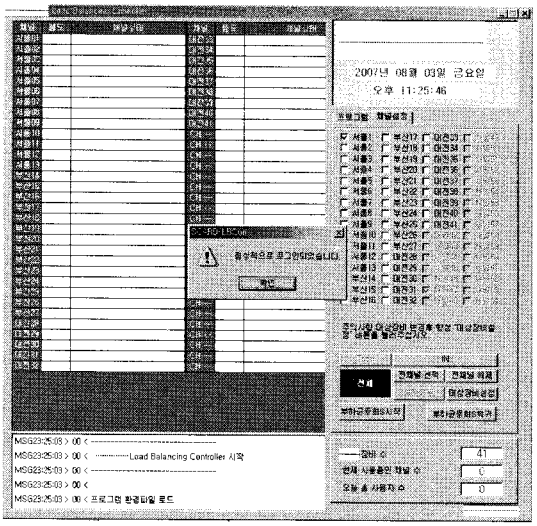
관리자의 시스템 보안을 위하여 인증을 성공해야 서비스를 구동할 수 있도록 구현하였다.



(그림 9) 구현된 부하 균등화 프로그램 1/6

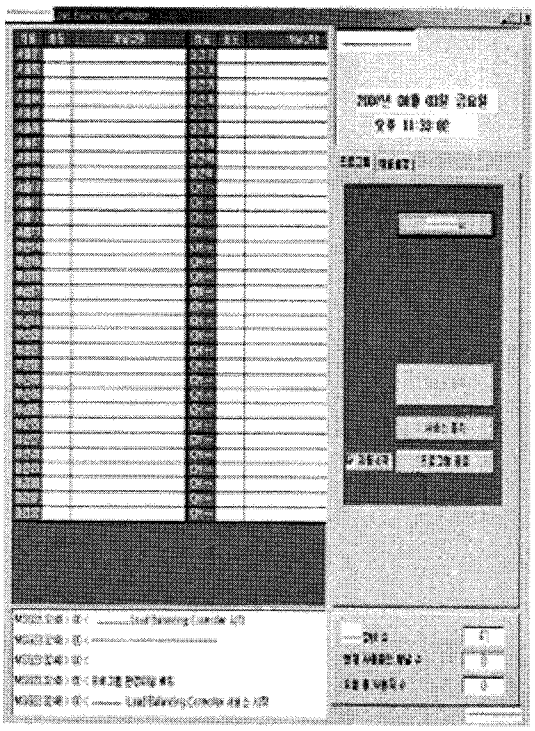


(그림 10) 구현된 부하 균등화 프로그램 2/6



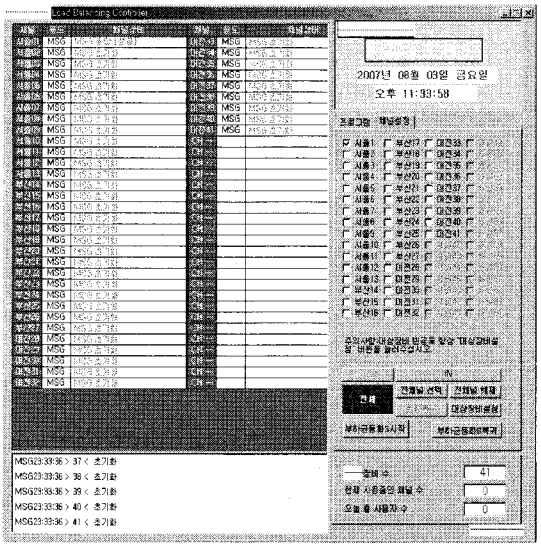
(그림 11) 구현된 부하 균등화 프로그램 3/6

그림 12와 같이 서비스는 환경 파일에서 자동 시작으로 설정하였다.



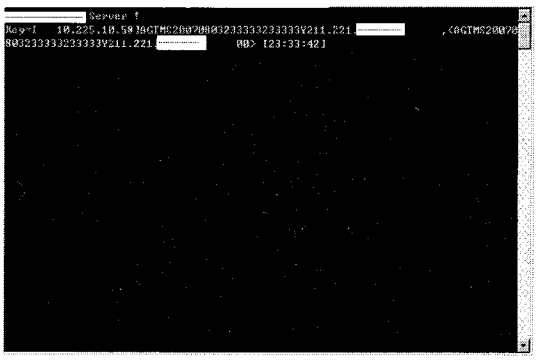
(그림 12) 구현된 부하 균등화 프로그램 4/6

시스템 1번이 부하로 인하여, 1번을 체크 후 부하균등화서비스를 시작했을 경우의 화면은 그림 13과 같다.



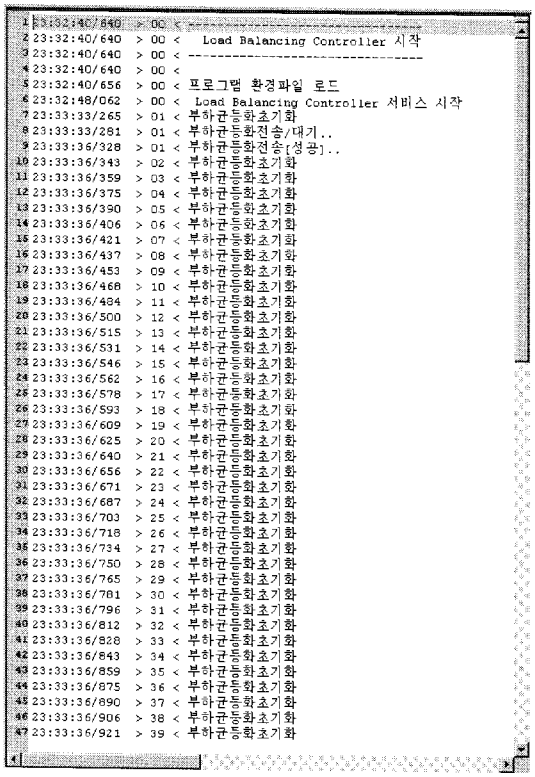
(그림 13) 구현된 부하 균등화 프로그램 5/6

그림 14는 부하균등화컨트롤러에서 부하 균등화 전문을 송신하였을 경우 각 서버에서 리시브된 메시지를 표현하였다.



(그림 14) 구현된 부하 균등화 프로그램 6/6

그림 15는 부하 균등화 컨트롤러에서 구현된 로거 화면이다. 시스템 처리 로그는 시간대별 채널별 서비스별로 기록하도록 구현하였다.



(그림 15) 부하 균등화 로거화면

5. 결론

본 논문에서는 분산된 서버를 효율적으로 관리하기 위하여 실시간 모니터링 시스템을 설계 구현하였다. 또한 관리자레벨에서 모니터링을 통하여 시스템 결함 혹은 장애가 발생, 부하가 발생하였을 경우에 대처할 수 있도록 부하 균등화 컨트롤러를 설계 구현하였다. 구현된 시스템은 방대한 하드웨어 및 소프트웨어의 서비스를 관리할 수 있으며, 부하문제에 대해 관리자 레벨에서 효율적으로 접근할 수 있다. 제안된 시스템은 금융권의 지점관리 시스템, 데이터베이스 서버 모니터링 시스템, 학교 및 법인의 학습실의 데스크톱 관리 등에 효율적으로 이용될 수 있다.

참 고 문 헌

[1] 한윤기, "CRM 설계 및 구현 기술 제안서", Digitalonnet, pp.2-20, 2002~2007.

[2] Steve Michaud & Pramod Ratwani, "Aspect Korea Market 세미나자료", Aspect & Digitalonnet , pp. 10-16, 2007.

[3] 이우중, SOAP 기반 미들웨어에 독립적인 SOAP노드의 실시간 성능 모니터링 방법에 대한 연구, 한양대학교대학원, pp. 9, 2004

[4] 송명남, "분산시스템에서 미래부하 예측을 통한 동적 부하 균등화 알고리즘", 서울여자대학교대학원, pp.4~5, 2000

[5] Douglas E. Comer, "Internetworking with TCP/IP", Prentice Hall, 1995.

[6] W. Richard Stevens, "TCP/IP Illustrated, Volume 1", Addision-Wesley, 1994.

[7] M. Mikhailov and C. Wills, "Embedded Objects in Web Pages", Tech Rep WPI-CS-TR-00-05, Worcester Polytechnic Institute, Worcester, MA, March 2000.

[8] M. Colajanni, P. Yu and V. Cardellini, "Dynamic Load Balancing on Web-Server Systems", IEEE Internet Computing, Vol.11, No.3, May/June 1999, PP.28-39.

[9] M. Aron, d. Sanders, P. Druschel, and W. Zwaenepoel, "Scalable Contents-Aware Request Distribution in Cluster-based Network Servers" in Proc USENIX 2000 Annual Techical Conference, San Die해, CA, PP.475-477, June 2000.

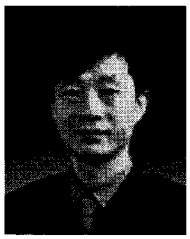
[10] Duncan Clarke, Thierry Jeron, Vlad, Elena Zinovieva, STG:A Symbolic Test Generation Tool, TACAS, 2002.

[11] A. Khoumsi, A method for testing the conformance of real-time systems, IEEE Intern. Symp. on Formal Techn. in Real-time and Fault-Tolsyst.(FTRTFT), Oldenburg, Germany, September 2002.

◐ 저 자 소 개 ◑

구 용 완(Yong-Wan Koo)

1976년 중앙대학교 전자계산학과 졸업(학사).
1980년 중앙대학교 대학원 전자계산학과 졸업(석사).
1988년 중앙대학교 대학원 전자계산학과 졸업(박사).
1983년~현재 수원대학교 IT대학 학장, 컴퓨터학과 교수.
관심분야 : 분산 및 운영체제, 임베디스 시스템, 실시간 리눅스 시스템, 시스템 네트워크 관리, 유비쿼터스 컴퓨팅 등.
E-mail:ywkoo@suwon.ac.kr



이 종 대(Jong-Dae Lee)

1989년 수원대학교 전자계산학과 졸업(학사)
1992년 수원대학교 전자계산학과 졸업(석사)
2008년 수원대학교 전자계산학과 졸업(박사)
1997년 ~ 현재 국제대학 컴퓨터학과 부교수
관심분야 : 분산처리시스템, etc.
E-mail : leejd3@kookje.ac.kr