클러스터 인식 응용 프로그램의 구현 기법에 관한 연구

조 익 성* · 임 재 홍**

A Study on Implementation Methodology of the Cluster Aware Application Program

Ik-Sung Cho* · Jae-Hong Yim**

Abstract

Recently, critical applications have been migrated to distributed computing networks, Internet or client/server environments.

High availability is essential for continuous operation of such business critical applications. Cluster, the high availability software delivers protection for most critical systems. By automatically monitoring, restarting, and recovering applications as soon as they falter, cluster gives end users continuous access to applications, data, and network resources across distributed network environments.

Unlike other solutions, cluster lets you relocate individual failed services from one server to another. This allows applications on the original server to continue running, unaffected by the failed service. Cluster provides the additional advantage of easy administration and scalability, and the cluster architecture provides a standard structure for scalable, cluster-aware applications in future.

So the purpose of this paper is to create a high-level cluster-aware application that provide advanced services such as automatic failover, faster error recovery, easy administration and scalability in clustering environment.

* 한국해양대학교 전자통신공학과 석사과정 전자 · 전산 전공

** 한국해양대학교 전자통신공학과 교수

1. 서 론

시스템의 성능이 높아지고 기능이 다양해질수록 그 동작이 복잡해지고 장애(fault)의 가능성이 높아지게 된다. 높은 가용성(availability)을 이룰 수 있는 한 가지 전통적인 하드웨어 구조는 완전히 똑같은 컴포넌트를 가진 시스템을 중복시키는 것이다. 예전의 컴퓨팅 환경, 즉 메인프레임을 이용하여 연산과 저장, 출력 등을 모두 처리하는 구성에서는 무정지 시스템을 구현하여 장애에 대비하였다[1]. 또한 이러한 하드웨어를 활용하기 위한 전통적인 소프트웨어 모델은 일차 시스템이 응용 프로그램을 실행하고 있는 반면, 다른 시스템은 일차 시스템의 에러가 발생하는 경우에 작업을 넘겨 받기 위하여 대기상태로 있는 것이다. 이 방식의 단점은 시스템의 산출에는 전혀 도움이 안되면서도 하드웨어 비용이 증가하며, 간헐적인 응용 프로그램의 에러에 대비할 수 없다는 것이다.

그러나 이제는 보편적으로 컴퓨팅 환경과 사용자의 마인드가 중앙집중식 환경에서 분산처리 환경 또는 클라이언트/서버 환경으로 변화되었다. 이에 따라 새로운 환경에서 장애에 대비할 새로운 방법이 필요하게 되었고, 그것이 바로 클러스터라는 개념이다. 메인프레임 한대에 모든 것을 의존하는 중앙집중식 환경과는 달리 분산처리 환경에서는 네트워크로 연결된 여러 대의 중소형 시스템에 자원과 부하를 분산하여, 전체 시스템을 구현하는데 드는 비용을 줄이면서 오픈 시스템을 표방하는 여러 벤더(vendor)의 하드웨어와 솔루션을 조합하여 각각의 장점을 모두 이용할 수 있다.

분산처리 환경에서는 작업에 따라 서버를 구분하여 처리 성능을 개선하도록 하는 클라이언트/서버 구현이 필연적이다. 이것은 예전에 메인프레임 혼자서 다 처리하던 연산, 저장, 출력, 통신 등과 같은 작업들을 여러 대의 서버가 각각 성능에 따라 하나 또는 그 이상씩 나누어서 전담하도록 하고 클라이언트에서는 사용자 인터페이스를 구동하여 필요한 작업을 해주는(서비스를 제공하는) 서버를 찾아 그 작업을 해주도록 요청하는 방법이다[2]. 클러스터는 이러한 환경에서 서버의 서비스가 중단되는 시간을 최소화하는 기법을 말한다.

본 논문에서는 이러한 자동적인 작업인계 기능, 신속한 복구, 쉬운 관리의 편의성과 높은 확장성을 위하여 클러스터를 충분히 활용하는 클러스터 인식 응용 프로그램의 구현에 관하여 논한다. 이를 위하여 클러스터 서버의 구조 및 알고리즘을 제시하고, 실제로 2대의 윈도우즈 NT 서버로 구성된 마이크로소프트 클러스터 서버 환경에서 하나의 서버가 다운되는 경우, 에러가 발생한 시스템의 작업을 나머지 서버가 복구하여 클러스터 내의 다른 시스템에 분산시키는 클러스터 인식 응용 프로그램의 구현기법에 관하여 논한다.

2. 클러스터 인식 응용 프로그래밍 기법

2.1 구현환경

본 논문에서는 클러스터 인식 응용 프로그램을 구현하기 위해 그림 1과 같이 시스템을 구성하였다.

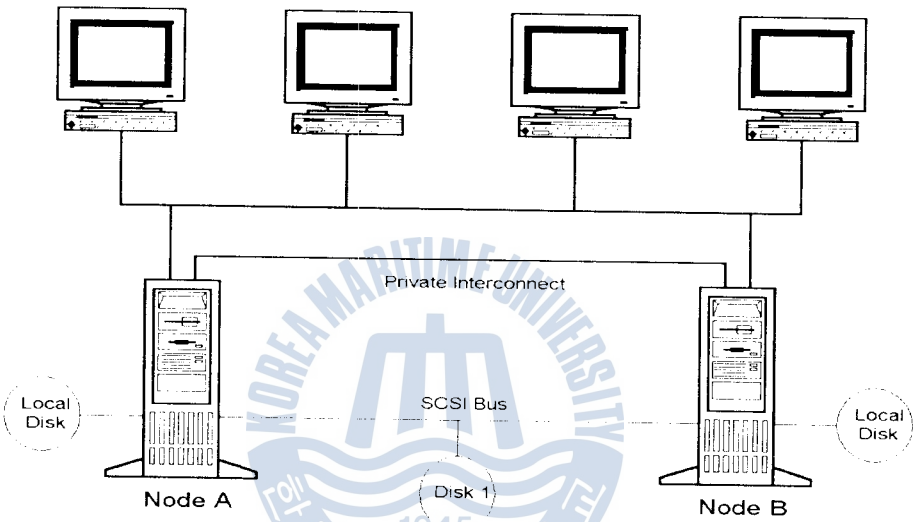

그림 1. 클러스터의 시스템 구성

SERVER 1	
Domain Name	OSDB_SERV
IP Address	203.230.252.80
Cluster Name	INS_CLUS
Cluster IP Address	203.230.252.84
Shared Disk	E:, F:
Location Of Cluster excutables	C:\winnt\system32
Quroum Disk	E:

그림 2. OSDB SERV 구성

CPU는 Pentium Pro 200MHz, 클러스터의 데이터가 저장될 디스크는 2개의 하드 디스크가 부착된 Adaptec AHA2940U2W(AIC-7890A)를 사용하였으며, 클러스터 서버를 구성하기 위해 2대의 윈도우즈 NT 엔터프라이즈 에디션 운영체제를 사용하였다. 클라이언트의 클러스터 서버의 접근을 위한 프라이비트 네트워크와 클러스터 서버간의 통신을 위한 엔터프라이즈 네트워크는 TCP/IP 프로토콜을 사용하였다. 클러스터 서버를 구성할 두 시스템을 OSDB_SERV, OSDB_BACK이라고 명명하였고, 그 구성은 그림 2, 3과 같다.

SERVER 2	
Domain Name	OSDB_BACK
Cluster Name	INS_CLUS
IP Address	203.230.252.81
Cluster IP Address	203.230.252.84
Shared Disk	E:, F:
Location Of Cluster excutables	C:\winnt\system32
Quroum Disk	E:

그림 3. OSDB_BACK 구성

2.2 구현 방법

본 논문에서는 클러스터 인식 응용 프로그램의 GUI 및 프로그래밍 개발 도구로써 비주얼 C++ 5.0을 사용하였고, 비주얼 C++로 간단한 TCP/IP 소켓 응용 프로그램을 실제 구현하고, 구현된 실제 소켓 응용 프로그램과 비주얼 C++ 5.0과의 연계를 위해서 마이크로소프트사의 클러스터 서버 SDK(Software Development Kit)를 이용하여, 다음과 같은 작업을 수행하였다.

(1) 클러스터를 지원할 응용 프로그램을 만든다.
(2) 응용 프로그램에 클러스터 기능을 첨가한다.
(3) 클러스터 서버가 클러스터 리소스로서 응용 프로그램과 서비스를 관리하도록 새로운 리소스 DLL을 만든다.

(4) 클러스터 서버가 커스텀 리소스 유형을 조정하도록 클러스터 관리 확장 DLL을 만든다.

```
//SverSok.cpp
//서버클래스 함수 소스
#include "stdafx.h"
#include "sversok.h"
//생성자
CSeverSock::CSeverSock()
{
     CAsyncSocket::CAsyncSocket();
}
//메세지를 전달할 윈도우 설정
void CSeverSock::SetWnd(HWND hwnd)
{
     m_pHwnd=hwnd;
}

//새롭운 클라이언트가 연결되었을때
//실행되는 함수
void CSeverSock::OnAccept( int nErrorCode )
{
      TRACE("Errocode = %d",nErrorCode);
      //클라이언트 클래스의 m_pChild로 연결시켜준다.
      Accept(m_pChild);
      //새로운 클라이언트가 연결되었다는
      //WM_ACCEPT_SOCKET 메세지를 전달합니다.
      SendMessage(m_pHwnd,WM_ACCEPT_SOCKET,0,0);
}
//OnAccept함수 실행후 새롭운 클라이언트와 연결된
//m_pChild 소켓을 넘겨줍니다.
CClientSock* CSeverSock::GetAcceptSocket()
{
     return &m_pChild;
}
```

그림 4. 서버 소켓 프로그램

```
#include "stdafx.h"
#include "Clientsok.h"


CClientSock::CClientSock()
{
     CAsyncSocket::CAsyncSocket();
}

void CClientSock::SetWnd(HWND hwnd)
{
     m_pHwnd=hwnd;
}


void CClientSock::OnReceive( int nErrorCode )
{
      TRACE("Errocode = %d",nErrorCode);
      SendMessage(m_pHwnd,WM_RECEIVE_DATA,0,0);
}
```

그림 5. 클라이언트 소켓 프로그램

클러스터 인식 응용 프로그램은 클러스터 노드에서 동작하고 클러스터 리소스로서 관리될 수 있는 응용 프로그램이다. 클러스터 인식 응용 프로그램은 클러스터가 제공하는 환경을 인식하고 클러스터의 특징을 이용하도록 설계된다. 본 논문에서 구현할 클러스터 인식 응용 프로그램을 직접 실험하기 위해서 그림 4, 5와 같이 서버, 클라이

언트가 서로 채팅을 할 수 있는 응용 프로그램을 만들었다. 이 응용 프로그램은 그림에서 보는 바와 같이 비동기(asynchronous)소켓을 이용하였고 클라이언트가 연결되었을 때의 함수로 "OnAccept"를, 서버의 메시지를 받을 함수로 "OnReceive"를 사용하였다.

2.3 클러스터 기능의 첨가

클러스터 인식 응용 프로그램이 클러스터 노드에서 동작하고 있다는 것이 감지되면, 클러스터 인식 응용 프로그램은 클러스터 API를 통하여 유용한 작업을 제공한다. 비록 응용 프로그램을 관리할 수 있는 리소스 DLL이 클러스터 서버에 의해 제공된다 하더라도, 자신의 새로운 리소스 DLL을 만드는 것이 좋다. 응용 프로그램에 적합한 리소스 DLL을 만듦으로써, 클러스터 서버가 제공하는 더 나은 제어와 유동성을 제공할 수 있다. 커스텀 리소스 유형을 만듦으로써 응용 프로그램에 적합한 방식으로, 리소스 모니터가 자신의 응용 프로그램을 관리할 수 있도록 한다. 또한, 리소스 API를 이용하여 리소스 DLL을 만들 수 있고, 클러스터 관리자가 응용 프로그램을 관리할 수 있도록 클러스터 관리 확장 DLL을 만들 수 있다. 리소스 DLL은 응용 프로그램과 리소스 모니터 사이의 통신을 제어하기 위해 리소스 API에 정의된 엔트리포인트 함수의 실행을 담고 있다.

클러스터 확장 DLL은 응용 프로그램이 클러스터 관리자에 의해 관리될 수 있도록 클러스터 확장 API의 인터페이스 메소드를 이용한다. 클러스터 확장 DLL은 네트워크 관리자의 작업을 단순화시킨다. 즉 응용 프로그램과 독립된 리소스, 그룹에 통합된 리소스를 사용자 인터페이스를 통하여 서로 동작시키도록 한다.

2.4 리소스 DLL과 관리 DLL의 설계 및 구현

클러스터 인식 응용 프로그램과 관련된 새로운 리소스 유형을 만드는 것은 가능하다. 새로운 리소스 유형(type)을 만들 때 네트워크 관리자는 클라이언트가 접근하는 가상(virtual) 서버를 구성할 네트워크 이름과, 그것과 관계된 리소스를 새로운 유형과 결합해야 한다. 네트워크 관리자는 그룹을 형성할 때 필요한 리소스에 관계된 의존성이 결여되었을 때 클러스터 관리 확장 DLL의 상태를 점검하고, 리소스 DLL을 적재함으로써 같은 그룹 내의 필요한 모든 의존성을 위치시켜야 한다. 예를 들면, 새로운 리소스 유형이 데이터 파일에 관련된 클러스터 응용 프로그램을 관리한다고 가정하면, 네트워크 관리자는 이 응용 프로그램과, 응용 프로그램의 데이터 파일과 관련된 네트워크 네임 리소스, IP 어드레스 리소스, 공유 디스크 리소스를 형성해야 한다[3].

2.4.1 리소스 유형 마법사

새로운 리소스 유형을 만들기 위해서 그림 6과 같이 비주얼 C++의 "Cluster Resource Type AppWizard"를 이용하여 리소스 이름을 Test라고 명명하였다. 그림 7과 같이"Resource Type AppWizard"를 이용한다.

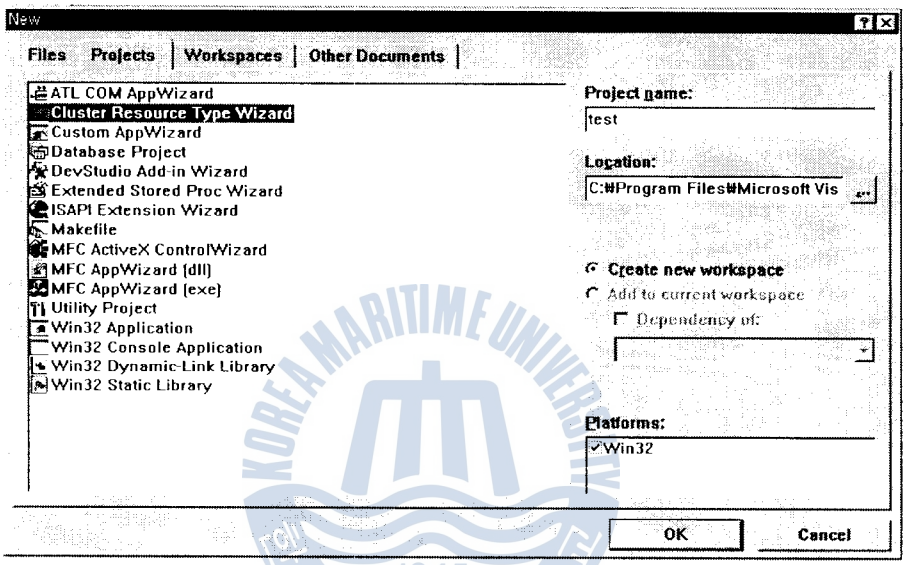

그림 6 . 클러스터 리소스 유형 마법사

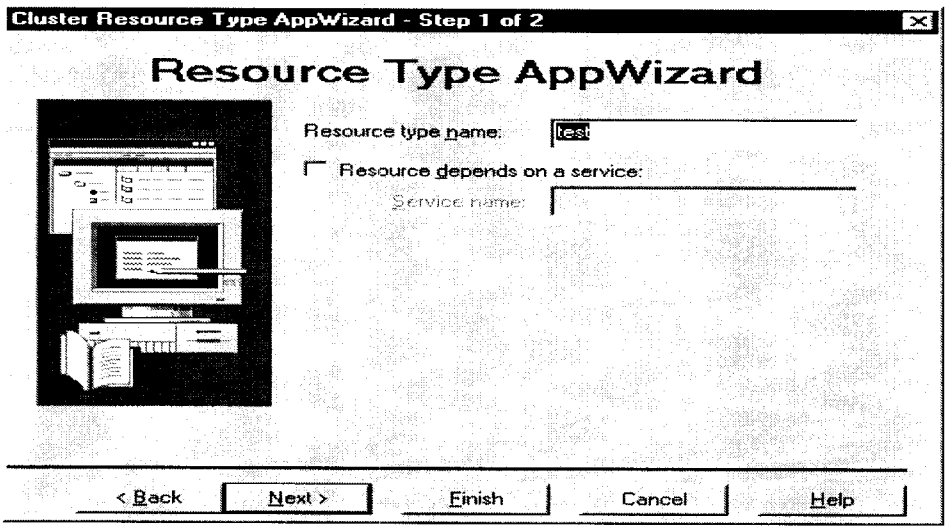

그림 7. 리소스 유형 응용 프로그램 마법사

2.4.2 파라미터 첨가

파라미터는 리소스 유형에 관계된 프라이비트(private) 속성이다. 네트워크 관리자가 새로운 리소스 유형을 만들기 위하여 클러스터 관리자를 동작시킬 때, 클러스터 관리 확장 DLL은 이 속성(property)에 대한 값을 리턴한다. 이 값들은 클러스터 데이터베이스에 저장된다. 이것은 엔트리포인트 함수와 인터페이스 메소드의 뼈대를 만든다. 그림 8에서 파라미터 이름을 MaxUser라고 주었고, Maxium value를 10이라고 주었는데 이것은 서버쪽에서 최대 사용자를 얼마나 받아들일 것인지를 나타낸다.

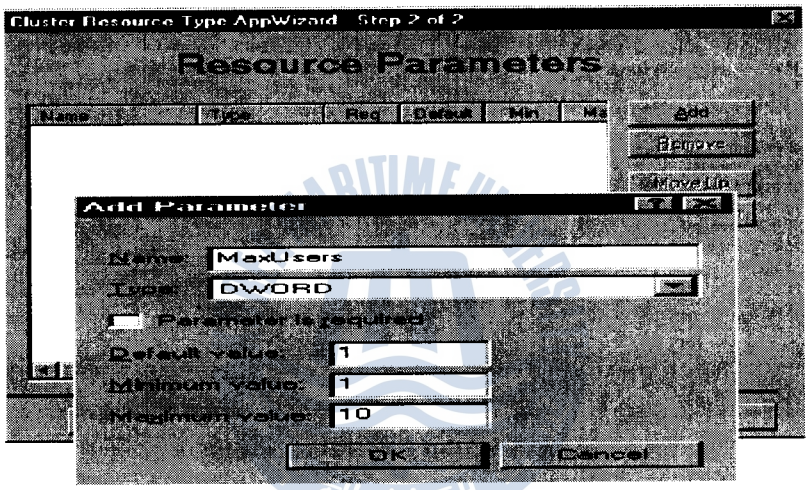

그림 8. 리소스 파라미터 첨가

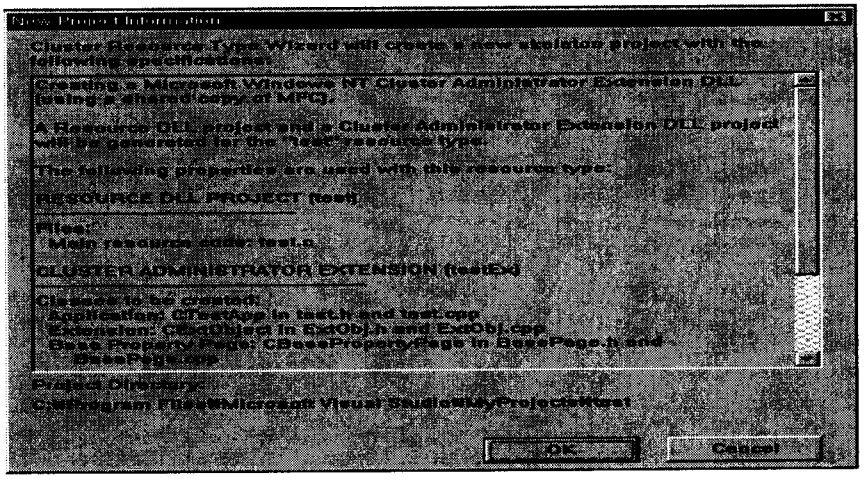

그림 9. 클러스터 프로젝트 정보

이렇게 해서 그림 9와 같이 클러스터를 지원하는 리소스에 대한 DLL, 즉 "cluster resource DLL"과 "cluster administration DLL"이 만들어진다[4].

2.4.3 헤더 파일 첨가

응용 프로그램이 클러스터를 인식하게 하기 위해서는 그림 10과 같이 응용 프로그램에 대한 정보를 담고 있는 헤더 파일, 즉 <server.h>, <sversok.h>를, 클러스터 행위를 정의하고 있는 <clusapi.h>, <resapi.h>, <stdio.h>다음에 첨가시켜야만 한다. 그리고 그림 11에서 보는 바와 같이 리소스에 관계된 IP 어드레스, 네트워크 이름, 공유 이름을 의존성으로 파라미터에 추가시켰다.

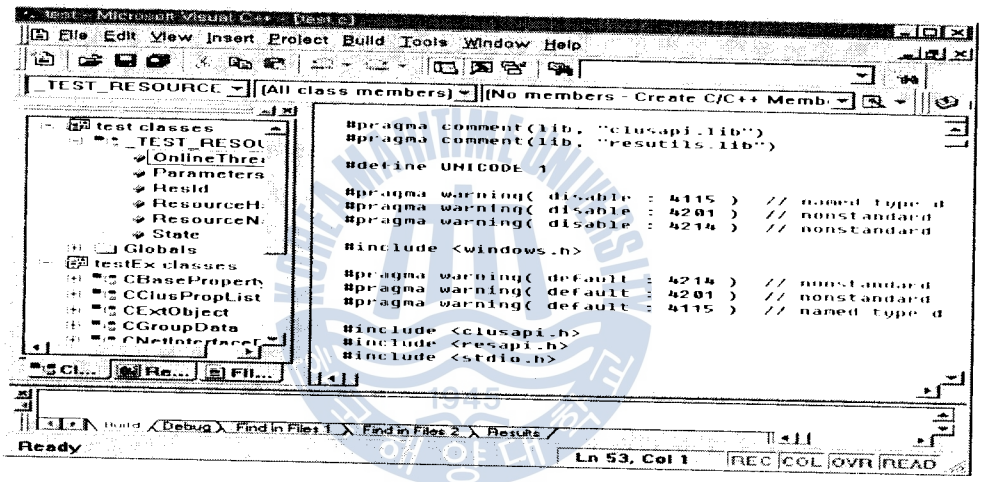

그림 10. 클러스터 리소스 DLL과 클러스터 관리 DLL

```
//
//  Type and constant definitions.
//

#define TEST_RESNAME    L"TEST Sample"
#define TEST_SUCNAME    TEXT("SocketServer")

#define DBG_PRINT printf

// ADDPARAM: Add new parameters here.
#define PARAM_NAME__SHARENAME L"ShareName"
#define PARAM_NAME__PATH L"IPAddress"
#define PARAM_NAME__REMARK L"NetName"


// ADDPARAM: Add new parameters here.
typedef struct _TEST_PARAMS {
    PWSTR                ShareName;
    PWSTR                IPAddress;
    PWSTR                NetName;
} TEST_PARAMS, *PTEST_PARAMS;
```

그림 11. 파라미터 정보

Ⅳ. 실험 결과 및 고찰

본 논문에서 구현한 클러스터 인식 응용 프로그램에 대한 리소스는 그림 12와 같이 "Resource DLL"과 "Cluster Administration Extension DLL"이다. 구현한 "Resource DLL은 "regcladm.exe"라는 클러스터 서버 응용 프로그램으로 클러스터 서버에 리소스 유형으로 등록하였고, 그룹을 형성할 의존성에 대한 리소스로서 네트워크 이름, IP address, 디스크 리소스를 만들었다.

이렇게 해서 그림 9와 같이 클러스터를 지원하는 리소스에 대한 DLL, 즉 "cluster resource DLL"과 "cluster administration DLL"이 만들어진다[4].

2.4.3 헤더 파일 첨가

응용 프로그램이 클러스터를 인식하게 하기 위해서는 그림 10과 같이 응용 프로그램에 대한 정보를 담고 있는 헤더 파일, 즉 <server.h>, <sversok.h>를, 클러스터 행위를 정의하고 있는 <clusapi.h>, <resapi.h>, <stdio.h>다음에 첨가시켜야만 한다. 그리고 그림 11에서 보는 바와 같이 리소스에 관계된 IP 어드레스, 네트워크 이름, 공유 이름을 의존성으로 파라미터에 추가시켰다.

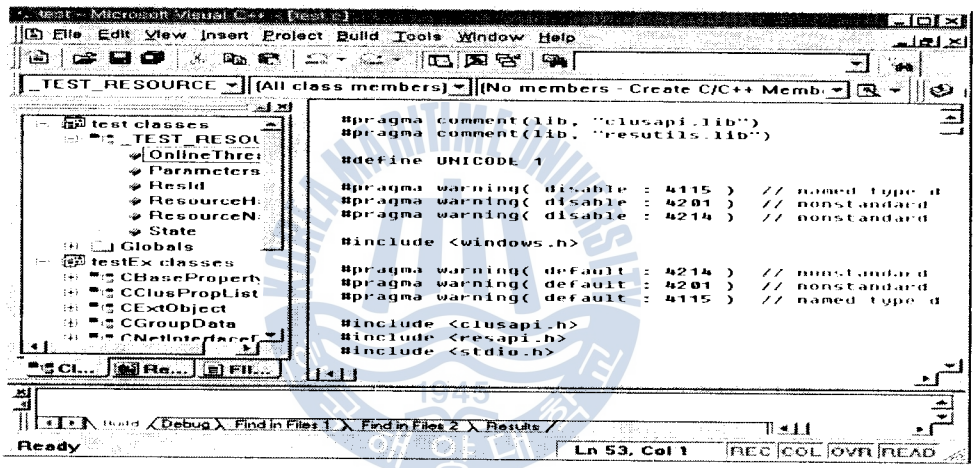

그림 10. 클러스터 리소스 DLL과 클러스터 관리 DLL

```
//
//  Type and constant definitions.
//

#define TEST_RESNAME    L"TEST Sample"
#define TEST_SVCNAME    TEXT("SocketServer")

#define DBG_PRINT printf

// ADDPARAM: Add new parameters here.
#define PARAM_NAME__SHARENAME L"ShareName"
#define PARAM_NAME__PATH L"IPAddress"
#define PARAM_NAME__REMARK L"NetName"


// ADDPARAM: Add new parameters here.
typedef struct _TEST_PARAMS {
    PWSTR               ShareName;
    PWSTR               IPAddress;
    PWSTR               NetName;
} TEST_PARAMS, *PTEST_PARAMS;
```

그림 11. 파라미터 정보

Ⅳ. 실험 결과 및 고찰

본 논문에서 구현한 클러스터 인식 응용 프로그램에 대한 리소스는 그림 12와 같이 "Resource DLL"과 "Cluster Administration Extension DLL"이다. 구현한 "Resource DLL은 "regcladm.exe"라는 클러스터 서버 응용 프로그램으로 클러스터 서버에 리소스 유형으로 등록하였고, 그룹을 형성할 의존성에 대한 리소스로서 네트워크 이름, IP address, 디스크 리소스를 만들었다.

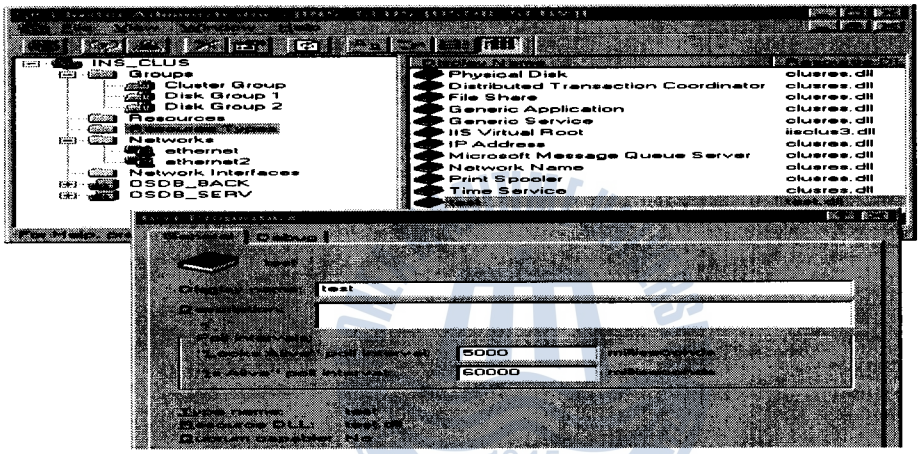

그림 12. 클러스터 인식 응용 프로그램

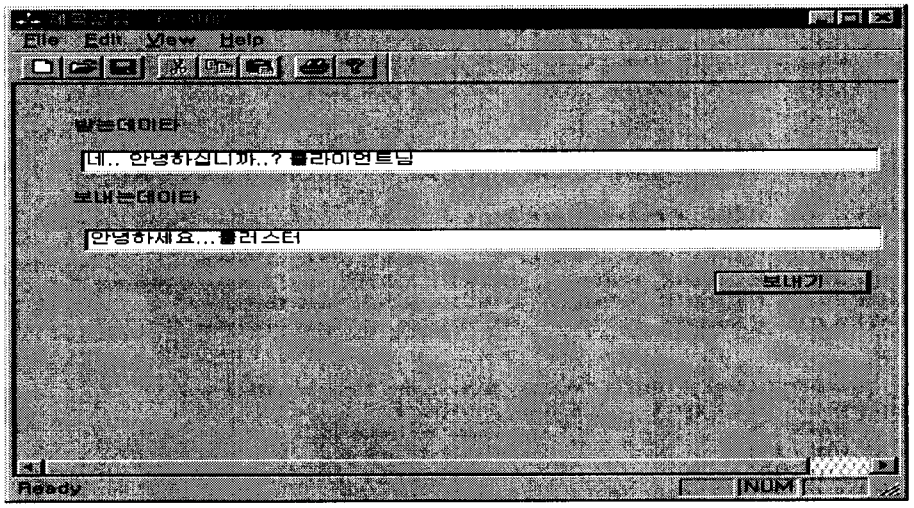

그림 13. 서버 응용 프로그램

실제 구현한 DLL의 시험을 위해 간단한 클라이언트 및 서버 소켓 응용 프로그램을 그림 13, 14에 나타내었다. 이 응용 프로그램은 클라이언트가 서버쪽으로 연결을 설정하여 채팅을 하는 응용 프로그램이다. 실제 동작을 확인하기 위하여 서버 채팅 프로그램을 수행하고 있는 INS_SERVER의 SOCKET_GROUP을 그림 15와 같이 수동적으로 클러스터 내의 나머지 서버로 이동시켰을 때, 수행되던 서비스는 그림 16과 같이 자동적으로 INS_BACK으로 작업인계가 이루어진다는 것을 확인할 수 있었다. 그리고 작업인계 시간은 약 30초 정도에서 이루어진다는 것을 알 수 있었다.

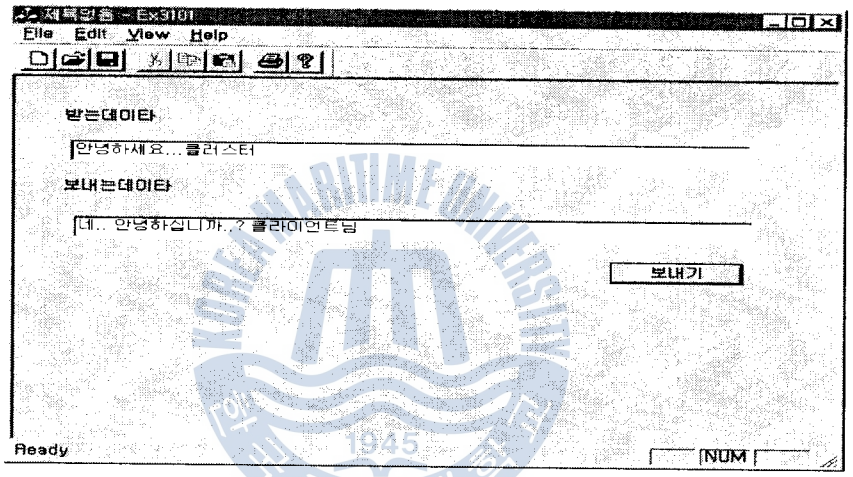

그림 14. 클라이언트 응용 프로그램

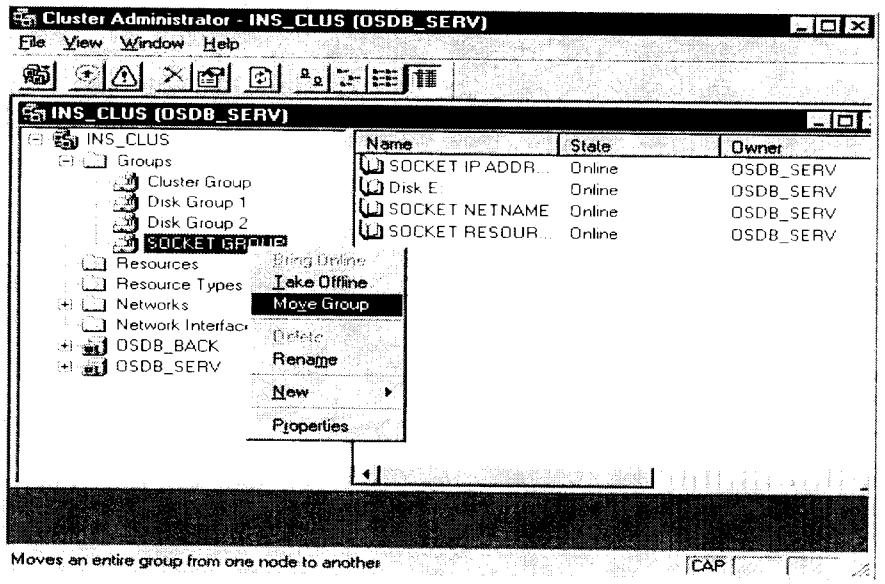

그림 15. 결함 전의 응용 프로그램

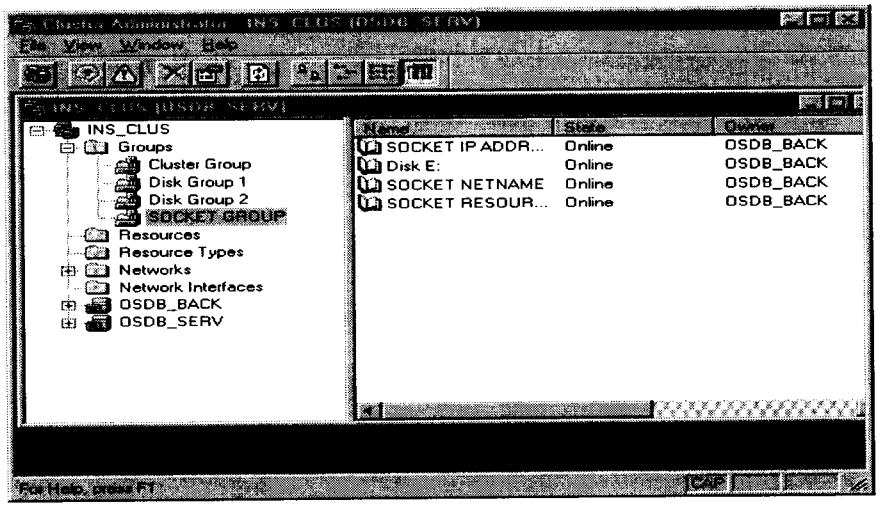

그림 16. 결함 후의 응용 프로그램

V. 결 론

현재 데이터베이스 서버와 웹 서버, 메일 서버와 같이 서버의 중요성이 높은 경우에는 클러스터의 작업인계 기능을 지원하고는 있지만 모든 서버 응응 프로그램이 클러스터 기능을 지원하지는 않는다.

본 논문에서는 응용 프로그램의 자동적인 작업인계 기능, 신속한 복구, 관리의 편의성과 높은 확장성을 위하여 클러스터 인식 응용 프로그램의 구현기법에 관하여 기술하였다. 이를 위하여 간단한 TCP/IP 소켓 응용 프로그램을 구현하고, 클러스터를 지원하게 함으로써 실제 마이크로소프트 클러스터 서버 환경에서 한 서버 내의 응용 프로그램이 다운되는 경우 다른 서버가 자동적으로 다운된 서버 내의 응용 프로그램을 작업인계한다는 것을 확인할 수 있었다.

클러스터가 대부분의 서버 응용 프로그램의 가용성과 관리의 편의성을 제공하지만, 클러스터 인식 응용 프로그램은 클러스터 환경의 확장성을 완전히 활용할 수 있다[5].

향후 연구로는 클러스터가 지원하는 노드의 수를 늘리고 확장성이 개선되어야 할 것이며, 웹 서버와 데이터베이스 서버가 이상이 발생했을 경우 빠른 서비스를 제공하기 위해서는 작업인계 시간이 향상되어야 할 것이다. 또한 보다 편한 GUI 환경을 지원하기 위한 OCX(Ole Control eXtensions)와 같은 컴포넌트를 이용한 클러스터 오토메이션 서버가 개발되어야 할 것이다.

참 고 문 헌

[1] http://www.mantech.co.kr/ha.html, High Availability

[2] Katzman., J.A., et.al., *A Fault-tolerant multiprocessor system*, March 28, 1989

[3] SharonJ, *Writing Microsoft Cluster Server (MSCS) Resource DLLs*, Microsft Corporation, 1997

[4] Scott H. Davis, *Clusters for Windows NT*, Microsoft Developer's Network Library, 1998

[5] Tom Phillips, *Designing and deploying cluster solution*, Microsoft corporation