다수의 터미널 서버 관리를 위한 네트워크 프레임워크 개발

곽준욱*, 조정훈*#, 김승광**
*경북대학교 전자전기컴퓨터학부
{bpwook, jcho}@ee.knu.ac.kr
**㈜위즈엔텍
kimsg@wizntec.com

Development of Network Framework for Managing Mass Terminal Servers

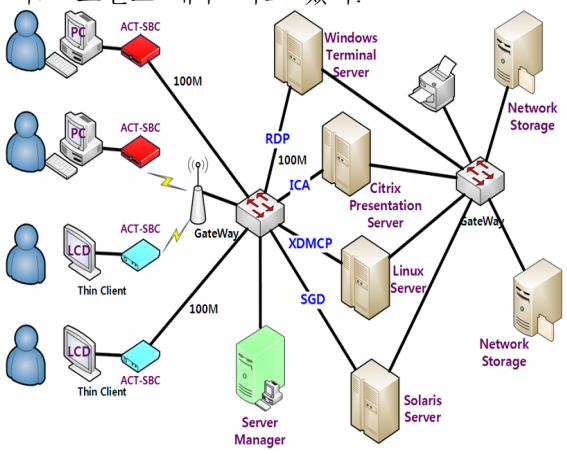
Jun-Wook Kwak*, Jeonghun Cho*#, and Sung-Kwang Kim**
*School of EECS, Kyungpook National University
**Wizntec co., ltd.

요 약

최근 각광 받고 있는 서버 기반 컴퓨팅(SBC, Server-based Computing)은 많은 수의 서버가 필요하기 때문에 많은 수의 서버를 관리할 수 있는 관리자 시스템의 필요성이 대두되고 있다. 이에 관리자 네트워크를 필요에 따라, 코맨드 센터(CC, Command Center), 서버 에이전트(SA, Server Agent), 관리자 인터페이스(Manager Interface), 유저 에이전트(UA, User Agent)등 네 가지 모듈로 분리하여 필요한 시스템 기능을 구현하는 것 만으로 저비용으로 신뢰성 있는 네트워크를 구축할 수 있는 네트워크 프레임워크를 개발하고, 그 성능을 측정하였다.

1. 서론

최근 기업에서는 저비용 유지 보수와 보안의 이유로 SBC 가 대두 되고 있다[1]. 또한 일반 사용자를 대상으로 소프트웨어나 컴퓨터 자원을 대여하여 주는 서비스 역시 웹 오피스의 부족한 기능을 보완하면서 저 비용으로 소프트웨어를 사용할 수 있어서 새로운 비즈니스 모델로 대두 되고 있다.



(그림 1) ACT-SBC 구성도

그림 1 에서는 현재 개발중인 위즈엔텍[2]사의 ACT-SBC 를 보여주고 있다. ACT-SBC 는 멀티 프로토콜 스택을 사용하여 이종 서버의 자원을 모두 활용할 수 있으면서도 사용자에게는 친숙한 윈도우 환경을 제공해주는 SBC 서비스를 목표로 개발 중이다. 현재는 Microsoft 의 터미널 서버(Terminal Server)[3]만을 이용하여 서비스가 가능한 상태이다. 개발된 네트워크 프레임워크는 이 ACT-SBC 의 터미널 서버 시스템들을 대상으로 관리자 네트워크를 구축하고 테스트 하였다.

일반적으로 SBC 시스템의 경우 많은 사용자가 한 대의 서버에서 동시에 작업을 진행하므로 서버의 성능이 매우 중요한 요소이다[4]. 따라서 SBC 시스템을 구축할 때에는 많은 수의 서버 컴퓨터를 사용해야만 한다. 이에 따라 많은 수의 터미널 서버를 효율적으로 관리할 수 있는 매니지먼트 네트워크가 필요하게 되었다. 이에 SBC 관리자가 자신이 필요한 기능만을 구현하여 손쉽게 매니저 네트워크를 구축할 수 있는 네트워크 프레임워크를 개발하였다.

이 논문에서는 2 절에서 관리 네트워크와 네트워크 프레임워크가 갖추어야 할 요구 사항과 개발 환경을 살펴보고, 3 절에서 전체 구조와 구현 방법에 대해서 설명한다. 4 절에서는 구성된 네트워크 프레임워크의 성능을 테스트하고 결과를 정리한 후, 마지막으로 5 절에서 결론을 맺는다.

교신 저자(Corresponding Author)
※ 본 과제(결과물)는 교육인적자원부, 산업자원부, 노동부의 출연금으로 수행한 산학협력중심대학육성사업의 연구결과입니다.

2. 개발 환경 및 요구사항

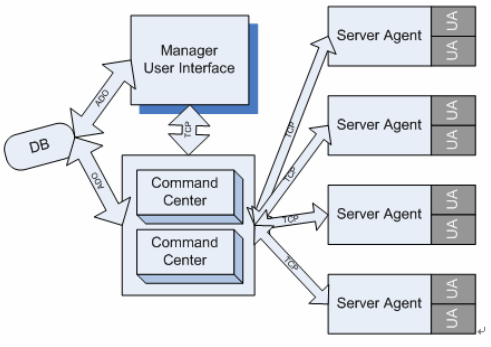
ACT-SBC 는 최신 버전인 터미널 서버 6.0 을 이용한다. 터미널 서버 6.0 이 RDC(Remote Desktop Connection)[5] 시스템을 SBC 로 사용할 때 유리한 여러 가지 기능을 제공하기 때문이다. 터미널 서버 6.0 은 Windows Server 2008[6]에서만 제공되므로 Windows Server 2008 Beta 를 사용하여 전체 네트워크 프레임워크를 개발하였다. RDC 클라이언트는 모든 버전의 Windows 시스템에서 무료로 사용 가능하다[7]. 전체 개발 환경의 편의성을 위해 데이터 베이스 서버나 미들웨어 플랫폼도 동일한 환경으로 설정하였다. Windows Server 2008 은 아직 정식으로 발매되지 않아서 베타 버전을 사용하였으며, 개발용 컴퓨터들에서는 Windows SDK for Longhorn Beta 를 사용하였다.

RDC 를 이용한 SBC 시스템에 가장 문제가 되는 부분은 한 대의 터미널 서버가 사용자 몇 명에게 원활한 사용자 환경을 제공해 줄 수 있는 가 하는 점으로, 서버의 시스템 사양과 운영체제의 종류 그리고 네트워크의 상태에 따라 다양한 경우의 수가 발생한다. ACT-SBC 에서는 서버당 최대 이용자를 15 명으로 설정하고, 동시 이용자 수의 1/12 이상의 터미널 서버를 활용하기를 권유한다.

이러한 요구 사항에 맞추어 관리자 시스템은 많은 수의 서버를 동시에 관리할 수 있으며 관리자 시스템이 터미널 서버에 끼치는 영향은 최소화 해야 한다. 또 전체 서버들을 감시하여 주기적으로 서버의 상태를 데이터 베이스에 기록하고 필요에 따라 관리자에게 보고할 수 있어야 한다.

3. 구조 및 구현

기본적으로 이 프레임워크는 다수의 터미널 서버 관리 시스템 구축을 목표로 제작하였다. 관리 시스템의 기본적인 전체 구조는 그림 2 와 같다. 네트워크 프레임워크의 유연성을 보장하기 위해 모든 모듈은 동적 연결 라이브러리(DLL, Dynamic Linked Library) 형태로 제작하였다.



(그림 2) 전체 구조도

일반적으로 이런 관리 네트워크 시스템은 각각의 터미널 서버로부터 필요한 정보를 주기적으로 데이터베이스에 기록하거나 관리자에게 전달하여 주는 중간 서버와 터미널 서버에서 필요한 정보를 입수하여 중간서버에 반환하는 서버용 프로그램, 그리고 관리자에게 편리한 환경을 제공하는 관리자 인터페이스로 구성되어야 한다. 이런 구조를 구축하기 위해 이 네트워크 프레임워크에서는 중간 서버 역할을 하는 CC 를 제작할 수 있는 CC 라이브러리와 터미널 서버에서 동작하며 중간 서버와 교신하는 SA 라이브러리, 세션 별로 동작하는 UA 라이브러리 그리고 관리자의 인터페이스와 CC 의 통신과 명령 전달을 책임지는 관리자 인터페이스 라이브러리를 제공하고 있다.

관리자 네트워크를 구축할 때 이 네 가지 라이브러리를 사용하여 필요한 형태로 최종 관리 네트워크를 구축할 수 있다. 이 네트워크 프레임워크는 각 모듈별로 네트워크 상의 연결과 각 연결의 신뢰성을 보장하고, 동적으로 실행되는 Plug-in 을 제작할 수 있는 라이브러리를 포함한다.

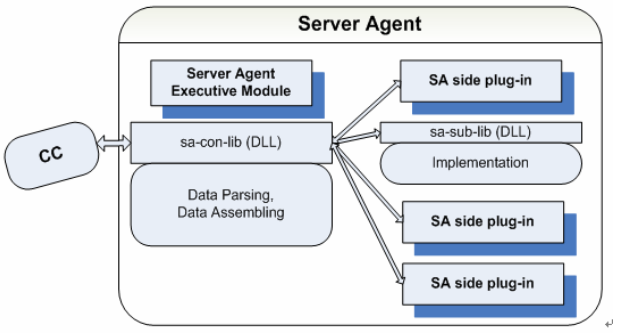
■ Network and Connectivity

이 네트워크 프레임워크가 제공하는 가장 중요한 기능은 많은 수의 터미널 서버를 관리할 수 있으며 신뢰성 있는 네트워크를 구축할 수 있는 환경이다.

이런 환경을 제공하기 위해 이 네트워크 프레임워크는 기본적으로 요청-응답 형태의 연결 구조를 사용한다. CC 와 SA, 관리자 인터페이스 각각의 모듈은 열린 포트를 가지고 있어서 다른 모듈로부터의 요청을 받거나 자신이 요청한 데이터에 대해 응답을 되돌려 줄 때 이 포트를 사용한다. 모든 연결에는 TCP 를 사용하고 Time out 을 설정하여 메시지 전달 실패 시에는 재전송을 요구한다. 또한 각 모듈들 스스로 오염된 메시지에 대해서는 재전송을 요구한다. 관리 대상인 터미널 서버들과 CC 는 같은 인트라넷에 구축되어 있는 것이 유리하다. 그렇지 않을 경우에는 Time out 조절을 통하여 네트워크 상태에 따른 유연성을 제공한다. 관리자 인터페이스는 기본적으로 인트라넷이 아닌 접속 지점을 가정하였으며 동시에 1 대의 관리자 인터페이스가 존재한다. 여러 명의 관리자에 대한 정책은 관리자 네트워크의 구현자가 적절히 정해야 한다.

■ Server Agent(SA) Library

SA 는 터미널 서버에서 동작하며 필요한 정보를 수집하여 CC 로 전달할 때 사용된다. 따라서 SA 라이브러리는 그런 여러 가지 기능들을 실행할 수 있는 환경을 제공하도록 제작되었다.



(그림 3) 서버 에이전트 라이브러리의 구조

SA 는 터미널 서버에서 동작하여 사용자의 환경에 직접적으로 영향을 끼친다. 따라서 메모리나 프로세서의 사용량을 최소화 하여 관리 시스템으로 인한 서버의 효율 저하를 줄여야 한다.

그림 3 에는 SA 의 구성 모습이 나타나 있다. SA 라이브러리는 크게 CC 로부터 명령을 받아서 적절한 Plug-in 모듈을 메모리에 탑재하여 실행시키고 그 결과를 다시 CC 로 반환하는 일련의 과정을 처리하는 SA Con Lib 와 Plug-in 을 작성할 수 있는 라이브러리로 이루어져 있다. 두 라이브러리에 필요한 형태의 실행 모듈을 추가하면 SA 가 된다.

Plug-in 은 필요할 때만 메모리에 탑재되어 사용되고 실행이 종료되면, 메모리에서 제거되므로 서버에 끼치는 영향을 최소화 할 수 있다. SA Con Lib 는 CC 로부터 명령을 전달 받아서 작업을 수행하며 모든 동작은 CC 의 명령에 의존적이다. 터미널 서버에서 주기적으로 일어나야 하는 작업은 SA 가 수행하는 것이 아니라 CC 가 주기적으로 작업 명령을 내려야만 한다. 이런 방법으로 SA 가 서버의 성능에 끼치는 영향을 최소화하였다.

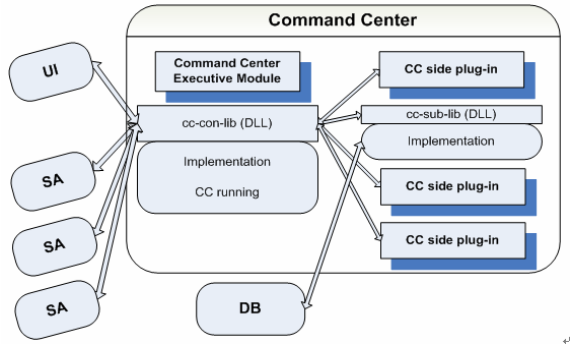
■ User Agent(UA) Library

UA 는 일반적으로 터미널 서버에 접속한 모든 세션마다 실행되며 관리자의 명령이나 미리 정해진 정책에 따라 CC 가 명령을 내리면 사용자의 작업을 강제로 중단하거나 관리자로부터 세션 사용자에게 메시지를 전달하는 작업 등에 이용된다. UA 라이브러리는 SA 라이브러리에 종속적으로 실행된다. 실제로 명령을 전달받거나 응답을 CC 또는 관리자 인터페이스로 전달 할 때는 SA 를 통해 처리한다. 따라서 UA 는 내부적으로 네트워크 관련 기능을 사용하지 않고 제작되었다. UA 는 마치 SA 의 Plug-in 처럼 SA 와 함수 호출을 통해 직접적으로 데이터를 주고 받는다.

■ Command Center(CC) Library

CC 는 이 네트워크 프레임워크의 가장 핵심적인 부분이다. 하나의 관리자 네트워크에 여러 개의 CC 가 존재할 수 있다. 특히, 관리해야 할 터미널 서버가 많아질 경우 CC 의 수를 늘림으로 시스템 효율을 손쉽게 증대할 수 있다. CC 는 (ㄱ) 관리자 인터페이스와 터미널 서버와의 연결 통로의 역할과 (ㄴ) 주기적으로 이루어지는 터미널 서버들의 상태 점검 역할 그리고 (ㄷ) 네트워크의 신뢰도를 관리하는 역할을 한다.

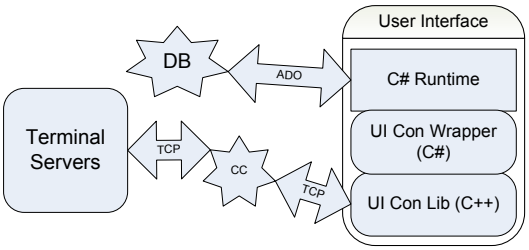
최초로 CC 가 동작할 때 관리자 네트워크에 사용될 CC 와 관리할 터미널 서버의 목록을 설정하여야 한다. 설정된 CC 와 터미널 서버 목록은 데이터 베이스를 통해 저장되어 있으며, 관리자가 수정할 수 있다. 만약 네트워크에 등록된 CC 가 2 개 이상일 경우 CC 끼리 동기화 채널을 구성한다. 동기화 채널을 통해 CC 끼리 서로 태스크 큐의 크기를 공유하며, 다음 태스크가 발생하면 가장 작은 크기의 태스크 큐를 가지고 있는 CC 가 작업을 전담하게 된다. CC 는 태스크 큐 정보를 주기적으로 등록해야 한다. 주기 이상으로 오래 태스크 큐 크기가 업데이트 되지 않는 CC 가 있으면 CC 끼리 작업을 분배하여 동작하지 않는 CC 에 대한 대응을 한다. 관리자 인터페이스 역시 이 동기화 채널을 공유하여 항상 태스크 큐가 가장 작은 CC 를 통해 태스크를 진행한다.



(그림 4) 코맨드 센터 라이브러리의 구조

그림 4 에서는 CC 의 개략적인 모습을 살펴볼 수 있다. CC Con Lib 는 SA 라이브러리의 SA Con Lib 와 통신한다. CC 역시 Plug-in 형태의 모듈을 제작하여 사용할 수 있다. CC 의 Plug-in 은 SA 의 Plug-in 과 비슷한 형태이지만 CC 의 기능을 확장하는 데 주로 사용된다는 차이점이 있다. 데이터 베이스에 접근하는 모듈이나 주기적으로 SA 에 명령을 내리는 모듈 등을 Plug-in 으로 제작하여 CC 의 기능을 확장할 수 있다.

관리자 인터페이스에서 내리는 명령은 반드시 CC 를 통해서 SA 로 전달 된다. 이 과정에서도 CC 는 네트워크의 무결성을 보장한다. 장시간 응답이 없는 서버는 재작동을 요구할 수도 있다. 또 Plug-in 으로 제작된 모듈들 역시 네트워크 프레임워크에서 안정적인 전달과 동작을 보장한다.



(그림 5) 관리자 인터페이스의 동작 구조도

■ Manager Interface Library

그림 5 는 이 프레임워크 테스트를 위해 제작한 관리자 인터페이스의 예시이다. 이 관리자 인터페이스는 C#으로 제작하였으며, 관리자 인터페이스 라이브러리를 이용하기 위해 Wrapper 인터페이스를 추가하였다

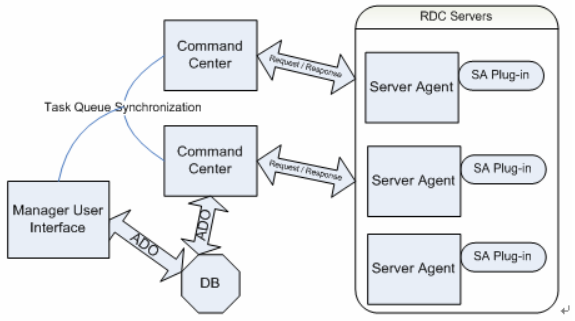
전체 CC 와 터미널 서버 목록을 등록하여 관리 네트워크 전체 구조를 설정하는 작업과 관리자에게 보고하는 기능이 관리자 인터페이스에서 가장 먼저 이루어져야 할 작업이다. 이 과정을 거친 후, 관리자는 일반적인 관리를 수행할 수 있다.

실행 모듈에서 원하는 터미널 서버의 상태를 확인하기 위해 특정 명령을 수행하도록 네트워크 프레임워크의 함수를 호출하면 매니저 인터페이스 라이브러

리의 UI Con Lib 가 이 작업을 처리한다. UI Con Lib 은 CC 의 동기화 채널을 공유하므로 태스크 큐의 크기가 가장 작은 CC 를 선택할 수 있다. 또한 필요하다면 정보를 받아서 실행 모듈로 돌려주는 작업도 매니저 인터페이스 라이브러리에서 이루어진다. 따라서 실행 모듈에서는 네트워크 경로나 CC 를 고려할 필요 없이 터미널 서버의 정보를 얻거나 명령을 내릴 수 있다.

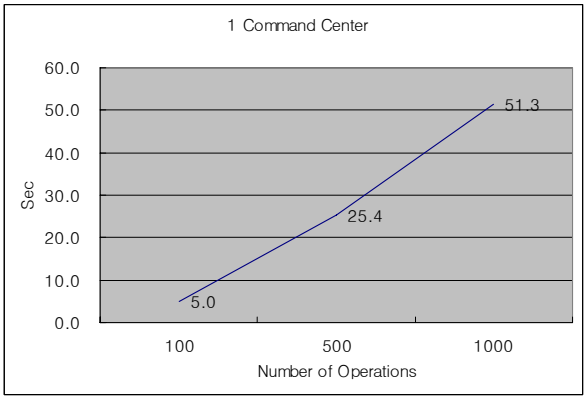
4. 테스트 및 결과

실제로 관리 네트워크를 구축하여 네트워크 프레임 워크의 성능을 테스트하였다. 100Mbps 로 구성된 인트라넷에 관리자 네트워크를 구축하였으며, CC 의 숫자를 변경시켜 가며 테스트 하였다. 데이터 베이스 서버와 CC 는 독립적으로 동작하며 ADO(Active Data Object)[8]를 이용하여 데이터를 주고 받았다. 터미널 서버는 총 51 대에 255 개의 SA 를 구동하였다. CC 와 SA 는 각각의 라이브러리를 구동하는 코드만으로 작성하였다. 관리자 인터페이스는 임의로 프로세스와 세션 정보를 읽어오는 단일 명령어를 설정하고 이 명령어를 수행하도록 SA 플러그 인을 생성하였다. 전체 서버의 도식은 그림 6 와 같다.



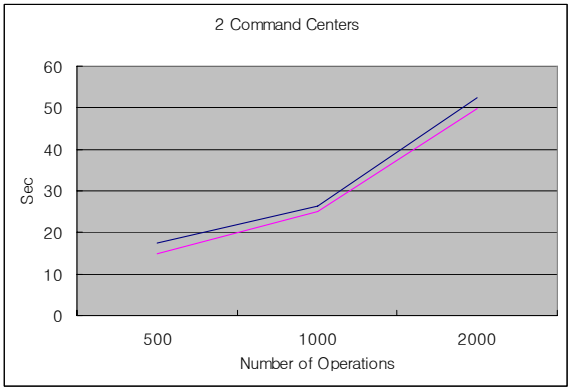
(그림 6) 테스트 네트워크 구조도

한 대의 CC 에 임의의 터미널 서버들로 상태 수집 명령 100 개/500 개/1000 개를 수행할 경우의 전체 수행 시간이 도표 1 에 나타나 있다. 이 결과는 1000 대의 서버 상태를 수집하는데 1 분 이내의 시간이 걸린다는 것을 시사한다.



(도표 1) 한 대의 코맨드 센터 성능 테스트

두 대의 CC 에 임의의 터미널 서버들로 상태 수집 명령 500 개/1000 개/ 2000 개를 수행한 경우 전체 수행 시간이 도표 2 에 나타나 있다. 작업은 비교적 균일하게 두 CC 로 분배 되었다. 두 CC 사이의 수행 시간 차이는 CC 1 의 경우 네트워크 재 설정이 불규칙하게 약간 더 많이 일어 나서 생긴 결과로 사실상 오차가 거의 없다고 생각할 수 있다. 이 결과에 따르면 관리할 서버의 수가 늘어날 때 CC 의 수를 늘리는 것만으로 전체 매니저 네트워크의 성능을 향상 시킬 수 있음을 알 수 있다.



(도표 2) 두 대의 코맨드 센터 성능 테스트

5. 결론

이 네트워크 프레임워크를 이용하면 짧은 개발 기간과 필요한 기능만 구현하는 정도의 저비용으로 신뢰할 수 있고 Plug-in 을 이용해 손쉽게 확장이 가능하며 다수의 터미널 서버를 관리할 수 있는 네트워크를 구축할 수 있다. 향후 실제로 서비스에 적용할 수 있는 매니저 네트워크를 이 네트워크 프레임워크를 이용하여 개발할 계획이다. 또 CC 라이브러리의 성능 향상을 위해 멀티 태스크 큐를 사용하는 방법을 고려중이며, 네트워크 프레임워크를 범용 다중 서버 관리 시스템에도 사용할 수 있도록 개선할 계획이다.

참고문헌

[1] “SBC(Server--Based Computing)을 이용한 정보보호 시스템” ㈜어울림정보통신 White Paper. June 2006.
[2] ㈜위즈엔텍. http://www.wizntec.com
[3] Terminal Server. http://www.microsoft.com/resources/sam/lic_cal_win2k_ term_services.mspx
[4] J. Nieh and S. J. Yang, *Measuring the Multimedia Performance of Server Based Computing* Proceedings of the Tenth International Workshop on Network and Operating System Support for Digital Audio and Video, Chapel Hill, NC, June 2000.
[5] Remote Desktop Connection. http://technet2.microsoft.com/windowsserver/en/library/a175573a-bb3d-4752-a412-8db391535a891033.mspx?mfr=true
[6] Windows Server 2008. http://www.microsoft.com/windowsserver2008
[7] Remote Desktop Connection Download http://www.microsoft.com/windowsxp/downloads/tools/rdclientdl.mspx
[8] Active Data Object. http://msdn2.microsoft.com/en-us/library/ms675532.aspx
[9] The ADAPTIVE Communication Environment. http://www.cs.wustl.edu/~schmidt/ACE.html