Journal of Digital Contents Society Vol. 10 No. 4 Dec. 2009(pp. 509-520)

X 프로토콜 기반의 애플리케이션을 통한 씬-클라이언트 프레임워크 설계

송민규*

요 약

네트워크 및 애플리케이션 기술의 발전은 컴퓨터, 모바일 시스템을 비롯한 정보기기의 활용에 커다란 변화를 야기시켰다. 60-70년대의 메인 프레임을 시작으로 80년대의 서버-클라이언트 패러다임을 거쳐 90년대 이후의 네트워크 컴퓨터 형태로 발전하는 과정에서 현재 컴퓨터 시스템은 독립적인 물리적 시스템에서 상호보완적인 네트워크 기반의 가상 시스템으로 진화하고 있다[1][2]. 네트워크 기반의 시스템에서 작업 수행에 필요한 애플리케이션과 데이터는 로컬 시스템에 해당하는 클라이언트가 아닌 서버에 저장된다[1]. 사용자는 네트워크를 통해 서버 상의 애플리케이션, 데이터를 마치 로컬 환경에서와 같이 활용할 수 있으며, 이러한 메커니즘에 의하여 클라이언트는 보다 경량화, 네트워크 친화적 시스템으로 발전해나가고 있다. 본 논문에서는 이러한 씬-클라이언트를 보다 효율적으로 구현할 수 있는 가능성 있는 방안에 대해 논의하기로 한다. 서버 상의 애플리케이션과 데이터를 마치 로컬 환경에서 활용할 수 있도록 본 논문에서는 X프로토콜을 활용하였다. 기존의 단일화 된 서버 시스템과는 달리 프락시를 미들-티어로 설계하여 QoS 및 세션의 영속성을 제고하였다. 씬-클라이언트와 서버에 각각 X서버, Xvfb(X virtual frame buffer)를 구현하였고 세션 관리를 위하여 XSMP(X Session Management Protocol)을 적용하였다. 이를 통하여 최종적으로 단순한 서버 디스플레이 전달을 넘어, 서버 상의 애플리케이션이 네트워크를 경유하여 씬-클라이언트에 원격 애플리케이션으로 전달되도록 하는 씬-클라이언트 프레임워크를 제안하였다.

A Design of Framework for Thin-Client by using X Protocol based Application

Min-Gyu Song*

Abstract

The advancement of network & application technology causes a major change for the use of IT(Information Technology) equipment, including computer and mobile system. In the process from beginning with main frame in the 1960s and 70's, through the server-client paradigm in the 1980s and toward the development of network computer since 90's, computer systems are now evolutioning from isolated physical system to complementary network based virtual system[1][2]. In network based computer system, application and data required for operation are stored at not client as local system, but at server[1]. User can use application & data on a server as if those are on a local client, and a client is now toward a developing thin and network friendly system. In this paper, we discuss possible ways for the efficient implementation of thin-client. For the use of remote application & data as if in local environment, we make use of X protocol. Unlike formal simple Client - Server paradigm, we design a Proxy for middle-tier server for the improvement of QoS and session persistence. X server, Xvfb(X virtual frame buffer) are implemented on thin client and Server, respectively and we applied XSMP(X Session Management Protocol) to our framework for session management. In the end, beyond simple transfer of server display, we suggest thin client framework for the transfter of remote server application over internet.

Keywords : Thin Client, X, NX, Pseudo Server, XSMP, Middle-Tier, Session Persistence

1. 서 론

네트워크 기술의 발전은 단순한 네트워크 형태의 진화만이 아닌, 컴퓨터 시스템의 형태 및 활용에 있어서도 막대한 변화를 야기시키고 있다. 네트워크로부터 격리된 컴퓨터는 제한된 성능, 용량의 독립적 시스템에 불과하지만 네트워크에 연결됨으로써 이러한 한계를 극복할 수 있다. 수많은 컴퓨터 시스템이 서로 연결되어 하나의 거대한 네트워크가 이루어지는 특성에 의하여 임의의 컴퓨터 시스템은 네트워크를 통하여 타 시스템의 기능 및 자원을 효율적으로 활용할 수 있다[1]. 성능, 안정성 측면에서 네트워크의 급격한 발전은 원격의 컴퓨터를 마치 로컬 컴퓨터와 같이 활용하는 것을 가능하게 하였다. 수천 km 떨어진 시스템이라도 양질의 네트워크에 연결되어 있다면 그는 더 이상 독립적인 시스템이 아닌, 네트워크를 통해 통합된 거대한 컴퓨터 시스템의 한 부분에 해당하며 다수의 시스템이 결집된 네트워크는 이미 그 자체가 컴퓨팅 플랫폼으로서의 기능을 수행한다[1][3]. 네트워크 상에서 각 시스템의 CPU, 메모리, 디스크가 하나의 네트워크 컴퓨터를 구성하는 리소스로 활용되는 것과 마찬가지로 타 시스템의 애플리케이션, 함수는 네트워크 기반 플랫폼에서 호출 가능한 애플리케이션, 함수에 해당한다. 이러한 특성으로 원격의 네트워크 상에 위치한 컴퓨터 시스템의 애플리케이션을 로컬에서와 같이 활용할 수 있으며, 이러한 메커니즘에서 클라이언트를 경량화하는 것이 가능하다[4]. 이러한 시스템은 씬-클라이언트로 불리고 있으며 본 논문에서는 이러한 씬-클라이언트를 보다 효율적으로 구현할 수 있는 접근법 및 방법에 대해 살펴보고자 한다.

본 논문에서는 씬-클라이언트 시스템을 구현하는 가능성 있는 한 방법으로 X 프로토콜을 활용하였다. X 프로토콜은 로컬에서 원격 시스템의 애플리케이션을 호출할 수 있도록 하는 그래픽 기반의 프로토콜로서 원격 애플리케이션을 마치 로컬 시스템 상의 애플리케이션과 같이 활용하는 것을 가능하게 한다[5][6]. 이러한 특성의 X 프로토콜을 기반으로 우리는 네트워크 상의 각 시스템에 X server, Xvfb(X virtual frame buffer)를 구현하였으며 이를 기반으로 원격 애플리케이션을 실행할 수 있는 시스템 아키텍처를 설계하였다. 원격 애플리케이션 호출에 있어 가장 큰 문제점은 네트워크 상태에 종속적이고 민감하다는 것이다[7]. 네트워크가 안정적이고 정상적으로 동작할 때에는 애플리케이션 호출에 큰 문제가 없지만 그렇지 않은 경우에는 원격 시스템 상의 애플리케이션 활용은 큰 문제점을 내재하게 된다. 씬-클라이언트의 가장 큰 취약점이기도 한 이러한 특성을 극복하기 위해서는 네트워크의 상태에 관계없이 애플리케이션을 지속적으로 활용할 수 있어야 한다. 본 논문에서는 그를 위한 방안으로서 프락시 기능을 수행하는 Xvfb를 미들-티어 서버 상에 구성하였으며 안정적인 X 프로토콜 통신이 수행될 수 있도록 동일한 LAN 상에 터미널 서버와 애플리케이션 서버를 구성하였다. 또한 X 애플리케이션에 대한 요청/응답을 처리하는 모듈과 X애플리케이션을 동일한 시스템 상에 구현함으로써, 네트워크의 상태의 영향을 최소화하였고 XSMP(X Session Management Protocol)을 활용하여 원격 애플리케이션 호출 세션이 지속적으로 유지될 수 있도록 하였다[8].

본 논문은 다음의 순서에 따라 구성되어진다. 본 서론에 이어 2장에서는 씬-클라이언트의 정의 및 동작 메커니즘에 대해 알아볼 것이며 3장에서는 현 씬-클라이언트의 문제점 및 개선방안에 대해 살펴보고자 한다. 4장에서는 X 프로토콜을 기반으로 한 아키텍처에 및 동작 메커니즘에 대해 알아볼 것이며 5장에서는 이를 실제적으로 시스템에 구현할 수 있는 각 요소 기술의 활용방안에 대해 논의할 것이다. 6장에서는 논의된 개선사항 및 요소기술을 기반으로 실제 씬-클라이언트 프레임워크를 설계할 것이고 마지막 장 결론에서 씬-클라이언트의 전망 및 향후 가능성을 진단하는 것으로 본 논문의 마무리 짓고자 한다.

※ 제일저자(First Author) : 송민규
접수일:2009년 11월 26일, 완료일:2009년 12월 11일
* 한국천문연구원 전파천문연구부
mksong@kasi.re.kr

2. 씬-클라이언트의 개념 및 동작 메커니즘

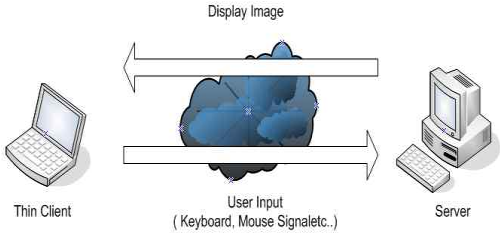
씬-클라이언트는 기존 클라이언트-서버 모델과는 달리 클라이언트의 기능을 경량화시킨 시스템으로서 작업 수행에 필요한 데이터 및 애플리케이션은 서버에 위치한다[1]. 본 장에서는 씬-클라이언트의 개념 및 동작 원리에 대해 서버 기반 컴퓨팅과 플러그-인 기반 컴퓨팅을 중심으로 살펴보고자 한다.

2.1 씬-클라이언트의 개념

씬-클라이언트는 그 이름에서 알 수 있듯이 작고 가벼운 기능이 최소화된 클라이언트를 의미한다. 기존 클라이언트/서버 환경에서 작업 수행에 필요한 애플리케이션과 데이터는 클라이언트에 위치하였고 사용자는 네트워크 연결에 관계없이 독립적으로 원하는 작업을 수행하였다[2]. 하지만 네트워크 기술의 급격한 발전에 따라 시스템의 거리에 관계없이 각 컴퓨터 시스템이 하나로 통합되는 것이 현실화되었고 이에 따라 원격 시스템에 위치한 애플리케이션을 마치 로컬에서와 같이 호출하는 것이 가능하게 되었다. 이에 따라 기존에 클라이언트에 있던 애플리케이션, 데이터는 서버로 이동되었으며 클라이언트는 사용자 입력 및 서버로부터의 결과를 처리하는 형태에 이르게 되어 씬-클라이언트로서 나타나게 되었다.

2.2 서버 기반 컴퓨팅

1981년 IBM에서 출시한 PC(Personal Computer)는 컴퓨터 시스템 활용에 있어 크나큰 진전을 불러일으켰다. CPU, HDD, 메모리 등 별도의 디바이스를 탑재한 PC의 등장으로 사용자들은 전산실이라는 제한된 환경에서 관리자의 승인을 받아 시스템을 활용하던 한계를 극복하게 되었다. 하지만 PC의 경우 세 가지 크나큰 단점을 가지고 있다. 먼저 초기 도입 비용이 크다는 것이 그 첫 번째이고 시스템 유지 관리, 소프트웨어 업그레이드 등의 TCO(Total Cost of Ownership)가 크다는 것이 두 번째, 그리고 데이터 보안에 취약하다는 것이 그 세 번째에 해당한다[4]. 일반 사용자의 경우 PC 상에서 수행하는 작업의 상당수가 웹 서핑, 메일 작성, 문서 작업 등이라는 점에서 기존의 PC는 오버 스펙이라 할 수 있다. 또한 대기업의 경우 한해 시스템 유지 관리 비용은 날이 갈수록 증가하는 추세에 있으며 주요 정보의 유출로 인하여 각 기업체에서는 이를 방지하기 위한 보안 강화에 역점을 기울이고 있는 상황이다[7]. 이에 따라 대두된 것이 서버 기반의 컴퓨팅으로서 네트워크 기술에 발전에 따라 그 구현이 가능하게 되었다.



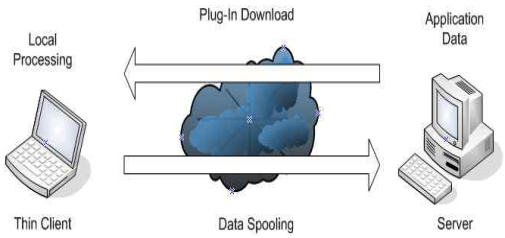
(그림 1) 서버 기반 컴퓨팅의 메커니즘

서버 기반의 컴퓨팅에서 애플리케이션과 데이터는 기존의 클라이언트/서버 방식과는 달리 서버에 저장된다. 클라이언트는 서버 측으로 사용자 입력을 전송하고 서버로부터 디스플레이 이미지를 전송받는다. 이러한 과정을 통하여 마치 로컬 상에서 실제 작업을 수행하는 것 같은 효과를 얻을 수 있으며 관련 프로토콜로 VNC(Virtual Network Computing), RDP(Remote Desktop Protocol), ICA(Independent Computing Architecture) 등이 활용되고 있다. 서버 기반 컴퓨팅은 시스템 유지, 보수의 측면에서도 최적의 선택이라 할 수 있는데 이는 서버상의 애플리케이션 업그레이드하는 것으로 네트워크 상의 모든 클라이언트가 최신의 애플리케이션을 활용하는 것이 가능하기 때문이다[4][7]. 서버 기반 컴퓨팅의 개략도를 도시하면 위 (그림 1)과 같다.

2.3 플러그-인 기반 컴퓨팅

서버 기반의 컴퓨팅에서 클라이언트의 입력신호는 서버로 전달된다. 모든 작업 및 연산처리는 애플리케이션, 데이터가 저장된 서버에서 이루어지며 서버의 디스플레이 이미지는 클라이언트로 전달되었다. 입력 신호와 디스플레이 이미지가 전달되는 서버 기반 컴퓨팅과는 달리 플러그-인 기반의 컴퓨팅에서는 실행 가능한 애플리케이션

이 클라이언트로 전달된다[9]. 일반적으로 이는 서버 상에 저장된 애플리케이션을 웹 애플리케이션의 형태로 호출하는 방식으로 실행되며 사용자는 네트워크를 경유하여 원하는 작업을 수행하게 된다. 모든 연산 처리가 전적으로 서버에 의존하는 서버 기반 컴퓨팅과 달리 플러그-인 기반의 컴퓨팅에서 클라이언트는 연산의 일부분을 담당하며 그에 필요한 임의의 애플리케이션이 웹 브라우저와 상호작용하여 설치 및 운용된다. 이에 관련된 네트워크 기술로 ActiveX, 자바 애플릿, 플래시 등이 있으며 클라이언트는 임의의 애플리케이션을 웹 브라우저 상에서 실행하는 것이 가능하다. 플러그-인 기반의 컴퓨팅에서 클라이언트는 서버 기반 컴퓨팅 방식보다 보다 많은 역할을 수행한다. 작업 수행에 필요한 임의의 프로그램은 플러그-인 모듈의 형태로 클라이언트에서 실행되며, 웹 브라우저가 클라이언트로서의 기능을 수행한다. 이러한 플러그-인 기술 기반의 컴퓨팅 개념을 그림으로 도시해보면 아래와 같다[9].



<그림 2> 플러그-인 기술을 활용한 씬-클라이언트 컴퓨팅

3. 씬-클라이언트의 문제점 및 개선 방안

씬-클라이언트는 네트워크 발전에 따라 등장한 시스템이기에 네트워크의 상태에 종속적일 수 밖에 없다[2][6]. 이는 씬-클라이언트의 장점으로 나타나기도 하지만 단점으로 작용하기도 하는데 본 논문에서는 씬-클라이언트의 이러한 단점 및 문제점에 대해 살펴보고 그를 극복할 수 있는 방안에 대해 알아보고자 한다.

3.1 씬-클라이언트의 단점 및 한계

씬-클라이언트는 네트워크에 종속적이다. 네트워크 성능의 증가에 따라 타 시스템 상에서 작업을 수행하고 그를 로컬 상에 디스플레이하는 형태가 나타나게 되었지만 네트워크가 고품질의 성능을 제공하지 못하거나 연결에 문제가 있다면 씬-클라이언트는 정상적으로 동작할 수 없다[6]. 네트워크 연결에 문제가 있다면 실제 작업을 수행하는 애플리케이션이 실행되는 서버 시스템에 접속할 수 없거나 진행 중인 작업의 내용이 유실되는 상황이 발생할 수 있다. 네트워크 환경이 불안정하고 장애가 발생하더라도 사용자가 기존 세션 정보를 유지하고 작업을 지속할 수 있기 위한 방안이 마련되어야 할 것이며 그를 지원할 수 있는 방향으로 씬-클라이언트 지원의 프레임워크가 구축되어져야 할 것이다.

더불어 단순히 서버의 디스플레이 이미지를 전송받는 기존 씬-클라이언트의 동작 메커니즘 역시 개선되어질 필요가 있다. 씬-클라이언트 환경에서 네트워크는 하나의 컴퓨팅 플랫폼이자 애플리케이션 전달 인터페이스에 해당한다. 기존과 같은 서버 디스플레이를 이미지 형태로 단순 전송하는 방식이 아닌 서버 상의 애플리케이션을 실제로 호출하여 네트워크를 통해 전달하는 새로운 방법이 마련되어야 하며 이러한 요건이 충족되었을 때 씬-클라이언트의 효율성은 더욱 배가될 수 있다.

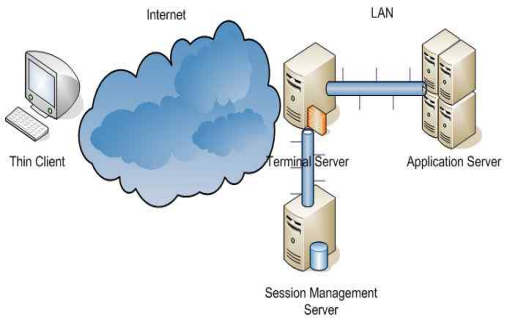
3.2 문제점 극복을 위한 시스템 설계 방안

상기 지적한 특성을 구현하기 위해서는 시스템 설계 과정에서 많은 변화가 필요하다. 단순한 클라이언트-서버 방식으로는 불안정한 네트워크 상에서 사용자 세션을 일관되게 유지할 수 없다. 또한 인터넷과 같은 WAN(Wide Area Network) 상에서 네트워크의 불안정성을 완벽히 제거하는 것도 사실상 불가능하다. 이러한 문제점을 극복하기 위한 방안으로 본 논문에서는 두 가지 방안을 사용하였다. 첫째, 서버 시스템을 단일 시스템을 넘어 그 역할 및 기능에 따라 크게 터미널 서버와 애플리케이션 서버, 세션관리 서버로 세분화하였다. 터미널 서버는 씬-클라이언트와 애플리케이션 서버 사이에 위치하는 일종의 미들-티어 서버로서 두 시스템 간의 통신을 관장하는 역할을 하며 애플리케이션 서버는 작업 수

행에 필요한 실제 애플리케이션이 실행되는 시스템에 해당한다. 터미널 서버의 주요한 특징으로 세션의 영속성을 들 수 있으며, 애플리케이션 서버의 X 애플리케이션에 대한 복사본을 활용하여 네트워크 장애가 발생하더라도 일관된 세션을 유지될 수 있도록 하였다. 네트워크 장애로 씬-클라이언트 상의 X 디스플레이가 중단된다 하더라도 터미널 서버에 복사본과 세션 정보가 보관되어 있기에 재연결 시 세션을 원래대로 유지/복구시키는 것이 가능하다.

둘째, 본 논문에서는 터미널 서버와 애플리케이션 서버를 서로 다른 네트워크가 아닌 동일한 LAN(Local Area Network) 상에 구축함으로써 통신에 있어서의 QoS가 대폭 개선될 수 있도록 하였다. 이더넷 기반의 LAN 상에서는 보다 많은 데이터를 빠르고 안정적으로 전송할 수 있다. 따라서 애플리케이션 서버의 X 애플리케이션과 터미널 서버 상의 X 디스플레이의 복사본 간의 신뢰성 있는 통신을 구현할 수 있으며 세션의 영속성을 보장할 수 있다. 뿐만 아니라 X 서버의 상태, 용량을 질의하는 데이터 패킷의 송수신이 상당한 대역폭을 제공하는 내부망에서 이루어짐으로 인터넷 상에서 전달되는 데이터 패킷을 효과적으로 감소시킬 수 있다.

상기 언급한 두 가지 개선 방안을 기반으로 구현하고자 하는 시스템의 대략적 개요를 도시해보면 아래와 같다.



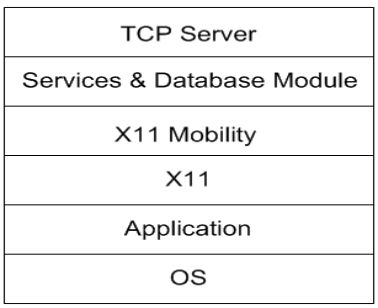
(그림 3) 씬-클라이언트 시스템의 기본 구성

4. X 프로토콜 기반의 씬-클라이언트 아키텍처

본 장에서는 X 프로토콜을 기반으로 씬-클라이언트를 구현하기 아키텍쳐 설계에 대해 알아보고자 한다. 그를 위한 계층화된 시스템 아키텍쳐에 대해 먼저 살펴보고 X 클라이언트, X 서버의 활용방안에 대해 논의하고자 한다.

4.1 기본 레이어 구성

씬-클라이언트는 로컬 환경이 아닌 원격에 위치한 서버 상의 애플리케이션을 호출하고 네트워크 상의 그 어디에서라도 일관된 컴퓨터 작업 환경을 구현하여야 한다[1]. 따라서 씬-클라이언트 시스템을 설계함에 있어 그 기능 및 역할에 따라 아키텍처의 세분화가 필요가 있다. 본 논문에서 구현하고자 하는 시스템은 TCP/IP 네트워크 상에서 X 프로토콜 사용을 기반으로 하며 서버 상의 애플리케이션을 데이터 유실 없이 안정적으로 클라이언트에게 전송할 수 있어야 한다. 또한 하나의 클라이언트가 아닌 다수의 클라이언트에 멀티플렉싱하는 기능도 갖추어야 할 것이다. 이와 같이 X 프로토콜을 기반으로 임의의 클라이언트에게 서버 상의 애플리케이션을 송신하고 세션을 유지 및 전달하기 위해서 본 논문에서는 그를 위한 가상 서버(Pseudo Server)를 지정하였으며, 이를 통하여 애플리케이션 서버의 애플리케이션을 클라이언트 상에서 보다 원활히 실행할 수 있다. 이러한 특성 및 필요성을 반영하여 씬-클라이언트 시스템 구축을 위한 계층적 아키텍쳐를 도시해보면 (그림 4)와 같다.

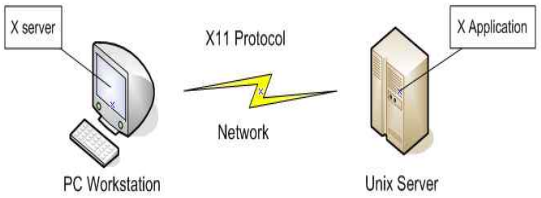


(그림 4) X 프로토콜 기반의 계층화된 씬-클라이언트 시스템 구성

씬-클라이언트 시스템 구현을 위한 기본 아키텍쳐는 위 그림에서와 같이 프로토콜 계층(X11), 확장 계층(X11 Mobility), 관리 계층(Services & Database Module), 애플리케이션 계층(Application)으로 분류된다. 프로토콜 계층은 X 윈도우 레벨을 의미하며 본 논문에서는 TCP/IP 네트워크 상에서 X 프로토콜을 활용하여 X 윈도우 방식으로 애플리케이션을 구동할 것이다. 확장 계층은 이러한 서버 상의 X 애플리케이션이 네트워크 상의 씬-클라이언트로 전달될 수 있도록 하며 일관된 세션상태를 유지할 수 있도록 지원하는 역할을 한다. 네트워크의 상태와 독립적으로 클라이언트와 애플리케이션 서버 간의 높은 QoS 세션이 유지될 수 있도록 확장 계층에서는 네트워크 상태, X 서버의 상태와 용량 등의 네트워크 자원 정보를 각 시스템에게 서비스하는 역할을 한다. 애플리케이션 계층은 네트워크 상에서 씬-클라이언트 프레임워크를 구축하는 일련의 시스템에 해당하며 씬-클라이언트를 운용하는 과정에서의 애플리케이션 실행, 데이터 전달을 관장한다. 나아가 X 서버/클라이언트 등의 소프트웨어 및 씬-클라이언트, 터미널서버, 애플리케이션 서버 등의 하드웨어들이 실제로 TCP/IP 네트워크 상에서 유기적으로 동작할 수 있도록 하며 그에 필요한 인터페이스를 사용자에게 제공하는 역할을 한다.

4.2 동작 메커니즘

X 프로토콜 기반의 시스템에서 원격의 사용자가 실행하는 서버 상의 애플리케이션은 X 애플리케이션(X 클라이언트)이다. 반면 클라이언트에서 서버 상의 애플리케이션을 실행, 디스플레이하는 객체는 X 서버(X 디스플레이)에 해당한다[5][6]. X 서버는 원격의 X 애플리케이션의 요청을 처리하여 해당 애플리케이션을 로컬 상의 스크린에 디스플레이 한다. 뿐만 아니라 로컬 상의 입력 장치(키보드, 마우스)로부터 받은 신호를 원격의 X 애플리케이션에 전달하여 실제 구동하는 역할을 한다. 본 논문에서 씬-클라이언트 구현에 있어 기반으로 할 X 윈도우 시스템의 기본 구성을 그림으로 나타내면 아래와 같다[5].



(그림 5) X 윈도우 시스템 클라이언트/서버 구성

X 서버와 X 애플리케이션은 본 절에서 우리가 구현하고자 하는 씬-클라이언트 구성에서 각각 클라이언트와 서버 시스템에 위치한다. X 서버를 구동하는 시스템이 클라이언트, X 애플리케이션이 실행되는 시스템은 서버에 해당한다. 이 두 시스템이 네트워크를 통해 연결되어 원격의 애플리케이션이 실행되기에 통신에 있어서의 QoS는 중요한 변수가 된다. 네트워크 장애 및 고장으로 두 시스템 간의 통신이 단절되면 세션, 데이터가 유실되고 X 애플리케이션은 좀비상태가 된다. 이러한 네트워크 상의 문제가 발생하더라도 X 서버와 X 애플리케이션 간 세션이 지속될 수 있어야 하며 이를 위하여 본 논문에서는 씬-클라이언트와 애플리케이션 서버 사이에 프락시로서의 기능을 수행하는 가상 서버를 지정하여 미들-티어 서버 상에서 수행되도록 하였다. 가상 서버는 씬-클라이언트의 X 서버와 유사한 형태로 동작하며 서버 상에서 실행되는 X 애플리케이션이 네트워크 상의 타 클라이언트로 이동 또는 리다이렉트 되는 것을 가능하게 한다. 이에 대한 한 예로서 네트워크 불안정으로 클라이언트 상의 X 서버가 셧다운 되었을 시 서버 상의 X 애플리케이션 또한 정지하는 경우를 들 수 있다. 기존의 씬-클라이언트에서는 이 경우 세션을 복원할 수도 없고, 서버 상의 애플리케이션을 다시 시작하여야 하였다. 하지만 본 논문에서 다음 장에서 제안할 방식에서는 네트워크의 불안정으로 세션이 소실되거나 애플리케이션이 정지될 필요가 없다. 기존에 실행하던 X 애플리케이션의 세션 정보를 XSMP 프로토콜 기반의 데이터베이스를 통하여 가상 서버가 가지고 있기에 네트워크가 유실되더라도 사용자는 새로운 X 서버 상에서 기존의 X 애플리케이션 작업을 지속하는 것이 가능하기 때문이다[8]. 네트워크 불안정으로 세션이 끊어졌을 때, 연결 상태(connection profile), 사용자 세팅 그리고 X 서버, X 애플리케이션 등의 각종 터미널, 디바이스

정보를 기반으로 세션을 재구성 수 있어야 한다. 이러한 네트워크 자원 정보를 저장할 데이터베이스가 필요하며 이를 기반으로 네트워크 회복 시 세션을 복구하는 것이 가능하다. X 서버와 X 애플리케이션의 세션이 단절되어졌을 시, 일단 기본세팅은 바로 직전에 연결되었던 클라이언트 시스템의 네트워크 정보로 지정하였다.

5. X 프로토콜 기반의 씬-클라이언트 구현을 위한 요소기술

본 장에서는 X 프로토콜 기반의 씬-클라이언트 구현에 있어 반영하고자 하는 요소기술에 대해 살펴보고자 한다. 우리가 구현하고자 하는 시스템은 단일 네트워크 또는 LAN 환경이 아닌 수많은 장비와 네트워크로 구성된 인터넷과 같은 WAN에서 실행된다. 때문에 시스템 설계에 있어 성능, 효율성을 극대화시킬 수 있는 기술에 대한 논의가 이루어져야 할 것이다. 이전 장에서 그를 위한 중추 시스템으로 미들 티어 상에서 별도의 시스템을 지정하였는데 본 장에서 소개할 Xvfb, XSMP, NX를 기반으로 우리가 원하는 씬-클라이언트 메커니즘을 설계하고자 한다.

5.1 Xvfb(X virtual frame buffer)

Xvfb는 X virtual frame buffer의 약자로서 메모리 상에서 실행되는 가상의 X 서버를 의미한다. 물리적 디스플레이, 입력 장치가 없이도 실행 가능하며 새로운 플랫폼 상의 X 서버 이식. 디스플레이 설정에 따른 서버 테스팅, 그리고 배치 프로세싱을 위한 백그라운드 렌더링 엔진 등의 용도로 활용된다. Xvfb는 클라이언트 상의 실제 X서버와 코드를 공유하며 X 애플리케이션의 요청을 처리하여 X 서버로 전달한다. 또한 X 서버로부터 해당 요청에 대한 결과는 물론 이벤트 발생, 에러 등의 정보를 X 애플리케이션에 전달하는 역할을 한다[10]. 이처럼 Xvfb는 X 애플리케이션과 X 서버 양측에 있어 각각 가상의 X 서버, X 애플리케이션으로서 프락시의 기능을 수행하며 본 논문에서는 이러한 특성에 준하여 미들 티어 상의 가상 서버로서 Xvfb를 활용하고자 한다.

가상 서버는 X 애플리케이션으로부터 스크린 출력에 관련된 요청을 수신함과 동시에 그를 X 서버 상에 디스플레이하는 데 필요한 픽셀, 컬러 등의 정보로 변환해 X 서버로 전달하는 역할을 한다. 또한 X 서버로부터 이 요청에 대한 응답과 에러 및 이벤트에 관련된 정보를 수신받아 X 애플리케이션으로 전달하게 된다. X 애플리케이션과는 분리된 미들-티어 시스템에 가상 서버로서 역할을 하는 Xvfb를 지정한 것은 작업에 있어서의 효율성에 근거한다. 본 논문에서 구현하고자 하는 시스템에서 우리는 다수의 씬-클라이언트 시스템이 중앙 서버의 애플리케이션을 공유하는 것을 가능하게 하고자 한다. 이는 X 애플리케이션이 보다 많은 X 서버와 통신 및 작업을 수행하여야 함을 의미하며 네트워크 상에서 연결되는 클라이언트(X서버)의 개수가 증가함에 따라 서버(X 애플리케이션)의 오버로드는 급격히 증가되게 된다. 이에 따라 X 애플리케이션의 기존 업무에서 통신 관련 업무를 분리하여 가상 서버가 실행되는 별도의 시스템에서 Xvfb가 실행되도록 설계하였다.

5.2 XSMP(X Session Management Protocol)

네트워크 상에서 서버(X 애플리케이션)와 클라이언트(X 서버)간 통신 세션을 저장, 유지하기 위한 프로토콜로서 본 논문에서는 XSMP(X Session Management Protocol)을 활용하고자 한다. X 서버와 통신하는 일련의 X 애플리케이션 그룹이 세션의 주체에 해당하며 각 X 애플리케이션은 자신만의 상태정보를 가지고 있다. 이러한 세션은 세션 관리자라 불리는 네트워크 서비스에 의하여 제어되며 X 애플리케이션의 세션 모니터링, 저장, 종료 등에 관련된 커맨드가 이 세션 관리자를 통해 전달된다[8]. 이에 따라 네트워크 상에서 세션이 단절되더라도 각 X 애플리케이션의 정보는 세션 관리자에 개별적으로 저장되어 유지될 수 있으며 X 서버는 이전의 X 애플리케이션 상태를 유지한 채 일관된 통신을 수행하는 것이 가능하다. X 애플리케이션은 XSMP를 활용하여 세션 관리자에 등록되며 초기 구동 시 및 실행 과정 동안 세션 관리자와 연결되어야 있어야 한다. XSMP는 X 컨소시엄의 ICE 프로토콜의 상위에 위치하여 세션 관리 및 인증을 수행한다[8]. X 애플리케이션은 세션

관리자가 제공하는 SESSION MANAGER 환경 변수를 통하여 세션 관리자에 접속할 수 있으며, 일련의 속성을 활용하여 각 X 애플리케이션은 세션 관리자에게 자신의 상태를 전달할 수 있다. 본 논문에서는 씬-클라이언트에 대한 일관된 세션 유지 및 네트워크 복원 후의 세션 정보 제공을 XSMP 프로토콜을 통하여 구현하고자 한다.

5.3 NX

X 윈도우 시스템에서 애플리케이션에 해당하는 X 클라이언트와 씬-클라이언트 상의 X 서버 간의 통신은 X 프로토콜에 의하여 이루어진다. 하지만 X 프로토콜은 본래 동일한 시스템 또한 LAN 상에서의 통신을 목적으로 개발되어진 프로토콜이기에 본 논문에서 지향하는 인터넷 상의 씬-클라이언트 시스템에 바로 적용하기에는 한계가 있다. 다수의 네트워크와 사용자로 인해 낮은 대역폭, 높은 지연이 상존하는 WAN 상에서의 X 프로토콜 통신은 네트워크 성능 저하로 이어지며 그 원인은 X 트래픽, 빈번한 라운드 트립, 캐시 활용 부재의 세 가지로 요약할 수 있다. 본 논문에서는 이를 극복하기 위한 기술로서 우리의 시스템에 NX를 접목하고자 하며 이를 통하여 X 트래픽 압축, 라운드 트립의 감소는 물론 캐시 활용 메커니즘을 구현할 것이다. 이를 통하여 보다 향상된 성능의 X 데이터 전달 뿐 아니라 보안의 측면에 있어서도 SSH 프로토콜 기반의 암호화, 인증을 활용하는 것이 가능하다.

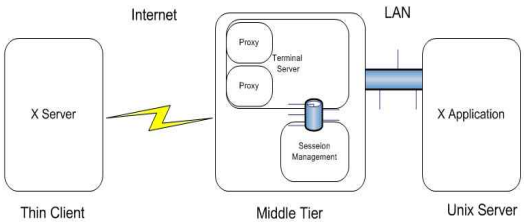
NX는 그 기능 및 역할에 있어 NX 프락시와 NX 에이전트로 구성된다[11]. 본 논문에서 우리는 X 애플리케이션 및 X 서버 상에 NX 프락시, NX 에이전트를 각각 활용하였고 이를 통하여 각 시스템 간에 안정적인 통신이 이루어지도록 하였다.

X 애플리케이션은 원격에 위치한 서버 상에서 실행되며 X 서버는 로컬 상의 씬-클라이언트에서 실행된다. 낮은 대역폭 기반의 네트워크라 하더라도 NX 프락시, NX 에이전트 컴포넌트 구현을 통하여 씬-클라이언트와 서버 시스템 간 신뢰성 있는 통신을 구현할 수 있다. 서버 시스템 상에서 NX 에이전트를 실행하여 X 애플리케이션과 X 서버간의 라운드 트립을 감소시킬 수 있도록 한 것은 씬-클라이언트의 QoS 제고에 있어 핵심적 요소로 작용한다.

6. 씬-클라이언트 시스템 프레임워크 설계

본 장에서는 지금까지 살펴본 씬-클라이언트 프레임워크 및 그 동작에 필요한 요소기술을 기반으로 씬-클라이언트 시스템을 실제 구현하기 위한 프레임워크를 설계해보고자 한다.

본 논문에서 구현하고자 하는 씬-클라이언트 시스템 프레임워크는 보다 많은 씬-클라이언트의 통신을 관장하며 낮은 QoS 특성의 네트워크 상에서도 X 세션을 지속적으로 유지할 수 있어야 한다. 기 지적하였듯이 X 프로토콜은 본래 내부 네트워크 LAN이나 동일한 시스템 상에서 구현하기 위하여 개발된 것으로서 인터넷과 같은 외부 네트워크에 적용하기에는 한계가 있다. 낮은 대역폭, 높은 지연 특성을 갖는 인터넷과 같은 외부망에서 X 프로토콜 데이터를 전달함에 있어 충족되어야 할 요건으로는 데이터 압축, 캐시 활용, 라운드 트립 트래픽 감소 등을 들 수 있다. 이를 구현하기 위하여 본 논문에서는 기존의 클라이언트와 서버라는 2 계층 모델을 지양하고, 그 사이의 멀티-티어에 가상의 X 서버(프락시, Psuedo 서버)를 두는 3 계층 모델을 설계의 시작점으로 지정하였다. 이를 그림으로 나타내면 아래와 같다.



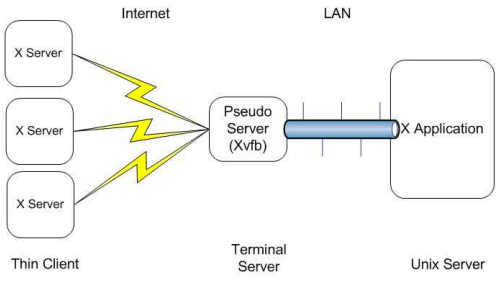
(그림 6) X 프로토콜 기반의 씬-클라이언트 아키텍쳐

기존 2계층 구조의 가장 큰 문제는 X 애플리케이션이 실행되는 서버 측에 일방적으로 부하가 가중된다는 것이었다. X 애플리케이션의 요청 처리를 다수의 X 서버로 전송하는 것은 물론, X 서버로부터 그에 대한 응답 및 이벤트, 에러 등의 정보를 수신하여 X 애플리케이션에 포워딩하는 것이 기존 서버의 역할이었다. 통신 뿐만 아니라 애플리케이션 관련된 부하까지 처리

하는 이러한 구조적 특성 하에서 서버 부담은 가중될 수 밖에 없었고 결과적으로 이는 네트워크와 애플리케이션 운용 모두에 있어 비효율을 야기시켰다. 이를 개선하기 위해서는 네트워크, 애플리케이션 두 종류의 부하를 서로 다른 시스템으로 분할하는 것이 필요하며 그에 따라 본 논문에서는 미들-티어를 지정하여 가상 서버를 구축하였다.

Xvfb를 기반으로 한 가상 서버는 클라이언트 측 방향으로는 가상의 서버, 서버 측 방향으로는 가상의 클라이언트로서 동작한다. 본 논문에서는 이러한 기능의 가상 X 서버를 씬-클라이언트와 서버 중간의 미들-티어에 구성하였고 네트워크 부하를 전담하도록 설계하였다. 가상 X 서버가 실제 X 서버에 대한 가상의 시스템으로서 디스플레이 출력 장치와 마우스, 키보드 등의 입력장치를 필요치 않는 특성을 고려하여 Xvfb(X virtual frame buffer)를 활용하여 미들-티어를 구축하였다. 가상 서버는 X 애플리케이션이 실행되는 서버와 동일한 네트워크 상에 존재하기에 X 애플리케이션이 실행되는 애플리케이션 서버와의 통신 QoS를 탁월이 향상시킬 수 있다. 또한 네트워크를 가로 질러 씬-클라이언트와 통신하지 않아도 가상 서버를 통해 서버의 상태, 용량 등의 파악에 관련된 라운드 트립 트래픽을 줄이는 것이 가능하다. 뿐만 아니라 그를 폭넓은 대역폭의 LAN 상에서 수행하니 세션의 안정화를 제고하는 효과도 얻을 수 있다.

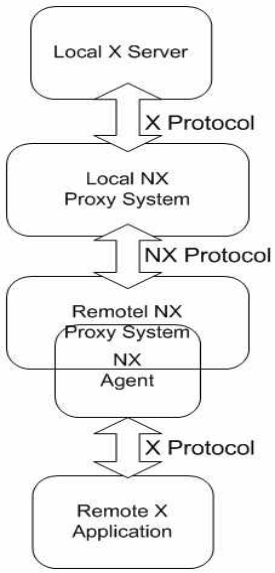
위에서 설명한 씬-클라이언트 상에서 가상 서버를 활용하여 X 애플리케이션, X 서버 간의 통신 메커니즘을 간략히 도시해보면 (그림 7)와 같다.



(그림 7) 가상서버(Xvfb)를 활용한 X 애플리케이션, X 서버 간의 통신 메커니즘

본 논문에서는 이러한 장점에서 나아가 실제 X 서버에 동기화된 복사본을 미들-티어에 구성하고자 하였다. 실제 X 서버에 동기화된 서버가 내부 LAN 상에서 실행된다면 X 프로토콜 통신에 있어서 지연의 가장 큰 원인으로 작용하였던 라운드 트립 트래픽을 획기적으로 경감할 수 있다. 그를 위한 기술로 본 논문에서는 NX기술을 활용하였다.

NX는 크게 NX 프락시, NX 에이전트의 두 컴포넌트로 구성된다. 씬-클라이언트와 서버 간의 통신은 X11 기반의 NX 프로토콜로 수행되며 우리가 미들-티어에 구현하고자 하는 동기화된 서버는 로컬 NX 프락시, 원격 NX 프락시, NX 에이전트를 설정함으로써 구현 가능하다. 그에 대한 시스템 구성을 그림으로 도시해보면 (그림 8)과 같다.



(그림 8) NX컴포넌트를 적용한 씬-클라이언트 아키텍쳐

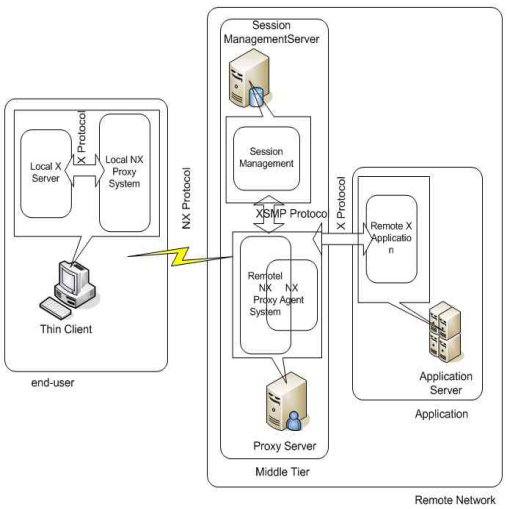
원격 NX 프락시는 X 애플리케이션 측에서 바라볼 때 마치 X 서버인 것처럼 동작한다. 따라서 굳이 외부 네트워크를 거치지 않고 동일한 LAN 상에서 가상 X서버와 X 애플리케이션 간의 메시지 전달이 가능하며 외부에서의 라운드 트립 트래픽 억제를 구현할 수 있다. 원격 프락시에는 X 서버의 기능을 통합 구현한 NX 에이

전트가 포함된다. 이는 LAN 상에서 X 애플리케이션에 대한 X 서버로서 동작하며 이를 활용함으로써 본 논문에서 목표로 하는 X 서버의 동기화된 복사본을 서버 시스템 상에 유지할 수 있게 된다. 씬-클라이언트 상의 로컬 프락시는 로컬의 X 서버와의 통신 및 외부의 NX 프로토콜을 내부의 X 프로토콜로 변환하는 역할을 한다. 서버 상의 원격 프락시는 내부 LAN 상에서 X 프로토콜을 활용해 X 애플리케이션과 통신을 수행한다.

씬-클라이언트 구현에 있어 X 프로토콜 데이터의 압축, 캐싱, 라운드 트립 트래픽 감소를 통한 성능 개선과 더불어 중요한 것이 일관된 세션유지이며 이전 장에서 그 요소기술로 XSMP를 활용하였다. XSMP는 서버 상에서 실행되는 다수의 X 애플리케이션을 관리하기 위한 프로토콜로서 각 X 애플리케이션은 세션 관리자와의 연결을 통해 상태 정보를 저장한다. X 애플리케이션의 실행 초기 시 세션 관리자는 전송 프로토콜, 호스트, 포트 넘버 등의 정보가 담긴 SESSION MANAGER이라는 환경 변수를 전달하며 서버 상의 각 X 애플리케이션은 이를 활용하여 세션 관리자와의 통신을 설정할 수 있다. 이에 따라 네트워크 장애로 씬-클라이언트와 X 애플리케이션 간의 연결이 상실되더라도 세션 관리자에 저장된 X 애플리케이션의 기존 정보를 활용하여 세션을 지속적으로 유지하는 것이며 씬-클라이언트는 시스템, 장소에 관계없이 일관된 통신을 수행하는 것이 가능하게 된다.

본 논문에서는 XSMP 기반의 세션 관리를 구현하기 위하여 별도의 리눅스 시스템을 지정하였고 해당 시스템 위에 씬-클라이언트와 통신하는 애플리케이션의 세션 정보를 관리하는 데이터베이스를 구축하였다. 애플리케이션 정보 저장을 위한 데이터베이스로는 리눅스에서 가장 범용적으로 쓰이는 MySQL을 활용하였으며, 리눅스 시스템 상에서 세션 관리자가 실행되도록 하였다. XSMP는 서버 상에서 실행되는 애플리케이션을 세션 관리자에 등록하는데 활용된다. XSMP를 통해 얻어진 애플리케이션 상태 정보가 리눅스 시스템의 데이터베이스에 저장될 수 있도록 제어 프로토콜, 리눅스 서버 IP, 포트넘버로 이루어지는 네트워크 주소를 서버에 제공하여 애플리케이션이 세션 관리자에 접속할 수 있도록 하였다. 데이터베이스 상에서는 세션 관리에 필요한 각 속성을 칼럼으로 하는 테이블을 구성하였다. 이를 통해 서버 원격 애플리케이션 호출에 필요한 Program, Client-ID, ProcessID는 물론 UserID, Current Directory, Previous-ID 등의 정보에 씬-클라이언트가 접근하도록 함으로써 세션이 중단되더라도 재연결시 복원될 수 있도록 하였다. 뿐만 아니라 Register Client, SaveYourselfRequest, Set Properties, RestartCo-mmand 등의 프로토콜 커맨드를 통해 애플리케이션의 상태 정보가 세션 관리자를 통해 효율적으로 반영되고 데이터베이스에 저장될 수 있도록 하였다.

네트워크 상에서 씬-클라이언트를 구현하기 위한 접근법으로서 우리는 각 시스템에서 X 애플리케이션을 공유할 수 있는 다양한 논의를 진행하였는데 이제까지 논의된 Xvfb, XSMP, NX 기술을 적용하여 씬-클라이언트 구현을 위한 아키텍처 설계를 그림으로 나타내면 (그림 9)과 같다.



(그림 9) 각 요소기술을 통합하여 설계된 씬-클라이언트 아키텍처

상기 그림에서 확인할 수 있듯이 본 논문에서 제안한 씬-클라이언트 아키텍처는 씬-클라이언트, 서버로 구성되던 기존 방식을 개선하였다. 보다 효율적인 데이터 전송, 안정적인 세션 관리가 이루어질 수 있도록 원격 네트워크 상에 프

락시 서버, 세션관리 서버로 구성되는 미들-티어를 구축하였다. X 프로토콜이 본래 내부 네트워크 상에서의 통신을 목적으로 제작된 것이기에 본 논문에서도 그를 내부 네트워크 용도로 제한하였다. 인터넷 상에서 X 데이터를 효과적으로 전송하기 위해서는 데이터 압축 및 캐싱 메커니즘이 구현되어야 하기에 서버 측 네트워크에 NX 컴포넌트를 미들-티어로 구성하였으며 그를 통해 라운드 트립으로 인한 통신 지연을 대폭 완화시킬 수 있었다. 네트워크 단절 시에도 이전의 상태가 복원되도록 세션관리 서버를 데이터베이스 형태로 리눅스 시스템에 구축하였으며, 애플리케이션 서버와의 통신을 기반으로 애플리케이션 정보가 세션관리 서버에 저장될 수 있도록 하였다.

본 논문에서는 씬-클라이언트와 애플리케이션 서버 간의 데이터 통신을 그 기능 및 역할에 따라 세분화하고 그에 관련된 각 프로토콜을 LAN에서 분산 시스템 형태로 구현함으로써 성능과 안정성 면에서 한층 개선된 씬-클라이언트 아키텍처를 구현할 수 있었다.

7. 결론

네트워크 및 애플리케이션 기술의 발전에 따라 네트워크 상에 위치한 타 시스템의 함수를 호출하고 원격에서 애플리케이션을 실행하는 것이 가능하게 되었다. 컴퓨터 시스템은 이제 더 이상 외부와 격리되어 독립적으로 실행되는 시스템이 아니라, 네트워크 상에서 타 시스템에 서비스를 제공하기도 하고 요청도 하는 가상 컴퓨터의 일부분으로 융합되기에 이르렀다. 이러한 시스템이 근래 씬-클라이언트, 네트워크 컴퓨터, 클라우드 컴퓨팅으로 회자되고 있으며 IT 기술의 진보에 따라 향후 이러한 추세는 더욱 심화될 것으로 예상된다.

본 논문에서는 네트워크 상에서 임의의 클라이언트가 타 시스템의 애플리케이션을 효율적으로 활용할 수 있는 씬-클라이언트 구현을 위한 프레임워크 설계 방안에 대해 알아보았다. 기반 프로토콜로서 X 프로토콜을 활용하였고 메커니즘 구현을 위한 요소기술로서 Xvfb, XSMP, NX를 활용하였다. X 프로토콜은 본래 동일 시스템 또는 네트워크 상에서 X윈도우 시스템을 활용하기 위하여 개발된 프로토콜로서 인터넷과 같은 WAN 환경에서 활용하기에는 적합하지 않다. 무엇보다 X 애플리케이션과 X 서버 간의 통신에 있어 방대한 양의 데이터 송수신, 라운드 트립 트래픽 발생은 씬-클라이언트로서 X 서버의 가능성을 저하시켰으며 성능에 있어 심각한 문제를 야기시켰다.

본 논문에서는 성능 및 효율성에 있어서 이러한 문제를 극복할 수 있도록 클라이언트-서버라는 단순한 2계층 모델을 지양하고 프락시 기능을 수행하는 별도의 미들-티어를 지정하였고 그를 서버와 동일한 네트워크 상에 구성하였다. 씬-클라이언트 상의 실제 X 서버에 대한 가상의 X서버로서 프락시가 그 기능을 수행하기에 WAN을 통해 전달되는 데이터를 효과적으로 경감시키는 것이 가능하다. 이와 더불어 인터넷을 통해 가로지르는 X 프로토콜 데이터 관련 데이터에 대한 압축, 캐싱 메커니즘 수행을 위하여 본 논문에서는 NX 컴포넌트를 활용하였다. NX 프락시, NX 에이전트를 씬-클라이언트와 서버 측 시스템에 구성하였고 NX 에이전트를 통하여 실제 X 서버의 복사본을 서버와 동일한 네트워크에 구성하여 X 프로토콜 데이터로 인한 라운드 트립 트래픽을 WAN에서 효율적으로 감소시킬 수 있었다.

이와 더불어 씬-클라이언트 구성에 있어 중요한 요소로서 일관된 세션 유지를 들 수 있다. 네트워크 장애 및 오류로 인해 씬-클라이언트와 서버 간의 연결이 단절되더라도 이전 상태로 복구할 수 있어야 할 것이다. 이를 위하여 본 논문에서는 XSMP 프로토콜을 기반으로 서버 상의 애플리케이션 상태를 저장 및 유지할 수 있는 세션 관리자 및 그를 위한 별도의 데이터베이스 활용을 제안하였다. 서버 상의 X 애플리케이션은 실행되는 동안 데이터베이스의 세션 관리자와 통신을 수행하여 자신의 상태 정보를 데이터베이스에 저장하게 된다. 향후 네트워크 및 씬-클라이언트의 문제로 인하여 세션이 중단되어도 씬-클라이언트는 세션 관리자를 통해 저장된 정보를 기반으로 이전의 세션을 복원할 수 있으며, 이를 통하여 보다 높은 QoS 환경에서 효율적으로 원격의 애플리케이션을 활용하는 것이 가능하다.

참 고 문 헌

[1] http://en.wikipedia.org/wiki/Thin_client
[2] White Paper, "Thin-Client/Server Computing", Citrix Systems, Inc, 1998
[3] Niraj Tolia, David G. Andersen, and M. Satyanarayanan "Quantifying Interactive User Experience on Thin Clients", IEEE Internet Computing, pp. 46-52, IEEE Computer Society, 2006.
[4] Andrej Volchkov, "Server-Based Computing Opportunities", IT Pro, pp. 18-23, IEEE Computer Society, 2002.
[5] http://en.wikipedia.org/wiki/X_Window_System_core_protocol
[6] Martin Mauve, "Protocol Enhancement and Compression for X-Based Application Sharing", University of Mannheim, pp. 11-12, , 1997
[7] Tristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood, and Andy Hopper, "Virtual Network Computing", Mobile Computing, IEEE Coimputer Society, pp. 33-38, 1998
[8] Mike Wexler, "X Session Management Protocol", The Open Group, pp. 1-3, 2002
[9] http://www.pulsewan.com/data101/thin_client_basics.htm
[10] http://en.wikipedia.org/wiki/Xvfb
[11] http://en.wikipedia.org/wiki/NX_technology

송 민 규

2001년 : 강원대학교 전기공학과 (공학사)
2005년 : 강원대학교 전자공학과 대학원 (공학석사)
2002년~현재 : 한국천문연구원 전파천문연구부
관심분야 : 분산 컴퓨팅, 씬-클라이언트, 네트워크 컴퓨터, 원격 데스크탑, 그리드