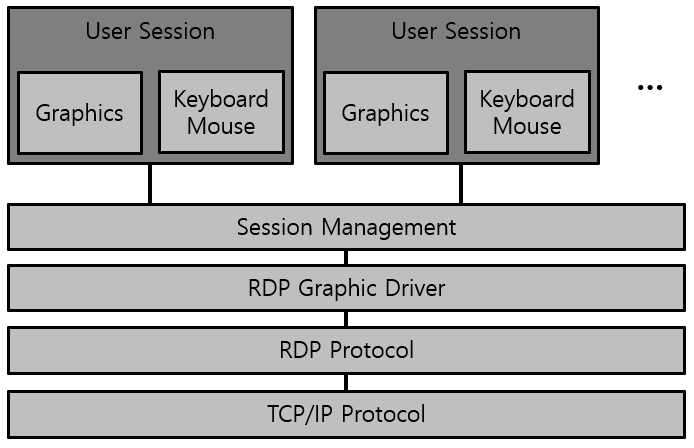
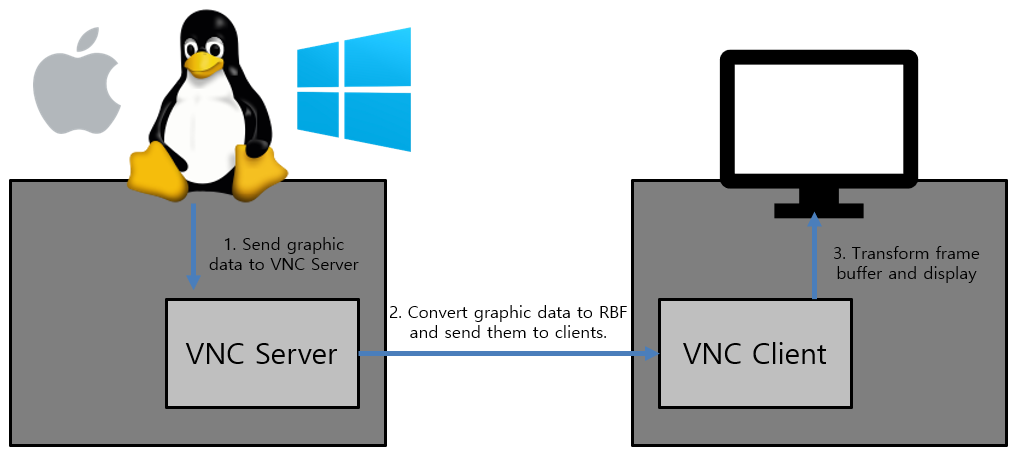
1. **Remote Access for NEMO-UX**
2. **Wayland**
3. **NEMO-UX & Weston Compositor**
4. **Remote Access Technologies**
   1. **RDP (Remote Desktop Protocol)**



RDP는 네트워크를 통한 원격 그래픽 인터페이스 제공을 위해 Microsoft에서 자체적으로 만든 프로토콜이다. 이 때문에 초기의 RDP 서버는 Window OS 환경에서만 제공이 가능했으나 현재는 Linux OS, Mac OS 등의 다양한 플랫폼 환경에서도 RDP 서버를 제공하는 벤더가 증가하는 추세이다. 반면, RDP 클라이언트는 초기부터 다양한 플랫폼에서 사용이 가능하였다. RDP 서버는 멀티 채널 세션 방식을 통해 각 클라이언트별 세션을 제공한다. 각 세션은 세션 관리 계층 상에서 클라이언트별 그래픽 인터페이스 출력 처리와 마우스, 키보드 입력 처리를 수행한다. 클라이언트에게 그래픽을 전송하는 것은 네트워크 Bandwidth 관점에서 봤을 때 수행 비용이 높기 때문에 그래픽 데이터를 전송할 때 효율적인 압축 알고리즘을 사용하는 것이 중요하다. RDP 그래픽 드라이버는 GDI나 GDI+와 같은 그래픽 인터페이스로부터 어플리케이션의 그래픽 커맨드를 얻는다. 즉, 유저 세션상의 어플리케이션이 디스플레이에 원하는 그래픽을 표현하기 위해 그래픽 인터페이스로 명령과 함께 비트맵 및 텍스트, 마우스 커서 등의 데이터들을 보내면 그래픽 인터페이스는 RDP 그래픽 드라이버로 그래픽 데이터를 포워딩한다. RDP 그래픽 드라이버는 그래픽 데이터를 압축하고 RDP 프로토콜 형식으로 변환해 클라이언트에게 그래픽 데이터를 전송한다. RDP 클라이언트는 받은 RDP 프로토콜 형식의 그래픽 데이터를 변환해 클라이언트 환경의 어플리케이션에 출력한다. RDP 프로토콜을 이용해 네트워크 원격 접속 서비스를 제공하는 오픈소스 소프트웨어로는 대표적으로 FreeRDP, Remmina, xrdp, Rdesktop 등이 있다.

* 1. **VNC (Virtual Network Computing)**



VNC 프로토콜은 호스트의 그래픽 유저 인터페이스에 네트워크를 통해 원격 접속하기 위한 프로토콜이다. VNC 프로토콜은 RFB(Remote Frame Buffer) 프로토콜에 기반해 동작한다. VNC 서버는 RFB 프로토콜 형식으로 디스플레이 데이터를 클라이언트에게 보내고, 클라이언트는 서버로부터 받은 RFB 데이터를 변환해 클라이언트 데스크톱 상에 설치된 그래픽 라이브러리를 이용해 디스플레이에 서버의 그래픽 디스플레이를 나타낸다. 반대로 클라이언트는 서버와 Authentication 과정을 통해 연결이 완료되면 키보드 디바이스나 포인트 디바이스의 표준 워크스테이션 모델에 기반한 프로토콜로 서버에게 입력을 전송할 수 있다. 클라이언트 디바이스에서 키보드 버튼이나 포인터 버튼을 누르거나 포인트 디바이스를 움직일 때 마다 해당 디바이스의 표준 프로토콜을 통해 클라이언트는 입력 데이터를 서버에 전송한다. VNC 프로토콜은 보통 클라우드에서 동작하는 서버 VM(Virtual Machine) 동작을 위해 사용한다. 예를 들어, 가상화 소프트웨어인 QEMU는 VNC 프로토콜 사용을 기본 옵션으로 제공한다. 하이퍼바이저 레벨에서 VNC 서버를 실행시킨다면 클라이언트는 별도의 VM에 VNC 서버 설치 없이 사용하는 VM에 VNC 접속을 할 수 있다. 하지만 VNC는 서버와 클라이언트 간의 데이터 전송 오버헤드와 클라이언트의 RFB 데이터를 변환하는 과정의 오버헤드 때문에 실제 서버 데스크톱의 반응 속도에 비해 클라이언트에서 느리게 반응한다는 단점이 있다. VNC 프로토콜을 이용해 네트워크 원격 접속 서비스를 제공하는 오픈소스 소프트웨어로는 대표적으로 TigerVNC, TightVNC, Guacamole, DWService 등이 있다.

* 1. **RDP와 VNC 비교**

RDP와 VNC 기술의 공통점으로는 네트워크를 사용하여 원격 컴퓨터의 디스플레이에 접근한다는 목적이 같다는 것이다. 또한, 두 기술 모두 서버와 클라이언트 단의 어플리케이션에서 프로토콜 통신에 대한 소프트웨어 지원이 필요하다.

반대로, RDP와 VNC 기술의 차이점으로는 먼저 RDP는 멀티 채널 세션 방식을 사용해 각각의 클라이언트마다 디스플레이 세션을 지원할 수 있다는 것이다. 즉, 클라이언트는 서버 시스템의 자원을 사용할 때, RDP 프로그램을 사용하여 각각의 디스플레이 인터페이스로 개별적인 작업을 수행할 수 있다. 반면, VNC 프로토콜은 각 유저마다 개인적인 디스플레이 세션을 지원하지 않는다. VNC 프로토콜을 사용하는 환경에서 모든 클라이언트는 같은 디스플레이 세션을 공유하게 된다. 즉, 서버 데스크톱을 포함한 서버에 연결된 모든 유저에서 보이는 디스플레이는 각 클라이언트 환경에 따른 오버헤드에 의한 속도 차이를 제외하면 동일한 디스플레이를 공유하게 된다.

또한, 대부분의 RDP 프로그램은 Window OS에서 동작하며 VNC 프로토콜은 여러 플랫폼에서 사용이 가능하다. 하지만 최근에는 몇몇 RDP 프로그램 또한 Linux OS, Mac OS 등을 지원하고 있는 추세이다.

1. **RDP Programs**
   1. **FreeRDP**
   2. **Remmina**
   3. **xrdp**
   4. **Rdesktop**
2. **VNC Programs**
   1. **TightVNC**



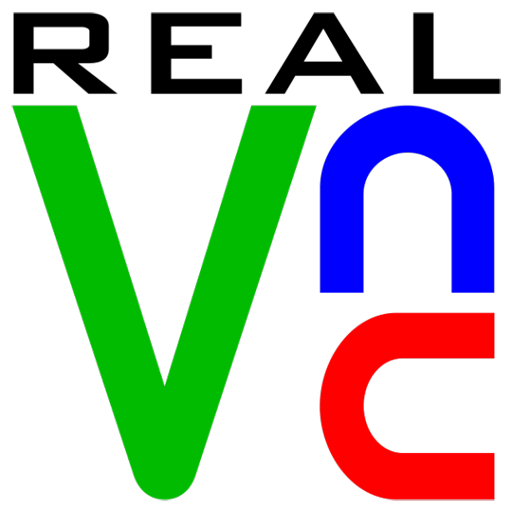
Tight VNC는 원격 데스크톱 컨트롤을 위한 오픈소스 소프트웨어 패키지이다. Tight VNC의 서버는 Unix 계열 OS와 Window OS를 모두 지원한다. 하지만 독자적인 자바 기반 클라이언트 어플리케이션을 사용하기 때문에 TightVNC를 사용하기 위해서 클라이언트는 TightVNC의 클라이언트 어플리케이션을 설치해야 한다. TightVNC는 RFB 프로토콜을 따르는 표준 VNC 소프트웨어로서 구현되었다. TightVNC는 그래픽 데이터를 제공할 때 JPEG와 zlib 압축 방식의 Tight Encoding이라는 독자적인 인코딩 방식을 사용해 낮은 Bandwidth를 제공하는 연결 상에서도 비교적 높은 성능을 제공한다. TightVNC는 이러한 효율적인 인코딩 방식을 사용해 기존의 느린 VNC 연결의 단점을 보완하였기 때문에 TightVNC를 사용하면 클라이언트는 비디오를 시청하거나 DirectX 기반의 그래픽 게임을 하는 등 원활하게 그래픽 중심의 어플리케이션을 사용할 수 있다. 하지만 TightVNC의 클라이언트가 아닌 다른 클라이언트 소프트웨어를 사용할 경우, 대부분의 클라이언트가 Tight Encoding을 지원하지 않기 때문에 양 쪽의 소프트웨어에서 모두 Tight Encoding을 지원해야만 TightVNC의 빠른 그래픽 데이터 변환의 장점을 활용할 수 있다. TightVNC는 2001년부터 릴리즈되었기 때문에 RemoteVNC, TurboVNC, TigerVNC 등의 여러 VNC 소프트웨어의 기반이 되었다.

* 1. **TigerVNC**



TigerVNC는 고성능의 크로스 플랫폼 원격 데스크톱 소프트웨어를 지원하기 위해 구현된 VNC 기반의 오픈소스 소프트웨어 패키지이다. TigerVNC는 RealVNC는 소스코드를 공개하지 않고 유료 서비스를 제공하며 TightVNC는 2006년 이후로 업데이트가 되지 않고 있는 등 당시의 문제를 해결하기 위해 개발되었다. TigerVNC는 TightVNC 기반으로 개발되었으며, Tight VNC의 성능적인 장점은 살리며 계속해서 릴리즈 되는 OS 버전에 호환되며 어떠한 플랫폼에서도 좋은 성능을 보장할 수 있도록 현재까지 개발되는 중이다. 따라서 클라이언트는 TigerVNC 뷰어 프로그램을 사용하면 어떠한 플랫폼에서도 TigerVNC가 설치된 서버에 원격 접속이 가능하다. 하지만 서버는 Window OS와 Unix/Linux OS만 지원한다. Linux OS를 위한 TigerVNC 패키지는 총 6개의 다른 프로그램으로 구성되어 있다. 먼저 vncviewer는

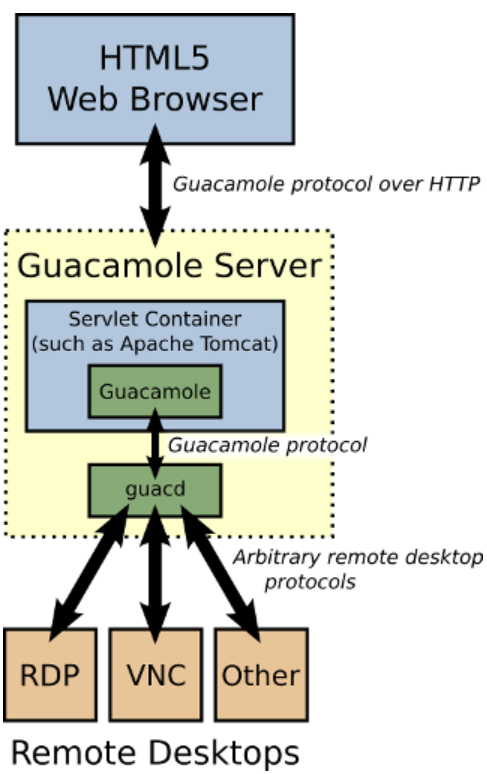
* 1. **RealVNC**



* 1. **Guacamole**



Guacamole는 아파치에서 개발한 원격 접속 웹 어플리케이션이다. Guacamole는 오픈소스 소프트웨어이다. Guacamole 클라이언트는 HTML5 기반의 웹 어플리케이션이기 때문에 HTML5를 지원하는 웹 브라우저를 사용하면 어느 플랫폼에서나 서버 데스크톱에 원격으로 접속할 수 있다. 또한, 클라이언트가 웹 기반으로 동작하기 때문에 HTML5를 지원하는 웹 브라우저가 설치된 환경이라면 모바일 환경에서도 서버 데스크톱에 원격 접속이 가능하다. Guacamole는 RDP, VNC, SSH 등의 원격 접속 프로토콜을 지원한다. 또다른 Guacamole의 장점으로는 다른 오픈소스 원격 접속 어플리케이션과 비교했을 때 API 기술 문서, 설치 매뉴얼 등이 상세하게 기술 되어있다는 장점이 있다. Guacamole 서버는 자바 기반 웹 서버인 Tomcat 7과 8에서 동작하며 Guacamole 클라이언트는 자바스크립트 기반으로 웹 브라우저에서 동작한다. 설정 정보 저장 및 유저 정보 저장을 위한 지원으로 MySQL 데이터베이스 및 XML 파일, JSON 파일 형식을 지원한다. 추가로 Guacamole는 웹 브라우저 상에서 관리자 유저를 생성할 수 있어 유저 관리를 웹 상에서 할 수 있다는 장점이 있다. 반면, Guacamole 서버는 많은 기능을 갖기 때문에 많은 라이브러리 및 어플리케이션에 의존성을 가져 서버 환경을 구축하기가 어렵다는 단점이 있다.



다음은 Guacamole 동작의 상세 설명이다. Guacamole 서버는 그림과 같이 하나의 프로세스로서 동작하는 것이 아니라 Guacamole 톰캣 서버와 guacd 데몬 프로세스가 Guacamole 프로토콜을 사용해 서로 통신하며 동작한다. 독자적인 Guacamole 프로토콜을 사용하는 이유는 사용 플랫폼에 독립적인 통신을 하기 위해서이다. Guacamole 클라이언트는 HTML5를 지원하는 웹 브라우저 상에서 Guacamole 톰캣 서버가 제공하는 자바스크립트 파일을 통해 HTTP 형식의 Guacamole 프로토콜을 사용해 Guacamole 서버와 통신한다. Guacamole 톰캣 서버는 Guacamole 클라이언트에게 원격 접속 프로토콜에 맞는 그래픽 데이터를 전송하며, Guacamole 클라이언트는 포인터 디바이스 및 키보드 디바이스 입력을 Guacamole 톰캣 서버로 전송하게 된다. Guacamole 톰캣 서버는 클라이언트와의 통신만을 목적으로 하며 원격 접속을 위한 실질적 역할은 서버 내부의 guacd 데몬 프로세스가 하게 된다. Guacamole 톰캣 서버는 guacd 데몬 프로세스와 Guacamole 프로토콜을 사용해 클라이언트로의 그래픽 데이터 전송 및 클라이언트의 캐릭터 디바이스 입력 수신을 위한 통신을 한다. guacd는 Guacamole 톰캣 서버와 원격 데스크톱 서버와의 프록시 서버 역할을 한다. guacd는 Guacamole 프로토콜과 원격 접속 프로토콜 간 변환을 통해 원격 데스크톱과 양방향 통신을 한다.

* 1. **DWService**



1. **RDP on Linux**
2. **VNC on Linux**
3. **Implementation Plan**
   1. **Remote Access through Web Browser**
   2. **Wayland Compositor가 Remote Access 지원**
   3. **Remote Access Program이 Remote Access 지원**