페이지폴트 인터럽트 루틴 후킹을 이용한 소프트웨어 스트리밍 시스템 구현

(Implementation of a Software Streaming System Using Pagefault Interrupt Routine Hooking)

김 한 국*, 이 창 조**
(Han-Gook Kim, Chang-Jo Lee)

요 약 소프트웨어 산업에 있어서 서버 관련 정보기술이 발전하고 네트워크 성능이 크게 향상된 반면, 이용 비용은 크게 높아짐에 따라 중소 사업장을 중심으로 ASP(Application Service Provider)와 같은 다양한 형태의 소프트웨어 사용 방식이 등장하게 되었다. 이를 가능하게 하는 요소 기술에는 여러 가지가 존재하는데, 본 연구에서는 컴퓨터상에서 수행에 필요한 최소한의 응용 소프트웨어를 중앙 서버로부터 분산 가상페이징 기술로 자동으로 끌어 와서 사용하는 소프트웨어 스트리밍 서비스 기술에 있어서 PageFault Interrupt Routine Hooking 방식을 응용한 새로운 개념의 소프트웨어 스트리밍 시스템 구현을 제안하였다. 이러한 방식을 이용하여 소프트웨어 스트리밍 시스템을 구현하게 되면, 보다 효율적으로 응용 소프트웨어를 관리할 수 있을 뿐만 아니라, 소프트웨어 전체를 인스톨할 필요가 없기 때문에 하드웨어의 저장 공간을 거의 사용하지 않는다. 또한, 기본적인 바이너리만을 받아서 로드하기 때문에 하드웨어 자원을 최대한 줄일 수 있게 된다.

핵심주제어 : 소프트웨어 스트리밍, PE파일, 페이지폴트

Abstract The need for ASP(Application Service Provider) has evolved from the increasing costs of specialized software that have far exceeded the price rage of small to medium sized businesses. There are a lot of technologies that make ASP possible, and software streaming service is one of them. Software streaming is a method for overlapping transmission and execution of stream-enabled software. The stream-enabled software is able to run on a device even while the transmission/streaming of the software may still be in progress. Thus, a user does not have to wait for the completion of the software's download prior to starting to execute the software. In this paper, we suggest the new concept of software streaming system implement using the PageFault Interrupt Routine Hooking. As it is able to efficiently manage application, we do not have to install the entire software. In addition, we can save hardware resources by using it because we load basic binaries without occupying the storage space of the hardware.

Key Words : Software Streaming, Portable Executable, Implementation, Pagefault

Ⅰ. 서 론

소프트웨어를 실행시키기 위해서는 기본적으로 인스톨이라는 설치 과정을 거치게 되는데, 일반적으로는 소프트웨어가 담긴 CD-ROM이나 DVD를 CD-ROM 드라이브나 DVD 드라이브에 삽입하거

* 우송대학교 IT경영정보학과(교신저자)
** 우송대학교 게임멀티미디어학과

나 해당 프로그램을 다운로드 하여야 한다.

하지만, 대규모 사업장에서는 기술의 발달로 인해 소프트웨어를 단순하게 각자의 컴퓨터에 인스톨하여 사용하는 것이 아닌 다른 여러 가지 사용방법이 나타나게 되었는데, 이러한 형태의 대표적인 예가 Application Service Provider(ASP)의 등장이다. 즉, ASP는 애플리케이션의 기능과 그것에 수반되는 각종 서비스를 데이터 센터로부터 인터넷이나 광역통신망을 통하여 배포 및 관리하고 있다[1].

이러한 어플리케이션을 제공해주는 업체는 주로 데이터통신을 이용하여 개인 사용자가 서비스를 제공하는 서버에 접속하여 서버에 있는 프로그램을 사용하는 것이다. 즉, 클라이언트 컴퓨터를 사용하는 사용자는 해당 프로그램을 다운로드 받지 않고 단순히 서버 컴퓨터에서 멀리 떨어져 있는 모니터의 역할을 함으로써 처리하고자 하는 데이터 값을 주고, 필요한 결과 데이터 값을 받아오는 것이다.

이러한 서버기반 컴퓨팅과 관련하여, 분산 가상페이징 기술을 응용한 소프트웨어 스트리밍이라는 기술이 등장하였는데, 다시 말해 컴퓨터상에서 수행에 필요한 최소한의 응용 소프트웨어를 중앙 서버로부터 분산 가상페이징 기술로 자동으로 끌어와서 실행할 수 있는 소프트웨어를 배포 받아 사용하는 것이다. 즉, 다운로드이나 인스톨 과정을 거치지 않으면서도 수행에 필요한 소프트웨어를 중앙서버로부터 전송받고, 분산 가상페이징 기술을 이용하여 최소한의 수행코드만을 클라이언트에 공급받아 실행하기 때문에 어플리케이션은 클라이언트 측에서만 수행된다[2].

소프트웨어 스트리밍은 오디오, 비디오 스트리밍에 비해서 기술적인 면에서 구현하기 어려운 부분들이 많고, 순차적 실행이 아니라는 점에서 기존의 스트리밍과 큰 차이점이 있다. 즉, 기존의 스트리밍은 음악이나 영화와 같이 순차적으로 재생을 할 수 있는 것이어서, 제일 처음 부분에 해당하는 데이터를 보내고 그것을 실행하는 동안에 나머지 부분을 전송하면 되는 것이었다[3].

하지만 소프트웨어 스트리밍의 경우 비순차적이며, 순서를 정하기 어려운 작업의 연속으로 이루어지기 때문에 기존의 스트리밍과 많은 차이점을 가지고 있다. 그렇기 때문에 클라이언트는 사용자가 요청하는 기능을 순간순간 서버에 요청하여야 한다. 하지만 요청하는 자료를 매번 서버에 요청하는 것은 시간적 낭비일 뿐만 아니라 사용자의 소프트웨어 사용에 불편을 가져오게 된다.

따라서, 본 논문에서는 운영체제 인터럽트 루틴을 후킹하여 로컬에 설치되어 있지 않은 코드 바이너리를 실시간으로 전송받아 실제 물리메모리 영역에 로드시키는 방식을 사용하였다. 즉, 본 논문은 PageFault Interrupt Routine Hooking을 이용하여 PE 파일의 구조와 그에 해당하는 바이너리를 전송받아 시스템의 메모리에 로드시키는 방식을 적용함으로써 보다 더 효율적인 소프트웨어 스트리밍 시스템을 구축하는데 그 목적을 두고 있다.

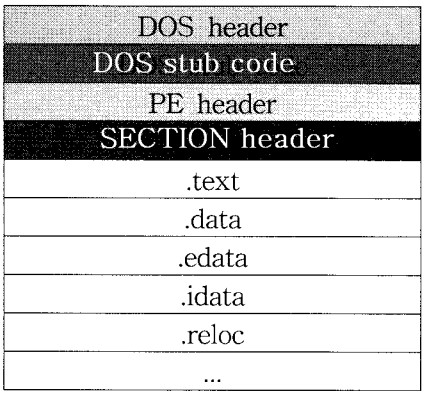
본 논문의 구성은 다음과 같다. 2장에서는 관련 기술 및 유사한 방식을 소개하고, 3장에서는 제안하는 기법인 페이지폴트 인터럽트 루틴 후킹 방식을 이용한 소프트웨어 스트리밍 시스템을 설명한 다음, 기존의 방법론과의 차이점을 밝힌다. 마지막으로 결론과 향후 과제는 4장에서 다루기로 한다.

II. 관련 기술

2.1. 실행파일(PE) 구조 및 원리

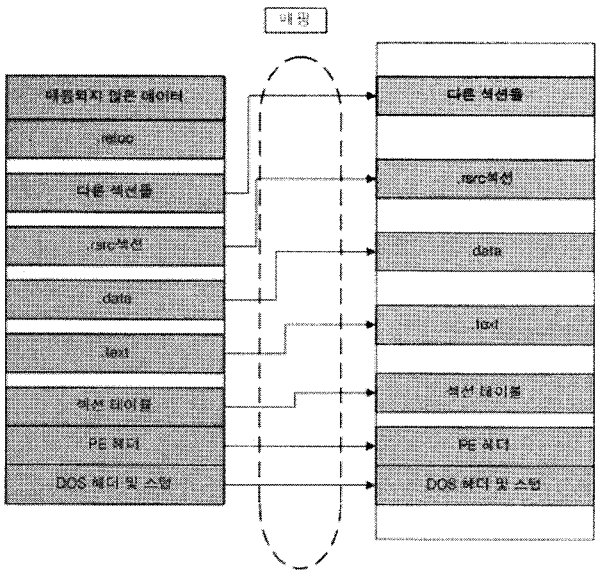
Windows 운영체제에서는 실행 코드와 데이터들을 저장하기 위하여 사용하는 포맷으로 PE (Portable Executable)라는 형태를 적용하고 있다. 이 PE 파일 포맷은 유닉스 시스템에서의 COFF (Common Object File Fomat) 파일 포맷에 기반하여 나온 것으로 Windows NT 개발당시 여러 CPU플랫폼에서 동일한 파일 포맷으로 실행이 가능하도록 하기 위한 용도로 만들어졌기 때문에 Win32플랫폼 하에서 공통적으로 사용이 가능하다는 것을 의미한다[4].

PE 파일은 크게 Dos헤더, Dos호환더미, PE헤더, Section헤더, Section으로 나누어져 있고 그 정보들은 프로세스 로더가 사용하여 프로그램을 실행하게 된다. PE 파일의 구조는 그림 1과 같다.



(그림 1) PE 파일의 구조

PE 파일이 실행될 때에는 우선 메모리에 로드되어야 하는데, 단순히 로드되는 것이 아니라 정확하게 표현하지면 가상 주소 공간에 매핑되는 것이다. 메모리에 로드될 때 윈도우의 프로그램 로드는 프로세스 커널 객체를 생성한다. 그리고 그 프로세스에 속하는 4G Byte의 가상 주소 공간을 만든다. 이렇게 가상 주소 공간이 만들어 지면 로더는 해당 PE 파일을 포함할 수 있는 충분한 공간을 그림 2와 같이 가상 주소 공간에 예약 한다.



(그림 2) PE의 VAS로의 맵핑

2.2 윈도우즈 커널영역

커널은 컴퓨터 운영체계의 가장 중요한 핵심으로서, 운영체계의 다른 모든 부분에 여러 가지 기본적인 서비스를 제공한다. 커널은 쉘과 대비될 수 있는데, 쉘은 운영체계의 가장 바깥부분에 위치하고 있으면서, 사용자 명령어에 대한 처리를 담당한다. 커널과 쉘이라는 용어는 IBM 메인프레임을 제외하고, 유닉스와 기타 몇몇 운영체계에서 자주 사용된다. 일반적으로 커널에는 종료된 입출력 연산 등, 커널의 서비스를 경쟁적으로 요구하는 모든 요청들을 처리하는 인터럽트 처리기와, 어떤 프로그램들이 어떤 순서로 커널의 처리시간을 공유할 것인지를 결정하는 스케줄러, 그리고 스케줄이 끝나면 실제로 각 프로세스들에게 컴퓨터의 사용권을 부여하는 수퍼바이저 등이 포함되어 있다[5].

또한, 커널은 메모리나 저장장치 내에서 운영체계의 주소공간을 관리하고, 이들을 모든 주변장치들과 커널의 서비스들을 사용하는 다른 사용자들에게 고루 나누어주는 메모리 관리자의 역할을 가지고 있다. 커널의 서비스는 운영체계의 다른 부분이나, 흔히 시스템 호출이라고 알려진 일련의 프로그램 인터페이스들을 통해 요청된다.

2.3 소프트웨어 스트리밍 기술과 유사 방식과의 비교

소프트웨어 스트리밍 기술은 스트리밍 기법을 통하여 서버에 존재하는 응용 프로그램 이미지 중 일부를 클라이언트에서 실행하는 기술로, 클라이언트는 응용 프로그램을 자신의 저장장치에 전체 다운로드 및 직접 설치하지 않고도 실행에 필요한 부분만 네트워크를 통해서 서버로부터 전송받아 실행시키는 방식을 말한다[6].

이러한 소프트웨어 스트리밍 기술과 유사한 방식들을 살펴보면 표 1과 같다[7].

서버 수행 방식은 서버에서 프로그램이 실행되고 실행결과가 클라이언트의 모니터에 전송되는 방식이다. 대표적인 예가 Windows 2000부터 지원되는 터미널 서비스이다. 즉 터미널 서비스는 터미널을 통하여 서버에 저장되어 있는 소프트웨어를 원격으로 사용할 수 있도록 하는 것을 말한다.

또한, 클라이언트 설치 방식은 프로그램을 서버로부터 다운로드 받아서 프로그램이 실행되는 방식이다. 대표적인 예가 인터넷 온라인 게임인데, 온라인 프로그램은 서버로부터 실제 구동되는 프로그램을 모두 다운로드 받고 이를 클라이언트의

PC에 설치함으로써 구동되는 원리이다.

<표 1> 소프트웨어 스트리밍 유사방식 비교

구분	서버수행방식	클라이언트 설치방식	소프트웨어 스트리밍 방식
서버	-실행화면의 화면전송 -S/W를 서버에 설치 및 실행하고 화면을 전송	-S/W의 원격 설치 -가능한 이미지 파일화 -관리자가 원격으로 클라이언트에 다운로드시킴	-S/W의 실시간 전송 -가능한 이미지 파일화 -클라이언트의 요구에 따라 스트리밍
전송	- 화면전송 방식 - Pull & Push	-다운로딩 방식으로 S/W를 원격 설치 -Pull & Push	-스트리밍 방식으로 S/W 전송 -Pull, 비순차적 전송
클라이언트	-Terminal Service -클라이언트에서 전송된 화면이 보여짐	-원격으로 클리아인트 설치 및 실행	-Launchpad(자체기술)로 클라이언트에서 실행 -Cache(옵션), 복제 불가능
대표 제품	-WBT(MS) -Metaframe (Citrix, 미국) -투스칸, 만파식적(한국)	SMS(MS) -Tivoli(IBM) -TCO!Stream (한국)	-Softricity(미국) -Exent(이스라엘) Z!Stream(한국)

III. 소프트웨어 스트리밍 시스템 구축

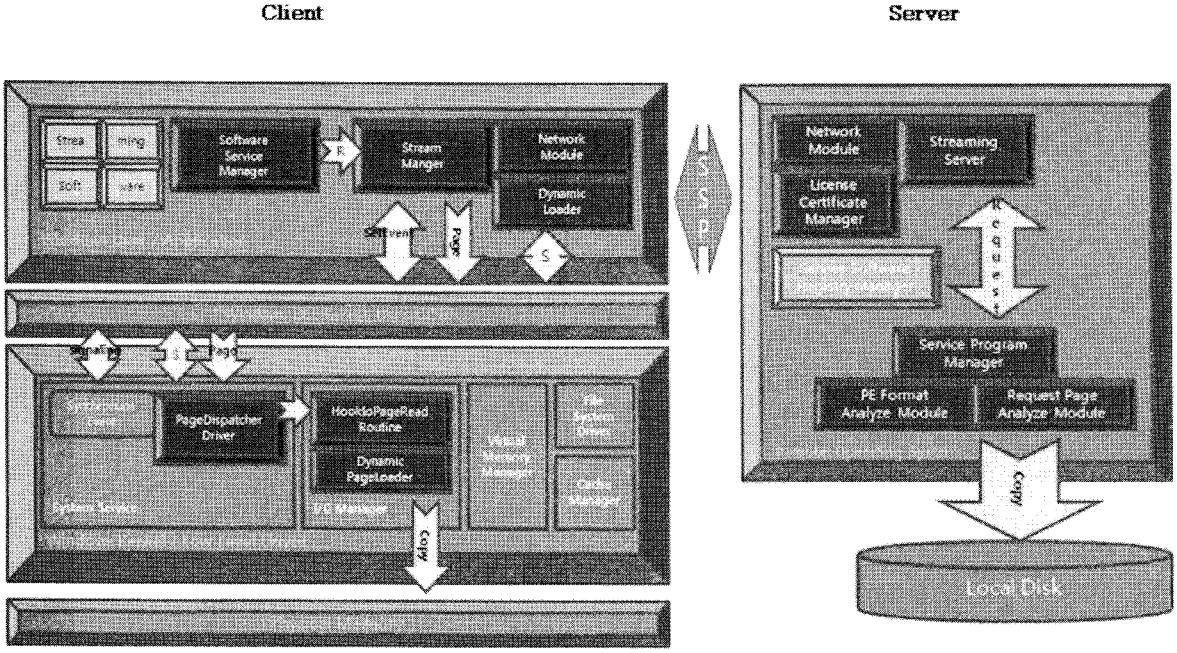
3.1 시스템 구축 구성도와 흐름

그림 3은 소프트웨어 스트리밍 시스템의 개략적인 구조를 표현한 System Architecture이다. 아키텍처는 크게 서비스 받을 소프트웨어를 요청하는 Client와 서비스할 소프트웨어를 제공하는 Server로 구성되어 있다.

먼저 Software Service Manager가 실행되는 순간부터 PageFault 인터럽트를 후킹하는 PageDispatcher Driver가 로딩 되어 PageFault 관련 I/O 루틴을 후킹하게 된다

또한, 사용자가 Software Service Manager 어플리케이션에서 원하는 소프트웨어를 클릭하는 순간부터 PageFault가 발생하고 Hooklo -PageRead 루틴이 선행되어 폴트 정보를 Synchronuos 메커니즘을 통해 User Level의 Stream Manager에게 폴트 정보를 전달한다.

한편, Stream Manager는 폴트 정보를 SSP (Software Streaming Protocol) 방식으로 Streaming Server로 전달하고 전달받은 SSP를 분석하여 해당 파일에서 요구된 페이지 Block을 클라이언트로 전달하게 된다.



(그림 3) System Architecture

마지막으로, Stream Manager는 서버로부터 수신한 SSP의 페이지 Block을 PageDispatcher로 전달하게 되고 전달받은 페이지는 실제 물리 메모리로 직접 로드되며 이 과정이 반복되면서 요청한 소프트웨어가 실행되게 된다.

3.2 PageFault Interrupt Routine 후킹

PageFault Interrupt Routine을 후킹하기 앞서 일단 Windows 시스템이 Interrupt Handler를 처리하는 과정을 아래의 그림 4를 통해 알아본다.

그림 4의 점선 영역이 디스크 상에서 페이지를 실제로 읽어오기 위해 최종적으로 호출되는 루틴이다. 실제 windbg를 통해 파일 읽기요청을 트레이싱하여 커널 스택 프레임을 통해 그 정보를 알아낼수 있다.

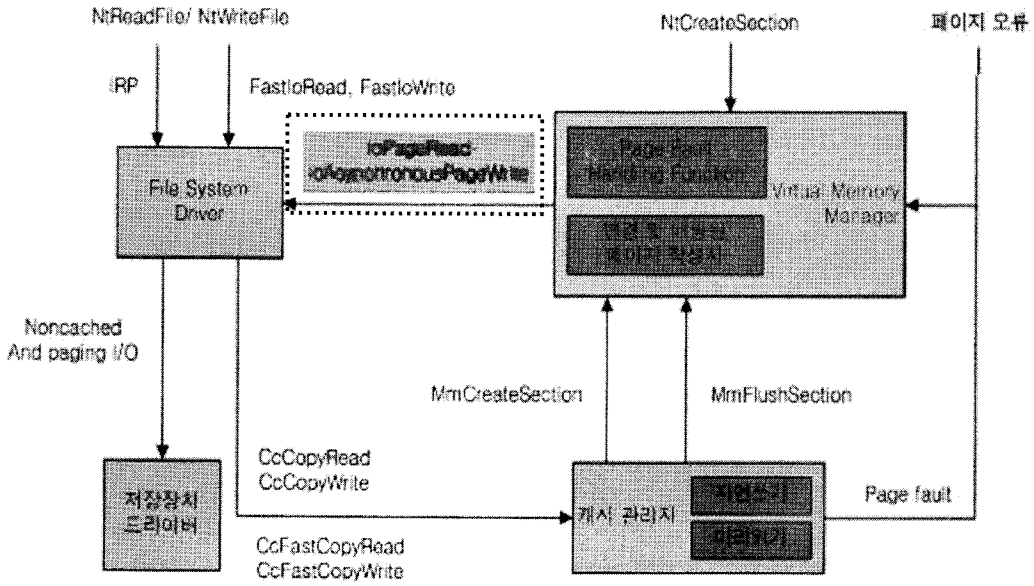
```
# ChildEBP RetAddr
00 f779fc88 804fcdf4 nt!IoPageRead+0x1b
01 f779fcfc 804eb7de nt!MiDispatchFault+0x274
02 f779fd4c 804e3ff1 nt!MmAccessFault+0xc09
03 f779fd4c 7c8647b7 nt!KiTrap0E+0xcc
04 0012f638 00401d68 0x7c8647b7
05 0012f6a0 5f4373fc 0x401d68
```

VAD는 캐시관리자의 읽기 중인 파일의 매핑된 뷰를 서술하며 따라서 MmAccess-Fault는 MiDispatchFault를 호출하여 유효한 가상 메모리 주소에서 페이지 폴트를 처리한다.

MiDispatch-Fault는 파일 개체를 가지고 I/O관리자의 IoPageRead함수를 호출하여 IRP_MJ_READ 타입과 IRP_NOCACHE 와 IRP_PAGING_IO 속성을 설정하여 파일개체를 가리키는 장치 개체를 소유한 FSD에게 전달한다.

실제로 IoPageRead루틴은 커널에 의해서만 호출되는 루틴이다. 즉 SSDT(System Service Descriptor Table)에서는 해당 루틴의 주소를 찾을 수 없기 때문에 여기에서는 우회 패칭을 적용한다. 즉, Non-paged되어 메모리에 상주된 코드를 직접 수정하는 것이다. 여기서 중요한 점은 Instruction레벨에서 다른 루틴으로 우회하도록 하고 필요에 따라서 다시 원래 루틴으로 돌아가도록 하는 것이다. IoPageRead 루틴이 동작 형태를 알아보기 위해 Windbg를 통해 Disassembly한 내요을 살펴보면 다음과 같다.

```
nt!IoPageRead:
804fd174 8bff          mov     edi,edi
804fd176 55            push    ebp
804fd177 8bec          mov     ebp,esp
804fd179 6a00      push    0
804fd17b ff7518    push    dword ptr [ebp+18h]
```



(그림 4) 파일 시스템의 캐시 관리자 및 메모리관리자의 상호 작용

```
804fd17e ff7514      push  dword ptr [ebp+14h]
804fd181 ff7510      push  dword ptr [ebp+10h]
804fd184 ff750c      push  dword ptr [ebp+0Ch]
804fd187 ff7508      push  dword ptr [ebp+8]
804fd18a e81fffffff  call  nt!IopPageReadInternal (804fd0ae)
804fd18f 5d          pop   ebp
```

위의 내용을 보면 전달받은 인수를 스택에 쌓아 다른 CoreRoutine을 호출한다. IoPageRead의 실제 코드는 804fd18a번지에 위치한 call 804fd0ae 번지에 있음을 추정할 수 있다.

3.3 Interrupt Routine 후킹 후 폴트 처리 과정

IoPageRead루틴 후킹은 일종의 광역후킹이다. 즉, 윈도우에서 일어나는 모든 페이지 폴트를 추적하기 때문에 스트리밍 서비스되는 여러 프로세스를 병렬적으로 처리할 수 있다는 점이 장점이며, 단점은 필요 이상의 오버헤드를 발생시켜 시스템 전체의 성능을 저하 시키는 원인으로 작용한다. 그렇기 때문에 타겟 프로세서에 대한 정확한 필터링이 필요하다.

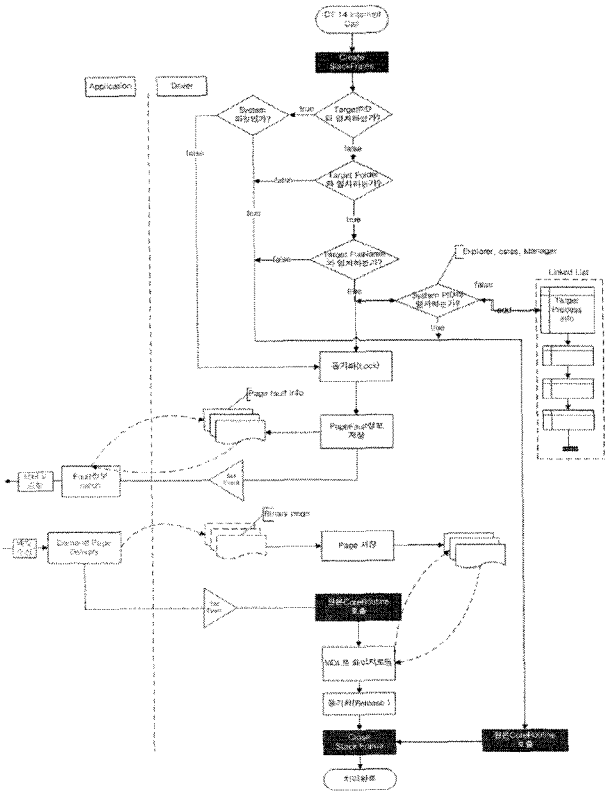
이를 통해, PageFault 정보를 저장하고 어플리케이션의 이벤트 객체를 신호화하여 드라이버의 Fault 정보를 전달하고 해당정보를 서버로 전송하게 된다.

또한, 요구 페이지를 서버로부터 수신하여 드라이버 저장 공간에 저장한 뒤, 드라이버 이벤트 객체를 시그널하여 정지된 컨텍스트를 다시 복구 시키고, 원본 Core루틴을 호출하여 Fault 처리를 정상적으로 마친 다음, 해당 페이지를 실제 메모리에 로드한다. 그림 5는 이러한 과정을 흐름도로 나타낸 것이다.

3.4 Streaming Server의 동작 실험

제안한 방법론을 기반으로 구현된 소프트웨어 스트리밍 시스템에 대한 동작 실험을 실시하였다. 그림 6에서 보는 바와 같이 먼저, Client 쪽에서 해당 소프트웨어를 클릭하면 Pagefault Status에서 확인 할 수 있듯이 Pagefault가 발생하고, HookIoPageRead 루틴이 선행되어 fault 정보를 Steam Manager에게 전달한다. Stream Manager는 fault 정보를 Streaming Server로 전달하고, 전달받은 SSP를 분석하여 요구되는 페이지 block을 Client에 전달하게 된다.

또한, 서버로부터 수신한 SSP의 페이지 block을 PageDispatcher로 전달하게 되는데, 이를 실제 물리 메모리로 직접 로드시킴으로써 요청했던 소프트웨어가 실행됨을 확인할 수 있었다.

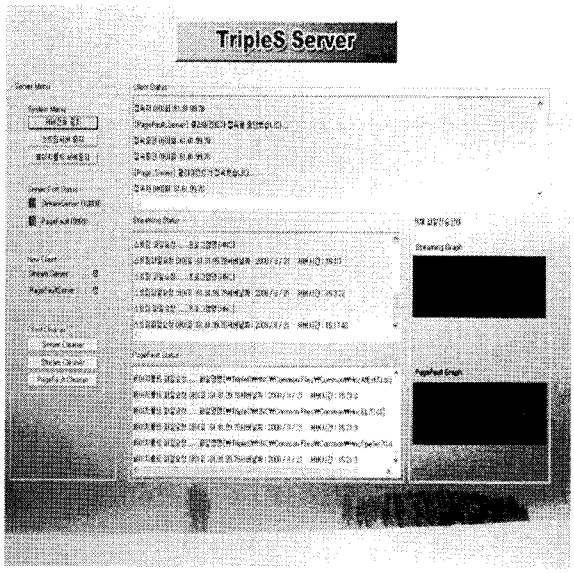


(그림 5) PageFault 처리 과정

그림 6은 Client 시스템과 Streaming Server를 연결하여 소프트웨어 스트리밍 시스템을 실험하는 그림이다.

Ⅳ. 결 론

본 논문에서는 기존의 On-Demand 소프트웨어의 기술인 NFS(Network File System) 방식, 즉 원격지 시스템의 파일시스템을 로컬파일시스템처럼 사용하는 방식이 아니라, 운영체제 인터럽트 루틴을 후킹한 다음, 로컬에 설치되어 있지 않은 코드 바이너리를 실시간으로 전송받아 실제 물리메모리 영역에 로드시키는 방식을 사용하였다.



(그림 6) Streaming 서버의 동작 실험

NFS 방식의 경우, 소프트웨어 실행에 필요한 기본적인 데이터를 전송받아 사용하지만, 본 논문에서 제안한 PageFault Hooking을 이용한 방식은 PE 파일의 구조와 그에 해당하는 바이너리를 받아와 시스템의 메모리에 로드시켜 이용하다는 차이점이 있다.

이러한 소프트웨어 스트리밍 서비스를 통해 보다 효율적으로 응용소프트웨어를 관리할 수 있을 뿐만 아니라, 소프트웨어 전체를 인스톨할 필요가 없기 때문에 하드웨어의 저장 공간을 거의 사용하지 않고 기본적인 바이너리를 받아서 로드하기 때문에 하드웨어 자원을 최대한 줄일 수 있는 것이다.

그리고 이용자는 자신의 ID만 등록해 놓으면 언제 어디서든지 소프트웨어를 이용 할 수 있어 장소의 제약이 없다는 장점이 있지만, 이로 인해 상용 소프트웨어가 불법적으로 사용될 수 있기 때문에 향후 이러한 소프트웨어 스트리밍에 대한 법적 대책도 필요할 것으로 생각된다.

참 고 문 헌

[1] 모리타 수수무, 새로운 비즈니스 모델의 패자 ASP, Pearson Education Korea, 2000.

[2] Kuacharoen, P. Monney V. J., and Madisetti V. K, "Software Streaming via Block Streaming", '03 Conference, Munich, Germany, IEEE and ACM SIGDA, pp.912-917.

[3] 윤선희, 이헌희, 신재호, 조용순, 이재철, "소프트웨어 스트리밍 방식의 법적 연구", 한국소프트웨어저작권협회 연구보고서, 2004.

[4] 이호동, Windows 시스템 실행 파일의 구조와 원리, 한빛미디어, 2005.

[5] 윤성우, 윈도우즈 시스템 프로그래밍, 한빛미디어, 2007.

[6] 최완 외 7명, "온디맨드 소프트웨어 스트리밍 기술현황 및 개발방향", 전자통신동향분석, 제19권, 제5호, 2004.

[7] 이상범, "대학소프트웨어 자원의 효율적 활용 사례, 대학 사용 S/W 운영 개선 방안", 2004년도 춘계 대학정보전산기관장 세미나 자료, 2004.

김 한 국 (Hanl-Gook Kim)

- 1999년 건국대학교 산업공학과
- 1999년~2000년 삼성전자 반도체총괄 사원
- 2003년 동경공업대학 경영공학전공
- 2007년 동경공업대학 경영공학 공학박사
- 2007년~2008년 한국과학기술정보연구원 선임프로젝트연구원
- 2008년~현재 우송대학교 IT경영정보학과 초빙교수
- 관심분야: 성과관리 방법론, e-Business, BPM

이 창 조 (Chang-Jo Lee)

- 종신회원
- 1989년 인하대학교 전자계산학과
- 1991년 인하대학교 대학원 컴퓨터과학전공
- 1996년 고려대학교 대학원 컴퓨터과학전공
- 1990년~1994년 한국과학기술연구원/시스템공학연구소(KIST/SERI) 소프트웨어공학연구부 선임연구원 (현, 한국전자통신연구원)
- 1994년 ~ 1996년 한국문화예술진흥원 문화정보사업본부 선임연구원
- 1996년 ~ 현재 우송대학교 게임 멀티미디어학과 교수
- 2005년 카네기멜론대학(ETC) 연수
- 관심분야 : 게임콘텐츠, 디지털콘텐츠융합기술

논문접수일 : 2009년 4월 1일
논문수정일 : 2009년 4월 24일
게재확정일 : 2009년 5월 8일