네트워크 고장감내 소프트웨어 스트리밍 기술의 설계 및 구현

Design and Implementation of Network Fault-Tolerant Application Software Streaming

심정민, 김원영, 최 완
한국전자통신연구원 온디맨드서비스연구팀
Jeong-Min Shim(jmshim@etri.re.kr), Won-Young Kim(wykim@etri.re.kr),
Wan Choi(wchoi@etri.re.kr)

요약

소프트웨어 스트리밍은 사용자가 자신의 컴퓨터에 설치되어 있지 않고 서버에 있는 응용 프로그램을 스트리밍 받아 바로 사용할 수 있게 하는 가상화 기술이다. 이 기술을 이용하면 사용자는 응용 프로그램이 마치 로컬 컴퓨터에 설치된 것처럼 바로 사용할 수 있으며, 별도의 다운로드나 설치 과정을 요구하지 않는다. 소프트웨어 스트리밍은 네트워크를 기반으로 하기 때문에 제공되는 서비스는 네트워크 성능 및 상태에 영향을 받는다. 특히, 네트워크 고장이 발생하면 스트리밍이 더 이상 불가능하기 때문에 스트리밍 중인 응용 프로그램이 고장 나거나 심한 경우 시스템 전체가 고장나게 된다. 파레토의 원리(Pareto Principle)에 의하면, 대부분의 사용자들은 자주 사용하는 몇 가지 기능을 주로 사용한다. 이러한 원리에 따라 네트워크 고장 감지와 지능적인 스트리밍 기술을 제공한다면, 네트워크 고장이 발생하더라도 사용자들은 이미 스트리밍된 응용 프로그램의 기능을 중단 없이 사용할 수 있을 것이다. 본 논문에서는 네트워크 고장이 발생하더라도 로컬 컴퓨터에 스트리밍된 기능을 사용자들이 지속적으로 사용 할 수 있게 하는 네트워크 고장감내 소프트웨어 스트리밍(에버그린)의 개념 및 기술을 제안한다. 또한, 에버그린 기술의 구현에 대한 자세한 내용에 대해 논의한다.

■ 중심어 : | 소프트웨어 스트리밍 | 네트워크 고장감내 | 소프트웨어 가상화 |

Abstract

Application software streaming is a virtualization technology that enables users to use applications without installation on her/his computer. With application streaming service, a client immediately starts and uses the application as if it were installed. The application can be executed while executable codes for the application may still be streamed. Since the software streaming is based on networks, its service is affected by network failures. Network failures may cause the streamed application to stop, and to make it worse, also the system may crash because executable codes for the application can't be streamed from the streaming server. Using the Pareto principle (80 vs. 20 rule), users can be served continuously with the minimum functions that are frequently used, pre-fetched and cached if we provide a more intelligent and fault-tolerant streaming technique. This paper proposes the concept and technique named Evergreen. Using the Evergreen technique, users can continue using the streamed application while a network failure occurs, although user can access only the streamed code. We also discuss the implementation of Evergreen technique in details.

■ keyword : | Software Streaming | Network Fault-tolerant | Software Virtualization |

접수번호 : #060608-001
접수일자 : 2006년 06월 08일
심사완료일 : 2006년 09월 22일
교신저자 : 심정민, e-mail : jmshim@etri.re.kr

I. 서 론

스트리밍은 오디오, 비디오 및 멀티미디어 컨텐츠를 네트워크 상에서 실시간 또는 주문형으로 배포하는 기술로 사용되어 왔다[1-3]. 최근, 스트리밍 기술은 소프트웨어에 적용되어, 소프트웨어의 새로운 배포 기술로 발전해가고 있다[4-11]. 스트리밍 기술을 이용하면 컴퓨터마다 개별적으로 응용 프로그램의 업데이트나 설치를 하지 않고, 한꺼번에 응용 프로그램을 설치하거나 업데이트 할 수 있다. 또한, 스트리밍 기술은 응용 프로그램의 모든 모듈이 로컬 컴퓨터에 설치 또는 다운로드 되지 않아도 응용 프로그램을 실행 할 수 있다.

스트리밍 기술은 네트워크를 기반으로 하기 때문에 스트리밍 서비스의 품질은 네트워크 상태에 영향을 많이 받는다. 멀티미디어 스트리밍 서비스에서 일시적인 네트워크 고장은 단지 멀티미디어 재생의 일시적인 지연만을 초래한다. 이와 비교해서 소프트웨어 스트리밍 서비스에서는 일시적인 네트워크 고장이라고 해도 응용 프로그램의 실행에 필요한 실행 코드의 부재로 인해 응용 프로그램의 실행이 중단되고, 심한 경우에는 시스템 전체의 고장이 발생하여 응용 프로그램의 상태를 복구시킬 수 없다. 따라서 소프트웨어 스트리밍 서비스에서 네트워크 고장은 심각한 문제이다.

파레토 원리(Pareto Principle : 80 대 20 법칙)에 따르면 대부분의 응용 프로그램 사용자들(80%)은 응용 프로그램의 일부(20%)만을 자주 사용할 것이라고 한다[12]. 따라서 기존의 소프트웨어 스트리밍 기술에 스트리밍된 실행코드의 관리 기술과 네트워크 오류 처리 기술이 더해진다면, 이미 스트리밍된 일부(20%)의 기능에 대해 네트워크 고장시에도 지속적인 서비스가 가능하다. 이것은 순차적으로 컨텐츠 전체를 스트리밍 해야 하는 멀티미디어 스트리밍에서 네트워크 단절이 응용 프로그램과 시스템에 지장을 주지 않지만, 서비스 중단을 초래하는 것에 비해 매우 유용한 서비스가 될 수 있다.

본 논문에서는 네트워크 고장시에도 응용 프로그램에 대해서 지속적인 서비스를 지원하는 에버그린 기술을 제시한다. 에버그린 기술은 스트리밍된 실행 코드뿐만 아니라 관련 정보도 관리하여, 네트워크 고장 시 스트리밍 오류로 인한 응용의 고장을 방지하여 지속적인 서비스를 제공하는 기술이다. 따라서 에버그린 기술을 기존 스트리밍 기술과 결합하여 사용자들에게 더욱 신뢰성 있는 소프트웨어 스트리밍 서비스를 제공 할 수 있다.

본 논문은 다음과 같이 구성되어 있다. 2장은 소프트웨어 스트림이 기술과 관련된 연구와 소프트웨어 스트리밍의 개념 및 시스템 구조에 대해 설명한다. 3장에서는 제안하는 에버그린 기술의 설계에 대해 논의한다. 4장에서는 에버그린 기술의 구현 방법에 대해 기술하고, 마지막으로 5장에서 결론을 맺는다.

II. 관련 연구 및 소프트웨어 스트리밍의 개념

이 장에서는 소프트웨어 스트리밍의 개념, 시스템 구조 및 장점에 대해 설명한다. 또한, 소프트웨어 스트리밍의 동작 원리에 대해 설명하고, 소프트웨어 스트리밍 기술의 관련 연구에 대해 간략하게 설명한다.

소프트웨어 스트리밍은 응용 프로그램 전체를 미리 다운로드 하거나 설치하지 않고 응용 프로그램이 필요할 때 바로 사용이 가능하며, 네트워크 환경에서는 어디에서나 응용 프로그램을 사용할 수 있는 새로운 기술이다.

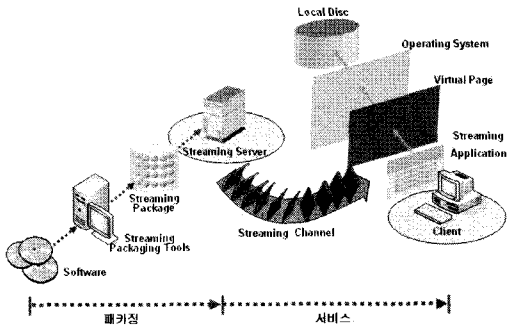


그림 1. 소프트웨어 스트리밍의 개념도

소프트웨어 스트리밍 서비스는 패키징과 서비스의 두 단계로 이루어진다. [그림 1]은 소프트웨어 스트리밍 서비스의 두 단계를 보이고 있다. 패키징 단계에서는 스트리밍 패키징 도구를 이용하여 기존의 응용 프로그램을 소프트웨어 스트리밍에서 사용할 수 있도록 변환한다.

스트리밍 패키징 도구는 응용 프로그램의 설치 정보와 실행 코드를 수집하고, 이 정보들을 이용하여 응용 프로그램 패키지를 스트리밍에 적합하도록 변환한다. 서비스 단계에서는 스트리밍 기술을 이용하여 사용자에게 응용 프로그램을 제공한다. 사용자가 네트워크를 통해 스트리밍 서버의 응용 프로그램을 요청하면, 스트리밍 서버는 스트리밍 패키징 도구에 의해 변환된 응용 프로그램의 초기 실행 코드를 클라이언트에게 보낸다. 사용자는 응용 프로그램의 초기 실행 코드가 도착하면 응용 프로그램 전체에 대한 실행 코드 없이도 응용 프로그램을 실행하여 사용할 수 있다. 게다가 클라이언트는 응용 프로그램에서 사용자가 사용하는 기능을 수행하는 필요한 실행 코드 및 데이터만을 스트리밍 한다.

소프트웨어 스트리밍 서비스를 위해서는 클라이언트 스트리밍 엔진과 스트리밍 서버가 필요하다. 클라이언트 스트리밍 엔진은 가상화 모듈과 통신 모듈로 구성되어 있다. 통신 모듈은 응용 프로그램을 실행하기 위해 가상화 모듈에서 요구하는 응용 프로그램의 실행 코드들을 스트리밍 서버로부터 전송 받는다. 가상화 모듈(Virtual Page)은 스트리밍 서버로부터 응용 프로그램의 실행 코드들을 전송받아 응용 프로그램을 실행한다. 응용 프로그램의 디스크 접근을 가로채어, 해당하는 실행 코드들이 클라이언트 엔진이 관리하는 로컬 디스크의 캐시에 있는지 확인한다. 만약 로컬 디스크의 캐시에 해당 실행 코드가 존재하지 않는다면, 통신 모듈에게 스트리밍 서버로부터 해당 실행 코드를 전송받도록 요청한다. 전송받은 실행 코드는 가상화 모듈에 의해 로컬 디스크에 임시 저장되며, 동시에 메모리에 적재 되어 응용 프로그램의 실행에 필요한 레지스트리를 가상으로 구성하며, 환경 설정들을 임시적으로 맞추어 준다. 응용 프로그램의 실행을 위해 설정된 환경 설정은 서비스가 종료되면 해제되며, 기존에 설치되어 있는 응용 프로그램과 충돌이 발생하지 않도록 관리한다. 스트리밍 서버는 스트리밍 패키징 도구에 의해 변환된 응용 프로그램 패키지를 유지하며, 클라이언트의 통신 모듈에서 실행 코드를 요청하면 해당하는 응용 프로그램의 실행 코드를 전송한다.

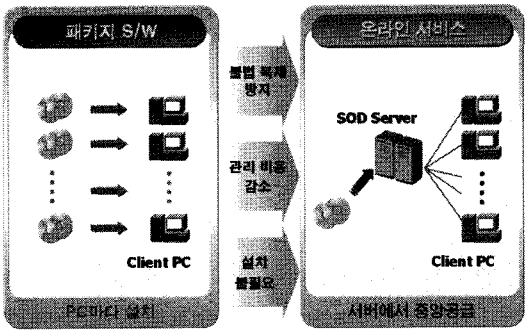


그림 2. 소프트웨어 스트리밍의 장점

[그림 2]는 기존의 설치를 기본으로 하는 패키지 프로그램과 비교하여 소프트웨어 스트리밍의 장점을 보이고 있다. 우선, 소프트웨어 스트리밍은 응용 프로그램을 중앙 집중적으로 관리하기 때문에 불법 복제를 방지할 수 있다. 둘째, 사용자들은 다운로드, 설치 및 환경 설정을 하지 않고도 응용 프로그램을 즉시 사용할 수 있다. 특히, 초보 사용자가 설치하기 힘든 응용 프로그램에 대해서는 더욱 유용한 기술이다. 마지막으로, 응용 프로그램을 서버에서 중앙 집중적으로 관리하기 때문에 응용 프로그램에 대한 유지보수 비용을 절감할 수 있다.

최근 여러 업체들이 소프트웨어 스트리밍 기술을 이용한 제품들을 출시하거나 연구하고 있다. ETRI[11]는 리눅스 환경에서 공개 소프트웨어 기반 온-디맨드 기술을 연구하고 있다. SoftOnNet[5]은 소프트웨어 스트리밍 시스템인 Z!Stream 제품을 제공하고 있으며, AppStream[4]은 스트리밍 기술을 이용하여 응용 프로그램의 배포, 관리 및 서비스를 지원하고 있다. Softricity의 SoftGrid 플랫폼[6]은 응용 프로그램의 배포 및 관리 비용을 절감하기 위해서 중앙 집중의 소프트웨어 온-디맨드 서비스를 제공하며, 서비스에 제공하는 윈도우 소프트웨어를 변환하기 위해 소프트웨어 가상화를 제공한다. 또한, Stream Theory[7]도 소프트웨어 온-디맨드 서비스를 제공하는 소프트웨어 스트리밍 기술의 대표 회사 중 하나이다. Kuacharoen[13]은 원격의 임베디드 시스템에서의 소프트웨어 스트리밍에 대해 연구했다. 또한, 이들은 소프트웨어 스트리밍과 블록 스트리밍에 대한 자원 이용에 대해서도 논의했다.

III. 에버그린 : 네트워크 고장감내 소프트웨어 스트리밍 서비스

소프트웨어 스트리밍은 네트워크를 기반으로 하기 때문에 네트워크 상태가 서비스의 품질에 많은 영향을 준다. 기존의 소프트웨어 스트리밍 서비스에서는 네트워크 통신으로 인한 실행 코드 전송에 따른 응용 프로그램의 실행 시간의 지연을 줄이고자 캐시 기법을 사용하고 있기는 하지만, 네트워크 고장으로 인한 문제는 고려하지 않았다. 따라서 기존의 서비스에서 네트워크 고장이 발생했을 때, 사용자가 요청한 기능의 실행 코드가 로컬 시스템의 캐시에 존재하지 않는다면, 클라이언트가 응용 프로그램의 실행에 필요한 페이지들을 스트리밍 서버로부터 받을 수 없기 때문에 응용 프로그램은 더 이상 동작할 수 없고, 사용자는 작업을 멈추어야만 한다[4-10]. 예를 들어, 사용자가 응용 프로그램에서 아직 스트리밍 되지 않은 "문서 출력" 메뉴를 눌렀을 때 갑작스런 네트워크 고장으로 해당 메뉴의 실행 코드를 스트리밍 할 수 없다면, 응용 프로그램은 응답 없이 그 상태로 멈추거나 응용 프로그램 또는 시스템 고장이 발생할 것이다.

여기에서 우리는 네트워크 고장이 발생하더라도 지속적인 스트리밍 서비스가 가능한 네트워크 고장감내 소프트웨어 스트리밍 기술(일명 : 에버그린)을 제안한다. 에버그린 기술은 아직 스트리밍 되지 않아 사용이 불가능한 기능에 대해 경고 메시지를 제공하거나 응용 프로그램의 해당 메뉴를 비활성화 함으로써 사용자들이 사용 불가능한 기능들이 무엇인지를 알 수 있도록 도와준다. 따라서 에버그린 기술은 네트워크의 상태에 따라 더 유용한 정보를 사용자에게 제공함으로써 응용 프로그램이나 시스템의 고장이 없이 소프트웨어 스트리밍 서비스를 지속적으로 제공할 수 있도록 도와준다.

1. 네트워크 고장감내 소프트웨어 스트리밍

에버그린 기술은 기능별 스트리밍과 네트워크 고장으로 인한 오류 처리가 복합된 기술이다. 기능별 스트리밍은 사용자가 원하는 기능별로 스트리밍을 수행하는 기술이고 네트워크 고장 처리 기술은 네트워크 고장이 발생한 것을 탐지하고 이로 인해 발생하는 오류들을 상황에 맞게 처리하는 기술이다. 네트워크 고장이 발생한 경우 지속적인 서비스를 제공하기 위해서는 현재 스트리밍된 실행코드들에 대한 정보가 필요하다. 즉, 사용자가 요청한 기능을 수행하기 위해 현재 그 실행 코드가 스트리밍되어 있는지를 판단할 수 있어야 한다.

먼저, 기능별 스트리밍에 대한 동작 방식을 기술한다. 기존의 서비스에서 클라이언트와 서버 사이의 전송 단위는 페이지인 반면, 기능별 스트리밍에서는 페이지 단위뿐 만 아니라 기능 단위(Functional Unit)로 전송한다. 기능 단위는 응용 프로그램에서 각 기능의 동작에 필요한 페이지들의 집합을 의미한다. [그림 3]은 기능 단위의 예를 보이고 있다. 기능 A에 대한 기능 단위는 페이지 4, 5, 6, 12, 15로 구성된다. 기능별 스트리밍을 위해서는 응용 프로그램의 모든 기능과 페이지들 사이의 관계를 기술한 기능별 스트리밍 정보 테이블(FSIT : Functional unit Streaming Information Table)이 클라이언트에서 필요하다. 임의의 응용에 대한 FSIT 정보는 3.2절에서 기술될 스트리밍 패키지 분석도구에서 생성되고, 해당 응용의 최초 스트리밍 시 서버로부터 최초에 전송되는 데이터이다. 클라이언트는 이 정보와 함께 스트리밍 된 기능들의 목록(s_flist)과 페이지들의 목록(s_plist)들을 유지한다. 사용자가 응용 프로그램의 "인쇄" 같은 기능을 선택하면, 클라이언트는 스트리밍 서버로부터 기능에 대응하는 페이지들의 집합인 기능 단위를 스트리밍하고 클라이언트에서 해당 응용에 대한 s_flist와 s_plist를 갱신한다.

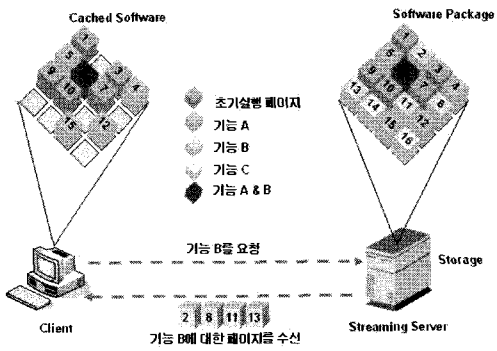


그림 3. 기능별 스트리밍

[그림 3]은 기능별 스트리밍에 대한 예를 보이고 있다. 만약 사용자가 기능 A를 사용하려 하면 클라이언트는 기능 A에 대응하는 페이지 4, 5, 6, 12, 15를 스트리밍 서버로부터 스트리밍하고 s_plist와 s_flist를 갱신한다. 다음에 사용자가 기능 B를 사용하려 하면 클라이언트는 s_plist를 확인하여 기능 B에 대응하는 페이지들 중에 이미 전송된 페이지가 있는지 검토한다. 이때, 기능 B에 대응하는 페이지들 중에 페이지 6이 이미 전송되었기 때문에 클라이언트는 페이지 6을 제외하고 기능 B에 대응하는 나머지 페이지 2, 8, 11, 13을 스트리밍 서버로부터 스트리밍 하고, s_plist와 s_flist를 갱신한다. 이처럼 사용자에 의해 사용되는 기능 단위로 클라이언트는 실행 코드 요청을 스트리밍 서버에게 전달하고, 서버로부터 전송된 실행 코드들에 대한 상태 정보는 클라이언트에서 관리된다.

네트워크 고장과 복합된 에버그린의 처리 방식을 기술하면 다음과 같다. 만약 네트워크 고장이 발생했을 때, 사용자가 기능 C를 사용하려 한다면, 클라이언트는 해당 기능의 페이지들이 스트리밍 되었는지 s_flist를 통해 확인한다. 만약 해당 기능이 스트리밍 되지 않았다면, 클라이언트는 이 기능을 사용할 수 없다는 경고 메시지를 표시하여 사용자에게 알린다.

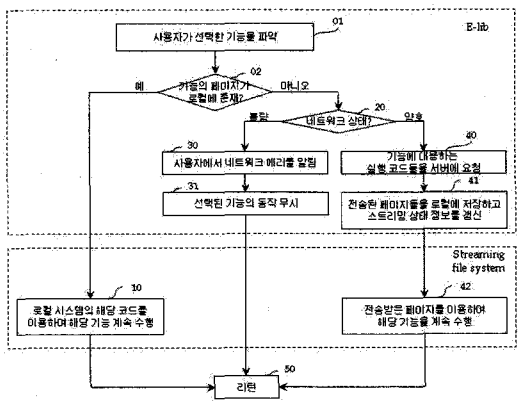


그림 4. 네트워크 상태에 따른 클라이언트의 서비스 흐름도

네트워크 상태에 따른 서비스 처리 과정을 [그림 4]에 도시하였다. 사용자가 응용 프로그램의 메뉴를 클릭하면, 클라이언트는 선택된 기능이 무엇인지 파악한다(01). 선택된 기능의 페이지들이 로컬 시스템에 존재하는지 확인하고(02), 만약 로컬 시스템에 존재하면 해당 기능은 계속 수행된다(10). 만약, 로컬 시스템에 해당 페이지들이 없다면, 네트워크 상태를 확인한다(20). 클라이언트는 주기적으로 네트워크 상태를 확인하여 그 정보를 유지한다. 만약, 네트워크가 끊어져 있다면 클라이언트는 사용자에게 해당 기능을 사용할 수 없다는 정보를 알려주고(30), 해당 기능의 동작은 무시된다(31). 만약 에버그린 서비스가 지원되지 않는다면, 이 시점에서 시스템이나 응용 프로그램의 고장이 발생할 것이다. 그러나 에버그린에서는 스트리밍된 실행 코드들과 네트워크 상태의 관리를 통해 사용자에게 서비스에 대한 자세한 상태 정보를 전달하고 지속적인 서비스를 제공한다. 만약 네트워크 상태가 정상이라면, 클라이언트는 선택된 기능에 대응하는 실행 코드들을 스트리밍 서버에게 요청한다(40). 그리고 스트리밍 서버로부터 스트리밍한 실행 코드의 페이지들을 로컬 시스템의 캐시에 저장하고, s_flist와 s_plist의 스트리밍 상태 정보를 갱신하며(41), 응용 프로그램은 서버로부터 전송받은 실행 코드를 이용하여 계속 동작한다.

2. 스트리밍 패키지 분석도구

앞 절에서 언급한 것처럼 에버그린 서비스를 위해서는 응용 프로그램에 대한 기능별 스트리밍 정보 테이블이 제공되어야 한다. 기능별 스트리밍 정보를 추출하는 방법은 두 가지가 있다. 하나는 응용 프로그램의 원시 코드 및 실행 코드를 분석하여 호출(Call) 관계가 있는 실행 코드들의 물리적 위치를 모으는 것이고, 또 다른 방법은 시뮬레이션을 통해 기능별 스트리밍 정보를 추출하는 것이다. 원시 코드 및 실행 코드를 분석하는 경우 프로그래밍 언어, 컴파일러 및 OS에 따라 내용이 달라지기 때문에 이러한 분석 방법에는 많은 노력과 비용이 요구된다. 따라서 우리는 시뮬레이션을 통해 응용 프로그램의 기능별 스트리밍 정보를 추출한다.

시뮬레이션을 위해서는 스트리밍 패키지 분석도구(SPA : Streaming Package Analyzer)와 같은 도구가 필요하다. 스트리밍 패키지 분석도구는 응용 프로그램

을 실행하면서, 각 기능과 기능에 대응하는 페이지 정보를 추출한다. 또한, 이 도구는 기존의 패키징 도구와 결합될 수 있다. 기존의 패키징 도구에서는 응용 프로그램을 스트리밍 패키지로 변환 할 뿐만 아니라 응용 프로그램의 실행을 위해 필요한 환경 설정 정보를 추출하거나 라이브러시와 같은 별도의 시스템 데이터도 스트리밍 패키지로 변환 할 수 있다.

스트리밍 패키지 분석도구는 응용 프로그램을 실행하기 위해 필요한 최소한의 초기 실행 코드 페이지들을 수집하고, 응용 프로그램의 각 기능들의 실행에 필요한 실행 코드 페이지들을 추출한다. 여기에서 추출하는 정보들은 기능의 이름과 페이지 정보이며, 페이지 정보는 기능과 관련된 파일 이름과 파일에서의 위치이다. 또한, 기능의 이름을 알아내기 위해 응용 프로그램이 사용하는 GUI 라이브러리를 수정하였다.

[그림 5]는 스트리밍 패키지 분석도구의 동작 개념을 보인다. 분석도구는 응용 프로그램을 실행할 때, 각 기능을 동작 시키는 이벤트를 가로채서 기능의 이름을 얻어내고, 기능의 실행을 위해 접근되는 페이지들을 수집한다. 이렇게 수집된 정보를 이용하여 기능별 스트리밍 정보 테이블을 생성한다.

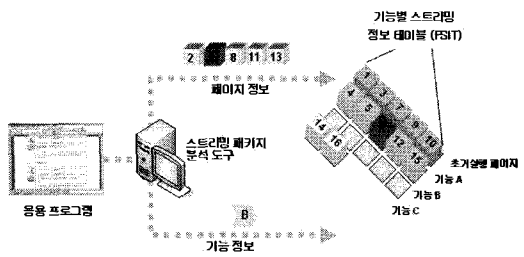


그림 5. 스트리밍 패키지 분석도구의 동작 개념

그런데, 여러 기능의 실행에 하나의 페이지가 공통적으로 사용되는 경우, 이러한 페이지 정보를 수집하는 것은 쉽지 않다. 예를 들어, 기능 A, B에 대해서 각각의 기능이 4, 5, 6, 12, 15와 2, 6, 8, 11, 13의 페이지 집합으로 이루어져 있다고 가정하자. 만약, 분석도구를 통해 기능 A를 실행하면 분석도구는 기능 A에 대응하는 페이지 4, 5, 6, 12, 15를 얻는다. 그 다음에 기능 B를 실행하면 분석도구는 기능 B에 대응하는 페이지 중에 페이지 6은 수집되지 않을 것이다. 왜냐하면, 페이지 6은 기능 A를 실행하기 위해 이미 메모리에 올라가 있기 때문이다. 따라서 기능 B에 대응하는 페이지는 2, 8, 11, 13으로 수집되고, 만약 실제 서비스에서 사용자가 기능 A보다 기능 B를 먼저 사용하려 한다면, 수집되지 않은 페이지 6은 클라이언트로 전송되지 않을 것이고, 이로 인해 응용 프로그램에 에러가 발생할 수 있다.

이러한 문제를 해결하기 위해서 두 가지의 접근 방식을 고려했다. 하나는 각 기능을 수행한 후에 메모리에 올라온 페이지들을 삭제하는 방법이다. 이 접근 방식을 위해서는 운영체제에 새로운 메모리 관련 인터페이스(함수)를 추가해야 한다(즉, 커널을 수정해야 한다). 다른 하나는 앞에서 실행했던 기능의 페이지들을 포함하여 페이지 정보를 수집하는 방법이다. 좀 더 자세히 설명하면, 처음에 수집된 기능의 페이지 정보는 다음에 실행하여 수집된 페이지 정보에 포함된다. 세 번째 기능의 페이지 정보는 해당 기능에 대응하는 페이지 정보뿐만 아니라, 앞에서 실행했던 처음과 두 번째 기능의 페이지 정보를 포함하는 것이다. 이 접근 방식은 실제 서비스에서 기능이 수집된 순서와 다르게 사용자가 기능을 사용할 경우 불필요한 페이지들도 전송해야 하기 때문에 비효율적일 수 있다.

시뮬레이션을 이용하는 방법은 공통 페이지 문제를 제외하더라도, 분석도구 사용자가 각 기능을 직접 실행해야 하기 때문에 어려운 작업일 수 있으나 다른 방법보다 비교적 수행 가능하다.

IV. 에버그린 서비스의 구현

앞 장에서 설명한 것처럼 에버그린 서비스를 위해서는 기존의 시스템에 클라이언트 스트리밍 엔진과 스트리밍 패키지 분석도구의 수정과 추가 개발이 필요하며 본 연구에서는 리눅스 환경의 사용자를 위해서 위의 두 가지를 동시에 개발하고 있다. 제안하는 시스템에서 스트리밍의 대상이 되는 소프트웨어는 리눅스용 소프트웨어이며, Fedora Core4와 한국전자통신연구원[11]에서

개발한 Booyo1.0 운영체제에서 개발하고 있다.

[그림 6]은 리눅스 환경의 클라이언트 스트리밍 엔진의 내부 구조를 보이고 있다. Launcher는 응용 프로그램을 시작하고 스트리밍 파일 시스템(streaming file system)은 스트리밍 서버로부터 실행 코드를 스트리밍하며 스트리밍된 페이지들을 관리한다.

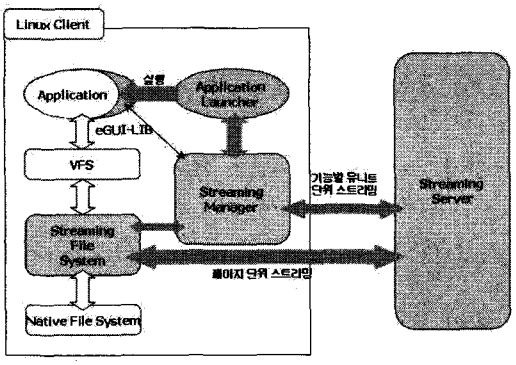


그림 6. 클라이언트 스트리밍 엔진 구조도

에버그린 서비스를 위해서 스트리밍 서버는 응용 프로그램의 기능별 스트리밍 정보 테이블의 실행 패키지를 유지한다. 만약 클라이언트로부터 응용 프로그램이 요청되면, 대응하는 FSIT을 클라이언트의 스트리밍 관리자에게 보낸다. 스트리밍 관리자는 사용자에 의해서 응용 프로그램이 필요로 하는 실행 코드를 기능 단위로 스트리밍하며, 스트리밍된 기능과 페이지 정보들을 유지한다. 만약 사용자가 응용프로그램의 기능을 사용하려고 하면, 그 이벤트는 에버그린 서비스를 위해 수정된 사용자 인터페이스 라이브러리(eGUI-LIB)를 통해 스트리밍 관리자에게 보고되고, 스트리밍 관리자는 해당 기능에 대응하는 모든 페이지들을 스트리밍 서버에서 스트리밍 한다. 동시에, 스트리밍 파일 시스템은 스트리밍된 페이지들을 이용하여 응용 프로그램을 동작시킨다. 네트워크에 이상이 있을 때 사용자에 의해 요청된 기능이 아직 스트리밍 되지 않은 기능이라면, 그 기능의 동작은 무시된다.

[그림 7]은 스트리밍 패키지 분석도구의 실행화면을 보이고 있다. 이것은 abiword[14]라는 응용 프로그램에 대한 FSIT의 일부정보를 보이고 있다. [그림 7]의 실행화면에서 첫 줄은 초기 실행을 위해 필요한 페이지들이며, 두 개의 파일에서 1394개의 페이지가 수집되었다. 각 기능에 대해서도 [그림 7]에서처럼 페이지 정보를 수집할 수 있으며, 전송 단위인 기능 단위는 분석도구 사용자가 조정할 수 있다.

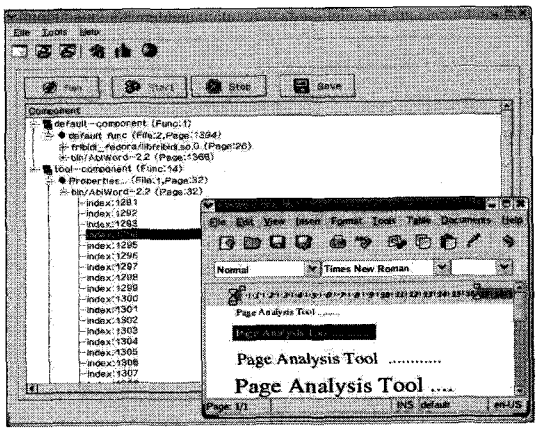


그림 7. 스트리밍 패키지 분석도구 실행 화면

V. 결 론

소프트웨어 스트리밍은 응용프로그램의 설치 없이 사용자가 원하는 기능을 온라인을 통해 많은 사용자들에게 경제적인 방법으로 소프트웨어를 배포하고 응용프로그램의 경제적인 사용을 지원하기 위한 새로운 기술로 등장했다.

소프트웨어 스트리밍은 네트워크를 기반으로 하기 때문에 스트리밍 서비스를 이용하여 동작하는 응용 프로그램은 네트워크의 영향을 받는다. 따라서 사용자가 사용하려고 하는 기능이 네트워크 고장으로 인해 스트리밍 할 수 없다면, 응용 프로그램은 종료되거나 시스템의 고장이 발생할 수 있다.

본 논문에서는 소프트웨어 스트리밍 서비스의 안정성과 신뢰성을 높이기 위해 네트워크 고장감내 소프트웨어 스트리밍 기술인 에버그린을 제안했다. 본 논문에서 설명한 에버그린 기술은 현재 개발 중인 시스템에 적용되고 있으며, 앞 절에서 설명한 것처럼 스트리밍 패키지를 분석하여 기능별 스트리밍 정보를 추출하는 스트리

밍 패키지 분석도구를 개발하여 시험하고 있다.

에버그린 기술은 시뮬레이션을 통해 응용 프로그램에서 제공하는 각 기능에 대한 실행 코드 정보를 수집하고, 이를 통해 기능 단위의 스트리밍을 제공한다. 또한, 네트워크의 상태를 감시하여 네트워크의 고장으로 인한 응용 프로그램과 시스템의 고장이 발생하지 않도록 한다. 따라서 사용자는 네트워크 에러로 인해 발생할 수 있는 응용 프로그램 또는 시스템 고장에 대한 우려 없이 응용 프로그램을 사용할 수 있다. 따라서 에버그린 기술은 항상 네트워크 단절이라는 문제를 갖고 서비스되는 소프트웨어 스트리밍에 대해서 더 안전하고, 지능적인 고장감내 서비스를 제공할 것이다.

참 고 문 헌

[1] http://service.real.com/help/library/index.html

[2] http://www.apple.com/quicktime

[3] http://www.microsoft.com/windows/windowsmedia/default.mspx

[4] http://appstream.com

[5] http://www.softonnet.com/korean/main.php

[6] http://www.softricity.com/home/index.asp

[7] http://www.streamtheory.com

[8] J. E, White, C. S. Helgeson, and D. A. Steedman, "System and method for distributed computation based upon the movement, execution, and interaction of processes in a network," US Patent 5603031, Feb. 1997.

[9] U. Raz, Y. Volk, and S. Melamed, "Streaming Modules," US Patent 6311221, Oct. 2001.

[10] D. Eylon, A. Ramon, Y. Volk, U. Raz, and S. Melamed, "Method and system for executing network streamed application," US Patent 6574618, Jun. 2003.

[11] http://www.etri.re.kr

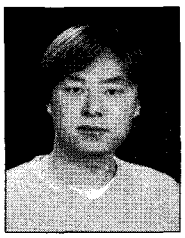
[12] http://en.wikipedia.org/wiki/80-20_rule

[13] P. Kuacharoen, V. J. Mooney, and V. K. Madisetti, "Software Streaming via Block Streaming", DATE'03 Conference, pp.912-917, 2003.

[14] http://www.abiword.org

저 자 소 개

심 정 민(Jeong-Min Shim) 정회원



- 2004년 2월 : 충북대학교 정보통신공학과(공학석사)
- 2004년 3월~현재 : 한국전자통신연구원 연구원

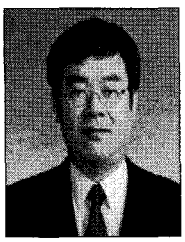
<관심분야> : 시스템 소프트웨어, 소프트웨어 공학, 소프트웨어 스트리밍

김 원 영(Won-Young Kim) 정회원

- 1998년 : KAIST(공학박사)
- 1999년~현재 : 한국전자통신연구원 연구원

<관심분야> : 데이터 마이닝, 데이터베이스 시스템, 메인-메모리 시스템

최 완(Wan Choi) 정회원



- 1983년 : KAIST(공학석사)
- 1985년~현재 : 한국전자통신연구원 연구원

<관심분야> : 소프트웨어 공학, 미들웨어, 소프트웨어 스트리밍