유비쿼터스 환경에서 응용 독립적 DIA를 위한 최적 트랜스코딩 경로의 CFG 기반 자동 탐색 방법

전 성 미[†] · 임 영 환[††]

요 약

유비쿼터스 장비를 통해 서버에 있는 디지털 아이템에 접근하기 위해서, 디지털 아이템은 시스템 환경과 장비 특성 및 사용자 선호에 따라 적응되어야한다. 유비쿼터스 환경에서 장비 의존적 적응의 요구 사항은 정적으로 결정되지 않으며 예측할 수 없다. 그러므로 특정한 응용에 대한 적용 절차는 일반적 디지털 아이템 적응 엔진에서 적용될 수 없다. 본 논문에서는 최소의 트랜스코더의 집합과 요구된 적응 요구를 위한 트랜스코딩 경로 생성기, 적응 스케쥴러를 갖는 응용-독립적 디지털 아이템 적응 구조를 제안한다. 또한 트랜스코딩 경로인 여러 단위 트랜스코더의 연결을 문맥 자유 문법을 사용하여 찾는 방법을 설명하고, 실험을 하였다.

키워드 : 디지털 아이템 적응, 유비쿼터스, 트랜스코더 구조

A CFG Based Automated Search Method of an Optimal Transcoding Path for Application Independent Digital Item Adaptation in Ubiquitous Environment

Chon Sungmi[†] · Lim Younghwan[††]

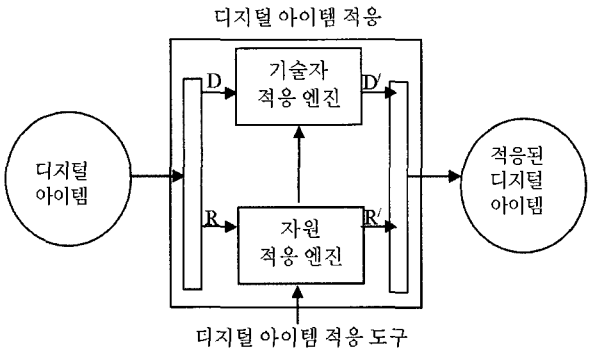
ABSTRACT

In order to access digital items in a server via ubiquitous devices, the digital items should be adapted according to the system environment, device characteristics and user preferences. In ubiquitous environment, those device-dependent adaptation requirements are not statically determined and not predictable. Therefore an application specific adaptation mechanism can not be applied to a general digital item adaptation engine. In this paper, we propose an application independent digital item adaptation architecture which has a set of minimal transcoders, transcoding path generator for a required adaptation requirement, and adaptation scheduler. And a CFG based method of finding a sequence of multiple unit transcoders called a transcoding path is described in detail followed by experimental results.

Key Words : Digital Item Adaptation, Ubiquitous, Transcoder Architecture

1. 서 론

디지털 아이템 적응(Digital Item Adaptation, DIA)은 MPEG21의 주요 부분 중 하나이다. DIA의 목적은 상호 운영적이며 투명한 멀티미디어 컨텐츠의 접근을 달성하는 것이다. 이것은 사용자에게 네트워크와 단말기의 설치, 관리와 구현 쟁점을 숨김으로써 가능하다. (그림 1)에서와 같이, 기술자 적응과 자원 적응의 조합은 새롭게 적응된 디지털 아이템을 생성한다. 디지털 아이템 적응 도구는 기술자 정보



(그림 1) 디지털 아이템 적응

에서 필요한 모든 자료를 저장한다[1]. 디지털 아이템 적응 프레임워크에서 다루어질 필요가 있는 디지털 아이템은 다

※ 이 논문은 2004년도 학술진흥재단의 지원에 의하여 연구되었음.(KRF-2004-005-D00198)
† 준 회 원 : 숭실대학교 정보미디어 기술연구소 연구원
†† 종신회원 : 숭실대학교 미디어학부 부교수
논문접수 : 2005년 1월 5일, 심사완료 : 2005년 5월 2일

양할 수 있다. 디지털 아이템의 성격에 따라, 자원 적응 엔진(Resource Adaptation Engine, RAE)의 구조나 구현 방법은 다를 수 있다. 자원 적응 엔진을 위한 현재의 연구는 하나의 미디움 트랜스코더에 초점이 맞추어져 있다. 이에 대한 예로는 형식 트랜스코더[2], 크기 트랜스코더[3], 프레임 율 트랜스코더[4], MPEG2에서 MPEG4로 변환하는 트랜스코더[5], 3D스테레오 스코프 비디오에서 2D 비디오로 변환하는 트랜스코더[6]등이 있다. 따라서 트랜스코딩이 필요한 각 경우에 대해 고정된 하나의 트랜스코더를 제공한다.

또한 특별히 MPEG 시리즈에 대한 디지털 아이템의 적응에 대한 많은 연구가 진행되고 있다. 즉, 이동 특성을 위한 메타 데이터[7], 기술자 도구[8], 메타 데이터 드리븐 적응[9] 등의 연구가 있다. 이것들은 트랜스코딩이 필요한 각 개별 경우를 위한 고정된 하나의 단위 트랜스코더를 대상으로 한 것이다. 이러한 방법에서는 근원지의 서비스 품질과 목적지의 서비스 품질이 변경될 경우, 이전 적응에 사용되었던 것과 동일한 트랜스코더를 다시 사용할 수 없다.

그리고 기존 연구중 하나에서는 프레임 율, 색상, 해상도 순으로 트랜스코더의 연결 순서를 제안하였다[10]. 이것은 모바일 환경의 멀티미디어 컨텐츠에 대해 네트워크 대역폭과 모바일 호스트로부터 필요한 대역폭을 고려한 것이다.

또 다른 연구에서는 비트율에 기반하여 프레임 율과 해상도 트랜스코딩을 동시에 하는 방법을 제안하였다[11]. 이것은 실시간 멀티미디어 서비스에서 네트워크 환경과 서버/클라이언트의 능력을 고려한 것이다. 그러나 두 방법 모두 실시간에 발생하는 디지털 아이템에 대한 사용자 선호를 고려하지 않았다. 또한 트랜스코더들간의 연결 순서를 결정하는 근거에 대한 명확한 설명이 없다.

이러한 연구들의 결정적 문제점은 주어진 트랜스코더가 모든 적응 요구 사항을 만족할 수 없다는 것이다. 즉, 근원지와 목적지의 적응 요구 사항이 변경될 때마다 종단간 서비스 품질이 변경되므로, 이전에 사용하였던 것과 동일한 트랜스코더는 사용할 수 없다. 예를 들어 MPEG21 DIA에서 3D/2D 비디오 변환은 오직 3D/2D 비디오 변환에만 사용할 수 있다. 또 다른 문제점은 한정된 트랜스코더들의 집합으로는 모든 트랜스코딩 요구 사항을 만족할 수 없다. 그 이유는 유비쿼터스 환경의 단말에서는 적응 요구 사항이 항상 변화하여, 서버와 유비쿼터스 기기간 연결이 이루어질때까지 필요한 트랜스코더를 예측할 수 없기 때문이다.

이러한 문제를 해결하기 위해서 응용에 독립적인 디지털 아이템 적응 구조(architecture)를 제안한다. 이 구조는 트랜스코딩 관리자와 일반적 자원 적응 엔진의 구현을 위한 트랜스코더들의 집합으로 구성되어 있다.

제안하는 응용 독립적 자원 적응 엔진은 하나의 트랜스코딩 기능만으로 제한된 단위 트랜스코더들의 집합을 갖는다. 또한 자원 적응 엔진은 근원지 컨텐츠인 디지털 아이템 입력 자원과 단말기에서의 적응 요구 사항을 분석하여 필요한 단위 트랜스코더의 적절한 순서를 찾는 메커니즘을 갖는다.

이렇게 트랜스코더들의 집합과 적응 요구 사항을 분리함으로써 트랜스코딩의 완전한 기능성과 구현의 유연성을 얻을 수 있다. 제안하는 자원 적응 엔진에서 트랜스코딩 운영은 트랜스코딩 경로 생성과 트랜스코딩 경로 응용의 두 단계로 구성되어 있다.

대부분의 멀티미디어 디지털 아이템은 텍스트, 이미지, 오디오, 비디오등과 같은 여러 스트림으로 구성되어 있으므로, 여러 트랜스코더들을 연결하여 근원지의 디지털 아이템이 목적지의 서비스 품질을 만족하는 디지털 아이템으로 트랜스코딩 되어 적용되도록 해야 한다.

주어진 디지털 아이템 적응 도구에서 트랜스코딩 경로를 생성하는 것의 문제점은 단말기나 네트워크 환경에 따라 다른 트랜스코딩 경로들을 다시 생성해야 한다는 것이다. 즉, 한정된 트랜스코딩 경로만으로는 디지털 아이템을 적응할 수 없다는 것을 의미한다. 대신 단말기와 네트워크 대역폭이 결정될 때마다, 적절한 트랜스코딩 경로가 동적으로 재생성되어야 한다.

본 논문에서는 주어진 디지털 아이템과 적응 요구 사항에 대하여 최소 지연 시간을 갖는 최적 트랜스코딩 경로를 생성하는 방법을 제안한다.

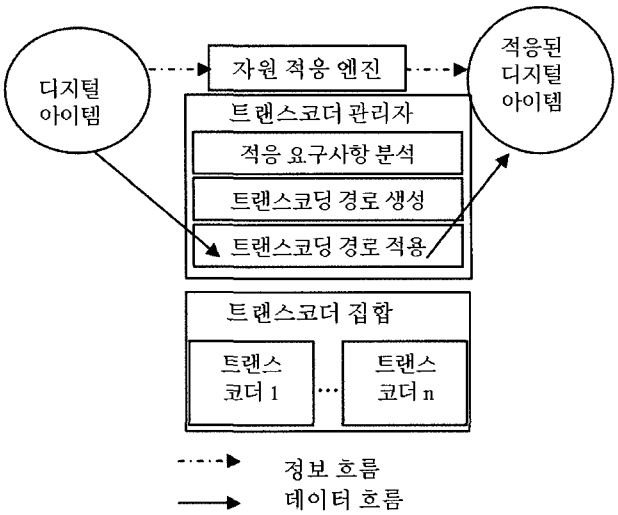
모든 가능한 트랜스코딩 경로를 생성하기 위하여 전통적인 브루트 폴스(brute force) 탐색 방법을 사용할 수 있다. 그런데 이 방법은 순환때문에 종료되지 않을 수 있으므로 트랜스코딩 경로 생성이 끝나지 않는 결과를 가져온다. 또한 이 방법은 모든 가능한 트랜스코딩 경로를 찾는 처리 시간이 많이 필요하다. 따라서 적절한 트랜스코딩 경로를 생성하는 과정에서 저장된 멀티미디어 데이터의 정보, 서버와 클라이언트 양쪽의 트랜스코더 집합, 단말기의 특성과 네트워크의 제한에 대한 지식을 통합하도록 제안한다.

그러한 지식을 통합하는 수단으로, 문맥 자유 문법(Context Free Grammar, CFG) 이론에 기반한 탐색 알고리즘을 제안한다. 문맥 자유 문법을 사용할 때 해결해야 할 핵심 문제는 주어진 상황에서 적절한 생성 규칙을 찾는 방법이다. 따라서 문맥 자유 문법 생성 규칙을 자동으로 생성하는 알고리즘과 최적인 하나의 트랜스코딩 경로를 찾는 알고리즘을 제안한다. 또한 최종적으로 실험에 대한 결과를 기술한다.

2. 응용 독립적 디지털 아이템 구조

유비쿼터스 환경에서는 유비쿼터스 기기의 특성, 네트워크 능력 및 사용자 선호가 언제나 변화한다. 그러므로 모든 적응 요구사항을 만족하는 응용 독립적 트랜스코더를 갖는 디지털 아이템 적응은 없다.

제안하는 자원 적응 엔진은 단위 트랜스코더라고 불리우



(그림 2) 제안하는 응용 독립적 자원 적응 엔진 구조

는 최소의 트랜스코딩 기능을 갖는다. 이때, 주어진 적응 요구 사항에 대하여, 단위 트랜스코더들의 순서화된 연결을 찾는 메커니즘을 개발할 수 있다. 이것으로 유비쿼터스 단말기가 요구하는 대로 디지털 아이템을 적응시킬 수 있다. 트랜스코더들의 집합과 적응 요구 사항을 분리하여 트랜스코딩의 완전한 기능성과 구현의 유연성을 얻을 수 있다.

제안하는 자원 적응 엔진의 주요 두 부분은 (그림 2)에서와 같이 트랜스코더와 트랜스코더 관리자이다.

트랜스코더들은 단위 트랜스코더들의 집합으로써 트랜스코딩 능력은 하나의 특정 기능에 제한되어 있다. 단위 트랜스코더의 추가, 삭제, 성능 향상과 같은 트랜스코더의 조작은 디지털 아이템의 특성과 적응 요구 사항에 대해 독립적으로 발생할 수 있다. 트랜스코더에 대해서 단위 트랜스코더를 위한 기능에 대한 기술, 시간에 대한 성능, 초당 비트인 처리율등과 같은 정보는 알 수 있다고 가정한다. 트랜스코더의 정보는 가능한 경로들 중 효율적인 트랜스코딩 경로를 선택하는 절차에서 효율성을 계산하는데 사용할 수 있다.

자원 적응 엔진의 트랜스코더 관리자는 근원지의 디지털 아이템과 유비쿼터스 단말기를 분석하여 서비스 품질에 대한 적응이 가능하도록 한다. 또한 서비스 품질에 적응하도록 필요한 단위 트랜스코더들의 적절한 순서를 찾으며, 최종적으로 근원지 디지털 아이템을 파이프라인 형태로 트랜스코더들의 연결을 적용하여 적응된 디지털 아이템을 생성한다. 트랜스코더 관리자는 적응 요구 사항에 대한 분석, 트랜스코딩 경로의 생성, 디지털 아이템에 대한 트랜스코딩 경로 적용의 세 가지 기능을 수행한다.

서비스 품질 전이도 기반 트랜스코딩 경로 생성 알고리즘이라고 불리우는 트랜스코더간 연결에 대한 기존 연구가 있다[12]. 이 연구에서는 브루트-폴스(brute-force) 방법을 사용하였다. 즉, 실시간에 컨텐츠의 파일 형식으로 수용 가능한 입력 형식의 모든 트랜스코더들의 연결을 시도하는 방법이다. 이때 이전에 연결된 트랜스코더들이 다시 연결되는 순환이 발생한다. 따라서 순환을 검사하고 제거하기 위한 절차가 필요하다. 이 방법은 많은 노드를 생성하며 트랜스코더들간 순서를 생성하는데 드는 시간도 길다. 이에 대해 제안하려는 방법은 주어진 적응 요구사항으로 최적 트랜스코딩 경로를 찾기 위한 절차에서, 저장된 디지털 아이템, 서버와 클라이언트에서의 트랜스코더들의 집합, 단말기의 특성, 네트워크 제한에 대한 정보를 통합시키는 것이다. 지식을 통합하는 방법으로 문맥 자유 문법(Context Free Grammar, CFG)의 이론에 기반한 탐색 알고리즘을 제안한다. 트랜스코딩 경로가 생성될 때, 문맥 자유 문법의 생성 규칙에 탐색 규칙과 상황적 지식이 포함된다. 이것은 적은 수의 노드가 생성되는 결과를 가져온다.

탐색 알고리즘에서 문맥 자유 문법을 사용하려고 할 때 해결해야 할 주요 문제는 주어진 상황에서 문맥 자유 문법의 적절한 생성 규칙을 찾는 방법이다. 본 논문에서는 문맥 자유 문법의 생성 규칙을 자동으로 생성하는 알고리즘을 제안한다. 즉, 단위 트랜스코더들의 집합에 종속적인 생성 규칙은 트랜스코딩 경로 탐색 절차를 시작하기 전에 생성될 수 있다는 점을 활용한 것이다. 이것은 트랜스코더들과 연관하여 잘 정의되고 최적화된 생성 규칙을 생성하는데 비실시간 모드에서 충분한 시간을 활용할 수 있다는 것을 의미한다. 실시간에 해야 할 작업은 단말기와 네트워크 환경에 종속적인 시작 생성 규칙의 집합을 생성하는 것 뿐이다. 그러므로 네트워크 환경과 단말기가 변할 때마다 다시 생성해야 할 규칙의 수를 제한하여 실시간에 트랜스코딩 경로를 생성하는 알고리즘을 제안할 수 있다.

3. 문맥 자유 문법 기반 서비스 품질 적응을 위한 트랜스코딩 경로 자동 탐색 방법

트랜스코딩 경로를 생성하기 위한 문법 $G_{\text{트랜스코딩 경로}}$는 다음과 같이 정의 한다.

$$G_{\text{트랜스코딩 경로}}=(S, V_N, V_T, P)$$

이때, S는 시작 기호, V_T 는 서버와 단말기에서 사용 가능한 트랜스코더를 표현하는 터미널 기호의 집합, V_N 은 트랜스코딩 경로 생성시 중간 기호를 표현하는 논 터미널 기호의 집합, 그리고 P는 생성 규칙의 집합이다.

트랜스코더는 다음과 같이 표기한다.

$$\text{입력 형식}\ \text{트랜스코더 종류}\ \text{출력 형식}$$

이때 트랜스코더 종류는 트랜스코더의 기능을 표현한다. 즉, 형식 트랜스코더는 FT로, 네트워크는 net로, 프레임 율

트랜스코더는 FrT로, 크기 트랜스코더는 ST로, 디코더는 DC로, 엔코더는 EC로 표시한다. 또한 입출력 형식은 트랜스코더의 파일 형식을 의미한다.

문맥 자유 문법을 사용하여 트랜스코딩 경로를 찾는 방법은 문맥 자유 문법을 생성하는 것에 비해 상대적으로 단순하다. 즉, 시작 기호에서 시작하여, 컴파일러 이론[13]에서의 좌단 유도를 적용하여 단말 기호로된 스트링을 생성한다. 유도된 스트링들의 집합이 찾고자 하는 트랜스코딩 경로이다.

(알고리즘 1) 트랜스코딩 경로를 위한 문맥 자유 문법 기반 탐색 방법

입력 문맥 자유 문법 $G_{\text{트랜스코딩 경로}}$, 디지털 아이템의 근원지 서비스 품질, 목적지 서비스 품질

출력 트랜스코딩 경로

단계 문맥 자유 문법 기반 트랜스코딩 경로 유도 단계

S에서 시작하여 좌단 유도를 계속하면, 결국 모든 논터미널이 터미널이 된다. 이것이 트랜스코딩 경로 중 하나이다.

예를 들어 <표 1>과 같은 디지털 아이템 적응 도구를 가정하고, 부록[1]과 같은 $G_{\text{트랜스코딩 경로}} = (S, V_{N\ src} \cup V_{N\ dest}, V_{T\ src} \cup V_{T\ dest}, P_{src} \cup P_{dest})$가 있다고 가정할 때, S로부터 시작하는 좌단 유도의 단계는 다음과 같다.

〈표 1〉 종단간 디지털 아이템의 서비스 품질의 예

근원지 서비스 품질	목적지 서비스 품질
30fps, QCIF, 24 비트 컬러, MPEG-1	5fps, CIF, 24 비트 컬러, MPEG-4

단계 1) S에서 시작하여 좌단 유도를 수행한다.

$S ::= <_{mpeg\text{-}1}FT_{src\ mpeg\text{-}4}>_{mpeg\text{-}4}net_{mpeg\text{-}4}<_{mpeg\text{-}4}FrT_{dest\ mpeg\text{-}4}><_{mpeg\text{-}4}ST_{dest\ mpeg\text{-}4}><_{mpeg\text{-}4}DC_{dest\ yuv}>$

$::= <_{mpeg\text{-}1}DC_{src\ yuv}><_{yuv}\ EC_{src\ mpeg\text{-}4}>_{mpeg\text{-}4}net_{mpeg\text{-}4}<_{mpeg\text{-}4}FrT_{dest\ mpeg\text{-}4}><_{mpeg\text{-}4}ST_{dest\ mpeg\text{-}4}><_{mpeg\text{-}4}DC_{dest\ yuv}>$

$::= _{mpeg\text{-}1}dc_{src\ yuv}<_{yuv}\ EC_{src\ mpeg\text{-}4}>_{mpeg\text{-}4}net_{mpeg\text{-}4}<_{mpeg\text{-}4}FrT_{dest\ mpeg\text{-}4}><_{mpeg\text{-}4}ST_{dest\ mpeg\text{-}4}><_{mpeg\text{-}4}DC_{dest\ yuv}>$

$::= _{mpeg\text{-}1}dc_{src\ yuv\ yuv}\ ec_{\ src\ mpeg\text{-}4\ mpeg\text{-}4}net_{mpeg\text{-}4}<_{mpeg\text{-}4}FrT_{dest\ mpeg\text{-}4}><_{mpeg\text{-}4}ST_{dest\ mpeg\text{-}4}><_{mpeg\text{-}4}\ DC_{dest\ yuv}>$

$::= _{mpeg\text{-}1}dc_{src\ yuv\ yuv}\ ec_{\ src\ mpeg\text{-}4\ mpeg\text{-}4}net_{mpeg\text{-}4\ mpeg\text{-}4}frt_{dest\ mpeg\text{-}4}<_{mpeg\text{-}4}ST_{dest\ mpeg\text{-}4}><_{mpeg\text{-}4}\ DC_{dest\ yuv}>$

$::= _{mpeg\text{-}1}dc_{src\ yuv\ yuv}\ ec_{\ src\ mpeg\text{-}4\ mpeg\text{-}4}net_{mpeg\text{-}4\ mpeg\text{-}4}frt_{dest\ mpeg\text{-}4\ mpeg\text{-}4}st_{dest\ mpeg\text{-}4}<_{mpeg\text{-}4}\ DC_{dest\ yuv}>$

$::= _{mpeg\text{-}1}dc_{src\ yuv\ yuv}\ ec_{\ src\ mpeg\text{-}4\ mpeg\text{-}4}net_{mpeg\text{-}4\ mpeg\text{-}4}frt_{dest\ mpeg\text{-}4\ mpeg\text{-}4}st_{dest\ mpeg\text{-}4\ mpeg\text{-}4}\ dc_{dest\ yuv}$

이 결과가 생성된 트랜스코딩 경로중의 하나이다.

단계 2) 두번째 시작 규칙인 S로부터 유도한다.

$S ::= <_{mpeg\text{-}1}FT_{src\ mpeg\text{-}4}><_{mpeg\text{-}4}FrT_{src\ mpeg\text{-}4}>_{mpeg\text{-}4}net_{mpeg\text{-}4}<_{mpeg\text{-}4}ST_{dest\ mpeg\text{-}4}><_{mpeg\text{-}4}DC_{dest\ yuv}>$

유사한 방법으로, 모든 논터미널은 터미널이 되어 $_{mpeg\text{-}1}dc_{src\ yuv\ yuv}\ ec_{\ src\ mpeg\text{-}4\ mpeg\text{-}4}frt_{src\ mpeg\text{-}4\ mpeg\text{-}4}net_{mpeg\text{-}4\ mpeg\text{-}4}st_{dest\ mpeg\text{-}4\ mpeg\text{-}4}\ dc_{dest\ yuv}$가 된다.

이 결과도 생성된 트랜스코딩 경로 중의 하나이다.

이러한 절차를 통해 얻어진 총 트랜스코딩 경로 수는 다음과 같이 두가지이다.

(트랜스코딩 경로 1)

$_{mpeg\text{-}1}dc_{src\ yuv\ yuv}ec_{src\ mpeg\text{-}4\ mpeg\text{-}4}net_{mpeg\text{-}4\ mpeg\text{-}4}frt_{dest\ mpeg\text{-}4\ mpeg\text{-}4}st_{dest\ mpeg\text{-}4mpeg\text{-}4}dc_{\ dest\ yuv}$

(트랜스코딩 경로 2)

$_{mpeg\text{-}1}dc_{src\ yuv\ yuv}ec_{src\ mpeg\text{-}4\ mpeg\text{-}4}\ frt_{src\ mpeg\text{-}4\ mpeg\text{-}4}net_{mpeg\text{-}4\ mpeg\text{-}4}st_{dest\ mpeg\text{-}4\ mpeg\text{-}4}dc_{dest\ yuv}$

그 후 최적 트랜스코딩 경로를 결정하기 위해 각 트랜스코딩 경로의 종단간 지연을 계산하여, 최소 지연의 최적 경로를 결정할 수 있다[12].

유도 단계가 용이한 반면, 문맥 자유 문법을 생성하는 것에 주된 노력을 해야 한다. 앞으로 그 방법에 대하여 기술할 것이다.

4. 문맥 자유 문법 중 생성 규칙의 자동 생성 방법

트랜스코딩 경로를 탐색하기 위해 사용할 문맥 자유 문법을 $G_{\text{트랜스코딩 경로}} = (S, V_N, V_T, P)$라고 가정할 때, 핵심 문제는 생성 규칙 P의 집합을 자동적으로 생성하는 방법이다.

생성 규칙을 생성하려고 할 때 생성 규칙 생성에 영향을 주는 두 가지 요소가 발견되었다. 하나는 서버와 단말기에 있는 트랜스코더들의 집합이고, 또 하나는 종단간 디지털 아이템의 서비스 품질 정보와 네트워크에 대한 정보를 가지고 있는 디지털 아이템 적응 도구이다. 전자는 시간 종속적 측면에서 상대적으로 정적이고, 후자는 동적이다. 즉, 서버와 단말기에 있는 트랜스코더들의 집합은 상대적으로 오랜 기간동안 고정되어 있으나, 연출에 대한 단말기와 네트워크, 사용자 선호는 가변적이다.

정적 요소와 관련있는 생성 규칙은 비실시간에 생성한다. 이것은 트랜스코더와 관련된 생성 규칙을 생성하고 최적화하는데 사용할 시간이 충분함을 의미한다.

실시간에 해야 할 일은 동적 요소에 관련된 생성 규칙만 생성하면 된다. 이러한 요소는 시작 생성 규칙에 통합된다.

(알고리즘 2) 문맥 자유 문법 $G_{\text{트랜스코딩 경로}}$ = (S, V_N, V_T, P)의 자동 생성

입력 디지털 아이템 적응 도구, 단말기의 수용력, 트랜스코더들의 집합

출력 문맥 자유 문법 $G_{\text{트랜스코딩 경로}}$= (S, V_N, V_T, P) 단, P = $P_{\text{정적}} \cup P_{\text{시작}}$

단계 1) 비실시간 모드에서
터미널 기호 V_T, 논터미널 기호 V_N,
트랜스코더 관련 생성 규칙 $P_{\text{정적}}$ 생성
2) 실시간 모드에서
시작 생성 규칙 $P_{\text{시작}}$ 생성

4.1 비실시간 모드에서 정적 규칙의 생성

트랜스코더가 시스템에 등록될 때 터미널 기호, 논터미널 기호 그리고 정적 생성 규칙이 자동적으로 생성될 수 있다.

시스템에 트랜스코더들의 집합 TR={$_{\text{mpeg-2}}ft_{\text{mpeg-4}}$, $_{\text{mpeg-2}}dc_{\text{yuv}}$, $_{\text{yuv}}ec_{\text{mpeg-4}}$}과 같은 세 가지 트랜스코더가 있다고 가정하자. 터미널 기호의 집합은 TR과 동일한 것을 알 수 있다. 그러나 논터미널 기호를 생성하는 것은 간단하지 않다. TR에서 MPEG-1 데이터를 MPEG-4 데이터로 트랜스코딩하는 한 가지 방법은 $_{\text{mpeg-1}}ft_{\text{mpeg-4}}$ 트랜스코더를 사용하는 것이다. 다른 방법은 두 개의 트랜스코더의 조합인 $_{\text{mpeg-1}}dc_{\text{yuv}}$ 와 $_{\text{yuv}}ec_{\text{mpeg-4}}$를 사용하는 것이다. 그러므로 동일한 트랜스코딩 절차의 방법은 하나 이상이 존재할 수 있다. 디지털 아이템 형식을 MPEG-1에서 MPEG-4로 변환하는 논터미널<$_{\text{mpeg-1}}FT_{\text{mpeg-4}}$>는 $_{\text{mpeg-1}}ft_{\text{mpeg-4}}$ 나 $_{\text{mpeg-1}}dc_{\text{yuv}}$ $_{\text{yuv}}ec_{\text{mpeg-4}}$으로 그 기능을 할 수 있다.

따라서 트랜스코딩 경로 생성을 위한 논터미널의 생성 규칙은 다음과 같이 표현할 수 있다.

$$<{}_{\text{mpeg-1}}FT_{\text{mpeg-4}}> ::= {}_{\text{mpeg-1}}ft_{\text{mpeg-4}} \mid {}_{\text{mpeg-2}}dc_{\text{yuv}}\ {}_{\text{yuv}}ec_{\text{mpeg-4}}$$

논터미널과 문맥 자유 문법의 시작 생성 규칙을 생성하는 방법은 다음과 같다.

논터미널 기호를 생성하는 방법은 트랜스코더 집합내의 각 트랜스코더들에 대해 동일 트랜스코딩 절차를 수행하는 또다른 방법이 있는지 확인하는 것이다.

예를 들어 트랜스코더 $_{\text{mpeg-1}}ft_{\text{mpeg-4}}$를 고려하고, 트랜스코더가 갖을 수 있는 모든 데이터 형식이 {mpeg-1, yuv, mpeg-4}의 하나라고 가정한다.

이때 논터미널의 후보 집합으로 {<$_{\text{mpeg-1}}FT_{\text{mpeg-1}}$>, <$_{\text{mpeg-1}}FT_{\text{yuv}}$>, <$_{\text{mpeg-1}}FT_{\text{mpeg-4}}$>, <$_{\text{yuv}}FT_{\text{mpeg-1}}$>, <$_{\text{yuv}}FT_{\text{yuv}}$>, <$_{\text{yuv}}FT_{\text{mpeg-4}}$>, <$_{\text{mpeg-4}}FT_{\text{mpeg-1}}$>, <$_{\text{mpeg-4}}FT_{\text{yuv}}$>, <$_{\text{mpeg-4}}FT_{\text{mpeg-4}}$>} 를 형성할 수 있다. 이것은 데이터 형식 {mpeg-1, yuv, mpeg-4}의 모든 가능한 조합이다. 그 중 의미가 없는 <$_{\text{mpeg-1}}FT_{\text{mpeg-1}}$> 형태의 것은 후보에서 제거한다. 각 후보에 대해 트랜스코더 집합을 사용하여 트랜스코딩 경로가 될 수 있는지 검사한다.

비 실시간에 위와 같은 방법을 적용할 때, 논터미널 후보 <$_{\text{mpeg-1}}FT_{\text{mpeg-4}}$>는 트랜스코딩 경로 $_{\text{mpeg-1}}ft_{\text{mpeg-4}}$와 $_{\text{mpeg-1}}dc_{\text{yuv}}$ $_{\text{yuv}}ec_{\text{mpeg-4}}$를 갖는 것을 알 수 있다.

그러나 다른 논터미널 후보들은 트랜스코딩 경로를 갖지 못한다.

따라서 다음과 같은 생성 규칙을 얻을 수 있다.

$$<{}_{\text{mpeg-1}}FT_{\text{mpeg-4}}> ::= {}_{\text{mpeg-1}}ft_{\text{mpeg-4}} \mid {}_{\text{mpeg-1}}dc_{\text{yuv}}\ {}_{\text{yuv}}ec_{\text{mpeg-4}}$$

이에 대한 형식화된 알고리즘은 다음과 부록[2]에 있다.

(알고리즘 3) 논터미널과 생성 규칙의 생성 방법

입력 트랜스코더들의 집합

출력 문맥 자유 문법의 터미널, 논터미널과 생성 규칙

단계 부록[2] 참조

[정리 1] 위 (알고리즘 3)의 시간 복잡도는 다음과 같다.
$T(N)=O(N_{TR}*|Domain(format)|^2* depth * N_{TR}^{depth})$
단 N_{TR}은 트랜스코더들의 총 개수이며, depth는 트랜스코딩 경로 트리의 깊이이다. 또한 format은 디지털 아이템이 갖을 수 있는 형식이며, Domain은 format의 집합이다.

[증명]
서비스 품질 전이도 기반 트랜스코딩 경로 생성 알고리즘의 시간 복잡도는 $T(N_{TR}) = O(depth * N_{TR}^{depth})$[12]이다. 그런데 (알고리즘 3)에서는 서비스 품질 전이도 기반 트랜스코딩 경로 생성 알고리즘을 N_{TR} x $|Domain(format)|^2$의 수 만큼 수행한다. 따라서 (알고리즘 3)의 전체 시간 복잡도는 $T(N)=O(N_{TR}*|Domain(format)|^2* depth * N_{TR}^{depth})$ 이다.

[정리 2] 문맥 자유 문법 기반 트랜스코딩 경로 생성 알고리즘의 모든 해는 서비스 품질 전이도 기반 트랜스코딩 경로 생성 알고리즘의 모든 해와 동일하다[12].

[정리 3] V_T, V_N, $P_{\text{정적}}$은 오직 트랜스코더 집합에 대해서 종속적이며, 디지털 아이템 적응 도구에 대해서는 독립적이다.

[정리 1]에서 보는 바와 같이 논터미널과 정적 생성 규칙을 생성하는데는 많은 시간이 필요하다. 그러나 이들의 생성은 [정리 3]에 의하면 비 실시간에 오직 한번만 생성하면 된다. 즉, 시스템에 트랜스코더들의 집합을 등록하는 비실시간에 논터미널과 그에 대한 정적 생성 규칙을 생성할 수 있다. 실시간에는 시작 생성 규칙만 생성하면 된다. 따라서 정적 규칙을 생성하는 시간을 고려하지 않아도 된다.

4.2 실시간에서 시작 생성 규칙의 생성 방법

시작 생성 규칙은 트랜스코딩 경로의 검색 공간상 최상위 레벨에서 전역적 안내를 하는 역할을 맡는다. 그러므로, 먼저 실시간에 근원지와 목적지간 서비스 품질 요소 중 서로 다른 것이 있는지 검사한다. 예를 들어, 디지털 아이템의 근원지 서비스 품질이 30fps, 24비트 컬러, QCIF, MPEG-1이고, 목적지 서비스 품질이 5fps, 24비트 컬러, CIF, MPEG-4라고 가정하자.

주어진 디지털 아이템 적응 도구에서, 프레임 률 트랜스코더, 크기 트랜스코더와 형식 트랜스코더가 필요함을 알 수 있다. 그러면 다음과 같은 여섯 개의 완전한 트랜스코딩 절차를 얻을 수 있다.

O1 : FrT-ST-FT, O2 : FrT-FT-ST, O3 : ST-FrT-FT
O4 : ST-FT-FrT, O5 : FT-FrT-ST, O6 : FT-ST-FrT

추가하여, 네트워크 대역폭에 대한 제약 사항인 네트워크 관련 서비스 품질이 고려되어야 한다. 이는 $QoS_{네트-관련}$ 으로 표기한다.

예를 들어 네트워크 대역폭이 64Kbps일 때, 5fps, 24비트 컬러, QCIF, MPEG-4인 디지털 아이템은 네트워크를 통과할 수 있지만, 서비스 품질이 30fps, 24비트 컬러, QCIF, MPEG-1인 경우는 통과할 수 없다.

그러므로 다음과 같은 아홉 개의 중간 서비스 품질 중에 어느 것이 네트워크 대역폭을 통과할 수 있는지 결정하여야 한다.

qos(30fps, 24비트 컬러, QCIF, MPEG-1)
qos(5fps, 24 비트 컬러, QCIF, MPEG-1)
qos(30fps, 24 비트 컬러, CIF, MPEG-4)
qos(30fps, 24 비트 컬러, QCIF, MPEG-4)
qos(5fps, 24 비트 컬러, CIF, MPEG-4)
qos(5fps, 24 비트 컬러, QCIF, MPEG-4)
qos(5fps, 24 비트 컬러, CIF, MPEG-4)
qos(5fps, 24 비트 컬러, CIF, MPEG-4)

네트워크 서비스 품질에 따라, 디지털 아이템이 트랜스코딩될 곳이 서버인지 클라이언트인지 결정된다. 그러므로 디지털 아이템의 근원지 서비스 품질과 네트워크 관련 서비스 품질간, 네트워크 관련 서비스 품질과 디지털 아이템의 목적지 서비스 품질간 필요한 트랜스코더들을 알 수 있다.

필요한 트랜스코더들은 순열 형태를 취하며, 선행/후행 트랜스코더간 출력/입력 형식이 동일한 것을 할당한다. 이에 대한 결과가 시작 생성 규칙이다.

위의 예로부터, 네트워크 관련 서비스 품질을 5fps, 24비트 컬러, QCIF, MPEG-4로 결정할 경우에는 근원지에서 필요한 트랜스코더는 프레임 율 트랜스코더와 형식 트랜스코더이며, 목적지에서는 크기 트랜스코더가 필요하다. 따라서 디지털 아이템이 네트워크를 지나가기 전에 서버에서는 FrT-FT 나 FT-FrT의 트랜스코더 순서가 적용되며, 크기 트랜스코더인 ST는 클라이언트에서 수행될 수 있다. 이러한 절차에 의해 다음과 같은 시작 생성 규칙이 기술 될 수 있다.

$$S ::= <_{mpeg\text{-}1}frt_{\ mpeg\text{-}1}><_{mpeg\text{-}1}ft_{\ mpeg\text{-}4}>net<_{mpeg\text{-}4}st_{\ mpeg\text{-}4}>$$
$$\mid <_{mpeg\text{-}1}ft_{\ mpeg\text{-}4}><_{mpeg\text{-}4}frt_{\ mpeg\text{-}4}>net<_{mpeg\text{-}4}st_{\ mpeg\text{-}4}>$$

이때 net는 네트워크를 표현하며, 디지털 아이템을 전송만 하는 터미널이다.

실시간에 시작 생성 규칙을 자동적으로 생성하는 방법은 다음과 같다.

(알고리즘 4) 시작 생성 규칙을 생성하는 방법
입력 디지털 아이템 적응의 도구
출력 문맥 자유 문법의 시작 생성 규칙
단계 부록[3] 참조

문맥 자유 문법을 사용하여 트랜스코딩 경로를 유도하기 전에, 컴파일러 이론에서의 사용되지 않는 기호나 S로부터 도달하지 못하는 기호는 제거되는 것과 같은 최적화 절차가 적용된다.

5. 실험 결과

실험의 목적은 다음과 같다.

첫째, 트랜스코딩 경로를 생성하는데 사용된 트랜스코더의 수에 따른 생성 노드 수를 검사한다. 둘째, 트랜스코딩 경로를 생성하는데 사용된 트랜스코더의 수에 따른 노드의 성공율을 검사한다.

실험에 사용된 디지털 아이템의 근원지 서비스 품질은 30fps, 24비트 컬러, QCIF, MPEG-1 파일이며, 네트워크를 통해 목적지에서는 5fps, 24 비트 컬러, CIF, MPEG-4로 디코딩되어야 한다.

이때 트랜스코딩 경로를 생성하는 네 번의 실험이 두 개의 알고리즘 - 기존의 서비스 품질 전이도 기반 방법과 제

안된 CFG 기반 방법 - 으로 부록[4]의 가용 트랜스코더 집합1, 2에 대해서 수행되었다.

<표 2>에 네 번의 실험에 따른 트랜스코딩 경로의 수가 있으며, 부록[5]에 생성된 트랜스코딩 경로가 있다.

〈표 2〉 실험 결과

트랜스코더 집합	서비스 품질 전이도 기반 알고리즘		제안한 문맥 자유 문법 기반 알고리즘	
	집합 1	집합 2	집합 1	집합 2
생성 노드 수	(실험 1) 50	(실험 2) 120	(실험 3) 14	(실험4) 32
성공 노드율	(16/50)*100 = 32.0%	(35/120)*100 = 29.17%	(14/14) *100 = 100%	(32/32) *100 = 100%
tp의 수	4	12	4	12

다음은 가용 트랜스코더 집합 1과 2에 대한 각 실험의 결과이다.

첫째, 생성된 노드의 수는 서비스 품질 전이도 기반 알고리즘에서 CFG 기반 알고리즘보다 네 배 가까이 생성되었다.

이것은 후자에서 디지털 아이템 적응 도구를 분석한 후에 시작 생성 규칙을 통해 입출력 형식이 결정되어 필요한 트랜스코더들만 배치되기 때문이다. 반면 전자는 불필요한 트랜스코더의 연결을 막기 위해 선행 트랜스코더와 동일한 출력 형식을 갖는 입력 형식의 가용 자원을 만날 때마다 연결을 생성한 후에 순환의 생성을 검사하기 때문이다.

둘째, 생성된 노드의 성공율은 문맥 자유 문법 기반 알고리즘에서 서비스 품질 전이도 기반 알고리즘보다 세배에서 세배 반 가까이 높았다.

이것은 두 알고리즘에서 생성된 노드의 수가 차이가 난 것과 동일한 이유에 기인한다.

셋째, 생성된 트랜스코딩 경로의 종류와 수는 문맥 자유 문법 기반과 서비스 품질 전이도 기반 알고리즘에서 동일하다. 이것은 두 알고리즘 모두 요구된 트랜스코더의 순열의 형태로 트랜스코더를 연결하기 때문이다. 또한 전자에서 사용된 논터미널은 후자에 의해 생성되기 때문이다.

넷째, 문맥 자유 문법 기반 알고리즘의 노드 성공율은 100%이다. 이것은 가용 자원 집합 1과 2에 대해 순환이 발생하지 않기 때문이며, 순환 경로가 발생하지 않는다면 실패 노드도 발생하지 않는다. 그 이유는 모든 가능한 입출력 데이터 형식을 갖는 논터미널이 먼저 생성되고, 반드시 필요한 가용 트랜스코더들이 시작 생성 규칙에서 논터미널과 터미널를 사용해 형성되기 때문이다.

6. 결 론

디지털 아이템의 종단간 서비스 품질이 다를 때, 근원지의 디지털 아이템은 단말기의 서비스 품질을 충족하는 디지털 아이템으로 트랜스코딩되어야 한다.

주어진 연출 환경에서 트랜스코딩 경로를 생성할 때 문제점은 단말기나 네트워크 환경에 따라 서로 다른 트랜스코딩 경로가 생성되어야한다는 점이다.

또한 실시간 디지털 아이템의 연출에서 모든 가능한 트랜스코딩 경로를 생성하는 것보다 수용가능한 시간 복잡도에서 하나의 최선의 트랜스코딩 경로를 탐색하는 것이 더 중요하다. 주어진 연출 정보와 트랜스코더들의 집합, 플랫폼 환경 정보를 가지고 문맥 자유 문법을 사용한 적은 수의 트랜스코딩 경로를 탐색하는 알고리즘을 제안하였다.

또한, 탐색된 각 트랜스코딩 경로 중에서 트랜스코딩 지연 시간을 비교하여 최선의 하나를 선택하는 방법과 문맥 자유 문법의 요소인 시작 생성 규칙, 생성 규칙, 논터미널과 터미널이 자동적으로 생성되는 방법을 제안하였다. 최종적으로 제안한 모든 방법을 스트림 엔진인 TransCore에 구현하였고, 서비스 품질 전이도 기반 트랜스코딩 경로 생성 알고리즘과 비교하였다.

실험 결과에서 제안한 알고리즘은 기존의 알고리즘보다 동일한 트랜스코딩 경로를 생성하는데 더 적은 노드를 생성하면서도 실제 경로에 사용한 노드 수는 더 많은 효율성을 보였다. 또한 제안 알고리즘은 기존 알고리즘과 달리 비실시간/실시간 작업을 구분하여 트랜스코딩 경로를 생성함으로써, 유비쿼터스 환경에서 다양한 단말이 요구하는 디지털 아이템의 적응 요구 사항을 실시간에 응용 독립적으로 지원할 수 있다.

제안 알고리즘은 유비쿼터스 환경하에서 디지털 방송용 서버의 디지털 아이템을 각종 장치(any device)에 적응하여 재생하려고 할 때 사용할 수 있다.

참 고 문 헌

[1] Harald K.: Distributed Multimedia Database Technologies Supported MPEG-7 and MPEG-21, CRC Press, (2004) 116.

[2] Seo, K., Heo, S., and Kim, J. : A Rate Control Algorithm Based on Adaptive R-Q Model for MPEG-1 to MPEG-4 Transcoding in DCT Domain, IEEE International Conference on Communications, (2002). Vol.1, 109-113.

[3] Zhijun, L., and Georganas, N., D. : H.263 Video Transcoding for Spatial Resolution Downscaling, Proceedings of the International Conference on Information Technology: Coding

and Computing, (2002). 425-430.

[4] Fung, K., T., and Siu, W., C. : Low Complexity and High Quality Frame-Skipping Transcoder, IEEE International Symposium on Circuits and Systems, (2001) .Vol.5, 29-32.

[5] 강의선 : 압축 상태에서 MPEG2 P 프레임을 H.263 P 프레임으로 변환하기 위한 Guided Search 방법 연구, 숭실대학교 대학원 컴퓨터학과 석사 학위 논문, (2001). 1-15.

[6] Kim, M., B., Nam, J., H., Baek, W.,H., Son, J.,W., and Hong J., W.: The Adaptation of 3D stereoscopic video in MPEG-21 DIA, Signal Processing Image Communication, (2003). 685-607.

[7] Zafer S., and Anthony V. : Mobility Characteristic for Multimedia Service Adaptation, Signal Processing:Image Communication, (2003) 699-719.

[8] Gabriel P., Anderas H., Jorg H., Hermann H., Harald Kosch, Christian T., Sylvain D., and Myriam A.: Bitstream Syntax Description: a tool for multimedia Resource Adaptation within MPEG-21, Signal Processing:Image Communication, (2003) 721-747.

[9] Laszlo B., Hermann H., Harald Kosch, Mulugeta L., and Stefan P. : Metadata Driven Adaptation in the ADMITS Project Signal Processing:Image Communication, (2003). 749-766.

[10] 이성진, 이화세, 박시용, 이승원, 정기동 : Mobile Multimedia Network에서 프록시 기반의 트랜스코딩을 이용한 대역폭 조절 기법, 한국정보과학회 학술대회, 제29권 2호, (2002) 293-304.

[11] 김제우, 김용환, 백종호, 최병호, 정혁구: 실시간 멀티미디어 서비스용 비디오 트랜스코딩 시스템 설계 및 구현 , 영상 처리 및 이해에 관한 워크샵, 제15권 제1호, (2003) 177-182.

[12] 전성미 , 임영환.: 근원지와 목적지에서 서로 다른 서비스 품질(QoS)을 필요로하는 멀티미디어 연출의 재생을 위한 서비스 품질 전이도 기반의 변환 경로 생성 알고리즘, 멀티미디어학회 논문지, 제6권 제2호, (2003) 208-215.

[13] 김강현, 박두순, 황종선 : 컴파일러 구성, 한국방송대학교 출판부, (1998), 96-112.

부 록

[1] G트랜스코딩 경로의 예

	근원지	목적지
S	$S::=<_{mpeg-1}FT_{src\ mpeg-4}>\ _{mpeg-4}net_{mpeg-4}$ $<_{mpeg-4}FrT_{dest\ mpeg-4}><_{mpeg-4}ST_{dest\ mpeg-4}><_{mpeg-4}DC_{dest\ yuv}>$ $\mid<_{mpeg-1}FT_{src\ mpeg-4}><_{mpeg-4}FrT_{src\ mpeg-4}>\ _{mpeg-4}net_{mpeg-4}$ $<_{mpeg-4}ST_{dest\ mpeg-4}><_{mpeg-4}DC_{dest\ yuv}>$	
V_T	$V_{T\ src}$ $=\{\ _{mpeg-1}frt_{src\ mpeg-1},\ _{mpeg-4}frt_{src\ mpeg-4},$ $_{mpeg-1}dc_{src\ yuv},\ _{yuv}ec_{src\ mpeg-4}\ \}$	$V_{T\ dest}$ $=\{\ _{mpeg-4}frt_{dest\ mpeg-4}\ ,\ _{mpeg-4}ct_{dest\ mpeg-4},$ $_{mpeg-4}st_{dest\ mpeg-4},\ _{mpeg-4}dc_{dest\ yuv},$ $_{yuv}ec_{dest\ mpeg-4}\ \}$
V_N	$V_{N\ src}$ $=\{<_{mpeg-1}FT_{src\ mpeg-4}>,<_{mpeg-4}FrT_{src\ mpeg-4}>,$ $<_{mpeg-1}DC_{src\ yuv}>,\ <_{yuv}EC_{src\ mpeg-4}>\}$	$V_{N\ dest}$ $=\{<_{mpeg-4}FrT_{dest\ mpeg-4}>,\ <_{mpeg-4}ST_{dest\ mpeg-4}>,$ $<_{mpeg-4}DC_{dest\ yuv}>\}$
P	P_{src} $<_{mpeg-1}FT_{src\ mpeg-4}>$ $::=<_{mpeg-1}DC_{src\ yuv}><_{yuv}EC_{src\ mpeg-4}>$ $<_{mpeg-4}FrT_{src\ mpeg-4}>::=\ _{mpeg-4}frt_{src\ mpeg-4}$ $<_{mpeg-1}DC_{src\ yuv}>::=\ _{mpeg-1}dc_{src\ yuv}$ $<_{yuv}EC_{src\ mpeg-4}>::=\ _{yuv}ec_{src\ mpeg-4}$	P_{dest} $<_{mpeg-4}FrT_{dest\ mpeg-4}>::=\ _{mpeg-4}frt_{dest\ mpeg-4}$ $<_{mpeg-4}ST_{dest\ mpeg-4}>::=\ _{mpeg-4}st_{dest\ mpeg-4}$ $<_{mpeg-4}DC_{dest\ yuv}>::=\ _{mpeg-4}dc_{dest\ yuv}$

[2] 논터미널과 생성 규칙을 생성하는 단계

// 초기화

V_T={ },

V_N={ },

Candidate V_N={ },

P={ },

TempTR=TR

Repeat Until TempTR == { }

Select a $tr_i \in$ TempTR

TempTR =TempTR - { tr_i }

// 터미널 생성

V_T= V_T ∪ { tr_i }

// 논터미널의 모든 후보의 생성

Candidate V_N = Candidate V_N ∪

{ $<_{in\_a}$ tr-type $_{out\_a}>$ |

for all in_a, out_a ∈ Domain(Format) }

Repeat Until Candidate V_N == { }

Select a nonterminal $<_{in\_a}$ tr-type $_{out\_a}> \in$ Candidate V_N

switch (tr-type)

{

CASE frame rate tr : qos_{src} (A, X, Y, in_a)

qos_{dest} (B, X, Y, out_a)

where A ≠ B

break;

CASE color tr: qos_{src} (X, A, Y, in_a)
qos_{dest} (X, B, Y, out_a)
where A ≠ B
break;
CASE size tr: qos_{src} (X, Y, A, in_a)
qos_{dest} (X, Y, B, out_a)
where A ≠ B
break;
CASE format tr : qos_{src} (X, Y, Z, in_a)
qos_{dest} (X, Y, Z, out_a)
where in_a ≠ out_a
break;
}
TP =CALL CreateQoSbasedTP(qos_{src}, qos_{dest}, TR)

let TP={tp_1 ⋯tp_n} and
let tp is a sequence of transcoders,
i.e. for all tp_i ∈ TP let tp_i=<tr_type_1, ⋯, tr_type_k>
if (TP <> EMPTY)
then
{
{V_N = V_N ∪ { < $_{in\_a}$ tr-type $_{out\_a}$ > };
P=P ∪ { < $_{in\_a}$ tr-type $_{out\_a}$ > ::= tr_1 tr_2 ⋯tr_k };
}

[3] 시작 생성규칙을 생성하는 단계

Let QoS of the source be $qos_{src}(X_1, X_2, \cdots, X_n)$,
and QoS of the destination be $qos_{dest}(Y_1, Y_2, \cdots, Y_n)$.

calculate Position Vector (i,⋯,k) where $X_i \neq Y_i, \cdots, X_k \neq Y_k$
generate Candidate $qos_{net\ related}$
$_1C_n$
Candidate $qos_{net\ related}(X_1, \cdots, X_{i-1}, Y_i, X_{i+1}, \cdots, X_n)$
⋯
Candidate $qos_{net\ related}(X_1, \cdots, X_{k-1}, Y_k, X_{k+1}, \cdots, X_n)$
$_2C_n$
⋯
$_{|Position\ Vector|}C_n$

if (*Workload* Candidate $qos_{net\ related}$ ≤ *Bandwidth*)
decide $qos_{net\ related}$ of Candidate $qos_{net\ related}$

Let the needed transcoders be TRX, TRY
between (qos_{src} and $qos_{net\ related}$) or ($qos_{net\ related}$ and qos_{dest}).
The permutation of th needed transcoders such as (TRX, TRY) and (TRY, TRX).

TR_{first}.*input_attr* = qos_{src}.format
TR_{second}.output_attr = qos_{dest}.format

$TRX_{output\_attr\ input\_attr}$ TRY
output_attr =*input_attr*,
output_attr, input_attr ∈Domain(format)

[4] 실험에 사용할 가용 트랜스코더들

	가용 트랜스코더 집합 1	가용 트랜스코더 집합 2
근원지 $V_{T\ src}$	{$_{mpeg-1}frt_{src\ mpeg-1}$, $_{mpeg-4}frt_{src\ mpeg-4}$, $_{mpeg-1}dc_{src\ yuv}$, $_{yuv}ec_{src\ mpeg-4}$}	{$_{mpeg-1}frt_{src\ mpeg-1}$, $_{mpeg-4}frt_{src\ mpeg-4}$, $_{mpeg-1}st_{src\ mpeg-1}$, $_{mpeg-4}st_{src\ mpeg-4}$, $_{mpeg-1}ft_{src\ mpeg-4}$}
목적지 $V_{T\ dest}$	{$_{mpeg-4}frt_{dest\ mpeg-4}$, $_{mpeg-4}ct_{dest\ mpeg-4}$, $_{mpeg-4}st_{dest\ mpeg-4}$, $_{mpeg-4}dc_{dest\ yuv}$, $_{yuv}ec_{dest\ mpeg-4}$}	

[5] 실험에서 생성된 트랜스코딩 경로

(실험 1과 3에서 생성된 트랜스코딩 경로)

$mpeg\text{-}1frt_{src}$ mpeg-1 $mpeg\text{-}1dc_{src}$ yuv $yuvec_{src}$ mpeg-4 $mpeg\text{-}4net_{mpeg\text{-}4}$ $mpeg\text{-}4st_{dest}$ mpeg-4 $mpeg\text{-}4dc_{dest}$ yuv

$mpeg\text{-}1dc_{src}$ yuv $yuvec_{src}$ mpeg-4 $mpeg\text{-}4net_{mpeg\text{-}4}$ $mpeg\text{-}4frt_{dest}$ mpeg-4 $mpeg\text{-}4st_{dest}$ mpeg-4 $mpeg\text{-}4dc_{dest}$ yuv

$mpeg\text{-}1dc_{src}$ yuv $yuvec_{src}$ mpeg-4 $mpeg\text{-}4net_{mpeg\text{-}4}$ $mpeg\text{-}4st_{dest}$ mpeg-4 $mpeg\text{-}4frt_{dest}$ mpeg-4 $mpeg\text{-}4dc_{dest}$ yuv

$mpeg\text{-}1dc_{src}$ yuv $yuvec_{src}$ mpeg-4 $mpeg\text{-}4frt_{src}$ mpeg-4 $mpeg\text{-}4net_{mpeg\text{-}4}$ $mpeg\text{-}4st_{dest}$ mpeg-4 $mpeg\text{-}4dc_{dest}$ yuv

(실험 2과 4에서 생성된 트랜스코딩 경로)

$mpeg\text{-}1frt_{src}$ mpeg-1 $mpeg\text{-}1st_{src}$ mpeg-1 $mpeg\text{-}1ft_{src}$ mpeg-4 $mpeg\text{-}4net_{mpeg\text{-}4}$ $mpeg\text{-}4dc_{dest}$ yuv

$mpeg\text{-}1frt_{src}$ mpeg-1 $mpeg\text{-}1ft_{src}$ mpeg-4 $mpeg\text{-}4net_{mpeg\text{-}4}$ $mpeg\text{-}4st_{dest}$ mpeg-4 $mpeg\text{-}4dc_{dest}$ yuv

$mpeg\text{-}1frt_{src}$ mpeg-1 $mpeg\text{-}1ft_{src}$mpeg-4 $mpeg\text{-}4st_{src}$ mpeg-4 $mpeg\text{-}4net_{mpeg\text{-}4}$ $mpeg\text{-}4dc_{dest}$ yuv

$mpeg\text{-}1st_{src}$ mpeg-1 $mpeg\text{-}1frt_{src}$ mpeg-1 $mpeg\text{-}1ft_{src}$ mpeg-4 $mpeg\text{-}4net_{m\text{-}peg\text{-}4}$ $mpeg\text{-}4dc_{dest}$ yuv

$mpeg\text{-}1st_{src}$ mpeg-1 $mpeg\text{-}1ft_{src}$ mpeg-4 $mpeg\text{-}4net_{mpeg\text{-}4}$ $mpeg\text{-}4frt$ dest mpeg-4 $mpeg\text{-}4dc_{dest}$ yuv

$mpeg\text{-}1st_{src}$ mpeg-1 $mpeg\text{-}1ft_{src}$ mpeg-4 $mpeg\text{-}4frt_{src}$ mpeg-4 $mpeg\text{-}4net_{mpeg\text{-}4}$ $mpeg\text{-}4dc_{dest}$ yuv

mpeg-1ft_{src} mpeg-4 mpeg-4net_{mpeg-4} mpeg-4frt_{dest} mpeg-4 mpeg-4st_{dest} mpeg-4 mpeg-4dc_{dest} yuv

mpeg-1ft_{src} mpeg-4 mpeg-4net_{mpeg-4} mpeg-4st_{dest} mpeg-4 mpeg-4frt_{dest} mpeg-4 mpeg-4dc_{dest} yuv

mpeg-1ft_{src} mpeg-4 mpeg-4frt_{src} mpeg-4 mpeg-4net_{mpeg-4} mpeg-4st_{dest} mpeg-4 mpeg-4dc_{dest} yuv

mpeg-1ft_{src} mpeg-4 mpeg-4frt_{src} mpeg-4 mpeg-4st_{src} mpeg-4 mpeg-4net_{mpeg-4} mpeg-4dc_{dest} yuv

mpeg-1ft_{src} mpeg-4 mpeg-4st_{src} mpeg-4 mpeg-4net_{mpeg-4} mpeg-4frt_{src} mpeg-4 mpeg-4dc_{dest} yuv

mpeg-1ft_{src} mpeg-4 mpeg-4st_{src} mpeg-4 mpeg-4frt_{src} mpeg-4 mpeg-4net_{mpeg-4} mpeg-4dc_{dest} yuv

전 성 미

e-mail : smchon@hananet.net

1980년 숭실대학교 정보과학대학 전자계산학과(학사)

2000년 이화여자대학교 교육대학원 교육공학과(석사)

2003년 숭실대학교 대학원 컴퓨터학과(박사)

1995년~1999년 LG CNS 기술대학원 전문과장

2004년~현재 숭실대학교 정보미디어 기술연구소 연구원

관심분야 : 멀티미디어 스트리밍, 멀티미디어 통신, 멀티미디어 컨텐츠 등

임 영 환

e-mail : yhlim@computing.soongsil.ac.kr

1977년 경북대학교 수학과(학사)

1979년 한국과학원 전산학과(석사)

1985년 Northwestern University 전산학과(박사)

1979년~1996년 한국전자통신연구소 책임. 연구원

1996년~현재 숭실대학교 미디어학부 부교수

관심분야 : 멀티미디어