NPT 사용 방법 개선의 필요성과 그 방안

김진구, 정문열
서강대학교 영상대학원 미디어공학
seventy@empal.com

The necessity of improvement in using NPT and its way

Kim Jin Goo, Jung Moon Ryul
Media Lab, Dept of Media Technology
Graduate School of Media Communications, Sogang University

ABSTRACT

특정한 컨텐츠의 시작과 끝까지의 시간을 표시해주는 NPT는 디지털 방송에서 비데오와 데이터의 동기화된 서비스를 제공하기 위해서 MHP에서 권고하는 규약이지만 널리 사용되고 있지는 않다. 그 이유는 기술적, 문화적, 경제적인 요인 등이 있을 수 있는데 이 글에서는 기술적 원인에 대해서만 다룬다. 현재 사용되고 있는 수신기에 NPT 기능이 구현되어 있지 않다는 점과 구현되더라도 NPT 스트림을 추가적으로 인코딩해야하는 번거로움 그리고 그 와중에 생기는 동기화의 오차등이 NPT 사용을 막는 가장 큰 기술적 원인들이다. 이런 상황에서 NPT를 사용하는 방법을 몇몇 논문에서 제시 했지만, 여전히 실제 방송 프로그램에 사용하기엔 미진한 점이 남아 있다. 그 이유는 동기화의 정확성이 검증되지 않은 점, 추가적인 대역폭 사용이 필요한 점 등으로 들 수 있는데 이 글에서는 그 문제점들에 대해서 고찰해본다. 또한, 그를 통해서 현재 사용하고 있는 방법의 근원적인 결함을 지적하고 새로운 방법인 EIT 테이블을 이용하는 방법과 그 가능성에 대해서 생각해본다.

1. NPT(Normal Play Time)

디지털 방송 어플리케이션(Xlet)[1]이 TV 쇼의 특정한 순간에 어떤 일을 해야 되는 경우 또는 방송스트림을 통해 전송된 어떤 스트림 이벤트를 처리해야 되는 경우, 다시 말해서 어플리케이션과 비데오 스트림이 시간적으로 동기화 시킬 필요가 있는 경우, 그 시간을 나타낼 수 있는 방법이 필요하다. 유럽식 디지털방송에서는 이 목적을 위하여 NPT(Normal Play Time)를 사용하도록 권고하고 있다. 비데오와 동기화된 스트림 이벤트을 Do It Now 형태와 Scheduled로 나눌 수 있는데 전자는 NPT가 필요하지 않고 후자는 NPT가 필요하다[1].

현재 상용화되어 있는 DVB-MHP 미들웨어에는 Xlet에서 현재순간의 NPT를 구하는데 사용되는 API인 getNPT()를 구현하고 있지 않다. 따라서 비데오와 긴밀하게 동기화된 scheduled 스트림 이벤트를 사용한 대화형 방송을 제작하기가 불가능하다. 이에 본 논문에서는 NPT를 미들웨어를 수정하지 않고 근사적으로 구현하는 방법들에 대하여 살펴보고, 이들의 정확도를 조사하여 이들을 어떤 목적으로 사용할 수 있는지를 살펴보고자 한다.

NPT는 본래 MPEG-2 DSMCC(Digital Storage Media Command & Control)의 규약인데, 특정 미디어의 시작과 진행을 제어하는데 사용하는 시간이다. 예컨데, 하루는 0에서 24시까지 흘러간다. 그것은 하루의 시작에서 끝까지의 시간을 나타내는데, NPT의 경우는 나타내는 것이 프로그램의 시작과 끝이라는 것이 다를 뿐이다. 즉, 어떤 프로그램(9시 뉴스나 다른 일반적인 TV 프로그램)의 시작은 0이고 끝은 임의의 시간 X 인 것이다.

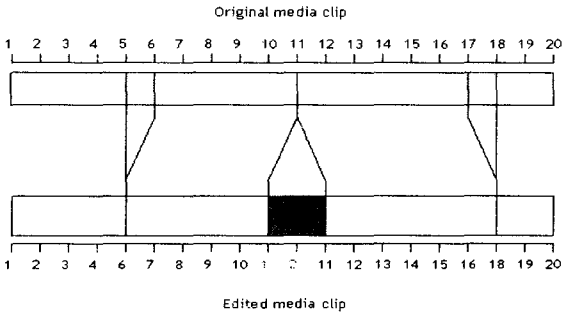

Figure 1. NPT 개념

1) Xlet은 유럽식 디지털 방송의 표준인 DVB-MHP에서 채택한 데이터 방송 어플리케이션 표준이다. Xlet은 기본적으로 Java를 기반으로 하고 있고, STB에서 실행되며 TV에서 시청자의 이벤트를 처리할 수 있다.

NPT는 재방송, 앞 방송 프로그램 종료 시간의 지연 등의 각종 불규칙적인 상황에서도 비데오 스트림과 데이타방송 어플리케이션(Xlet)을 동기화 시켜줄 수 있다. NPT는 특정 프로그램에만 관계된 시간이기 때문이다[1][2]. 그림1은 NPT의 장점을 보다 명확하게 보여준다. 편집된 미디어(방송 프로그램)은 두 가지 측면에서 변화가 일어났다. 첫째는 미디어의 앞부분과 뒷부분 중 일부가 잘려나간 것이고 두 번째는 미디어의 가운데 부분이 어떤 이유 때문에(광고의 삽입, 뉴스 속보등) 늘어난 것이다. 이 경우 NPT 값은 미디어의 변동 후에도 같은 지점을 가리키고 있는 것에 유의하자. 이런 장점은 실제 방송 환경에서 매우 강력한 장점이 된다[3].

그런데, 이런 NPT를 사용하기 위해서는 원래의 방송 프로그램을 구성하고 있는 비디오, 오디오 스트림 이외에도 그 방송 프로그램과 평행한 시간을 가지는 NPT 참조 서술자 스트림[2]을 추가적으로 만들어 방송스트림속에 삽입해 주어야 한다. 그리고 수신기에서는 그 NPT 참조 서술자 스트림에 포함되어 있는 NPT 값을 읽어와 정확한 NPT를 재구성해야 한다.

서론에 언급한 대로 상용화 된 DVB-MHP 셋톱박스의 미들웨어에 아직 현재순간의 NPT를 구하는 API getNPT()가 구현되어 있지 않다. 그러나 비데오와 긴밀하게 동기화된 데이터 방송을 실험적으로 구현하고자 하는 노력의 일환으로 NPT를 근사적으로 구현하는 방법들이 시도되었다[8,1,9]. 그러나 이 방법들이 얼마나 정확한 것이지에 대한 체계적인 분석이 이루어지지 않았다. 본 논문에서는 이 분석과 이 분석에 필요한 실험을 제안하며 마지막으로 NPT를 구하는 새로운 방법을 제안한다.

2. 근사적인 NPT 구현방법

논의의 출발점은 NPT를 정의하고 있는 규약에서 시작해야 한다. NPT는 앞에서 언급했듯이 수신기에서 재구성되어야 하며 그 재구성식은 식1과 같다. 식1에서 STC_Ref와 NPT_Ref는 NPT 참조 서술자 스트림 안에 들어 있는 값으로, 그 두 값은 같은 시간을 나타낸다. STC_ref 는 전송스트림 전체에 대한 일종의 절대시간이고, NPT_ref는 특정프로그램에 대한 일종의 상대시간이다. 수신기에서 NPT를 재구성하는 식은 식 1 과 같다.

$$NPT = (STC / 300 - STC\_Ref) + NPT\_Ref$$

식 1. 규약의 NPT 재 구성식

식1의 방법은 NPTRD= (STC_Ref, NPT_Ref) 를 적당한 빈도로 생성, 전송 스트림에 삽입한다는 가정을 가지고 있다. 이 두 쌍의 시간 값은 STC_Ref가 어떤 값을 가질 때, 이 때의 시각이 NPT_Ref와 같은 시각을 나타낸다는 것을 의미한다. 식 1은 수신기의 애플리케이션 (Xlet)이 수신기의 STC를 읽어 올 수 있으면 구현하는데 큰 어려움은 없다. 그러나 현재 TV API에는 STC를 얻어올 수 있는 API가 정의되어 있지도 않다.

이런 상황에서 [1]에서는 NPT를 재구성하기 위해서 식2와 같은 방법을 사용했다.

$$NPT = (CurSys - Prev) + NPT\_Ref$$

식 2. 현재 사용되고 있는 식

여기서 CurrSys는 NPT를 구하고자 하는 현재시간인데, 구체적으로는 현재순간의 Java System Time이다. Xlet은 전송스트림을 통해 NPTRD가 도착할 때 마다 그 순간의 Java System Time을 구하는데 그중에서 가장 최근의 것이 Prev이다. 따라서, CurSys - Prev는 식1의 STC - STC_Ref와 같은 의미를 가진다고 볼 수 있다.

[8]에서 개발한 NPTRD 생성기는 NPTRD의 STC_Ref 의 값을 TS안에 이미 들어 있는 PCR을 이용하여 구한다. 즉, PCR이 들어있는 전송 패킷을 찾아, 그 때의 NPT_Ref를 구하고 그 순간의 STC_Ref로서 PCR값을 쓸 수 있다. 이렇게 해서 (STC_Ref, NPT_Ref) 쌍을 구하면 원하는 NPTRD가 만들어 진 것이다. 이것을 현재 PCR이 들어 있는 패킷으로부터 가장 가까이 떨어져 있는 미래의 널 패킷에 삽입하면 된다.

그런데, 이 때 NPTRD의 위치는 본래의 PCR 위치로부터 A만큼 떨어져 있기 때문에, PCR 값은 그 만큼 달라져야 한다. TS는 고정된 전송 비트레이트를 사용하기 때문에 TS상의 위치를 알면 그 위치의 시각을 알 수 있다. 이 점을 사용하여, 두 패킷 간의 거리 A로부터 그것에 해당되는 시간을 구해, 그 시간만큼 NPTRD의 (STC_Ref, NPT_Ref)에 더한다.[8] 한편, 식 2에서 Java System Time의 단위와 NPT_Ref 의 단위는 서로 조정을 해야한다.

3. 실험

식1 과 2의 방법은 모두 방송국에서 (STC_Ref, NPT_Ref) 쌍을 적당한 빈도로 생성, 전송스트림에 삽입한다는 가정을 가지고 있다. 그러나, 식 2의 방법을 [8,9]에서 구현할 때는 성분 스트림에서 전송 스트림을 생성하는 도구를 사용할 수 없는 상황이어서 이미 생성된 전송스트림 (TS)을 사용하였다. 따라서 NPTRD를 기존 TS의 널패킷에 삽입할 수 밖에 없었다. 따라서 가용한 널 패킷의 분포에 따라 NPTRD를 충분한 빈도로 삽입할 수 없는 경우도 자주 발생했다. 제대로 하려면 원래 TS를 생성할 때 충분한 빈도와 간격으로 널 패킷이 생성되도록 TS를

2) NPT 값을 담는 구조는 참조 서술자Reference Descriptor 라고 하며 NPTRD라고한다. 따라서,

생성하든지, NPTRD 스트림을 다른 성분 스트림과 동시에 고려하여 TS를 만들든지 해야 한다.

그림 3은 단일 채널의,그림 4는 다채널의 방송 프로그램이 들어 있는 전송 스트림에 NPTRD이 어떻게 분포하고 있는가를 보여주고 있다.

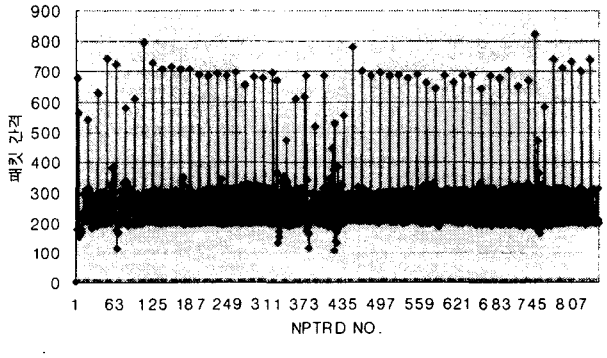

Figure3. 단일 채널의 NPTRD 분포도

그림 3의 경우 X 축은 NPTRD가 등장하는 순서이고, Y축은 현재 NPTRD 와 다음 NPTRD 이 들어있는 TS 패킷간의 간격이다. 패킷 간격은 200개 정도에서 평균을 이루고 있다.

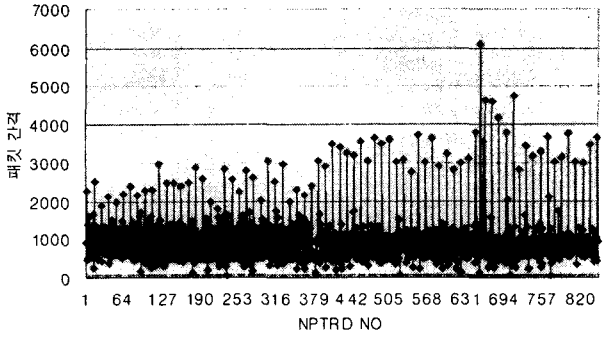

Figure4. 다채널 TS안의 NPTRD 분포도

그림 4는 다채널 하에서의 NPTRD 패킷 간의 거리를 나타낸 것이다. 두 경우 패킷 간격에 상당한 차이가 있는 것을 알 수 있다. 이것은 실험에 사용된 다채널 TS인 경우, 널 패킷의 분포가 좋은 NPTRD 스트림을 생성하기에 적합하지 않았다는 말이 된다.

4. 정확성 측정의 실험 방법

동기화를 위한 기준시간인 NPT 구현에서 가장 중요한 것은 역시 정확성이다. STC를 구할 수 없는 현 상황에서 어떻게, 어떤 방법의 정확성을 측정해야 할까. 식1의 방법은 STC를 구할 수 없다는 점에서 현실에서의 응용 가능성은 없다. 따라서 현재 실험 제작에서 이미 쓰이고 있고, 본 연구실에서 제작하고 있는 각종 NPT관련 제작 도구에서 사용하고 있는 식2가 실제로 얼마만큼의 정확성과 신뢰도를 아는 것이 중요하다. 신뢰도 측정을 위해서는 STC를 가장 근사적으로 아는 방법이 필요한데, 이를 위해서 본 논문에서는 모든 패킷에 PCR을 넣는 방법을 제안한다.

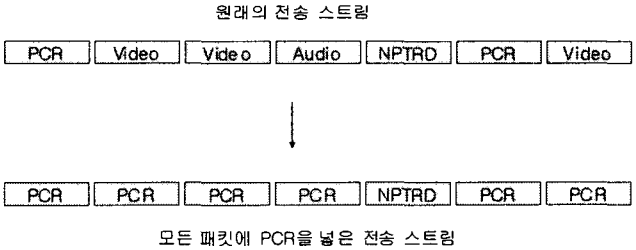

Figure6. PCR 값 삽입

디지털 방송 전송 스트림의 모든 패킷에 PCR 값을 넣는다면 어플리케이션에서 어느 순간 PCR을 얻어 오더라도 그 값은 거의 STC라고 가정해도 무방하다.

MPEG2 System에서는 전송 스트림을 원하는 시간에 재생시키기 위해서 인코더와 디코더의 기준 시간을 동기화 시킬 필요가 있다. 송신기의 STC 샘플링한 값을 전송 스트림 안에 전송하여 수신기의 STC를 동기화 시키는데 그 때 쓰이는 값이 PCR이다. 이 때 송신기와 수신기 사이에 전송 스트림이 도달하는 일정한 지연 시간이 있는데, 이를 무시하고, 인코더에서의 STC가 곧 디코더에서의 STC라는 가정하는데, 이는 그것이 모든 패킷에 공통된 지연 시간이기 때문이다. 즉, 수신기에 도달한 PCR은 곧 수신기의 STC라고 봐도 좋다. 따라서, 모든 패킷에 PCR을 넣는다면 그것은 곧 근사적으로(로딩 타임등의 시간을 무시한다면) 수신기가 얻을 수 있는 최대한도의 정확한 STC 값과 일치한다고 볼 수 있다.

모든 패킷에 사용자 섹션Private Section으로 에 PCR 값을 넣어야 하는데, 이는 애플리케이션에서 이 값에 억세스 할 수 있게 하기 위해서이다. 이 작업이 끝나고 나면 이제 특정한 빈도수를 정해서 NPTRD를 삽입한다. 그렇게 되면 스트림은 이제 그림7과 같은 형태를 가지게 된다.

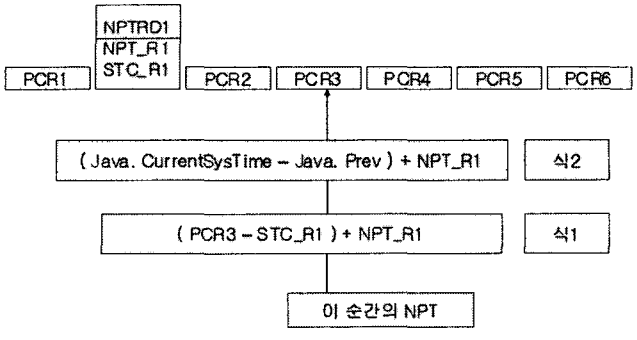

Figure7. 두 식의 비교

그림7은 이를 이용해서 Xlet에서 NPT를 계산하는 것을 나타내고 있다. 매 패킷마다 전송되는 PCR을 STC로 가정

해서 구하는 것이 식1의 방법이고, 띠엄 띠엄 간격을 두고 전송되는 NPT를 이용해서 구하는 것이 식2의 방법이다. 두 방법을 동시에 구할 수 있으므로 두 값을 비교할 수 있고, 결론적으로 현재 사용하고 있는 식2의 정확성의 범위와 한계를 알 수 있게 된다.

5. 제약사항

TS는 늘 다른 TS와 리먹싱될 수 있는데, 본 논문의 방법은 이와 관련하여 몇가지 문제가 있다. 리먹싱과정 중에 NPTRD를 포함하고 있는 패킷의 PID가 변할 가능성이 있다. 어플리케이션에서 (Xlet) NPTRD를 읽을 때, 이것이 포함된 패킷의 PID 값을 알고 있어야 하는데, 이것이 변할 경우, 어플리케이션의 코드도 동시에 고치야 하는 번거로움이 있다.

6. EIT와 RST를 이용한 방법

EIT(Event Information Table)는 방송되는 이벤트(즉, 하나의 방송 프로그램-9시 뉴스, 스포츠 중계등)에 대한 정보를 담고 있는 테이블이다. 그 정보에는 그 이벤트의 시작 시간과 종료 시간을 비롯한 각종 정보를 담고 있다. 또한, 디지털 방송 어플리케이션(Xlet)에서 그 값을 읽어 올 수도 있다.[6][7] 따라서, NPT를 재구성하는데 이를 이용할 수 있다.

만약, Xlet이 어떤 이벤트의 시작과 끝 시간을 안다면 그 시간에 적당한 숫자를 곱해서 적절한 NPT를 생성할 수 있다. 디지털 방송의 초당 프레임 수는 29.970이니까 (현재 시간 - 이벤트의 시작 시간) * 30을 하면 초당 30 프레임에 해당하는 값을 가지는 NPT를 만들어 낼 수 있다. 여기서 이벤트 시작 시간은 EIT 테이블에서, 현재 시간은 Java 시스템 타임을 이용한다.

그러나, EIT에 기록된 시작시간과 종료시간은 다분히 추상적인 시간이다. 즉, 예정 방송 시간이 3시인 이벤트가 3시 3분에 방송될 수도 있다는 의미이다. 그런 점에서 EIT 테이블의 단순 사용을 통해서 NPT를 구성하는 것은 의미 없는 작업일 것이다. 동기화라는 것은 Tight하거나 Loose하거나 상관없이 분 단위 이상의 오차를 발생시키는 것을 의미하지는 않기 때문이다.

그런데 같은 DVB-SI 규약 내에는 RST(Running Status Table)라는 것을 정의하고 있다. 이는 간단히 말하자면 EIT의 시작시간 값을 실제 방송의 시간으로 정확하게 바꿀 수 있게 하는 기능을 가지고 있다. 이 기능은 방송 스케줄의 변화로 이벤트의 시작시간과 종료시간이 변하는 경우에 필요한 기능이다. 이 테이블은 예약 녹화 따위의 기능을 보다 정확하게 수행할 수 있는 목적으로 설계되었다. 즉, 사용자가 어떤 이벤트의 녹화를 예약했을 경우 수신기는 전송되는 RST를 참조하여 그 이벤트의 '진정한' 시작시간에 녹화를 시작할 수 있게 된다.

RST의 사용을 통해서 정확한 시작시간을 알 수 있다면 이를 통하여 NPT를 생성하는 것은 의미를 가진다. 본래 NPT라는 것은 어떤 이벤트의 시작시간과 끝 시간을 기준으로 한 시간이기 때문이다. 물론 이런 방법을 통해서 구하는 NPT 스트림의 정확성이 문제시 될 수 있다.

그런 점에서 4장에서 행하는 실험의 결과는 의미가 깊다. 4장에서의 실험에서 식2의 방법이 정확성을 충분히 충족시킨다면, 식2의 방법을 여러 가지로 개선해서 사용하면 되겠지만, 식2의 결과가 충분히 EIT를 사용한 방법과 정확성이 비슷하다면 EIT를 사용하는 것이 다음과 같은 이유에서 장점을 가지기 때문이다.

NPT는 널 패킷에 들어간다. NPT는 기존 전송 구조와는 별개로 Private Section이라는 독립된 구조로 보내야 하기 때문에 하나의 NPT 값을 보내기 위해서 188byte에 달하는 TS Packet하나를 전부 소비해야 한다. NPT가 PCR과 NPT 값 그리고 그 외에 부가적인 정보 몇 바이트를 넣는 것을 생각해 볼 때 이 자체로 이미 큰 낭비가 된다.

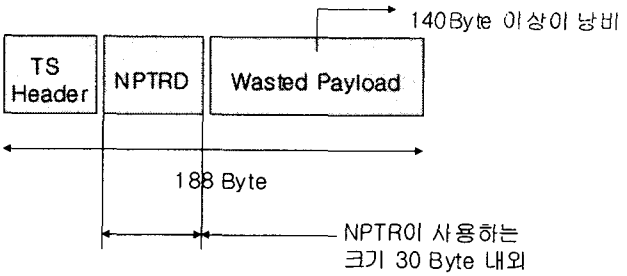

Figure 7. 낭비되는 대역폭

예를 들어 50분짜리 드라마에 NPT를 사용할 경우 얼마만큼의 대역폭을 차지하는지 생각해 보자. 대략 1초에 25번의 NPT를 보낸다고 생각하면 초당 25 * 188 Byte 만큼 사용하게 되고 이는 곧 4.58 Kbyte이다. 더구나, 그 사용량의 상당 부분은 아무것도 쓰지 않은 낭비되는 값이다. 전파는 공공재라는 점에서 이러한 대역폭 낭비는 간과하기 힘든 것이다. 가령, 10개의 채널에서 NPT에 관련된 방송을 한다고 한다면 대략 45Kbyte에 달하는 대역폭이 NPT에 관련된 정보로 소모되는 것이다. 하지만, EIT는 원래 디지털 방송에 들어가는 테이블이다. 즉, 추가적인 소모가 없다.

7. 결론

본 서에서는 현재 NPT가 쓰이지 않는 원인을 여러 가지로 분석했다. 검증되지 않은 정확성, 대역폭 소모, 리먹스 상에서 작용하는 각종 변화 요소들이 NPT의 사용을 막는 기술적인 원인이 되고 있다. 특히, 정확성에 관해서는 규약의 방식과 현재 사용되고 있는 방식의 정확성을 검증해야 할 필요가 있다.

디지털 방송 시대가 도래해도 시청자들이 그 효과를 충분히 누리지 못한다면 그것은 낭비일 뿐이다. 다양한 양방향 방송 프로그램이 디지털 방송의 가장 큰 장점 중 하나라는 점에서 그것을 이룰 수 있는 NPT에 관련된 것들은 계속 연구되어야 하고 개선되어야 할 것이다.

7. Reference

[1] 스트림 이벤트를 이용한 동기화된 데이터방송 애플리케이션 제작. 김세훈, 서강대학교 영상대학원 미디어공학과 석사 논문, 2003

[2] ISO/IEC 13818-6 Generic Coding of Moving Picture and Associated Audio : Digital Storage Media and Control

[3] MHP Tutorial, Http://www.mhp-interactive.org/index.shtml

[4] Issue in Data Embedding and Synchronization and Synchronization for Digital Television, IBM Research, J.Brunheroto외, 2000 IEEE

[5] MPEG-2 Overview of the systems layer, BBC, P.A sarginson외, 1996

[6] 디지털 방송을 위한 전자프로그램 가이드의 설계 및 구현, 오단비, 서강대학교 영상대학원 미디어 공학과 석사 논문, 2002, 2002년도 HCI

[7] JavaTV API Technical Overview. http://java.sun.com/products/javatv/

[8] 핫스팟을 지원하는 디지털 TV 데이터 방송용 비트스트림 제작, 박계철외, 서울 시립대학교 전자전기 컴퓨터공학부, 2002, 2002년도 방송공학회 학술대회

[9] 디지털 방송에서 T-Commerce을 위한 Video와 동기화된 Data Stream 생성과 해석기 구현, 이규혁, 2002년 서강대학교 영상대학원 미디어공학