계층화 비디오의 RAPTOR 부호화를 통한 적응적 전송 방법

*최 범 석, **박 광 훈, ***김 규 헌, ****서 덕 영

경희대학교 전자정보학부

*dcman@khu.ac.kr

Adaptive Transmission of Scalable Video using RAPTOR Code

*Beom Seok Choi, **Gwang-Hoon Park, ***Kyuheon Kim and ****Doug Young Suh

Kyunghee University, College of Electronics and Information

요약

본 논문에서는 Raptor 코드를 이용하여 계층화 비디오를 적응적으로 전송하는 방법을 보여 준다. 클라이언트는 정보를 수신 할 뿐만 아니라 수신한 정보를 다른 클라이언트와 교환함으로서 서버 의존도를 줄여 손실 상황에 대한 강인함을 지니게 된다. Raptor code의 특징을 이용하여 peer-to-peer 환경에서 효율적인 비디오 전송이 가능하다. 이러한 방법을 이용하여 실시간 UCC 비디오와 같은 콘텐츠를 자원이 제한적인 환경에서도 효율적으로 전송 할 수 있는 방법을 보인다.

1. 서론

네트워크의 품질이 향상되고, 멀티미디어 콘텐츠의 교류가 활발해짐에 따라 멀티미디어 콘텐츠의 전송은 더 이상 서비스 업체만의 몫이 아니다. 기존에 정보를 수신만 하던 클라이언트가 수신뿐 만 아니라 정보를 전달하는 클라이언트로 변화되어가고 있다. 특히 클라이언트에서 인코딩 되는 실시간 동영상 서비스가 증가하고 있고, 이러한 서비스는 많은 자원을 요구한다. 이동식 단말의 경우 연산량과 메모리는 제한적이며, 네트워크 대역폭 또한 제한적이다. 이러한 환경에서 실시간 인코딩된 비디오를 안정적으로 전송하는 것은 어렵다. 그러므로 안정적인 서비스를 제공하기 위해서는 자원사용에 부담이 적고, 전송 신뢰도가 높은 기술이 필요하다.

Raptor code는 비동기식 이동망의 표준화 기구인 3GPP(The 3rd Generation Partnership Project, www.3gpp.org)의 MBMS(Multimedia Broadcast/Multicast Service) 표준에 포함된 패킷 손실을 복원하기 위한 장치이다. 이는 LDPC(Low Density Parity-check Code)와 같이 샘물코드(fountain code)의 일종이며 샘물코드는 무율(rateless)한 특성을 가지고 있어서 패리티 패킷을 무한히 많이 생성할 수 있고, 수신측에서도 패리티를 임의로 만들어낼 수 있다. RS(Reed Solomon) 코드와 비교시 계산량이 매우 적고, 원본 패킷의 수에 추가로 몇 패킷만 더 수신되면 디코딩이 된다는 것이 장점이다. 이 기술은 모바일 단말기에 매우 적합한 손실에 대한 패킷 보호를 할 수 있는 기술이다.

Peer-to-peer 데이터 전송은 현재 많은 서비스에서 이용되고 있는 데이터 전송 방식이다. 총 서비스를 담당하는 서버의 네트워크 부하를 줄일 수 있으며, 서버와 클라이언트간 네트워크 상황이 좋지 않더라도 클라이언트간 정보의 교환을 통해 대안을 마련해줄 수 있다. 특히 실시간 멀티미디어 전송을 MANET 과 같은 망에서 사용하는 경우 다양한 라우팅을 통해 서버와 클라이언트간 1:1 전송보다 좋은 효율을 보장할 수 있다.

본 논문은 Raptor code와 peer-to-peer 재전송을 이용하여 안정적인 실시간 비디오 전송방법을 제시 한다.

2. Raptor code의 특성 이용

가. 손실에 대한 강인함을 위한 Raptor code 이용

기존의 RS(Reed Solomon)는 멀티미디어 데이터를 위한 FEC(Forward Error Correction)로서 매우 유용 하지만, 총 생성된 패러티 패킷이 많아지면 연산량이 급격히 높아지는 단점이 있다. 표 1.은 Raptor와 RS의 자원요구량을 보여준다(coding rate = k/n).

	Raptor	RS
메모리	C x k	C x n
연산량(인코딩)	F(1)	F(k)
연산량(디코딩)	F(k)	F(k)

<표 1> Raptor와 RS의 자원 요구량 비교

Peer-to-peer 전송을 하면서 유동적으로 패킷 손실에 대한 보호

를 한다면 각 클라이언트는 FEC 인코딩 연산량과 FEC 디코딩 연산량이 모두 요구된다. Raptor code를 이용하면 FEC 인코딩 연산량과 메모리 사용에 있어서 Reed Solomon에 비해 월등히 유리하다.

나. Raptor 코드의 rateless 특성 이용

기존의 peer-to-peer 비디오 전송은 패킷을 한 클라이언트에서 다른 클라이언트에게 전송하기 전에 상대 클라이언트가 어떠한 패킷을 필요로 하는지 알아야 한다. 만일 확인 하지 않는다면 추가로 받는 패킷의 중복성이 매우 높아진다. 하지만 FEC 의 패러티 패킷을 크게 늘린다면 경우의 수가 증가하게 되고, 이미 수신된 패킷에 대한 정보를 상대에게 제공하지 않더라도 중복성은 낮아진다. Raptor code는 FEC 인코딩 연산량이 F(1)이며 메모리 사용 또한 k와 비례하여 n이 크더라도 자원 사용에 큰 영향이 없다. 디코딩도 F(k) 로 k 에 비례하므로 n 과 관계가 없다. 이러한 특성은 Raptor code를 단순히 FEC로만 사용하는 것이 아닌 패킷의 다중적인 표현(multiple description)을 위한 용도로도 사용할 수 있게 해준다.

다. Raptor 코드의 패킷 재생성

클라이언트는 k개 이상의 패킷을 수신하면 Raptor 디코딩을 시도할 수 있다. 디코딩이 성공하여 원 소스 패킷이 복원되면, 해당 클라이언트는 Raptor 인코딩을 통해 원하는 만큼의 Raptor 패킷을 재생성 할 수 있다. 재생성된 Raptor 패킷은 n 개를 순서대로 생성해야 하는 것은 아니며 랜덤하게 선택적으로 생성 가능하다. 예를 들어 재전송량 만큼의 패킷만을 재생성함으로서 자원을 절약할 수 있다. 이러한 방법으로 생성된 패킷은 어디에서 전송하였더라도 모두 같은 Raptor 패킷으로 처리된다. 예를 들면 클라이언트 X는 서버 Y로부터 수신 받은 패킷을 재생성하여 또 다른 클라이언트 Z에게 전달한다. Z는 X로부터 받은 재생성된 Raptor 패킷과 Y로부터 받은 Raptor 패킷을 합쳐서 디코딩 할 수 있다.

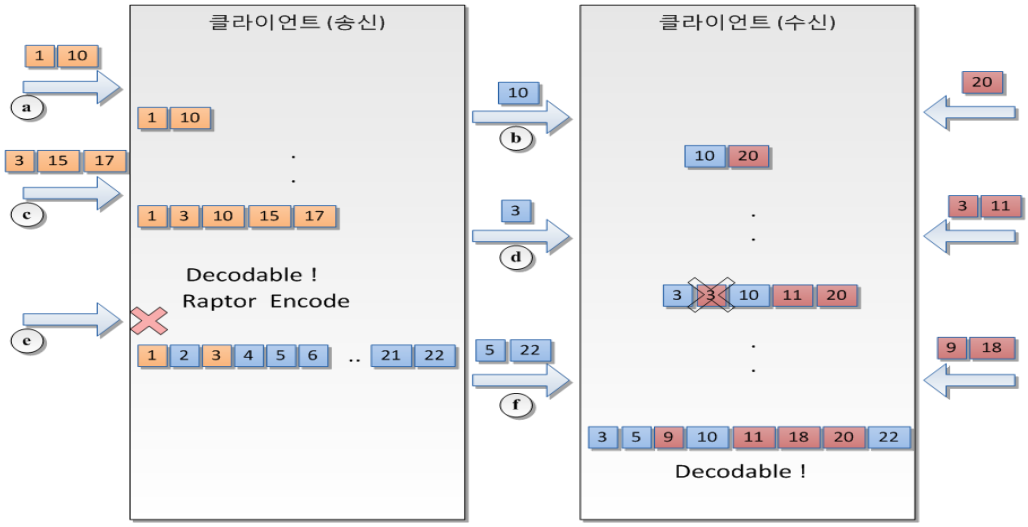
3. Raptor 패킷의 재전송

가. 처리 구조

Raptor code를 이용한 peer-to-peer 재전송 시스템의 단말은 크게 세 종류로 나눌 수 있다. 패킷을 제공만하는 서버, 전송받은 패킷을 재전송하는 클라이언트 또는 서버, 그리고 전달받은 패킷을 사용하기만 하는 클라이언트가 있다. 수신 및 재전송을 하는 클라이언트와 수신만 하는 클라이언트간 패킷이 어떻게 처리되는지 그림 1을 통하여 설명 한다.

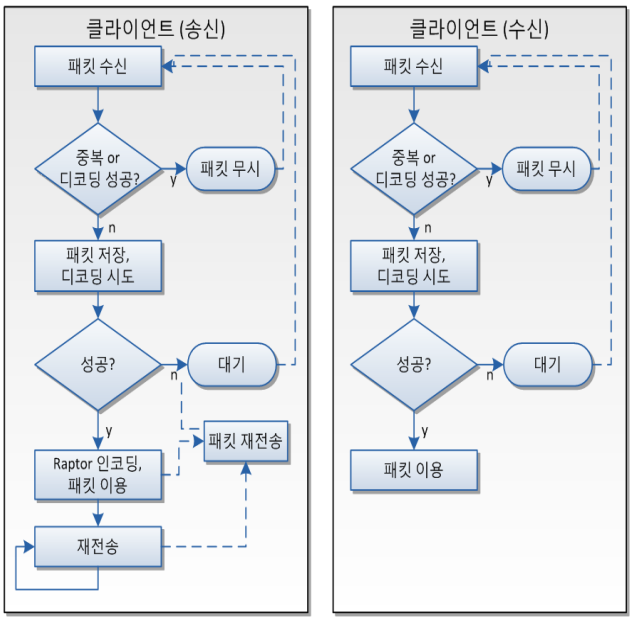
좌측 클라이언트는 수신 및 재전송을 수행하는 클라이언트다. 수신된 정보를 디코딩하여 사용하고, 또한 재전송을 한다. 우측은 수신받은 패킷을 사용하지만 재전송은 하지 않는다. 두 클라이언트 모두 수신 및 재전송을 한다면, 양 쪽 모두 좌측과 같다. 좌측 끝단 화살표는 서버 측에서 전송받는 것이며, 중앙 화살표는 두 클라이언트간 패킷 전송, 그리고 우측 끝단은 또 다른 곳(서버 또는 또 다른 재전송 클라이언트)으로부터 전송 받는 것을 나타낸다. 좌측 클라이언트는 ⓐ와 같이 Raptor 패킷을 수신한다. 같은 시간에 우측 클라이언트도 다른 클라이언트나 서버로부터 패킷을 수신한다. 수신한 패킷은 모두 동일한 Raptor 인코딩 단위다. '인코딩 단위'는 동일한 k개의 데이터 패킷을 이용하여 만들어진 패킷들을 의미한다. 즉 이 단위가 계층화된 비디오의 한 계층의 GOP(Group of Pictures)가 될 수 있다. 좌측 클라이언트는 재전송을 담당하는 클라이언트로서 현재 수신된 패킷 ⓐ의 일부를 ⓑ와 같이 재전송할 수 있다. 재전송시 어느 클라이언트에게 어느 패킷을 보내었는지는 기록하여 중복전송을 줄인다.

우측 클라이언트는 좌측 클라이언트로부터 받은 Raptor 패킷과 다른 송신측에서 받은 Raptor 패킷을 종합하여 동일하게 처리한다. 그림에서 우측 클라이언트는 좌측에서 10번과 다른 송신측으로부터 20번 패킷을 받은 것을 보여준다. 이 단계에서 우측 클라이언트는 Raptor 패킷이 부족하여 디코딩 시도를 하지 않는다. 좌측 클라이언트도 디코딩을 시도할 수 있는 만큼의 Raptor 패킷을 수신하지 못한다. 그러므로 ⓒ와ⓓ에서 추가적으로 패킷을 계속 수신한다. 만일 수신된 패킷의 수가 k개 이상이 된다면 디코딩을 시도한다. 디코딩이 성공한다면 우측 클라이언트는 ⓔ와 같이 동일한 인코딩 단위의 패킷은 더 이상 수신하지 않는다. 그리고 디코딩된 원본 정보를 이용하여 Raptor



<그림 1> 클라이언트간 Raptor 패킷의 전송

code 패킷을 재생성한다. 재생성된 Raptor code 패킷은 상황에 따라 그 개수를 유동적으로 정할 수 있다. Raptor 인코딩을 수행한 경우 우측 클라이언트에게 ⓕ 와 같이 더욱 다양한 패킷을 전송할 수 있다. 다른 peer-to-peer 방식과 달리 Raptor 인코딩된 패킷을 클라이언트간 전달시, 필요한 패킷에 대한 정보를 교환 할 필요가 없다. 그 이유는 중복된 패킷의 수신하여도 중복율을 낮게 n을 크게 할 수 있기 때문이다. 우측 클라이언트도 디코딩이 성공하였다면 좌측 클라이언트와 동일하게 같은 Raptor 인코딩 단위의 패킷은 더 이상 수신하여 저장하지 않고 폐기한다ⓓ. 좌측 클라이언트와 우측 클라이언트가 다른 점은 재인코딩을 통해 재전송을 하느냐이다.

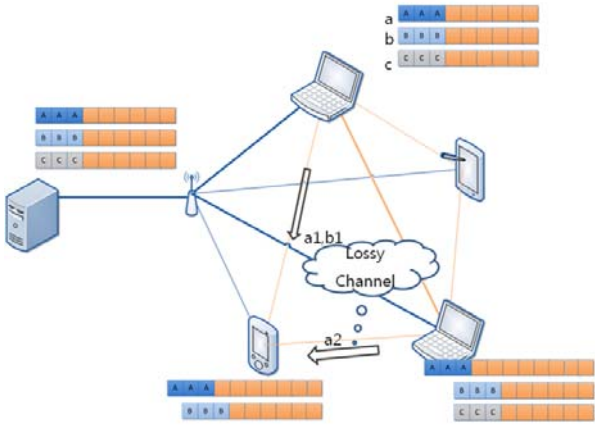


<그림 2>클라이언트의 Raptor 패킷 처리에 대한 블록도

그림 2는 이 구조에 대한 블록도를 보여준다. 이와 같은 블록은 각 Raptor 인코딩 단위별로 사용된다. 즉 t개의 Raptor 인코딩 단위를 동시에 처리 한다면 블록은 t개가 같이 병렬로 처리 돼야 한다. t가 클수록 재전송의 안전성은 높아진다. 하지만 자원이 많이 필요하고, 지연도 증가한다. 요구되는 최대지연과 사용가능한 자원에 따라 유동적으로 t를 설정하여야 한다. 블록도에서 두 클라이언트 모두 중복된 패킷은 저장하지 않고 폐기하며, 디코딩이 성공하면 추가로 수신받은 동 Raptor 인코딩 단위의 패킷은 폐기한다. 좌측 클라이언트와 우측 클라이언트 간 대표적인 차이는 재전송의 유무, 그리고 디코딩 성공 후 일정시간동안 재전송을 계속 수행하느냐 이다.

나. 전송 구조

Peer-to-peer 전송은 매우 유동적인 전송구조다. 다양한 라우팅을 통하여 정보를 전송하며, 유니캐스트 전송방식에서도 멀티캐스트방송과 같은 효율적인 대역폭 사용률을 가진다. Raptor code 는 이러한 전송방식에서 좋은 성능을 발휘한다. 앞에서 설명하였듯이 Raptor code의 rateless한 특성을 이용하면 (n, k)에서 n이 k에 비해 매우 크더라도 인코딩이 Reed-solomon 에 비해 매우 적고, n개중에 랜덤으로 전송을 여러 소스에서 전송하더라도 패킷의 중복은 적다. 그림 3은 Raptor code를 이용한 peer-to-peer 전송을 하는 시나리오를 예로 보여준다.

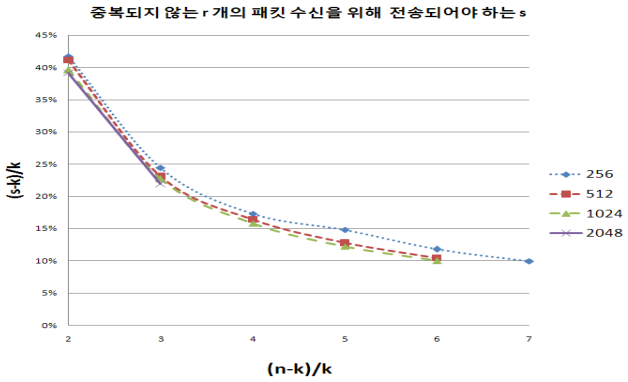


<그림 3> Raptor code를 이용한 비디오의 peer-to-peer 전송

그림 3의 왼쪽 끝은 패킷을 전송만 하는 서버이며, 랩탑 컴퓨터는 전송받은 패킷을 디코딩하여 이용과 동시에 재전송도 하는 클라이언트이고, PDA는 전송받은 패킷을 이용만 한다. 클라이언트는 다른 클라이언트들의 주소와 단말의 종류에 대한 정보를 안다. 서버는 계층화 비디오를 A, B, C 로 각각 Raptor 따로 인코딩을 한다. 이 비디오는 각 계층별 중요도가 1:1:1 이라고 가정한다. 비디오의 품질은 A+B, B+C, A+C 모두 동일하고 A+B+C는 더 좋은 품질이다. 서버는 좋은 사양을 가진 랩탑 컴퓨터에는 A,B,C 모두를 전송하며, PDA에게는 A와 B 만 전송한다. 밑쪽 PDA를 기준으로 보았을 때, 패킷을 전송하는 곳은 서버, 위쪽 랩탑, 그리고 우측 랩탑이 있다. 만일 peer-to-peer 전송이 없다면 밑쪽 PDA는 서버쪽에서만 수신을 받아야 한다. 그 경로가 손실이나 지연이 크다면 이 클라이언트는 디코딩을 성공할 가능성이 낮을 것이다. 하지만 peer-to-peer 전송이 이용된다면 PDA는 다양한 경로로 패킷을 전송받을 수 있기 때문에 디코딩 확률이 높아진다. 앞서 설명하였듯이 수신받은 패킷에 대한 정보를 각 클라이언트에게 보낼 필요가 없으므로 지연이 줄어들 것이다. 그림 3에서 PDA는 서버로부터 Raptor 인코딩 단위 A인 패킷을 전달 받아야 했으나 손실과 지연이 많아 전송받지 못하고, 다른 클라이언트에게서 전달받는 패킷만 가능한 환경이라 가정을 한다. 디코딩을 성공하기 위해서는 나머지 재전송 클라이언트들에게 의존을 해야한다. 만일 PDA는 N_{RCV}개의 패킷을 수신 받아야 원하는 기대치의 디코딩 성공률을 갖는다면, 재전송을 수행하는 두 클라이언트는 총 N_{RCV}개의 패킷을 수신 받을 수 있도록 해야 한다. PDA가 각 클라이언트에게 N_{RCV}개의 패킷을 보내주기를 요청하였다. 하지만 각 경로별 네트워크 상황이 틀리다. 이런 경우 클라이언트별 전송량을 보장하기 위해서는 랩탑은 네트워크상황에 적응적인 전송을 수행해야 한다. 그림 3에서는 우측 랩탑에서 오는 경로가 손실이 발생하고 있다. 그래서 A의 패킷을 추가로 더 많이 보내어 자신의 할당량 만큼 PDA가 수신 할 수 있도록 하고 있다. 그리고 네트워크 상황이 좋지않아 A+B 의 품질을 보장하기가 어려워 최소한 A의 품질이 유지 될 수 있도록 A에 대한 차별적 보호를 한다. 이렇게 비디오의 계층별 차별적인 보호를 통해 최소품질에 대한 보장을 할 수 있다. 위 시나리오는 A인코딩 단위를 B보다 많이 보내어 최소한 A가 디코딩을 성공할 확률이 높아지도록 할 수 있다는 것을 보여주고 있다.

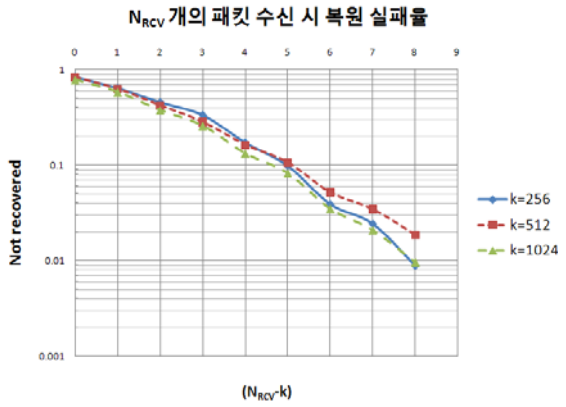
다. Peer별 전송량의 최적화

각 peer간 네트워크 상황이 고르지 못하고, peer의 수가 많다면 전송방법을 최적화하기가 복잡해진다. 하지만 각 peer별 전송 네트워크 상황을 알 수 있다면 수신측 클라이언트가 요청하는 패킷 수신량인 N_{RCV}개를 정할 수 있다. 우선 수신 받는 클라이언트는 각 peer의 개수와 각 경로별 네트워크 상황을 알아낸다. 만일 3개의 peer가 전송 해주고, 각 해당 경로가 손실율이 p0, p1, p2, 라 하고, 각 peer로부터 $N_{RCV}0$, $N_{RCV}1$, $N_{RCV}2$, 개의 패킷을 수신받아 총 N_{RCV}를 수신하려 한다면, 1번 peer가 전송 해야 하는 패킷의 수 s는 $s \leq N_{RCV}1 * (1+p)$이다. 하지만 이 경우 패킷의 중복이 존재한다.



<그림 4> 전송된 패킷수와 중복되지 않는 패킷의 수의 관계

그림 4는 n이 k의 4배 이상이면 k와 상관없이 대략적으로 10% 정도의 패킷을 더 보내면 된다는 것을 보여준다. 중복된 패킷의 수신을 고려하여 n이 k의 4배가 되도록 Raptor 인코딩을 하고, 패킷을 랜덤하게 전송을 한다면 1번 peer는 $s=N_{RCV}1 \leq N_{RCV}1 * (1+p+0.1)$ 개의 패킷을 전송하면 된다.



<그림 5> 수신 받은 Raptor 패킷에 따른 디코딩 실패율

Raptor code 는 Reed-solomon과 달리 k개의 패킷을 수신하면 무조건 디코딩이 되는 것이 아니다. k개의 패킷보다 많이 받을수록 디코딩실패율이 급격히 줄어들게 된다. 그림 5는 그 관계를 보여준다. 대략적으로 디코딩 보장율 99.99%를을 유지하려면 k보다 8개 이상의 패킷을 받아야 한다. 그러므로 N_{RCV}는 k+8이 된다.

Raptor 디코딩 확률이 w 일 때 N_{RCV}는 k+a 라면, N_{RCV} 와 s의 관계를 정리하면 아래와 같다

$$N_{RCV} * (1.1+p) \leq s$$
$$(k+a) * (1.1+p) \leq s$$

4. 결론

Peer-to-peer 비디오 전송은 이제까지 크게 TCP 전송과 Reed-solomon을 이용한 손실에 대한 강인함에 의존 하였다. Raptor code를 이용함으로서 에러에 대한 강인성, 차별화된 패킷보호, 지연, 연산량, 그리고 네트워크 상황에 따른 유동성 모두 Reed-solomon을 이용한 방식보다 월등히 많은 가능성을 보인다는 것을 알 수 있다. 많은 개선을 통하여 이러한 시스템을 최적화 시킨다면 Raptor code 가 FEC의 새로운 패러다임을 열었듯이, Raptor code를 이용하여 MANET 과 같은 Ad-hoc 네트워크에서 DVC나 MDC 비디오를 전송하는 시나리오 또한 비디오 전송에서의 새로운 패러다임을 열 수 있을 것이다.