유선망에서의 RTP/UDP/IP 헤더 압축 설계

김민영*·홍고르촐*·신병철**·이인성*

Design of RTP/UDP/IP Header Compression Protocol in Wired Networks

Min-Yeong Kim* · Khongorzul D.* · Byung-Cheol Shinn** · Insung Lee*

이 논문은 2004년도 한국과학재단 목적기초연구과제(R01-2003-000-11620-0)지원으로 수행되었습니다.

요 약

RTP(Real-Time Transport Protocol)는 실시간 데이터인 오디오/비디오나 IP텔레포니, 멀티미디어 서비스 등을 위한 인터넷 표준 프로토콜이다. 20 ms 프레임 단위로 코딩하는 8Kbps(또는 1K bytes/sec) 음성 코덱의 경우 패킷당 데이터 크기는 20바이트가 되며 RTP/UDP/IP계층을 거치면서 각 계층의 헤더가 추가되어 전체 헤더 크기는 최소한 40 바이트나 되어 많은 부담이 된다. 이를 해결하기 위하여 point-to-point 상에서 여러 가지 압축방법이 제시되었으며, 이 방법에서는 IP계층까지 헤더 압축을 하기 때문에 end-to-end 방식에서는 사용을 할 수 없다.

본 논문은 현재 라우터 기반의 유선망에 헤더 압축 기술을 적용할 수 있도록 기존에 설계된 헤더 압축 기법을 수정하여 성능을 분석하였다.

ABSTRACT

Real Time Transport Protocol (RTP) is the Internet standard protocol for transport of real time data audio/video IP Telephony, Multimedia Seivece. In case of 8Kbps voice codec, the size of packet per data is 20bytes and become more large to minimal 40bytes with adding each layer's header in RTP/UDP/IP. To solve this problem, various header compression skill were suggested on point-to-point network. But it compress even IP header and cannot be suitalbe to apply to end-to-end network. Thus, We will renew header compression protocol to apply wired router-based network.

키워드

Compressed RTP/UDP/IP, 헤더 압축, 라우팅

Ⅰ. 서 론

컴퓨터 통신망에서 최근에 실시간 서비스 트래픽의 수요가 증가하고 있는 추세이다. 이러한 요구를 해결하기 위해 RTP/UDP/IP의 프로토콜들이 사용되고 있다. 그러나 이 프로토콜은 프레임 시간이 8 Kbps인 음성 코덱의 경우, 데이터의 기본 크기는 20바이트이고 이에 반해 RTP 프로토콜 헤더는 12바이트, UDP 8바이

* 충북대학교

** 교신저자: 충북대학교, bcshin@cbu.ac.kr

접수일자 : 2005. 5. 9

트, IP 20바이트가 합쳐진 총 40바이트의 오버헤드를 가지고 있다. 이러한 비효율성을 극복하기 위한 방법으로 활발히 논의 되고 있는 연구가 바로 헤더 압축이다. 저속 링크 환경의 경우 TCP/IP에 관한 헤더 압축[1]과 IP계층에서의 헤더 압축[2], RTP/UDP/IP 계층에서 총체적으로 헤더 압축[3]하는 방법이 연구되고 있다. 최근에는 ROHC (Robustness Header Compression)[4] 헤더 압축 기법이 무선망을 중심으로 연구되고 있다[7][8].

현재 실시간 멀티미디어 서비스에 이용되는 RTP/UDP/IP/Ethernet 프로토콜을 이용할 경우, 이더넷 헤더 26바이트의 크기와 더불어 66바이트를 차지한다. 따라서 음성과 같은 저속 트래픽의 실시간 전송의 경우에는 전송되는 데이터의 크기가 헤더에 비해 작다는 것을 알 수 있다. FTP(File Transport Protocol)와 같이 헤더에 비해 일반적으로 데이터의 크기가 큰 경우에는 큰 문제가 되지 않지만, 많은 경우 실시간 서비스에서는 헤더에 비해 데이터의 크기가 적다. 여기에 헤더 압축 기술을 적용하여 많은 오버 헤드를 줄일 수 있다.

그러나 이 헤더 압축 기술은 현재 라우터 기반으로 구성되어 있는 망에서 직접적으로 사용이 가능하지는 않다. 그 이유는 IP 프로토콜의 압축으로 목적지주소 필드가 없어지기 때문에 라우팅이 불가능해 지기 때문이다. 이를 보완하기 위해, 라우터에서 인식할 수 있도록 헤더 필드의 압축방식을 재수정할 필요가 있다. 따라서 본 논문에서는 라우팅과 관련된 필드의 원본값을 유지해가는 상황에서 헤더 압축 기법을 적용해 보고자 한다.

Ⅱ. 헤더 압축 기술

2.1. 기존의 헤더 압축 기술

기존의 헤더 압축 기술은 먼저 Van Jacobson의 TCP/IP Header Compression으로 기본적인 헤더압축 표준이 제시되었다. 저속의 Link상에IPv4/TCP 패킷들을 압축하여 성능을 향상시킬 수 있도록하는 기본적인 방법이 소개되었다. 기본적인 헤더 압축 개념은 TCP연결이 성립된 후에 패킷들이 교환되는 과정에 앞뒤 프레임사이에서 40바이트의 일반적인 TCP/IP[9] 헤더중에서 많은 필드의 헤더 정보들이 중복된다는 점과, 필드값의 차이가 일정한 규칙에 따라 바뀐다는 점에 착안하여 헤더를 압축할 수 있다는 내용이다. 이와 같이 같은 필드끼리 연속되는 프레임 사이에 differential coding을 수행하면 그 필드값을 나타내기 위한 바이트를 줄일 수 있다. 따라서 헤더 압축 기법을 적용하려면 각 헤더 필드의 특성을 파악해야 한다. 아래는 Van Jacobson이 제안한 TCP/IP Header Compression[1]에서의 헤더 타입이다.

- Compressed_TCP : 압축된 TCP 헤더.
- Uncompressed_TCP : 압축되지 않은 헤더.
- Non-TCP : 압축할 수 없는 TCP프로토콜이 아닌 헤더.
- TYPE_ERROR : 에러 발생시 사용하는 헤더.

그리고 이러한 헤더 압축 기법은 IPv4의 option, fragmentation필드, IPv6의extension headers, 그리고 IPsec(Security)에서의 인증 헤더 등을 고려한 IP Header Compression[2]에서 좀 더 특성화 되었다고 할 수 있다. 이 IP Header Compression 방식에서는 송/수신단에서 헤더 정보를 저장하는 Context라는 테이블의 개념을 도입하고 수신측에서의 에러를 줄이기 위한 방안으로 Twice Algorithm을 적용하였다[2]. 또한 압축 가능한 필드들을 NOCAHNGE, DELTA, RANDOM, INFERRED 등의 카테고리로 좀 더 세분화하여 기존의 방식에 비해 더욱 압축 효율을 높혔다. 다음은 IP Header Compression에서 제안한 헤더 타입과 그에 대한 내용이다:

- FULL_HEADER : 압축되지 않은 패킷으로 초기화 또는 에러후에 refresh를 위해 사용.
- Compressed_NON_TCP : 압축된 헤더를 통해서 NON_TCP 패킷을 표시하는데 사용한다.
- Compressed_TCP : Context ID를 포함하며 압축된 TCP헤더를 표시한다.
- CONTEXT_STATE : 디컴프레서가 컴프레서로 부터의 동기를 잃었을 경우 각각의 Context를 맞추기 위해 보내지는 패킷.

2.2. 실시간 서비스를 위한 헤더 압축 기술

Compresed IP/UDP/RTP[3]는 실시간 서비스를 위한 RTP(Realtime Protocol) 프로토콜 중심의 헤더 압축 기법으로 Compressed RTP 또는 cRTP라고 불리운다. 또

한 링크 상태에 따른 헤더의 부가 전송을 이용해 에러 복구를 향상시킨 ECRTP[5]가 있다. 처음에 제시된 TCP/IP 헤더 압축 표준에서의 differential coding을 사용하고 있으며, IP Header Compression에서 새롭게 제안된 "context"나 "twice algorithm" 등을 이용한다. 실시간에서의 패킷당 페이로드의 크기가 평균 20바이트인 것에 비해 헤더의 크기는 40바이트로 거의 2배의 오버헤드가 발생하는 것을 알 수 있다. CRTP는 20바이트의 IP헤더와 8바이트의 UDP헤더, 그리고 최소 12바이트의 크기를 차지하는 RTP헤더가 합쳐진 최소 40바이트의 전체헤더를 최대 2바이트까지 압축이 가능한 기법이다. Compressed RTP에서 추가된 헤더 타입은 다음과 같다:

- Compressed_RTP : RTP/UDP/IP 모두 압축된 패킷 타입.
- Compressed_UDP : IP의 조각화 표시.

현재까지의 헤더 압축 기술은 라우터를 거치지 않는다는 가정하에서 제안되었다. 즉 중간에 라우터가 중계하지 않는 환경에서 헤더를 압축하는 것이다. 라우터에서 압축된 헤더를 인식하기 위해서는 압축/복원의 능력을 가지고 있거나, 아니면 압축된 헤더를 인식할 수 있는 소프트웨어를 설치해야 한다. 이러한 문제를 해결하기 위해 MPLS상에서의 헤더압축 기법이 제안되었다. MPLS는 Best-Effort 기반인 인터넷망에서 라벨을 이용해 고속의 패킷스위칭이 가능한 방식이다[6]. Ingress-router에는 compressor, egress-router에는 decompressor를 추가하여 MPLS 망내에서 라우팅 문제를 해결하고자 하고 있다. 그러나 각 라우터가 MPLS 프로토콜을 지원하여야 하며, 이를 위해 라벨을 인식하는 장치가 설치되어야 한다.

Ⅲ. 라우팅을 위한 헤더 압축 설계

본 연구는 현재 IPv4기반의 Best-Effort망에서 사용될 수 있도록 라우팅이 가능한 압축된 프로토콜을 설계하고자 한다. 실시간 서비스를 위해서 IP/UDP/RTP의 헤더 필드를 분석하고, 각 필드의 특성 및 패킷과 패킷사이의 변화값을 조사하여 각 클래스별로 나누어 보고, end-to-end 서비스를 위해 라우팅이 될 수 있도록 헤더 부분 중 라우팅에 필요한 필드를 제외하고 다른 필드는 페이로드로 대체하는 방안을 제시해본다.

3.1. 헤더 필드의 분류

Compressed RTP에서는 IP 근원지주소와 IP 목적지주소, UDP 근원지 포트번호와 UDP 목적지 포트번호, 그리고 SSRC(Synchronization Source)필드를 조합하여 세션 정보를 담은 session context로 표시한다. 위에서 언급한 필드들은 RTP세션 연결이 유지되는 동안에는 항상 같은 값을 유지하는 필드들이기 때문에 session context에서 한번만 표시하면 된다. 나머지 필드들 중에서는 일부 일정한 값으로 증가하는 필드들이 있으며, 하부계층에서의 정보로부터 계산을 통해 그 값을 알아낼 수 있는 필드도 있다. RTP 헤더의 각 필드들은 다음과 같은 특성들을 가지고 있다. Version, Padding, Extension, 그리고 SSRC(Synchronization Sour ce Identifier)필드는 그 특성상 변동이 없는 필드로 구분될 수 있으며 Sequence Number와 T(Timestamp)는 변동값이 일정하여 추론이 가능한 필드로 구분될 수 있다. CC(CSRC Count)는 원래 데이터를 보낸 사람의 갯수를 식별하는데 사용되며 그 값은 가변적이지만, 원래 데이터를 누가 보냈는지를 알려주는 CSRC 필드로부터 그 값의 추론이 가능하다. 따라서 CSRC Count나 CSRC list는 다른 값을 이용하여 추론이 가능한 필드로 분류할 수 있다. M(Marker)필드는 RTP를 이용하는 응용계층에서 데이터 경계를 파악하기 위해 사용되는 필드이며, PT필드는 Payload Type, 즉 RTP를 이용하는 응용계층의 데이터타입을 표시하는 필드이다. 이 필드들은 압축을 할 수가 없다. 이러한 특징을 통해 RTP의 여러 필드들을 다음과 같이 세가지 클래스로 나누어 볼 수 있으며 각각에 속하는 필드는 표 1에 나타내었다.

- STATIC : 일정한 값을 가지는 압축 가능필드.
- INFERRED : 다른 헤더 필드들을 통해 추론이 가능하여 압축 가능한 필드.
- CHANGING : 랜덤한 값을 가지는 필드로서 압축이 어려운 필드.

3.2. 라우팅을 고려한 헤더 압축 설계

표 1에서의 헤더 필드의 분류를 근거로 하여 RTP

프로토콜의 각 필드에 대해 압축 가능 여부를 표시하였다. RTP프로토콜 에는 압축이 불가능한 필드가 CSRC(Contribution Source), M(Marker), 그리고 PT (Payload Type)으로 분류될 수 있는데, 특히 CSRC의 경우는 CC(CSRC Count)필드와 관련이 있는데 CC필드는 4비트로 그 값에 따라 CSRC의 ID리스트 갯수가 바뀌게 되어 CC필드의 변화에 따라 0의 값을 가질때는 CSRC가 없는 경우를 뜻하게 되며, 1~15의 값을 가질 때는 CSRC가 여러개임을 뜻하게 된다. 여기서는 RTP 헤더의 크기를 12바이트로 가정하고 있는데 이는 CC의 값이 0임을 뜻한다. 즉 RTP패킷이 중간에 Mixer에 의해 혼합되어지지 않는다는 가정을 하고 있다.

총 헤더의 크기는 66바이트로 이더넷헤더 26바이트, IP헤더 20바이트, UDP헤더 8바이트, RTP헤더 12바이트이다. 여기서 이더넷상의 최소 페이로드 크기는 76바이트이다. 표 1에서 볼 수 있듯이, UDP헤더는 8바이트가 모두 압축이 가능하고 IP헤더는 라우팅을 고려하지 않는다면 16바이트가 압축이 가능하다. 라우팅이 가능하도록 하려면 IP의 각 필드가 어떻게 처리되는지 파악하여야 한다. 그리고 라우터가 이용하는 필드는 원래의 값을 유지시켜야 한다. 라우터에는 들어온 패킷을 처리하도록 하는 여러 모듈이 있는데 그림 1은 IP를 처리하는 모듈들을 보인 것이다[9]. 이 모듈에서는 목적지 주소, TTL 필드, 그리고 버전 필드와 전체 길이 필드를 처리하게 된다.

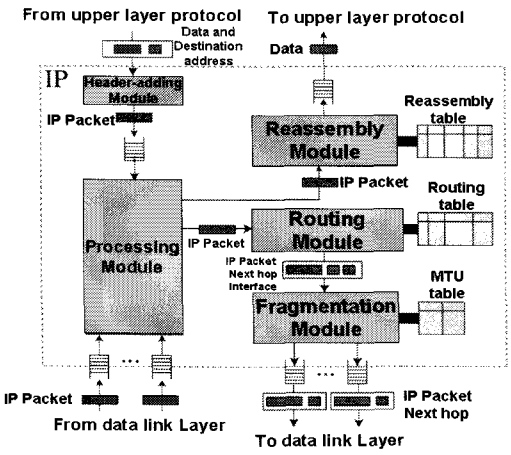


그림 1. IP 패킷 구성 모듈
Fig 1. IP Packet construction module

우선 버전필드를 확인하여 그 값이 4가 아니라면 다른 프로토콜로 인식하고, 전체길이 필드가 맞지 않다면 중간에 에러가 난 것으로 인식하여 무조건 버리게 된다. TTL필드를 통해서 그 패킷의 수명을 확인할 수 있으며, 목적지 주소를 통해 루프백 패킷인지 혹은 패킷이 최종 목적지에 도착하였는지 검사한다. 이러한 처리 후에 이 패킷들은 라우팅 모듈로 넘겨지게 되고, 이 후에 단편화 모듈과 재조립 모듈로 넘겨지게 된다.

이로부터 파악할 수 있는 것은 IP 처리모듈이 활용하는 필드는 압축할 수 없으며, 그와 더불어 전체 길이 필드를 맞추어야 한다는 것이다. 이러한 IP필드의 분석을 통해 그림 2와 같이 압축 가능한 헤더 필드를 판단할 수 있다.

그림 2에서 흰색으로 표시된 필드는 헤더압축이 불가능한 필드로 분류된다. 그리고 데이터 필드에서 압축되는 IP 필드만큼 데이터가 옮겨지게 된다.

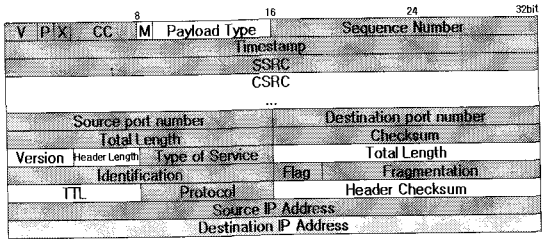


그림 2. 라우팅을 고려했을 경우 제거 가능한 필드
Fig 2. Erasable field when routing is considered

RTP헤더는 원래의 헤더크기인 12바이트에서 11바이트를 제거할 수 있으며, UDP헤더의 경우는 8바이트 중에서 모두를 제거할 수 있다. IP헤더에서 Version필드와 Header Length, Total Length, TTL(Time to Live), Header Checksum 그리고 Destination IP address는 라우터의 각 모듈에서 처리하기 위해 필요한 필드이다. 총 20바이트중 위에서 언급한 부분을 제외한 나머지 필드 10바이트를 압축할 수 있다.

표 1은 IP/UDP/RTP 스택에서 사용되는 헤더 필드를 압축될 수 있는 필드 종류(STATIC, INFERRED, CHANGING 등)에 따라 구분하여 정리한 것이다.

표 1. 헤더 필드의 분류
Table 1. Classification of header fields

RTP Protocol		
Protocol Field	size(bits)	STATE
V(Version) P(Payload Type) X(Extension) SSRC(Sync Source)	2 1 1 32	STATIC
CC(CSRC Count) Sequence Numner Timestamp	4 16 32	INFERRED
CSRC M(Marker) PT(Payload Type)	0~480 1 7	CHANGING
UDP Protocol		
Protocol Field	size(bits)	STATE
Source Port Number Destination Port Number	16 16	STATIC
UDP Length Checksum	16 16	INFERRED
IP Protocol		
프로토콜 필드	size(bits)	STATE
Version Header Length TOS(Type of Service) TTL(Time To Live) Protocol Flags Fragment offset Source IP Address Destination IP Address	4 4 8 8 8 3 13 32 32	STATIC
Total Length Header Checksum	16 16	INFERRED
Identification (if fragment)	16	CHANGING

그림 3은 리눅스 운영체제에서 압축 헤더가 생성되는 과정을 보여주고 있다. 먼저 소켓 버퍼를 수정해야 되기 때문에 원본 소켓 버퍼인 sk_buff 1을 임의로 생성된 sk_buff 2로 복사하고, 이를 초기화하여 헤더 및 데이터를 제거한다. 데이터를 제거한 후에 정의된 헤더 필드에 맞게 소켓 버퍼를 생성을 하게 된다. 즉

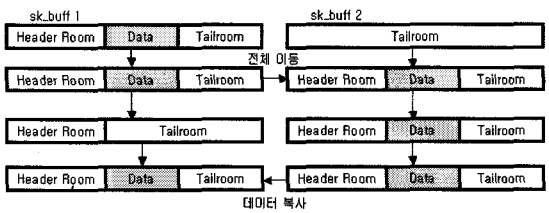


그림 3. 압축 헤더 생성 과정
Fig 3. Compressed header production processing

sk_buff 1을 sk_buff 2에 복사하고 sk_buff 1의 헤더 부분에 위에 분류한 필드에서 CHANGING state는 남겨두고 나머지 필드는 제거하게 된다. 제거된 필드에는 데이터를 넣게 되어 전체적으로 패킷의 사이즈가 줄어들게 되는 것이다. 헤더를 제거하는 방법은 pull() 함수를 이용하여 현재 있는 데이터의 앞부분부터 데이터를 제거하고, 그 반대로 헤더를 다시 생성할 때는 push() 함수를 이용하여 현재의 데이터의 앞부분에 새로운 데이터를 추가할 수 있다.

압축전의 소켓 버퍼에 저장된 구조를 살펴보면 다음과 같다. Header room은 디바이스 헤더를 위해 미리 할당된 공간이며, 그 다음에 IP 헤더, UDP 헤더, RTP 헤더, 그리고 데이터가 저장되고 아직 할당되지 않은 Tail Room이 남아있게 된다. 압축 후에는 그림 4(b)와 같이 디바이스 헤더인 Header room은 그대로 변화가 없고, RTP 헤더의 일부분과 IP 헤더의 일부압축된 부분에 데이터가 대신 들어가게 되고 UDP 헤더와 RTP 헤더는 압축된 헤더로 변하게 된다. 수신단에서는 역으로 압축된 헤더를 받아서 context table을 이용하여 원래의 헤더로 복원하게 된다.

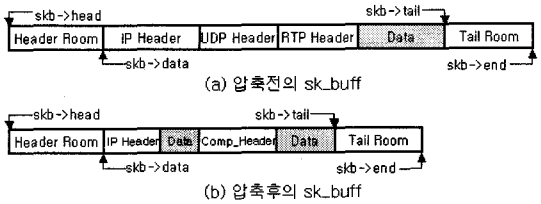


그림 4. 압축 전후의 sk_buff
Fig 4. Original and compressed sk_buff

Ⅳ. 헤더 압축 설계 분석

헤더 압축 설계의 성능을 분석하기 위해서 대역폭의 사용 효율을 고려해 볼 수 있다. 그 중에서 음성 서비스에서 각 음성 코덱마다 압축되지 않은 헤더와 제안된 압축 헤더사이의 대역폭을 비교해 보고자 한다. 매 20msec 마다 한개의 패킷 프레임이 생성된다고 가정하며, 각 패킷의 페이로드 크기는 프레임시간이 20ms 일 때의 경우로 설정하였다. 식의 단순성을 위해 여기에서 사용되는 변수를 다음과 같이 정의하였다.

- H_e : Ethernet header size.
- H_r : RTP/UDP/IP header size.
- H_cr : Compressed RTP/UDP/IP header size.
- P_v : Voice payload size.

위의 변수를 이용하여 다음과 같이 대역폭을 계산할 수 있다.

· voice packet size = H_e + H_r or H_cr + P_v
· N=codec bit rate/P_v[pps(packet per second)]
· bandwidth (per call) = voice packet size × N

예를 들어 헤더 압축이 적용된 상황에서 G.729 (8Kbps codec) 음성 코덱에서 필요한 대역폭을 계산해 보면 다음과 같다.

(a) 원래의 패킷을 적용한 경우
· voice packet size (bytes) = (H_e 26bytes) + (H_r 40bytes) + (P_v 20bytes) = 74bytes
· voice packet size (bits) = (74bytes) × 8 bits per byte = 592bits
· N = (8Kbps codec Bit Rate) / (160 bits) = 50 [pps] -160 bits = 20bytes(P_v) × 8(bits per byte)
· bandwidth = voice packet size (360bits) × 50[pps] = 29.6Kbps

(b) 헤더 압축 기법이 적용된 경우
· voice packet size (bytes) = (H_e 26bytes) + (H_cr 11bytes) + (P_v 20 bytes) = 57bytes
· voice packet size (bits) = (57bytes) × 8 bits per byte = 456bits
· N = (8Kbps codec Bit Rate) / (160 bits) = 50 [pps] -160 bits = 20bytes(P_v) × 8(bits per byte)
· bandwidth = voice packet size (456bits) × 50[pps] = 22.8Kbps

위의 계산을 통해 G.711, G.726, G.728, G.729, 그리고 G.723.1 코덱에서의 대역폭을 파악해 볼 수 있으며, 헤더 압축 전의 대역폭과 헤더 압축 후의 대역폭을 아래의 표 2에 정리하였다.

표 2. 압축 전후의 대역폭
Table 2. Bandwidth of original and compressed RTP

compression technic	payload size	Bandwidth full hader	Bandwidth with cRTP
	(Bytes)	(Kbps)	(Kbps)
G.711(64Kbps)	160	90.4	78.8
G.726(32Kbps)	60	67.2	51.8
G.726(24Kbps)	40	63.6	46.2
G.728(16Kbps)	40	42.4	30.8
G.729(8Kbps)	20	29.6	22.8
G.723.1(6.3Kbps)	24	23.6	16

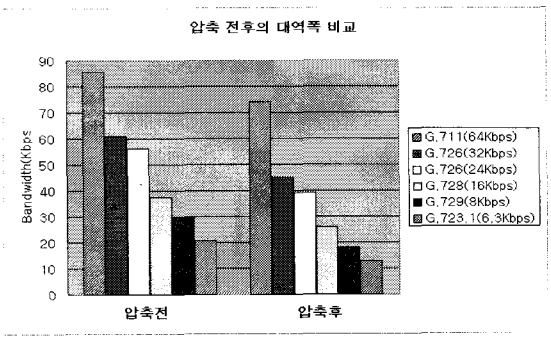


그림 5. 압축 전후의 대역폭 비교
Fig 5. Comparison of bandwidth of original and compressed packet

표 2 에서 보는 바와 같이 헤더 압축 기법이 적용되지 않은 원래의 RTP 패킷에서 라우팅을 고려한 헤더 압축 기법이 적용된 RTP 패킷보다 더 큰 오버 헤드율을 보이는 것을 알 수 있다.

그림 5에서는 RTP/UDP/IP 패킷의 압축전과 압축후에 대한 대역폭의 비교를 보여주고 있다. 코덱에 따라 차이는 있지만, 압축전에 비해 차지하는 대역폭이 감소하여 향상된 모습을 볼 수 있다.

V. 결 론

실시간 패킷 통신에 RTP는 필수적인 프로토콜이 되었다. 하지만 그 데이터양에 비해 헤더가 너무 크기 때문에 상당히 비효율적이다. 이를 극복하기 위해 point-to-point 헤더 압축 방안이 제시되었지만, 이는 라우팅

을 고려하지 않은 문제로 현재의 유선망에서의 적용이 불가능하다. 본 논문은 기존의 RTP 헤더 압축 기법을 라우팅이 가능하도록 설계하였으며, 그에 대한 성능을 분석하였다. IP헤더 필드 중에 라우터에서 인식하는 필드는 압축하지 않고, 그 외의 압축되어 제거된 필드의 자리에는 데이터 페이로드를 삽입하는 방안을 제시하였다. 이러한 방식을 택하여, 전체 IP길이를 맞추고 라우팅이 가능하도록 설계하였다. 이를 통해 기존에 사용되던 RTP/UDP/IP의 총 헤더 크기인 40바이트로부터 RTP에서 11바이트, UDP에서 8바이트, 그리고 IP에서 10바이트를 압축할 수 있었다. 즉, 헤더의 크기만 놓고 볼 경우, 40바이트에서 11바이트로 압축이 가능하다는 결론이다. 이러한 헤더 압축 방안은 앞으로의 실시간 패킷 통신에서의 대역폭 효율을 상당히 향상시킬 수 있을 것으로 기대된다.

참고문헌

[1] V. Jacobson, "Compressing TCP/IP Headers for Low-Speed Serial Link," *IETF RFC 1144*, Feb. 1990.

[2] M. Degermark, B. Nordgern and S. Pink, "IP Header Compression," *IETF RFC 2507*, Feb. 1990.

[3] S. Casner and V. Jacobson, "IP/UDP/RTP Headers for Low_Speed Serial Links," *IETF RFC 2508*, Feb. 1999.

[4] C. Borman et. al., "Robust Header Compression (ROHC)," *IETF RFC 3095*, July 2001.

[5] T. Koren, S. Casner, J. Geevarghese, B. Thompson, "Enhanced Compressed RTP (CRTP) for Links with High Delay, Packet Loss and Reordering," IETF RFC 3545, July 2003.

[6] Jerry Ash, Bur Goode, Jim Hand, "Protocol Extensions for ECRTP over MPLS," IETF Internet Draft, May 2004.

[7] L. Schwiebert, G. Richard, and Jiao. Changli, "Adaptive header compression for wireless networks," *in Proc IEEE Int. conf. Local Computer Networks*, pp.377-378, 2001.

[8] Le. Khiem, C. Clanton, Liu. Zhigang and Zheng. Haihong, "Efficient and robust header compression for real-time services," *in Proc IEEE Int. conf. WCNC*, vol.2, pp.924-928, 2000.

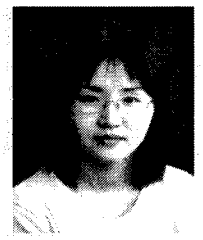
[9] Behrouz A. Forouzan, TCP/IP Protocol Suite, 2/e, MG-GrawHill, 2003.

저자소개

김민영(Min-Yeong Kim)

충북대학교 전파공학과 학사졸업
충북대학교 전파공학과 석사 졸업예정

※관심분야 : 광대역 네트워크, BcN, 휴대인터넷



Khongorzul D.

Bachelor of Mathematics in Nat. Univ. of Mongolia (NUM), Mongolia
Master of Math. in NUM, Mongolia
Ph.D Student of Radio Eng. in Chungbuk Nat. Univ., Korea

※Interest: A Mobile/Multimedia Communication Syst.

신병철(Byung-Cheol Shin)

KAIST 전자공학과 석사
KAIST 전자공학과 박사
KAIST 부교수
충북대학교 교수

※관심분야 : 멀티미디어통신, 무선랜

이인성(In-Sung Lee)

한국통신 연구 개발단 전임연구원
Texas A&M Univ., Dept. of Electrical Eng.
ETRI 이동통신기술연구단 연구원
충북대학교 부교수

※관심분야 : 이동/위성통신 시스템, 신호처리