UDP 헤더압축 구현 및 성능분석

나종민 · 이종범 · 신병철

충북대학교

Implementation and Performance Analysis of UDP/IP Header Compression

Jong-min Na · Jong-bum Lee · Byung-cheol Shin

*Chungbuk National University

E-mail : joyer@just.chungbuk.ac.kr

요 약

현재의 인터넷 환경은 실시간 서비스와 멀티미디어 데이터의 요구들이 계속 늘어나고 있는 추세이다. 이에 반해, 현재 널리 쓰이는 UDP/IP 프로토콜에서는 상당한 오버헤드가 존재하고 있다. 즉, 같은 패킷 스트림 안에서 연속적인 패킷의 헤더 필드 사이에 중복되는 오버헤드이다. 헤더 압축은 바로 이러한 오버헤드를 최소화하여 전송 효율을 높이는 방법이다. 거의 변화하지 않는 필드 정보를 최초에 한번 보내고 계속 유지함으로써 그 다음에 예상되는 필드 값을 대치하게 되는데, 이렇게 함으로 계속 필드 정보를 보내는 헤더의 크기를 최소화 할 수 있다.

본 논문에서는 특히, 이더넷 환경에서 UDP/IP 프로토콜의 헤더에서 발생하는 오버헤드를 줄였다. 대부분의 UDP/IP 헤더 패킷은 7 바이트 정도로 압축될 수 있으며, 헤더 압축 시스템은 리눅스 환경에서 디자인되고 구현되었다.

ABSTRACT

Recently, the demands for real-time service and mutimedia data are rapidly increasing. There are significant redundancies between header fields both within the same packet header and in consecutive packets belonging to the same packet stream. But there are many overheads in using the current UDP/IP protocol. Header compression is considered to enhance the transmission efficiency for small size of payload. By sending the static field information only once initially and by utilizing dependencies and predictability for other fields, the header size can be significantly reduced for most packets. This work describes an implementation for header compression of the headers of IP/UDP protocols to reduce overhead on Ethernet network. Typical UDP/IP Header packets can be compressed down to 7 bytes and the header compression system is designed and implemented on the Linux environment. Using the designed Header compression system between a server and a client have the advantage of effective data throughput in network.

키워드

헤더 압축, TCP, ROHC, RTP

Ⅰ. 서 론

최근 과학 기술의 발전과 더불어 컴퓨터를 이용한 통신 기술의 발전으로 광섬유를 전송 매체로 하는 고속통신망의 등장과 함께 트래픽의 크기도 작은 텍스트 중심였던 예전과 다르게 대용량의 멀티미디어 트래픽이 증가하고 있다. 현재 많이 사용되고 있는 컴퓨터 데이터용 TCP/IP나 실시간 서비스에 널리 사용되는 RTP/UDP/IP프로토콜은 고속으로 변화된 망 하부 구조의 속도를 제대로 이용하지 못하는 실정이다. 이 프로토콜들이 설계될 당시에는 에러가 많고 대역폭이 적은 망 환경특성에 맞게 설계되어 에러 제어가 주된 관심사였으나, 에러가 극히 적고 대역폭이 매우 큰 고속망 환경에서는 복잡한 에러 제어 방법 때문에 망이 제공하는 빠른 속도를 제대로 활용하지 못하고 있는 실정이다[1][2]. 이러한 제한된 대역폭에서 문제점을 해결하기 위한 기술들이 유선 망보다는 대역폭의 사용비용이 높은 무선망에서 활발히 연구되고 있다. 대표적인 기술은 Header Removal/Generation, Header Striping/ReGeneration, 헤더 압축기술[11] 등이 있으며, 여

러 가지 기술 중에서 가장 이슈가 되는 기술이 헤더 압축 기술이다. 이 헤더 압축 연구는 PPP프로토콜 기반 환경에서의 연구가 중심이 되고 있다. 유선망환경에서 TCP/IP프로토콜헤더 압축[13][14], RTP/UDP/IP헤더 압축[15]이 연구되었으며, 무선망에서는 ROHC헤더 압축[16]을 중심으로 연구되고 있다. TCP/IP프로토콜이나 실시간 통신에서 사용되는 RTP/UDP/IP프로토콜에서 대부분의 헤더 필드 값들이 몇 가지 특징을 가지고 있다. 일정한 필드 값, 증가폭이 일정한 필드 값, 다른 필드 값을 통해 추론이 가능한 필드 값등 이러한 특성을 이용하여 각 프로토콜의 오버헤드를 줄이게 된다. 헤더 압축 기술을 이용하여 이더넷 기반의 유선 망 환경에 알맞게 적용을 하면, 전송되는 오버헤드들을 감소시켜서, 네트워크 망의 부하를 줄여서, 효율을 증가시킬 수 있다. 특히 전송되는 데이터가 작은 실시간 서비스는 헤더의 오버헤드가 상대적으로 크다고 할 수 있는데, 이런 상황에서 헤더 압축은 더욱 효율적이다.

본 논문에서는 UDP/IP 헤더를 이더넷 기반의 유선망 환경에 맞도록 헤더 필드들을 클래스별로 분류하고 헤더 압축을 설계하여 리눅스 플랫폼에서 구현하였다. 논문의 구성은 다음과 같다. 2장에서는 현재 사용하고 있는 UDP/IP 프로토콜의 문제점 및 해결 방안으로써 헤더 압축의 기본 개념과 최적화시키기 위한 헤더 필드 분류와 기존의 헤더 압축 기술에 대해서 설명하고, 3장에서는 헤더 압축 구현의 기반이 되는 리눅스 커널의 전송/네트워크 계층의 송수신 부분을 분석하고, 송수신되는 패킷을 관리하는 소켓 버퍼의 구조에 대해서 살펴본다. 4장에서는 헤더 압축 설계와 구현에 대해서 설명하며, 5장에서는 헤더 압축에 따른 성능 평가를 하기 위하여, 기존의 UDP/IP프로토콜을 이용한 채팅 서비스와 헤더 압축을 이용한 채팅 서비스를 일정 시간동안 서비스한 후에 채팅에 사용된 데이터를 수집하여 실험 결과를 평가하고, 6장에서는 실험 결과에 대한 고찰과 앞으로의 연구 방향으로 결론을 맺는다.

II. 헤더 압축의 기본개념

(1) 기존 UDP/IP 프로토콜의 문제점

현재 사용하고 있는 UDP/IP/이더넷 프로토콜을 이용해, 서비스를 하는 경우, 헤더의 크기는 UDP 8바이트, IP 20바이트, 이더넷 14바이트이다. 그림 1을 보면 이더넷의 최소 전송 크기인 46바이트[12]에서 UDP/IP헤더의 크기를 제외하면, 18바이트가 남으며, 전송하는 데이터가 1~17바이트까지는 패딩이 붙게 된다. 데이터가 42바이트 이하이면, 데이터 보다 큰 오버헤드가 붙는다는 것을 알 수 있다. 이더넷의 최소 전송 데이터의 46바이트인 경우 UDP/IP/이더넷의 42바이트의 헤더와 18바이트의 데이터로 데이터의 크기의 2배 이상의 오버헤드가 전송되게 된다.

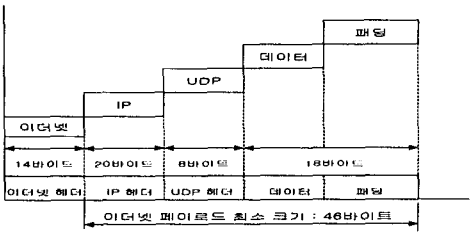

그림 1. 이더넷 최소 전송 크기

FTP와 같이 헤더에 비해 데이터의 크기가 큰 경우에는 큰 문제가 되지 않지만, 대부분의 실시간 서비스에서와 같이 헤더에 비해 데이터가 적은 서비스의 큰 오버헤드는 네트워크 효율 저하시키는 원인이 된다. 이와 같이 데이터에 비하여 헤더가 큰 경우 헤더 압축 기술을 이용하면 많은 오버헤드를 줄일 수 있으며, 네트워크에도 효율에도 도움이 된다.

(2) 헤더 필드의 분류

표 1. IP 헤더 필드의 분류

Filed	size(bits)	Class
Version	4	STATIC-KNOWN
Header Length	4	STATIC-KNOWN
ToS	8	CHANGING
Total len	16	INFERRED
IP-ID	16	CHANGING
Reserved flag	1	STATIC-KNOWN
May Fragment flag	1	STATIC
Last Fragment	1	STATIC-KNOWN
Fragment offset	13	STATIC-KNOWN
TTL	8	CHANGING
Protocol	8	STATIC-KNOWN
Header Checksum	16	INFERRED
Source Address	32	STATIC-DEF
Destination Address	32	STATIC-DEF

표 1은 ROHC[16]에서 IP헤더 필드를 분류한 것이다.

헤더 필드를 분류하는 데 있어서 효율적인 헤더 압축을 위하여, 표 2처럼 통계적으로 나타난 데이터를 이용하여 헤더 필드들을 분류[17]하기도 한다.

(3) 헤더 압축 관련 연구

현재까지 헤더 압축에 대한 연구는 무선망 환경과 유선망에서 PPP프로토콜 환경에 대해서 연구가 되고 있다. 무선망 환경의 연구에서 중요시하는 요소는 에러율, 효율, robustness이다. TCP 프로토콜의 HC 연구로는 VJHC[13], IPHC[14], ROHC등의 HC 알고리즘을 에러율에 따른 효율을 비교[3][7]하는 연구와 HC 알고리즘의 일부를

표 2. 헤더 필드들의 통계적인 확률 수치

Field	Class	Probability
Version Header Length Reserved flag Last Fragment Fragment offset Source Address Destination Address	STATIC	100%
Header Checksum Protocol Total len	INFERRED	100%
TTL	STATIC	99%
	CHANGING	1%
ToS May Fragment flag	STATIC	99.9%
	CHANGING	0.1%
ID	INFERRED	99%
	CHANGING	1%

수정하여, 에러 복구 효율 및 robustness를 보완하는 연구가 있으며[4], RTP프로토콜의 HC 연구로는 CRTP[15], ROHC등 HC 알고리즘에 대하여 에러율에 대한 패킷 손실 및 에러율에 대한 평균 헤더 크기를 비교[6][8][10]하는 연구와 에러율에 따라 에러 정정할때와 에러를 정정하지 않을 때에 패킷 손실를 비교하고[9], 풀 헤더와 압축 헤더의 중간 개념의 세미 압축 헤더로 패킷을 세분화[5]하여 효율을 높이는 연구가 있다. 헤더 압축에 관한 유/무선환경의 대표적인 기본적 모델이 되는 연구로는 Compressing TCP/IP Headers for Low-speed serial Links[13], IP Header compression[14], Compression IP/UDP/RTP Headers for Low-speed Serial Links[15], Robust Header Compressio(ROHC)[16]등이 있다.

III. 헤더 압축 설계 및 구현

(1) 헤더 압축 설계

전송 계층의 프로토콜에 따라 IP계층에서 IP헤더를 생성하는 함수와 헤더 생성 후 경유하는 함수는 다르다. 서로 다르게 헤더를 생성한 후에 모두 IP 계층의 ip_output함수를 지나가게 되고, 하위 계층으로 전송된다. 그림 2에 설계된 헤더 압축 시스템은 IP 계층의 ip_build_xmit에서 IP헤더의 필드들을 생성하고, 소켓 버퍼에 전송 후, 디바이스 계층으로 내려가기 전에 ip_header_comp에서 소켓 버퍼에 있는 데이터와UDP/IP헤더를 복사 후에 데이터를 압축한 후에 다시 소켓 버퍼로 복사하고, 이더넷을 통해 전송을 하게된다. 패킷을 받은 ip_rcv에서도 압축된 헤더를 소켓 버퍼에 복사한 후 원래의 헤더로 복구한 후에 다시 소켓 버퍼에 복사한 후 ip_rcv_finish함수로 전달하게 된다. 목적지 IP 주소에 따라 자신의 주소인 경우 UDP 계층으로 보내고, 목적지가 자신이 아닌 경우 ip_input_route함수와 ip_output함수를 통해 목적지로 전송하게 된다. ip_forwarding이 HC를 지원하지 않는 경우에는 압축된 헤더의 decompression과정이 없이 라우팅 테이블을 통해 ip_forwarding 하게 된다.

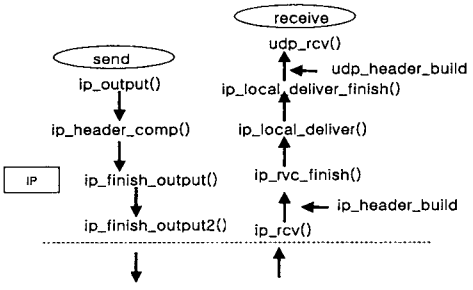

그림 2. 설계된 헤더 압축의 흐름

(2) 필드 분석 및 분류

헤더 필드의 분류는 대부분 무선망에서의 헤더 필드 분류와 비슷하며, 차이점은 무선망에서 대부분 디바이스 계층으로 PPP를 사용하기 때문에 라우팅에 별 문제점이 없지만, 유선 망의 환경에서 이더넷을 사용할 경우, 라우터를 통과해야하기 때문에 HC을 지원하지 않는 경우에는 라우팅 테이블에 관련된 정보를 모두 전송하여야 한다. 실제 ftp서비스나 채팅 서비스 등 TCP/IP프로토콜과 UDP/IP프로토콜의 패킷들을 캡쳐하여 분석한 결과 대부분은 무선망에서 정의한 필드들과 같았지만 몇몇 필드의 경우에는 기존 헤더 분류의 경우와 다른 경우가 발생하였다. IP프로토콜의 경우에 헤더 패킷을 분석 해보면 표 3과 같다. 대부분 필드에 할당된 클래스가 무선망에서의 할당된 필드 클래스와 비슷하지만 ToS는 리눅스나 윈도우 기반의 시스템에서는 지원이 안되고 있으며, IP-ID 필드의 경우에는 TCP는 1씩 증가 하지만, UDP와

표 3. 실제 시스템에서 IP 헤더 패킷 분석

Filed	size(bits)	Class
Version	4	STATIC
Header Length	4	STATIC
ToS	8	STATIC
Total len	16	INFERRED
IP-ID	16	STATIC
frag_offset	16	STATIC
TTL	8	STATIC
Protocol	8	STATIC
Header Checksum	16	INFERRED
Source Address	32	STATIC
Destination Address	32	STATIC

의 경우에는 0값을 갖는다. 그러므로 추론이 가능하거나 STATIC값을 갖는다. Frag_off의 경우에도 STATIC값을 가지고 있다.

헤더 체크섬의 경우에 무선망 환경에서는 에러율이 높은 관계로 헤더 체크섬 필요하지만, 유선망에서의 무선망의 환경과 달리 에러율이 낮다. 표 4[20]의 유선망에서 각 프로토콜의 에러율에 보는 봐와 같이 체크섬을 필요없을 정도로 에러율이 낮다. 만약에 필요하다면 현재 사용하고 있는 체크섬 16비트 보다 적은 비트로 설정하여 오버헤드를 줄이는 것이 좋을 것이다. 무선망에서도 전송 에러가 낮은 경우에는 체크섬 없이 전송하며, 전송 에러가 높은 경우에는 3~4비트 정도의 체크섬[18]을 전송하고 있다.

표 4. 유선망에서 각 프로토콜에서의 에러율

프로 토콜	전송된 패킷	CRC 에러	에러율(%)
Ethernet	170,000,000	446	2.62E-05
IP	170,000,000	14	8.23E-08
UDP	170,000,000	6	3.57E-08
TCP	170,000,000	1983	1.16E-05

UDP 헤더 필드에 대해서 살펴보면, 응용 계층에서 서비스가 시작해서 끝나기까지 포트 번호는 일정하다. UDP 길이 필드는 추론이 가능한데, 만약 IP의 전체 길이 값을 알고 있다면, TCP와 달리 대부분 IP option이 전송되지 않기 때문에 전체 길이에서 IP헤더 길이를 제외하면 UDP 길이 필드가 되며, 여기에 UDP헤더 길이를 제외하면, 순수 데이터만 남게 된다. UDP의 체크섬은 옵션으로 설정되어 있기 때문에 에러율이 낮은 경우 제거 할 수 있다.

표 5. 실제 시스템에서 UDP 헤더 패킷 분석

Field	size(bits)	Class
Source port	16	STATIC
Destination port	16	STATIC
Length	16	INFERRED
Checksum	16	INFERRED

헤더 압축 설계는 기존의 IP/UDP헤더 필드들을 이용해 헤더 압축을 하였으며, ipv4에 대해서만 고려하여 설계하였다. ip_forwarding을 하는 라우팅 시스템에서 헤더 압축을 지원하는 경우와 지원하지 않는 경우의 두 가지 모델로 나누었으며, 각 모델에 따라 위에서 정의한 필드들의 분석을 토대로 전송이 필요한 필드들은 다음과 같다.

- ip_forwarding을 HC이 지원하는 경우(Model I) 에는

서비스 구별 : CID(1바이트) + 발신지 IP 주소.
UDP 길이 필드와 소켓 버퍼를 추론 : 전체길이.
나머지 UDP/IP 필드 제거.

그림 3은 ip_forwarding을 HC이 지원하는 경우(Model I)에 제거 능한 UDP/IP헤더 필드들을 보여 준다. UDP헤더 필드는 8바이트와 IP헤더 13바이트를 제거 할 수 있으며, 제거되는 헤더 필드 대신에 데이터가 전송되어 전송효율을 높이게 된다.

- ip_forwarding을 HC이 지원하지 않는 경우 (Model II)에는

ip_rcv에서 필요한 필드 : IP 버전+IP 헤더 길이+IP 체크섬.

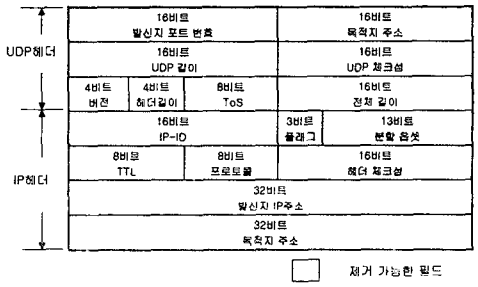

그림 3. Model I의 경우 제거 가능한 헤더 필드

라우팅에 필요한 필드 : 발신지/목적지 IP 주소, ToS, TT.
서비스 구별 : CID(1바이트).
UDP 길이 필드와 소켓 버퍼를 추론: 전체 길이.
나머지 UDP/IP 필드 제거.

그림 4는 ip_forwarding을 HC이 지원하는 않는 경우(Model II)에 제거 능한 UDP/IP헤더 필드들을 보여 준다. UDP헤더 필드는 8바이트와 IP 헤더 4바이트를 제거 할 수 있다. 제거되는 헤더 필드 대신에 데이터가 전송된다.

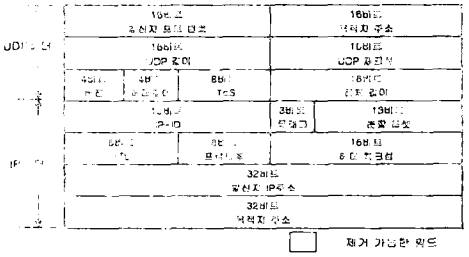

그림 4. Model II의 경우 제거 가능한 헤더 필드

(3) 헤더 압축 구현

본 논문에서 헤더 압축의 성능을 평가하기 위하여 와우 리눅스 7.1(커널 2.4.2)기반의 환경에서 ip_forwarding의 HC지원 여부에 따라 두 가지의 모델로 나누어 코드 구현후 성능 평가를 하였다. 구현 환경은 표 6에서 같이 하나의 서버와 두 종류의 클라이언트 모델로 ip_forwarding이 HC을 지원 모델의 경우(Model I) : 클라이언트1 -> 서버 중간에 라우팅 없이 전송되며, ip_forwarding

지원하지 않는 경우(Model II) : Client2 -> Server 중간에 IP주소가 210.125.158.1의 주소를 가지는 시스템의 라우팅 테이블을 거치게 된다.

구현 범위는 IPv4에만 한정을 하였고, 에러을 을 고려하지 않고 구현을 하였다. 디바이스로는 Ethernet II을 사용하였기 때문에, 최소 페이로드 단위는 46바이트[13]로 IP헤더 20바이트, UDP헤

표 6. 헤더 압축 구현 환경

name	Server	Client1	Client2
OS	WOW linux7.1	WOWlinux 7.1	WOWlinux 7.1
Kernel	2.4.2	2.4.2	2.4.2
IP 주소	210.125.151.114	210.125.151.121	210.125.147.124
용도	Server	Client	Client
위치	실험실	실험실	실험실

더 8바이트, Ethernet II헤더 14바이트가 된다. 그러므로 데이터가 18바이트 이하일 때 헤더 압축을 하는 경우에는 패딩이 붙게 되므로, 헤더 압축은 의미가 없게 된다. 그래서 데이터가 19바이트 상인 경우에 한하여, 헤더 압축을 하게된다.

헤더 압축의 전체적인 흐름을 보면 그림 5와 같이 IP의 전체길이를 확인하고, 전체 길이가 46 이하이면, 헤더 압축 없이 ip_finish_output함수를 호출하여 전송하게 되고, 46초과하면 헤더 압축을 하게된다. 소켓 버퍼를 수정해야 되기 때문에 먼저 원본의 소켓 버퍼를 소켓버퍼2로 복사하고, 소켓 버퍼를 초기화하여 소켓 버퍼에 있는 헤더 및 데이터를 제거한다. 데이터를 제거 후 정의된 헤더 필드에 맞게 소켓 버퍼를 생성한다. 생성 후 ip_finish_output함수로 전달하게 된다.

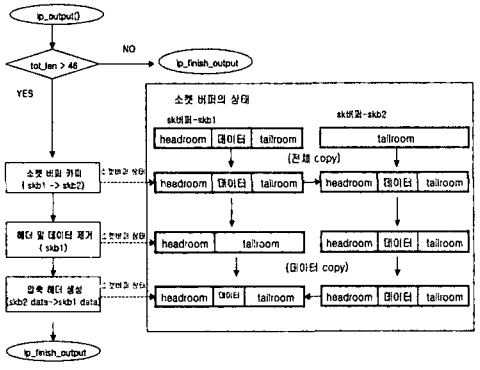

그림 5. 구현된 헤더 압축의 블록도

첫 번째로 ip_forwarding을 HC이 지원하는 경우(Model I), IP 계층에서 압축전의 소켓 버퍼에 저장된 구조를 살펴보면 그림 6과 같다. 앞의 headroom은 디바이스 헤더를 위하여 미리 할당된 공간이며, 그 다음 IP헤더, UDP헤더, 응용 계층에서 내려온 데이터가 저장되어있고, 아직 할당되지 않은 tailroom이 남아있게 된다.

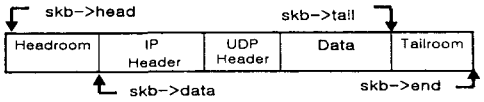

그림 6. IP 계층에서의 소켓 버퍼

저장된 소켓 버퍼를 헤더 압축 후에는 그림 5와 같이 Headroom의 경우에는 변함이 없으며, IP헤더의 필드들은 위에 정의된 것과 같이 IP헤더 일부 대신 데이터가 들어가고, UDP헤더 또한 제거되어 데이터가 들어 가게되어 Tailroom의 크기가 커지게 된다.

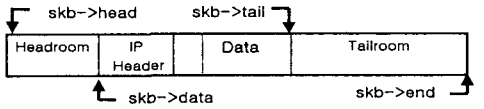

그림 7. IP 계층에서 압축된 소켓 버퍼

디바이스 계층에서는 위의 그림 7의 소켓 버퍼를 받아서 전송하게 된다. 그 이후의 흐름은 앞의 헤더 압축 시스템 흐름에서 설명한 것과 같이 ip_forwarding이 되는 서버에서는 ip_rcv 함수에서 ip 헤더를 생성하고 IP주소에 따라 라우팅 테이블을 이용하여 라우팅 정보를 얻고, 다시 헤더 압축 후에 ip_output을 거쳐서 다시 목적지로 전송하게 된다. 최종 수신측에서도 마찬가지로 ip_rcv에서 압축 헤더를 원래의 헤더로 복원하고, 상위 계층인 udp_rcv함수로 보내서, UDP 헤더를 다시 만들게 된다.

두 번째로 ip_forwarding을 HC이 지원하지 않는 경우(Model II), 압축된 헤더를 생성하는 절차는 똑같고, 다른 점은 ip_forwarding을 하기 위해서는 많은 제약이 따르게 된다. 라우팅되는 시스템에서 drop되지 않고, 라우팅 테이블을 이용하려면, 많은 정보를 전송해주어야 된다. 먼저 ip_rcv에서 drop되지 않기 위해서는 3가지의 조건을 확인하게된다. IP 버전, IP 헤더 길이를 확인하고, 전체길이와 버퍼의 길이를 비교하여 전체 길이 보다 버퍼의 길이가 작은 경우, 패킷의 일부가 손상 되었을거라고 판단하고 drop하게된다. 마지막으로 IP 체크섬을 계산하여, 에러를 확인하게된다. 위의 3가지 조건을 모두 만족시켜주도록 하기 위해서는 IP 버전과 IP 헤더 길이는 전송해주어야 하며, 전체 크기를 헤더 압축후의 소켓 버퍼보다 크지 않도록 재조정을 해줘야하며, IP 헤더 필드 값들을 수정후 IP 체크섬을 다시 생성해야한다. 그리고 라우팅 테이블을 이용하기 위하여 필요한 필드들은 IP 발신지/목적지 주소, TTL, ToS이며, 위와 같은 필드들은 전송되어야만 한다.

IV. 성능평가

본 논문에서 구현된 헤더 압축에 따른 성능 평가를 하기 위하여, 기존의 UDP/IP프로토콜을 이용한 채팅 서비스와 헤더 압축을 이용한 채팅 서비스를 일정 시간동안 서비스한 후에 채팅에 사용된 데이터를 수집하여 데이터에 대한 오버헤드 비율로 성능을 측정하였다.

첫 번째로 ip_forwarding을 HC이 지원하는 경우(Model I : 210.125.151.114 <- 210.125.151.121), 그림 10의 Model I 그래프를 보면 기존 UDP/IP 프로토콜 헤더의 경우 19바이트부터 데이터 크기에 따라 패킷 크기가 증가하지만, 헤더 압축을 한 경우에는 40바이트부터 패킷 크기가 증가하게 된다. 40바이트부터 증가하게되는 이유는 헤더 압축을 통해 UDP/IP헤더 21바이트를 감소시킬 수 있기 때문이다.

성능을 평가하기 위하여 UDP 채팅 서버를 5분간 서비스를 하여 1분 단위로 전송되는 패킷 데이터를 수집 분석하였다. 표 7을 보면 전송된 패킷 수는 분당 전송되는 패킷의 수를 말하며, IP 전체 길이 합은 전송된 패킷의 IP 전체길이 필드의 합을 말한다. 전송된 데이터는 패킷 중 데이터 부분의 크기를 말하며, 전송된 헤더는 UDP/IP 프로토콜의 헤더의 크기의 합을 말한다. 전송 효율을 평가하기 위하여 오버헤드율(%)을 정의하였는데, 오버헤드율(%)은 전송된 데이터에 대한 오버헤드 비율로 데이터의 길이가 이더넷의 최소 페이 로드보다 작아 붙는 패딩도 데이터로 취급하였다. 오버헤드 비율이 적을수록 데이터 전송 효율이 높아진다.

표 7. Model I 환경에서 채팅 서버의 전송 패킷 분석

	1분	2분	3분	4분	5분	평균
전송된 패킷수	24	22	21	25	21	23
IP 전체 길이 합(byte)	1200	1080	1001	1212	1023	1103
전송된 데이터(byte)	528	464	413	512	435	470
전송된 헤더(byte)	672	616	588	700	588	633
전송된 HC 헤더(byte)	404	391	399	456	378	406
기존 UDP/IP오버헤드율(%)	127	133	147	137	135	135
HC UDP/IP 오버헤드율(%)	77	84	97	89	87	87

그림 10은 표 7의 시간에 대한 오버 헤드율을 그래프로 나타낸 것으로 그래프를 보면 헤더 압축을 하게되면 시간마다 조금씩 차이는 있지만 약 50%정도의 오버헤드를 줄일 수 있게되며, 패킷당 평균 10바이트 정도의 헤더를 감소시킨다. 최대 패킷 감소 값인 21바이트와는 많은 차이를 나타내는데, 이유는 전송되는 패킷의 데이터 크기에 따라 헤더 압축 효율이 많이 달라지기 때문이다. 최대 패킷 감소 값과 비슷한 효율을 가지려면 대부분의 전송되는 패킷의 IP 전체 길이 값이 67바이트 이상의 값을 가져야되지만 실제 실험에서는 67바이트 보다 작은 데이터 전송이 많기 때문에, 작은 데이터 전송의 빈도에 따라 효율이 달라지게 된다.

두 번째로 ip_forwarding을 HC이 지원하는 않는 경우(Model II : 210.125.151.114 <- 210.125.158.1 <- 210.125.147.124), 그림 8의 Model II 그래프를 보면 전송되는 데이터가 18바이트가 되기 전 까

지는 패딩을 포함시켜 이더넷의 최소 페이로드인 46바이트가 전송된다. 일반적인 UDP/IP헤더의 경우 19바이트부터 데이터의 크기에 비례하여 증가하지만, 헤더 압축을 한 경우에는 31바이트부터 데이터의 크기에 비례하여 증가하게 된다.

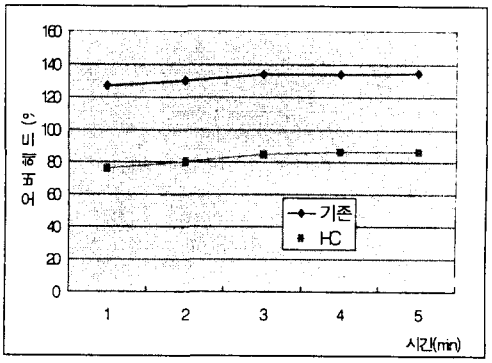

그림 8. Model I 환경에서 채팅 서버의 오버헤드율(%)

위의 첫 번째 실험과 동일한 방법으로 UDP 채팅 서버를 5분간 서비스를 하여 1분 단위로 전송되는 패킷의 데이터를 분석하였다. 표 8의 결과를 보면 기존의 UDP/IP프로토콜의 오버헤드율이 평균 161%인데 반하여, 헤더 압축을 한 경우 123%의 오버헤드율이 발생하여 약 40%의 오버헤드를 감소 시켰다.

그림 11은 표 8의 결과를 그래프로 나타낸 것으로 헤더 압축을 하게 될 경우 약 40%의 오버헤드율이 감소되면, 이를 바이트로 환산하면 패킷당 평균 6.6바이트 정도의 헤더를 감소시킨다. 이 실험도 역시 최대패킷 감소 값인 12바이트 값과 차이를 보이는데, 위의 실험에서와 마찬가지로 전송되는 패킷의 IP 전체 길이 값이 58바이트보다 작은 데이터 전송이 많기 때문이다.

표 8. Model II 환경에서 채팅 서버의 전송 패킷 분석

	1분	2분	3분	4분	5분	평균
전송된 패킷수	24	26	25	27	23	25
IP 전체 길이 합(byte)	1128	1129	1168	1189	1071	1137
전송된 데이터(byte)	456	401	468	433	427	437
전송된 헤더(byte)	672	728	700	756	644	700
전송된 HC 헤더(byte)	497	571	517	599	484	534
기존 UDP/IP오버헤드율(%)	147	182	150	175	151	161
HC UDP/IP 오버헤드율(%)	109	142	110	138	113	123

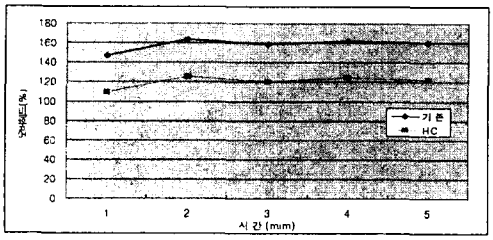

그림 9. Model II 환경에서 채팅 서버의 오버헤드율(%)

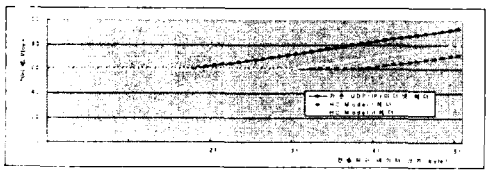

그림 10. Non-HC와 HC경우의 전송 헤더 크기 비교

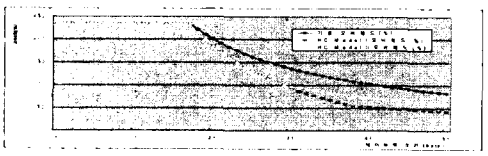

그림 11. Non-HC와 HC경우의 오버헤드율(%) 비교

V. 결 론

실시간 서비스에서 데이터의 크기가 작은 경우에는 붙는 헤더가 일정하기 때문에 상대적으로 오버헤드가 커지게 된다. 이런 이유에서 헤더 압축을 하게 되는데 유선 망에 비하여 대역폭이 상대적으로 비싼 무선망에서 활발히 연구되고 있다. 본 헤더 압축에 대한 연구에서는 무선망에 이용되는 헤더 압축을 이더넷기반의 유선 망에 맞게 헤더 필드들을 정의한 후, 두 가지 환경에 따라서 헤더 압축을 한 결과 헤더의 크기를 10~20byte정도 감소시킬 수 있으며, 오버헤드를 데이터의 40~50% 정도 감소시키는 결과를 보였다. 작은 데이터가 빈번하게 발생하는 경우에 사용하게 된다면 매우 효율적인 기술이 될 것이다. 무선망에서의 연구와는 유선 망과 이더넷이라는 환경 적인 요소로 인하여 효율적인 면에서 차이가 있지만 좀 더 연구를 하여 지금 까지 구현한 헤더 압축 시스템을 바탕으로 헤더 압축 범위를 RTP프로토콜까지 확장한다면 무선망에서의 연구와 비슷한 효율을 얻을 수 있을 것이라 생각한다. 차후 연구 방향으로는 IP sec을 고려하여 압축하는 것과 헤더 압축 범위를 RTP프로토콜로 확장을 하여 RTP/UDP/IP를 실시간 서비스에 동작하도록 구현하는 것이다. RTP/UDP/IP 헤더 압축과 보안을 통한 실시간 음성 서비스를 하게 된다면 다른 실시간 서비스관련 응용 프로그램의 개발하는데 있어서 기초 및 연구자료가 될 수 있다.

참고문헌

[1] W. Doeringer, D. Dykeman, M. Kaiserwerth, B. Meister, H. Rudin, and R. Willianmson, "A Survey of Light-Weight Transport Protocols for High-Speed Network," *IEEE Trans. on Communication*, vol.38, no.11, pp.2025-2039, Nov 1990.

[2] T. Porta and M. schwartz, "Architecture, Features, and Implementation of High-Speed Transport Protocols," *IEEE Network Magazine*, pp.14-22, May 1991.

[3] A. Giovanardi, G. Mazzini, M. Rossi and M. Zorzi, "Improved Header Compression for TCP/IP Over Wireless Links," *Electronics letters*, vol.36, no.23, pp.1958-1959, Nov 2000.

[4] G. Boggia, P. Camarda and V.G. Squeo, "ROHC+: a new header compression scheme for TCP streams in 3G wireless systems," *in Proc IEEE Int. conf. Communications*, vol.5, pp. 3271-3278, 2002.

[5] 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.

[6] A. Kondoz, A. Sadka, S. Worrall, S. Fabri and A. Cellatoglu , "Robust header compression for real-time services in cellular networks," *in Proc 3G Mobile Communication Technologies Int. conf.*, pp. 124-128, 2001.

[7] W.S. Filippo, M.W Ritter, R.J Friday and A. Srivastava, "A study of TCP performance over

wireless data networks," *in Proc IEEE Int. conf.* VTC, vol.3, pp.2265-2269, 2001.

[8] K Svanbro, L.-E. Jonsson, H. Hannu and L.-A. Larzon, "Efficient transport of voice over IP over cellular links," *in Proc IEEE Int. conf. GLOBECOM,* vol.3, pp.1669-1676, 2000.

[9] 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.

[10] M. Degermark, L.-E. Jonsson, H. Hannu and K. Svanbro, "Wireless real-time IP services enabled by header compression," *in Proc IEEE Int. conf. VTC,* vol.2, pp.1150-1154, 2000.

[11] TSG-SA-WG2, TSG-GERAN and TSG-RAN, "Header Compression for Optimized Voice Bearers," *Joint Meeting, 3GPP,* Aug 2001.

[12] C. Hornig, "A Standard for the Transmission of IP Datagrams over Ethernet Networks," *IETF RFC 894,* April 1984.

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

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

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

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

[17] Price et. al., "Efficient Protocol Independent Compression," *IETF Internet Draft,* Feb 2001.