효율적인 데이터 통신을 위한 Sliding Window 크기에 따른 패킷 헤더 압축 성능 개선 방안

차혜진*, 김강석*, 홍만표*
*아주대학교 대학원 지식정보공학과
e-mail : {wls8668, kangskim, mphong}@ajou.ac.kr

Performance of Packet Header Compression with Sliding Window Size for Efficient Data Communication

Hyejin Cha*, Kangseok Kim*, Manpyo Hong*
*Dept. of Knowledge Information Engineering, Graduate School of Ajou University

요 약

패킷 네트워크에서 음성, 영상 트래픽이 IP, UDP, RTP를 이용하여 전송 될 때 중복 헤더가 사용되어 통신의 비효율을 초래한다. 이를 방지 하기 위해 ROHC(Robust Header Compression)가 적용된다. 이는 인접 패킷 헤더 간의 차이 값이 규칙적으로 증가하는 영역은 그 차이 값만을 전송한다. 차이 값은 WLSB 인코딩 과정을 거쳐 전송이 되는데 인코딩 시 Sliding Window 값이 사용된다. Sliding Window 크기에 따라 헤더 압축률이 변하고 대역폭에도 영향을 미치게 된다. 본 논문에서는 효과적인 Sliding Window 값을 구하여 기존 ROHC 에 적용 하여 네트워크 통신의 효율성을 향상 시키고자 한다.

1. 서 론

네트워크 기술 향상과 인터넷의 발전으로 IP (Internet Protocol)를 이용한 데이터, 음성, 영상 등의 서비스를 패킷 네트워크를 이용하여 제공 할 수 있게 되었다. 특히 실시간 서비스라는 특징을 가진 음성과 영상 트래픽은 주로 UDP(User Datagram Protocol), RTP(Real Time Protocol)를 이용하여 전송이 이루어진다. 이때 IP 헤더의 크기는 20바이트이고 UDP 헤더의 크기가 8바이트, RTP 헤더 크기가 12바이트로 헤더 크기 합만으로도 40바이트가 된다.

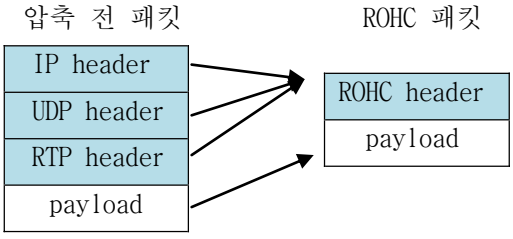
특히 VoIP의 경우 음성 페이로드(Payload)크기가 33바이트에 그치는 데 반해 헤더 크기가 40바이트 이상을 차지하게 되므로 무선 네트워크에서 대역폭 낭비를 초래하게 된다[1]. 이러한 비효율적인 헤더 사용을 줄이기 위해 헤더를 압축하는 여러 기법들이 제시 되었다.

그 중 IETF(Internet Engineering Task Force)가 채택한 RFC 3095의 ROHC(Robust Header Compression) 기법이 있다[2].

ROHC는 압축기에서 복원기로 (그림1)과 같이 서로 다른 헤더 간의 중복되는 정보는 제거하고 헤더 간 차이 값이 일정하게 증가하는 영역인 Dynamic 헤더 필드만을 보낸다. 전송을 할 때 압축기는 전체 비트를 보내는 것이 아니라 그 값을 식별 가능하게 하는 비트만을 보내어 헤더 크기를 줄인다.

복원기에서 원래의 값으로 복원 할 때도 원본을 식별 가능하게 하는 비트 값이 사용된다. 그 값을 구할 때 Sliding Window가 사용된다.

본 논문은 기존 헤더 압축 방식인 ROHC기법에서 인코딩 시 사용되는 Sliding Window를 패킷 손실에 따라 조절하여 압축률과 복원 실패율 두 측면에서 효과적인 결과를 보이고자 한다.



(그림1) ROHC의 주요 원리

본 연구는 미래창조과학부 및 한국인터넷진흥원의 "고용계약형 지식정보보안 석사과정 지원사업"의 연구결과로 수행되었음 (과제번호 H2101-13-1001)

2. 관련연구

2.1 ROHC 인코딩 방식 - Sliding Window

ROHC는 Dynamic 헤더 필드의 높은 압축률을 위해 기존의 LSB(Least Significant bits) 방식을 개선한 WLSB(Window Least Significant Bit)인코딩 방식을 사용한다.

LSB방식은 헤더의 중복된 값을 제외하고 최하위 비트인 k개의 비트만을 전송한다.

WLSB방식은 여러 개의 패킷이 손실 되는 경우에도 복원이 가능하도록 연속적인 값이 존재하는 Sliding Window를 사용한다. 즉 패킷 손실이 Sliding Window 값을 초과하지 않는 한 복원 실패가 일어나지 않는다[3].

압축기는 빈 상태의 Sliding Window로 시작한다. 그 후 압축에 성공하여 복원기로 전송된 value가 Sliding Window에 쌓이게 된다. 이때 Sliding Window의 최소값을 $vref_{\min}$, 최대값을 $vref_{\max}$ 이라 한다. 압축기는 아래의 공식을 이용하여 복원기로 전송하는 값 v를 식별 가능하게 하는 k값을 구하게 된다[4].

$$k = \max(g(vref_{\min}, v), g(vref_{\max}, v)) \quad (1)$$
$$= \max(floor(\log_2(v \ XOR \ v_{\min})) + 1,$$
$$floor(\log_2(v \ XOR \ v_{\max})) + 1)$$

복원기에서는 압축기가 전송한 값 중 복원에 성공한 마지막 값을 이용하여 interpretation interval 범위 내에서 헤더 필드의 실질적 값을 복원하게 된다. interpretation interval은 함수 f(vref, k)로 정의되며 다음과 같다[5].

$$f(vref, k) = [vref, vref + 2^k - 1] \quad (2)$$

예를 들면 다음과 같다. 압축기에서 reference value가 $11000_2, 11001_2, 11010_2, 11011_2$인 경우 Sliding Window값은 4가 된다. 이때 11100_2 를 복원기에게 전송하는 경우 식(1)를 통해 k는 3비트가 되고 최하위 3비트에 해당하는 100_2 만 전송된다.

복원기 측은 reference value로 11000_2 이 존재하고 11001_2 , 11010_2 , 11011_2 값은 손실된 상황을 가정한다. reference value 는 11000_2 이고 k=3일 때 수식(2)에 의하여 interpretation interval이 24(11000_2)부터 31(11111_2)사이의 값이 된다. 이 범위에서 k=3 (100_2)과 같은 최하위 비트 3개에 대해서 100_2 값을 갖는 값으로 28(11100_2)을 찾게 되고 복원에 성공하게 된다.

2.2 헤더 압축 복원 실패 시 복구 기법 - PHR

헤더 압축 시 압축기와 복원기는 정보를 공유하고 동기화되어야 한다. 네트워크 혼잡으로 인해 패킷 손실이 발생하면 동기화에 실패하고 기존에 보내진 패킷 들이 폐기되어 복원에 실패하게 된다. 이런 경우 복구를 위해 Periodic Header Refresh(PHR)기법이 사용된다[6]. 이는 주기적으로 Full Header를 보내어 패킷 손실 상황에서도 복원에 필요한 정보를 얻을 수 있도록 한다.

2.3 Adaptive Adjustment of Sliding Window

이 기법은 압축과 복원 시 네트워크 상황을 반영하여 Sliding Window을 조절한다.

한 세션 동안 계속 동일한 값을 유지하는 STATIC 헤더 필드의 복원이 실패하면 STATIC NACK피드백이 압축기로 보내진다.

변화 규칙이 일정한 동적 필드인 Dynamic 헤더 필드의 복원이 이루어지지 않으면 NACK피드백이 압축기로 보내진다. 이 피드백 메시지를 이용하여 Sliding Window을 조절한다[3].

STATIC NACK, NACK피드백을 받은 경우는 연속적인 패킷 손실을 뜻하므로 Sliding Window값을 증가 시킨다. 반대로 일정 시간 동안 STATIC NACK, NACK피드백이 없는 경우는 Sliding Window값을 감소 시킨다.

3 개선 방안

PHR기법은 패킷의 손실을 줄일 수 있는 장점이 있으나 대역폭 사용의 증가라는 단점이 있다.

Adaptive Adjustment of Sliding Window 기법은 네트워크 상황에 따라 조절하여 효율적이지만 Sliding Window값을 감소시키는 기준이 되는 일정시간에 불명확성이 존재하였다.

본 연구는 Adaptive Adjustment of Sliding Window기법의 장점을 받아들이고 개선 시키고자 한다.

실험의 가정은 Sliding Window값을 작게 설정하면 헤더 크기를 더 작게 압축 하는 것이 가능하나 복원 실패율은 커질 것이라는 것이다.

이를 위해 Sliding Window값을 유동적으로 변화시키고 그에 따른 압축률의 증가와 복원 실패율의 감소라는 효과를 얻고자 한다.

Sliding Window값을 초과하는 패킷 손실이 있을 시 Sliding Window값을 증가시키고자 한다. 패킷 손실의 측정을 위해 Biaz알고리즘을 이용한다[7]. $T_{\min}$ 은 패킷이 도착하는 시간 간격 중 최소값이다. x개 패킷의 손실을 가정 할 때 $packet_i$ 는 정상적으로 받은 마지막 패킷이고 $packet_{i+x+1}$ 은 패킷의 손실이 발생 한 이후에 받기 시작한 비순차적 패킷이다. 이때 $packet_i$ 와 $packet_{i+x+1}$ 의 도착 간격 시간은 T_i 이다.

$$(x+1)T_{\min} \le T_i < (x+2)T_{\min} \quad (3)$$

수식(3)을 만족하면 무선 전송 에러로 x개의 패킷이 손실되었음을 뜻한다[7]. 에러로 손실된 패킷의 수 x가 Sliding Window값보다 큰 경우 Sliding Window값을 증가 시킨다.

복원기가 압축기로 부터 ACK피드백을 받은 경우는 헤더 복구에 대한 정보를 충분히 전달 받은 경우로 이런 경우에 Sliding Window 값을 감소시킨다.

제안 방식의 실험을 위해 시뮬레이터와 Launchpad.net의 ROHC프로젝트의 공개 ROHC 라이브러리를 이용 한다. 라이브러리를 통해 압축된 ROHC헤더와 페이로드로 구성된 PCAP파일을 얻게 된다. 이를 분석하여 압축률과 복원실패율, 대역폭을 구하는데 사용한다.

압축률은 다음 수식으로 구한다.

$$(1-\frac{RH+P}{H+P})*100 \qquad (4)$$

여기서 H 는 압축 전 헤더 크기이고, RH는 압축 후에 ROHC 헤더 크기 P는 페이로드의 크기이다.

복원 실패율은 다음과 같다.

$$(\frac{FPNUM}{PNUM})*100 \qquad (5)$$

FPNUM은 복원에 실패한 패킷의 수이고 PNUM은 전체 패킷의 수이다. 감소된 헤더 크기를 이용하여 아래 수식을 이용하여 대역폭을 구한다[8].

$$\frac{(PAYLOAD\ SIZE + AVERAGE\ \ HEADER\ SIZE)}{SAMPLE\_PERIOD} \qquad (6)$$

수식(6)의 AVERAGE HEADER SIZE는 전송한 전체 헤더 사이즈를 전송한 패킷의 수로 나눈 것이다.

4. 결론

본 연구의 가정은 Sliding Window값이 작으면 헤더 크기를 더 작게 압축 하는 것이 가능하다는 것이다 즉 이는 압축률의 증가를 뜻한다. 또 하나의 가정은 Sliding Window값이 작으면 복원 실패율은 커질 것이라는 것이다.

이러한 가정을 바탕으로 네트워크 상황에 따라 Sliding Window을 조절하여 압축률과 복원실패율 두 인자에서 효과적인 결과를 얻고자 한다.

향 후 실험을 통해 얻은 데이터를 이용하여 Sliding Window값과 헤더크기 값은 비례관계를 가지고, Sliding Window값과 복원실패율은 반비례 관계를 가짐을 증명하려 한다.

또한 개선된 방안으로 대역폭 사용의 감소를 보이고자 한다.

참 고 문 헌

[1] Roshni Srinivasan, Jeff Zhuang, Motorola, "Draft IEEE 802.16m Evaluation Methodology Document", IEEE 802.16m-07, 2007

[2] C.Borman, "Robust Header Compresion", IETF,RFC 3095, 2001.

[3] Kazem Sohraby, Chonggang Wang, “Performance of Packet Header Compression And Context Replication in Wireless/Mobile Systems”, Military Communicatio ns Conference, 2006

[4] Ana Raquel Silva Faria,"Robust Header Compression over IEEE 802 Networks", 2009

[5] Mohammed Al-Obaidi, Harshavardhan Kittur, "Hardware Acceleration of the Robust Header Compression Algorithm", in Circuits and Systems IEE E International Symposium, 2012

[6] 신병철, 백봉, 김민영, 홍고르출, "IPV6기반 헤더 압축 에러 복구 기법의 성능 분석", 컴퓨터 정보통신연구지, 2006

[7] 류성윤,"다중흐름에서 혼잡 손실과 무선 손실의 구분에 관한 연구", 인하대학교 석사 논문, 2005

[8] 홍진우, 장원갑, "All-IP 환경에서의 RTP헤더 압축 및 다중화 기법", 대한 전자 공학회 하계종합학술대회 논문집, 2002