대용량 네트워크에서의 HTTP 파싱 기법 연구

유경민, 박민호*
숭실대학교

Yoojkm38@gmail.com, *mhp@ssu.ac.kr

A Study on HTTP Parsing Techniques in Large-Capacity Network

Yoo Kyeong Min, Pack Min Ho*
Soongsil Univ.

요 약

전세계 네트워크 트래픽은 점차 증가하는 추세이며 요구되는 연산량도 급격하게 증가한다. 이에 따라 소프트웨어적 성능을 높이는 방안으로 고성능의 패킷 파서 알고리즘이 요구된다. 그러나 기존의 패킷 파서는 패킷 원본을 버퍼에 저장함에 따라 처리 속도와 메모리 효율이 떨어진다는 문제점이 존재한다. 이를 개선하기 위하여 본 논문에서는 패킷의 HTTP 식별을 빠르게 수행하고 최소한의 버퍼를 사용하기 위해 한 바이트씩 입력받아 처리하는 HTTP 파싱 알고리즘을 구현하였다. 또한, 실험을 통해 처리량에 따른 처리 속도, 메모리 효율의 향상을 확인하였다.

Ⅰ. 서 론

IDC(Internet Data Center)나 통신사 등에서 운영하는 네트워크는 성능 모니터링, 방화벽, 라우팅과 스위칭 등의 네트워크 기능 지원을 위해 여러 종류의 패킷 처리 작업을 수행한다. 이러한 패킷 처리에는 패킷 파싱, 라우팅 및 스위칭, 혼잡 제어, 세션 관리, 보안 검사 등이 수행된다. 이중 패킷 파싱은 패킷에서 네트워크 프로토콜 헤더들을 분리해 네트워크 기능에서 사용할 정보들을 추출하는 작업으로, 패킷 처리 과정 중 가장 우선으로 처리되는 과정이다[1].

전 세계의 네트워크 트래픽 총량은 점증적으로 증가하는 추세이며, 차세대 고성능 네트워크들은 400Gbps 등의 높은 대역폭을 요구하고 있다. 특히 스트리밍, SNS 등의 사용률이 높아짐에 따라 웹 트래픽의 총량은 지속해서 증가하고 있다. 이에 따라 패킷 처리 작업에 필요한 연산량 또한 급격하게 증가하고 있으며, 요구되는 하드웨어, 소프트웨어적 성능도 점차 증가하고 있다. 이러한 상황 속에서 고성능 요구에 맞추기 위해 알고리즘 디자인과 하드웨어 기반의 연구가 진행되고 있다[2].

본 논문에서는 차세대 대용량 네트워크의 고성능 처리를 위한 HTTP 파서 디자인을 제안하고, 기존 알고리즘 대비 성능 확인을 수행하였다.

Ⅱ. 관련 연구

HTTP 대용량 트래픽 처리를 위한 파서에 관한 기존 연구는 다음과 같다[3].

1. strcmp 접근법: 각 HTTP 헤더 라인을 순회하며 memchr, memmem, strncmp과 같은 표준 문자열 함수를 사용하여 특정 필드를 검색한다.

2. PCRE(Perl Compatible Requlars Expressions) 접근법 : Perl 호환 정규 표현식을 사용하여 HTTP 헤더 필드에서 특정 패턴을 식별한다.

3. Flex(Fast lexical analyzer genrator) 접근법 : 각 패킷을 특정한 규칙에 따라서 긴 문자열로 처리한다. 필요한 HTTP 필드를 식별하기 위해 처리된 문자열을 유한 오토마톤을 사용하여 토큰화한다.

기존의 HTTP 파서는 파싱을 위해 원본 패킷을 버퍼에 저장하는 과정이 수행된다. 이는 패킷이 분할되는 경우 패킷의 일부만으로는 구문에 대한 분석이 불가능하기에 수행되는 과정이다[4]. 이는 후속 패킷이 도달하기 전까지 대기 시간을 가지므로 처리 속도가 저하되는 문제점이 존재한다. 또한, 버퍼를 지속적으로 사용함에 따라 트래픽이 증가할수록 메모리 효율이 낮아지는 문제점이 존재한다.

Ⅲ. 제안기법

제안 방안은 원본 패킷을 저장하는 과정을 제거하기 위해 유한 오토마톤을 사용하여 처리 상태를 저장해가며 구문을 즉시 처리한다. 구문은 문자 단위로 처리되며 공백을 기준으로 각 항목을 식별한다. 처리 상태의 변화는 그림 3-1과 같다.

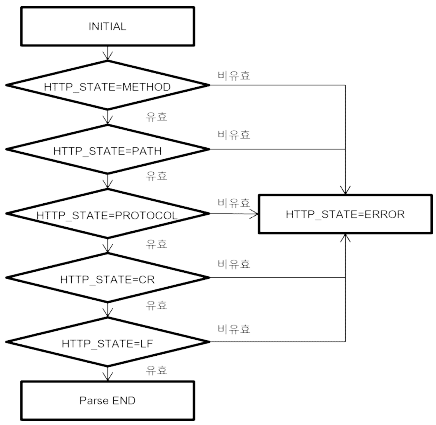

그림 3-1 처리 상태 변화

패킷 진입 시 INITIAL 상태에 돌입하며 초기화 작업을 수행한다. 초기화 작업이 종료되면 HTTP_STATE를 변경하고 구문 처리를 수행한다. 구문 처리 중 공백이 등장하면 처리 상태에 해당하는 구문에 대한 유효성을 확인한다. 유효하지 않으면 HTTP가 아니라 판단하고 HTTP_STATE를 ERROR로 변경하고 No HTTP를 리턴한다. 유효한 경우 필요 정보를 저장하고 다음 처리 상태에 진입 후 다음 문자 처리를 수행한다. 구문의 마지막에서 CRLF 등장 시 식별 결과를 리턴하며 파싱을 종료한다. 각 처리 상태 진입 시 동작은 그림 3-2와 같다.

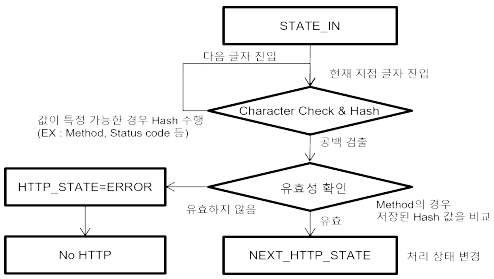

그림 3-2 처리 상태 진입 시 동작

처리 상태에 진입하면 구문을 문자 단위로 처리한다. 처리는 처리 상태에 따라 동작이 결정되며 Method, Protocol 등 패턴이 특정 가능한 경우를 제외하고 공백 식별만을 수행한다. Method와 같은 패턴 특정이 가능한 경우 해싱을 수행하여 처리 값을 저장한다. 문자가 처리될 때마다 저장된 해시값에 곱 연산을 수행하여 새로운 해시값을 추출한 뒤, 이를 버퍼에 저장한다. 공백이 등장하면 해싱된 값에 해당하는 Method를 식별하여 유효성을 확인한다. 해싱된 값에 해당하는 Method가 존재하는 경우 다음 처리 상태로 변경하며 없는 경우 HTTP가 아니라 판단한다. Protocol의 경우 HTTP/1.0과 HTTP/1.1만을 식별하면 되므로 HTTP/1.의 등장 후 0, 1이 다음으로 오는지를 식별한다.

이처럼 패킷을 저장하는 과정 없이 즉시 처리에 진입함으로써 HTTP의 빠른 식별이 가능하다. 식별된 값은 구현하기에 따라 특정 값 추출로 이용할 수 있다. 또한, 패킷 원본이 아닌 처리 상태, Method의 해시값 등 필요 최소한의 정보만을 저장하므로 사용되는 버퍼를 줄일 수 있다. 명세되 있는 항목 외에 추가 식별 조건이 필요한 경우 HTTP_STATE를 추가하는 것으로 파싱을 수행할 수 있다.

Ⅳ. 실험 결과

시뮬레이션은 각 알고리즘을 C 코드로 구현하고 임의의 HTTP 메시지를 일정 횟수 입력하여 HTTP 파싱에 소요되는 시간을 측정하였다. 그림 4-1은 Flex 접근법과 제안 알고리즘의 구문 처리 속도에 대한 시뮬레이션을 수행한 결과이다. 시뮬레이션 결과 처리량이 적을 때 두 알고리즘의 처리 속도가 유사하였으나 처리량이 증가함에 따라 소요 시간에 격차가 급격히 증가하였다.

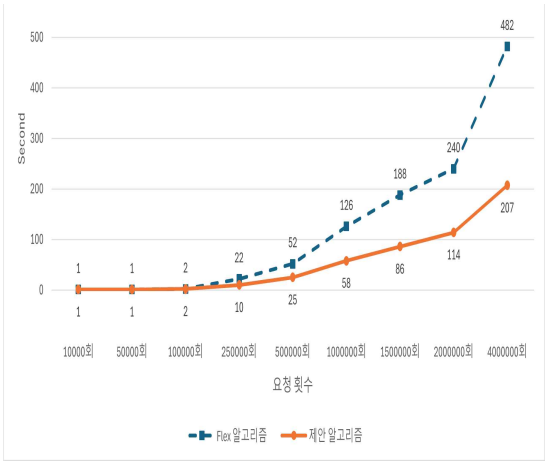

그림 4-1 구문 처리 속도 시뮬레이션 결과

Ⅴ. 결론

본 논문에서는 HTTP 파싱 중 버퍼 저장 과정 발생에 따른 처리 지연 및 메모리 효율 감소 문제를 확인하였다. 이를 해결하기 위해 원본 패킷을 저장하지 않고 패킷 유입 즉시 처리하여 처리된 정보 중 최소한의 정보만을 저장하는 방안을 제안하였다. 제안 방안의 성능을 확인하기 위하여 제안 방안과 기존 알고리즘의 처리 성능 비교 결과, 처리 용량이 증가할수록 기존 알고리즘 대비 처리 속도가 향상됨을 확인하였다.

참 고 문 헌

[1] Dongryeong Kim, & Jangwoo Kim (2021). A High Performance Packet Parser Design for Packet Processing Accelerator. 한국정보과학회 학술발표논문집, 개최지.

[2] Young-Sun Kim, & Hyuk-Chul Kwon (2021). Implementation of Information Retrieval System by Table-parsing. 한국멀티미디어학회 학술발표논문집, 개최지.

[3] Petr Velan, Tomáš Jirsík & Pavel Čeleda. Design and Evaluation of HTTP Protocol Parsers for IPFIX Measurement. 2013 IFIP International Federation for Information Processing

[4] Sang-Jo Youk, Myung-Ho Park, & Geuk Lee (2003). IDS Evasion Detection System with Packet Reassemble Funtion. Journal of Digital Contents Society, 4(1), 101-113.