



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2010-0003442
(43) 공개일자 2010년01월11일

(51) Int. Cl.

H04L 12/54 (2006.01)

(21) 출원번호 10-2008-0063342

(22) 출원일자 2008년07월01일

심사청구일자 2008년07월01일

(71) 출원인

충남대학교산학협력단

대전광역시 유성구 공동 220번지 충남대학교

(72) 발명자

이영석

대전 유성구 관평동 대덕테크노밸리 809-602

김기수

대전 동구 가양2동 149-9 삼덕빌라 가 101호

(뒷면에 계속)

(74) 대리인

김원준

전체 청구항 수 : 총 5 항

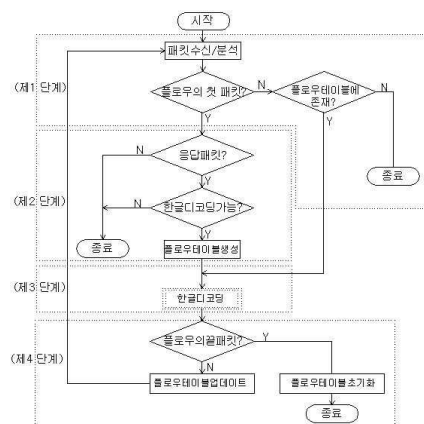
(54) HTTP 패킷에서 한글 디코딩 방법 및 장치

(57) 요약

본 발명은, (A) 수신된 패킷의 헤더를 분석하여 상기 패킷이 플로우의 첫 패킷인지를 확인하여 ① 첫 패킷이 아니면 상기 패킷에 대한 플로우테이블이 존재하지 않으면 종료하고, 존재하면 하기 제3단계로 이동하며, ② 첫 패킷인 경우 하기 제2단계로 이동하는 제1단계(플로우분석단계); (B) 상기 패킷의 HTTP 헤더 또는 바디의 정보를 참조하여 상기 패킷이 HTTP 응답패킷이면서 동시에 2바이트 한글 디코딩이 가능한 패킷이면 플로우테이블을 생성한 후 하기 제3단계로 이동하고, HTTP 응답패킷이 아니거나 2바이트 한글 디코딩이 불가능한 패킷이면 종료하는 제2단계(적합성분석단계); (C) 상기 패킷의 HTTP 바디의 스트링으로부터 한글을 디코딩하는 제3단계(디코딩단계); 및 (D) 상기 패킷이 플로우의 마지막 패킷인 경우 플로우테이블을 초기화하고 종료하며, 마지막 패킷이 아닌 경우 플로우테이블을 업데이트하고 상기 1단계로 이동하는 제4단계(플로우테이블관리단계);를 포함하여 이루어지는 HTTP 패킷에서 한글 디코딩 방법 및 이를 위한 장치에 관한 것이다

본 발명에 의해, HTTP 패킷에 인코딩되어 있는 한글 문자열을 탐지하여 해독할 때 한글 완성형 코드 값과 비교하여 그 값에 해당하는 한글 문자를 출력하는 것이 가능하게 된다. 또한 본 발명에 의해, 탐지된 한글 문자열 정보를 이용할 수 있게 되어 HTTP 트래픽의 상세한 분석을 가능하게 한다.

대표도 - 도1



(72) 발명자

손현구

대전 유성구 궁동 474-7 308호

권택근

대전 유성구 신성동 대림두레아파트 106-902

이 발명을 지원한 국가연구개발사업

과제고유번호 C1090-0801-0016 / A1100-0801-2758

부처명 정보통신연구진흥원 / 지식경제부

연구사업명 대학 IT연구센터 육성 지원사업 / IT원천기술개발사업

연구과제명 해킹·바이러스 대응기술연구 / 초정밀 측정 및분석기술연구

주관기관 충남대학교산학협력단

연구기간 2003. 8. 1~2011. 12. 31 / 2008. 3. 1~2009. 2. 28

특허청구의 범위

청구항 1

플로우정보 등이 저장되는 플로우테이블이 개재된 한글 디코딩 방법에 있어서,

- (A) 수신된 패킷의 헤더를 분석하여 상기 패킷이 플로우의 첫 패킷인지를 확인하여 ① 첫 패킷이 아니면서 상기 패킷에 대한 플로우테이블이 존재하지 않으면 종료하고, 존재하면 하기 제3단계로 이동하며, ② 첫 패킷인 경우 하기 제2단계로 이동하는 제1단계(플로우분석단계);
- (B) 상기 패킷의 HTTP 헤더 또는 바디의 정보를 참조하여 상기 패킷이 HTTP 응답패킷이면서 동시에 2바이트 한글 디코딩이 가능한 패킷이면 플로우테이블을 생성한 후 하기 제3단계로 이동하고, HTTP 응답패킷이 아니거나 2바이트 한글 디코딩이 불가능한 패킷이면 종료하는 제2단계(적합성분석단계);
- (C) 상기 패킷의 HTTP 바디의 스트링으로부터 한글을 디코딩하는 제3단계(디코딩단계); 및
- (D) 상기 패킷이 플로우의 마지막 패킷인 경우 플로우테이블을 초기화하고 종료하며, 마지막 패킷이 아닌 경우 플로우테이블을 업데이트하고 상기 1단계로 이동하는 제4단계(플로우테이블관리단계);를 포함하여 이루어지는 것을 특징으로 하는 HTTP 패킷에서 한글 디코딩 방법.

청구항 2

제 1 항에 있어서,

상기 플로우테이블의 필드는 플로우를 식별할 수 있는 정보 및 삭제예약플래그, 플로우유지시간 및 1바이트 저장공간을 포함하는 것을 특징으로 하는 HTTP 패킷에서 한글 디코딩 방법.

청구항 3

제 1 항 또는 제 2 항에 있어서,

상기 제2단계(적합성분석단계)에서,

- (a) ① HTTP 헤더에 "HTTP/1.1 200 OK" 스트링이 존재하고, ② "Content-Encoding:" 스트링은 존재하지 않음;
- (b) ① HTTP 헤더에 "Content-Type: text" 및 HTTP body에 "<?xml="1.0" encoding="이 모두 존재하거나
- ② HTTP 헤더에 "Content-Type: text" 및 "charset="이 모두 존재하거나
- ③ HTTP 헤더에 "Content-Type: application" 및 HTTP body에 "<?xml="1.0" encoding="이 모두 존재하거나
- ④ HTTP 헤더에 "Content-Type: application" 및 "charset="이 모두 존재함;

의 경우에 플로우테이블을 생성하고 제3단계로 이동하는 것을 특징으로 하는 HTTP 패킷에서 한글 디코딩 방법.

청구항 4

제 1 항 또는 제 2 항에 있어서,

상기 제3단계(디코딩단계) 및 제4단계(플로우테이블관리단계)에서,

- (a) 상기 패킷(n)이 플로우의 마지막 패킷이 아닌 경우 한글 디코딩을 수행하고 마지막 1바이트가 잔류하는 경우 이를 플로우테이블의 1바이트 저장공간에 임시로 저장하고,
- (b) 다음 패킷(n+1) 처리시 상기 임시저장된 1바이트 정보를 다음 패킷(n+1)의 바디 전단에 첨부한 후 디코딩하는 것을 특징으로 하는 HTTP 패킷에서 한글 디코딩 방법.

청구항 5

제 1 항에 의한 방법을 수행하기 위한 장치로서,

- (A) 수신된 패킷의 IP헤더를 분석하여 단편화된 패킷인지를 확인하는 헤더분석부;
- (B) 상기 HTTP 패킷헤더와 바디의 일부정보를 참조하여 HTTP 응답패킷인지, 한글 디코딩이 가능한 패킷인지를

판단하는 적합성분석부;

(C) 한글 디코딩이 가능한 패킷의 바디부분을 파싱을 통하여 디코딩하는 디코딩처리부; 및

(D) 플로우정보를 생성, 관리하는 플로우테이블관리부;를 포함하는 것을 특징으로 하는 HTTP 패킷의 한글 디코딩 장치.

명세서

발명의 상세한 설명

기술 분야

<1> 본 발명은 트래픽 모니터링 장비에서 HTTP 트래픽 분석을 수행할 때 수신된 HTTP 패킷 페이로드 내에 한글이 인코딩되어 있는지를 탐지하고 이를 해독하는 방법 및 장치에 관한 것이다.

배경 기술

<2> 현재 HTTP에서 쓰이는 한글 인코딩 방법은 한글완성형코드(KS완성형 표준한글코드)를 이용한 인코딩, EUC-KR, UTF-8, UTF-16로 구분할 수 있다.

<3> 한글완성형코드는 1987년 우리나라 국가표준코드로 지정된 코드로 2바이트 완성형 코드이며 2350자의 한글을 지원한다(KS C 5601). EUC-KR은 Bell Lab.에서 유닉스상에서 영문자 이외의 문자를 지원하기 위해 제안한 확장 유닉스 코드(Extended UNIX code)중 한글 인코딩 방식으로서 영문은 KS C 5636(새 이름은 KS × 1003)으로 처리하고 한글은 KS C 5601(새 이름은 KS × 1001)로 처리한다. 여기서 KSC5636은 영문자에 대한 표준으로서 한국공업표준 정보처리분야(C)의 5636번 표준안을 말하며 기존 ASCII Code에서 역슬래쉬(\)를 원(W) 표시로 대체한 코드이다. 즉, EUC-KR은 KS C 5601과 KS C 5636을 합친 코드를 사용하는 8비트 문자 인코딩이다.

<4> UTF-8과 UTF-16은 유니코드를 위한 가변길이 문자 인코딩 방식의 하나이다(ISO/IEC 10646). UTF-8인코딩은 유니코드 한 문자를 나타내기 위해 1바이트에서 4바이트까지를 사용한다. 예를 들면 U+0000부터 U+007F 범위에 있는 ASCII 문자들은 UTF-8에서 1바이트만으로 표시된다. 마찬가지로 U+0080부터 U+07FF까지는 2바이트이며, U+0800부터 U+FFFF사이에 들어가는 한글은 3바이트로 인코딩된다. UTF-16은 기본 다국어 평면(BMP, Basic Multilingual Plane)에 속하는 문자들은 그대로 16비트 값으로 인코딩하고 그 이상의 문자는 특별히 정해진 방식으로 32비트로 인코딩하는 방식이다.

<5> 인터넷으로 대표되는 네트워크의 활성화에 따라 네트워크의 속성과 특징을 정확히 이해하고, 각종 네트워크상에서 발생하는 문제(트래픽 문제, 보안문제 등)의 원인을 명확히 밝히고 해결하기 위하여 트래픽분석 또는 패킷분석이 이루어지고 있다.

<6> 현재 인터넷 상에서는 패킷분석을 쉽고도 정확하게 해주는 다양한 툴(Ethereal, Wireshark, Sniffer 등)들이 존재한다. 그러나 이러한 툴들은 패킷의 페이로드의 내용을 ASCII코드 문자열로 보여주는 것에 그치고 있어 한글로 된 문자열이 HTTP 패킷에 포함되어 있을 경우 그 내용을 알 수 없다. 특히 한국의 경우 인터넷상의 정보는 대부분 한글로 인코딩된 웹 콘텐츠이기 때문에, 인터넷 트래픽의 내용을 정밀하게 분석하기 위해서는 ASCII 코드가 아닌 한글 문자열로 자동으로 인식하고 해독하는 것의 필요성이 증대되고 있다.

발명의 내용

해결 하고자하는 과제

<7> 본 발명은 HTTP 패킷에서 한글이 인코딩되어 있는지를 확인하고, 인코딩되어 있는 경우 이를 한글 문자열로 디코딩하여 표현(출력)할 수 있는 방법을 제공하는 것을 목적으로 한다.

<8> 또한 본 발명은 상기 한글 디코딩을 수행하는 장치를 제공하는 것을 목적으로 한다.

과제 해결수단

<9> 전술한 과제를 해결하기 위한 본 발명은, (A) 수신된 패킷의 헤더를 분석하여 상기 패킷이 플로우의 첫 패킷인지를 확인하여 ① 첫 패킷이 아니면서 상기 패킷에 대한 플로우테이블이 존재하지 않으면 종료하고, 존재하면 하기 제3단계로 이동하며, ② 첫 패킷인 경우 하기 제2단계로 이동하는 제1단계(플로우분석단계); (B) 상기 패

킷의 HTTP 헤더 또는 바디의 정보를 참조하여 상기 패킷이 HTTP 응답패킷이면서 동시에 2바이트 한글 디코딩이 가능한 패킷이면 플로우테이블을 생성한 후 하기 제3단계로 이동하고, HTTP 응답패킷이 아니거나 2바이트 한글 디코딩이 불가능한 패킷이면 종료하는 제2단계(적합성분석단계); (C) 상기 패킷의 HTTP 바디의 스트링으로부터 한글을 디코딩하는 제3단계(디코딩단계); 및 (D) 상기 패킷이 플로우의 마지막 패킷인 경우 플로우테이블을 초기화하고 종료하며, 마지막 패킷이 아닌 경우 플로우테이블을 업데이트하고 상기 1단계로 이동하는 제4단계(플로우테이블관리단계);를 포함하여 이루어지는 HTTP 패킷에서 한글 디코딩 방법에 관한 것이다.

- <10> 별도의 상세한 부가설명이 없더라도, 본 발명의 특성상 통신상의 하자 등에 의해 더 이상 수신 또는 분석할 패킷이 없는 경우 상기 한글 디코딩 방법의 수행이 중단됨은 당연하다.
- <11> 본 발명에서 상기 플로우테이블의 필드는 플로우를 식별할 수 있는 정보 및 삭제예약플래그, 플로우유지시간 및 1바이트 저장공간을 포함하는 것이 바람직하다. 만일 플로우유지시간 필드가 있는 경우, 후속 패킷이 소정의 플로우유지시간(예컨대 30초) 내에 수신되지 않으면 디코딩을 종료하도록 할 수 있을 것이다. 삭제예약플래그 필드가 있는 경우, 초기 플로우분석단계에서 패킷이 마지막 패킷인 경우(즉 패킷의 TCP 헤더에 FIN 플래그가 설정된 경우) 이를 삭제예약 플래그에 기록해 두고, 이 기록의 유무를 확인함으로써 패킷이 마지막 패킷인지를 확인하도록 할 수도 있다. 즉, 플로우테이블에 설정된 삭제예약 플래그가 있는지, 또는 소정의 플로우유지시간을 넘긴 플로우가 있는지를 체크하여 있다면 플로우테이블을 삭제(초기화)한다.
- <12> 본 발명의 상기 제2단계(적합성분석단계)에서, (a) ① HTTP 헤더에 "HTTP/1.1 200 OK" 스트링이 존재하고, ② "Content-Encoding:" 스트링은 존재하지 않음; (b) ① HTTP 헤더에 "Content-Type: text" 및 HTTP body에 "<?xml="1.0" encoding="이 모두 존재하거나 ② HTTP 헤더에 "Content-Type: text" 및 "charset="이 모두 존재하거나 ③ HTTP 헤더에 "Content-Type: application" 및 HTTP body에 "<?xml="1.0" encoding="이 모두 존재하거나 ④ HTTP 헤더에 "Content-Type: application" 및 "charset="이 모두 존재함;의 경우에 플로우테이블을 생성하고 제3단계로 이동하도록 할 수 있다.
- <13> 본 발명의 상기 제3단계(디코딩단계) 및 제4단계 (플로우테이블관리단계)에서, (a) 상기 패킷(n)이 플로우의 마지막 패킷이 아닌 경우 한글 디코딩을 수행하고 마지막 1바이트가 잔류하는 경우 이를 플로우테이블의 1바이트 저장공간에 임시로 저장하고, (b) 다음 패킷(n+1) 처리시 상기 임시저장된 1바이트 정보를 다음 패킷(n+1)의 바디 전단에 첨부한 후 디코딩하는 것을 특징으로 하는 HTTP 패킷에서 한글 디코딩 방법.
- <14> 또한 본 발명은 전술한 HTTP 패킷에서 한글 디코딩 방법을 수행하기 위한 장치로서, (A) 수신된 패킷의 IP헤더를 분석하여 단편화된 패킷인지를 확인하는 헤더분석부; (B) 상기 HTTP 패킷헤더와 바디의 일부정보를 참조하여 HTTP 응답패킷인지, 한글 디코딩이 가능한 패킷인지를 판단하는 적합성분석부; (C) 한글 디코딩이 가능한 패킷의 바디부분을 파싱을 통하여 디코딩하는 디코딩처리부; 및 (D) 플로우정보를 생성, 관리하는 플로우테이블관리부;를 포함하는 HTTP 패킷의 한글 디코딩 장치에 관한 것이다.

효 과

- <15> 본 발명에 의해, HTTP 패킷에 인코딩되어 있는 한글 문자열을 탐지하여 해독할 때 한글 완성형 코드 값과 비교하여 그 값에 해당하는 한글 문자를 출력하는 것이 가능하게 된다.
- <16> 또한 본 발명에 의해, 한글 문자열이 여러 개의 패킷에 걸쳐 있을 경우 동일한 출발지와 목적지 IP 주소/포트번호를 갖는 연속적인 HTTP 패킷에 대해서도 인코딩 정보가 없더라도 한글 문자열을 탐지하고 해독할 수 있도록 한다.
- <17> 또한 본 발명에 의해, 탐지된 한글 문자열 정보를 이용할 수 있게 되어 HTTP 트래픽의 상세한 분석을 가능하게 한다.

발명의 실시를 위한 구체적인 내용

- <18> 이하 첨부된 도면을 참조하여 본 발명을 보다 상세히 설명한다. 그러나 이러한 도면은 본 발명의 기술적 사상의 내용과 범위를 쉽게 설명하기 위한 예시일 뿐, 이에 의해 본 발명의 기술적 범위가 한정되거나 변경되는 것은 아니다. 또한 이러한 예시에 기초하여 본 발명의 기술적 사상의 범위 안에서 다양한 변형과 변경이 가능함은 당업자에게는 당연할 것이다.
- <19> 도 1은 본 발명에 따른 한글 디코딩 방법의 일예를 보여주는 전체 흐름도이다.
- <20> (A) 제1단계(플로우분석단계)에서는 수신한 TCP/IP 패킷이 여러 개의 HTTP 패킷들 중에서 이미 받은 HTTP 응답

패킷들과 관련된 나머지 패킷인지(즉, 플로우에 해당하는지)를 판단한다. [물론 플로우의 첫 패킷이면서 적합성(한글디코딩가능성)이 확인된 경우에는 플로우테이블이 생성된다.]

- <21> 플로우(flow)란 공통의 주소쌍(송신자 주소, 송신자 포트 번호, 수신자 주소, 수신자 포트), 호스트 쌍(송신자 호스트 주소, 수신자 호스트 주소), 네트워크 주소 쌍(송신자 네트워크 주소, 수신자 네트워크 주소), AS 번호 쌍(송신자 AS 번호, 수신자 AS 번호) 등으로 주어지는 조건을 만족시키는 제한된 시간 내에 도착하는 IP 패킷들의 흐름으로 정의된다. 따라서 수신된 패킷의 헤더를 분석하면 그 패킷이 플로우에 속하는지, 속한다면 플로우의 몇 번째 패킷인지 또는 마지막 패킷인지 등을 확인할 수 있다. IP 패킷 헤더분석은 종래 널리 알려진 사항이므로 상세한 설명을 생략한다.
- <22> 본 단계에서는 수신된 패킷이 플로우의 첫 패킷이 아니면서 상기 패킷에 대한 플로우테이블이 존재하지 않으면 HTTP 응답패킷이 아닌 플로우거나 한글디코딩이 불가능한 플로우임이 첫 패킷에 의해 이미 확인된 것이므로 추가 조치 없이 종료한다. 물론 헤더 정보가 없는 패킷이면서 플로우테이블에도 없는 패킷이라면 어떠한 처리 없이 종료한 후 바로 다음 패킷이 있는지를 조사하는 식의 순서를 따른다. 첫 패킷이 아니면서 해당 플로우테이블이 존재하면 HTTP 응답패킷이면서 한글디코딩이 가능한 플로우임이 첫 패킷에 의해 이미 확인된 것이므로 바로 제3단계(디코딩단계)로 이동한다.
- <23> 수신된 패킷이 플로우의 첫 패킷이면서 상기 패킷에 대한 플로우테이블이 존재하지 않으면 제2단계(적합성분석단계)로 이동한다.
- <24> (B) 제2단계(적합성분석단계)에서는 플로우의 첫 패킷이면서 상기 패킷에 대한 플로우테이블이 존재하지 않는 수신 패킷이 한글 디코딩이 가능한지를 확인하고, 가능하다면 그에 대한 플로우테이블을 생성하는 단계이다.
- <25> 즉, 상기 패킷의 HTTP 헤더 또는 바디의 정보를 참조하여 상기 패킷이 HTTP 응답패킷이면서 동시에 2바이트 한글 디코딩이 가능한 패킷이면 플로우테이블을 생성한 후 제3단계(디코딩단계)로 이동한다. 한편 HTTP 응답패킷이 아니거나 2바이트 한글 디코딩이 불가능한 패킷이면 추가적 조치없이 종료한다.
- <26> 상기 플로우테이블에는 플로우를 식별할 수 있는 정보(출발 IP 주소, 목적 IP 주소, 출발 port, 목적 port, 수신한 시간, 식별자, 플래그, 단편화옵셋 등의 전부 또는 일부) 및 삭제예약플래그, 플로우유지시간 및 1바이트 저장공간 등의 필드로 구성되도록 할 수 있다. 플로우테이블에 삭제예약 플래그 필드가 있는 경우, 초기 플로우분석단계에서 패킷이 마지막 패킷인 경우(즉 패킷의 TCP 헤더에 FIN 플래그가 설정된 경우) 이를 삭제예약 플래그에 기록해 두고, 이 기록의 유무를 확인함으로써 패킷이 마지막 패킷인 경우 플로우테이블을 삭제(초기화)한다. 또한 만일 플로우유지시간 필드가 있는 경우, 후속 패킷이 소정의 플로우유지시간 내에 수신되지 않으면 디코딩을 종료하도록 할 수 있을 것이다.
- <27> 한편, 본 발명은, 상기 플로우테이블을 제2단계 전에 생성하였다가 적합성이 없는 것으로 판단되면 초기화하는 것을 배제하지 않는다.
- <28> 제2단계에 대해서는 뒤에서 다시 상세히 설명한다.
- <29> (C) 제3단계(디코딩단계)는 한글 디코딩 적합성이 확인된 상기 패킷의 HTTP 바디의 스트링으로부터 한글을 디코딩하는 단계이다. 뒤에서 다시 상세히 설명한다.
- <30> (D) 제4단계(플로우테이블관리단계)는 디코딩이 완료된 패킷이 플로우의 마지막 패킷인 경우 플로우테이블을 초기화하고 종료하며, 마지막 패킷이 아닌 경우 플로우테이블을 업데이트하고 상기 1단계(플로우분석단계)로 이동하는 단계이다.
- <31> 플로우테이블에 삭제예약 플래그 필드가 있는 경우, 초기 플로우분석단계에서 패킷이 마지막 패킷인 경우(즉 패킷의 TCP 헤더에 FIN 플래그가 설정된 경우) 이를 삭제예약 플래그에 기록해 두고, 이 기록의 유무를 확인함으로써 패킷이 마지막 패킷인지를 확인하도록 할 수도 있다. 즉, 플로우테이블에 설정된 삭제예약 플래그가 있는지, 또는 소정의 플로우유지시간을 넘긴 플로우가 있는지를 체크하여 있다면 플로우테이블을 삭제(초기화)한다.
- <32> 이상의 과정을 통하여 수신된 패킷들을 연속해서 한글 디코딩화가 가능하게 되며, 마지막 패킷이 디코딩처리된 후에는 플로우테이블의 모든 정보가 삭제(초기화)되는 것이다.
- <33> 도시하지는 않았지만, 전술한 한글 디코딩 방법은, (A) 수신된 패킷의 IP헤더를 분석하여 단편화된 패킷인지를 확인하는 헤더분석부; (B) 상기 HTTP 패킷헤더와 바디의 일부정보를 참조하여 HTTP 응답패킷인지, 한글 디코딩이 가능한 패킷인지를 판단하는 적합성분석부; (C) 한글 디코딩이 가능한 패킷의 바디부분을 파싱을 통하여 디

코딩하는 디코딩처리부; 및 (D) 플로우정보를 생성, 관리하는 플로우테이블관리부;를 포함하는 HTTP 패킷의 한글 디코딩 장치에 의해 수행될 수 있다.

- <34> 도 2는 본 발명의 제2단계의 일예를 보여주는 세부 흐름도이다.
- <35> 본 발명의 제2단계(적합성분석단계)에서는 플로우의 첫 패킷이면서 상기 패킷에 대한 플로우테이블이 존재하지 않는 수신 패킷이 한글 디코딩이 가능한지를 확인하고, 가능하다면 그에 대한 플로우테이블을 생성한다.
- <36> 먼저 HTTP 응답패킷인지를 확인하고, 응답패킷인 경우 패킷이 압축되어 있는지를 확인해야 한다.
- <37> 즉, HTTP 헤더에 "HTTP/1.1 200 OK"라는 스트링을 찾아 HTTP 응답 패킷 여부를 조사한다. 참고로 TCP 연결을 위한 3 단계 핸드셰이킹(3-way handshaking) 방법에 따라 HTTP 요구 패킷, 그에 대한 서버측의 ACK가 온 뒤에 사용자가 요청한 HTTP 응답 패킷이 오게 된다. 이어서 HTTP 패킷 헤더 정보에 시그니처1("Content-Encoding") 예를 들면, "Content-Encoding: gzip/r/n"과 같이 gzip이나 deflate와 같은 압축 알고리즘으로 HTTP 패킷이 압축되어 있는지를 판단한다. 압축이 되어 있는 경우 재조합 패킷들을 재조합하고 해당 압축을 역으로 압축을 풀어주고 난 다음에 한글 디코딩을 할 수 있기에 이러한 패킷을 배제하는 것이다.
- <38> 한글 인코딩 방식은 헤더에서 "charset=" 시그니처 다음이나 HTTP 바디에서 xml 태그 시작을 알리는 "<?xml="1.0" encoding=") 시그니처에서 "encoding="를 통해서 알 수가 있다. 따라서 하기 표 1에 나타난 시그니처2~5 중 어느 하나라도 가지는 패킷인 경우 해당 패킷의 정보를 플로우테이블에 저장(생성)하고 디코딩 적합성분석단계를 종료하고 다음의 디코딩단계로 이동하고, 시그니처 2~5 모두 존재하지 않는 경우 아무런 조치 없이 종료한다.

표 1

시그니처1	HTTP 헤더에 "Content-Encoding"
시그니처2	HTTP 헤더에 "Content-Type: text" 및 HTTP body에 "<?xml="1.0" encoding="
시그니처3	HTTP 헤더에 "Content-Type: text" 및 "charset="
시그니처4	(HTTP 헤더에 "Content-Type: application" 및 HTTP body에 "<?xml="1.0" encoding="
시그니처5	HTTP 헤더에 "Content-Type: application" 및 "charset="

- <39>
- <40> 시그니처 중 "Content-Type"은 패킷이 어떠한 type의 정보를 담고 있는가를 나타내는 시그니처이다. 파일 포맷에 따라 MIME에서는 text, image, audio, video, application, message, multipart와 같은 7 가지를 제공하고 있다. 또한 각각의 Content-Type 포맷 뒤에는 sub-type이 따라온다. 몇 가지 예를 들자면 text/html, text/xml, text/plain, image/jpeg, image/gif, application/x-shockwave-flash와 같은 것들이 있다. HTTP에서 많이 쓰이는 Content-Type에는 text, image, application같은 것들이며 HTTP 응답 패킷 중 거의 대부분을 차지한다. 그러나 한글 디코딩과 관련하여 필요한 Content-Type은 text이다. 그리고 application content-Type이라고 할지라도 text 포맷을 나타낼 수도 있다. 대표적으로 cdma/wcdma망에서 application/vnd.oma.dd+xml과 같은 것들은 text 포맷으로 되어있으므로 이런 경우도 고려되어야 한다.
- <41> 정리하면, 본 발명에서 디코딩 적합성분석단계에서는 한글 디코딩이 가능한지 여부를 조사하는 단계로서, 응답 패킷인지, 해당 패킷이 압축되어 있는지, 또는 해당 패킷의 포맷이 한글을 포함할 수 있는 포맷인지와 한글을 어떤 인코딩방식을 써서 인코딩했는지를 알아내는 단계이다.
- <42> 도 3은 본 발명의 제3단계(디코딩단계) 및 제4단계(플로우테이블관리단계)의 일예를 보여주는 세부 흐름도로서 플로우분석단계 및 디코딩 적합성분석단계를 모두 통과하고 플로우테이블이 생성된 패킷에 대해서 실제의 한글 디코딩 알고리즘을 적용하여 한글로 출력하는 과정을 보여준다. 도에서 각 단계에 표시된 첨부문자 1)~12)의 의미는 하기 표 2와 같다.

표 2

1)	플로우테이블에 H0가 저장되어 있는가?	7)	서브스트링(T_L)에 더 읽을 문자가 있는가?
2)	T : 전체 스트링	8)	H1 : 인코딩된 한글의 하위 1바이트
3)	$T = T_L + '<' + T_R$	9)	다음 1바이트만큼 포인터 이동
4)	i는 해당 서브스트링(T_L)의 현재검사 위치	10)	"H0H1" 2바이트가 한글 완성형 코드표에 존재하는가?
5)	M은 왼쪽 서브 스트링(T_L)의 길이	11)	T_R 을 "<"로 더 분리할 수 있는가?
6)	H0 : 인코딩된 한글의 상위 1바이트	12)	H0을 플로우테이블에 저장

<43>

<44>

한글 디코딩처리부로 들어온 패킷에 대해서 우선 HTTP의 헤더유무를 조사하여 한글 문자열이 시작하는 위치를 찾는다(300). 즉 HTTP 헤더가 없는 패킷은 처음부터 시작하고, HTTP 헤더가 있는 패킷은 이전의 처리 과정이었던 한글 적합성분석부에서 조사한 이후의 HTTP 바디부분부터 시작한다. 한글 문자열의 시작 위치를 고정하고 플로우테이블에 저장된 한글 인코딩 타입을 참조하여 역으로 한글 디코딩을 시작한다.

<45>

한글 인코딩타입은 이전 4.1에서 설명했듯이 한글 완성형 인코딩 타입(KS C 5601)과 EUC-KR인코딩 타입, 유니코드의 인코딩 타입 중 하나인 UTF-8, UTF-16이 있다. 이 중에 한글 완성형 인코딩 타입(KS C 5601)과 EUC-KR 인코딩 타입에 대해서만 처리하는 걸로 알고리즘을 한정한다. 이 두 인코딩 타입은 위에서 설명한 바와 같이, 역슬래쉬(\)를 원표시(₩)로 대체한 코드이다. 따라서 이러한 두 인코딩 타입은 한글 디코딩 관점에서는 같은 알고리즘으로 처리될 수 있다(305).

<46>

EUC-KR 혹은 KS C 5601로 인코딩될 수 있는 한글은 2350자로 2바이트로 표현되며 B0A1에서 C8FE의 범위를 갖는다. 위에서도 언급한 바와 같이 이러한 2바이트로 인코딩된 한글 문자열의 상위 1바이트와 하위 1바이트가 나뉘어 서로 다른 패킷의 페이로드에 실려 전달될 경우가 생긴다. 이러한 경우를 감안하여 플로우테이블에 상위 1바이트가 있는지를 체크하는 것이 필요하다. 따라서, 한글 디코딩 타입을 설정했다면(305) 그 다음은 플로우테이블에서 해당 플로우의 한글 인코딩 상위 1바이트가 저장되어 있는지를 확인하고, 상위 1 바이트가 존재하면 현재 패킷의 맨 처음에 복사한다(310). 물론 전 패킷에서 디코딩 후 마지막 1 바이트가 남게되면 이를 플로우테이블에 저장해 둔다.

<47>

이렇게 디코딩할 스트링이 최종적으로 결정됐다면 전체 스트링을 최초의 '<'를 기준으로 서브 스트링으로 분리한다(315).

<48>

즉, 전체 스트링을 T라하면

<49>

1) $T = T_L + '<' + T_R$

<50>

2) $T = T_R$

<51>

3) $i = 0, M = |T_L|$

<52>

여기서 i는 해당 서브스트링(T_L)의 현재검사 위치를 가리키며 처음엔 서브스트링의 맨 앞을 가리키게 된다. M은 왼쪽 서브 스트링(T_L)의 길이를 나타내며 해당 서브스트링의 종료 지점을 찾는데 쓰인다(315).

<53>

위와 같은 식으로 '<'의 왼쪽 서브 스트링(T_L)을 입력으로 하여 한글 디코딩 알고리즘을 수행하고, 문자열 T_L 을 검사했다면 식2)에서 전체 스트링 중 검사하지 않은 T_R 로 설정하며 다시 1)을 수행하는 식의 반복을 하게 된다.

<54>

위에서 나눈 서브스트링을 차례대로 처음부터 검사해 나가는데 EUC-KR 혹은 KS C 5601에서 정한 한글 디코딩 값의 B0A1 ~ C8FE의 범위를 우선 상위 1바이트(H_0)에 적용해본다(325). 즉, 처음 읽은 1바이트의 값을(320) 한글

문자열의 범위를 나타내는 B0 ~ C9내의 값인지 비교해 본다. 'B0' <= H₀ < 'C9'의 조건에서 범위 내에 H₀가 존재한다면 서브스트링 중 다음에 읽을 바이트가 있는지를 확인한다(335). 이것은 하위 1바이트를 찾아서 한글 완성형 테이블과 비교하여 한글 문자열의 유무를 판단하기 위함이다.

<55> 만약 H₀가 'B0' ~ 'C9'의 범위 내에 있지 않다면 해당 스트링의 위치를 가리키는 i는 1만큼 증가하여 다음 문자를 입력으로 하여(330) 'B0' <= H₀ < 'C9' 비교를 하게 된다(320). 만약 1만큼 증가한 값이 서브스트링의 길이인 M과 같다면 해당 서브스트링에는 더 이상 한글 디코딩을 할 만한 문자가 없기 때문에 (315)단계를 통해 새로운 서브스트링을 구하게 된다.

<56> 다시 (335)단계에서 서브스트링의 현재 위치에서 1 바이트 더 읽을 수 있다면(340), 즉 하위 1 바이트(H₁)를 가져 읽을 수 있다면(340), "H₀H₁"를 한글 코드 값이 저장된 KS_C_table]에서 검색한다(345). 해당 테이블에 "H₀H₁"의 값이 저장 되어 있다면 그 인덱스와 같은 값을 갖는 한글 문자를 출력하게 된다. 그리고 서브스트링의 현재 위치에서 2바이트만큼 이동하여 이미 읽은 인코딩된 한글(2바이트)의 다음 1바이트를 가리키게 된다(350). 이렇게 2바이트를 이동한 다음에 현재 서브스트링에서 더 읽을 문자가 있는지를 체크하게 된다. 그래서 더 읽을 문자가 있다면 방금 전 한글 디코딩 알고리즘에 의한 순차적인 흐름을 또 다시 거치게 되는 것이다. 그런데, 만약 "H₀H₁"값이 한글 인코딩 테이블에 없다면 1바이트만큼 이동하여 다음 바이트를 읽는다. 즉 H₁이 H₀값으로 대체되어 위에서 한 과정을 다시 반복하게 된다(355).

<57> 마지막으로 'B0' <= H₀ < 'C9'의 범위 내에 H₀가 존재하지만(325) 해당 서브스트링에서는 더 이상 읽을 문자가 없을 경우(335)에는 다음과 같이 진행된다.

<58> 위와 같은 상황에서 새로운 서브스트링을 만들 수 있다면(360), 즉 아직 시그니처 '<'로 나뉘어질 부분이 남아 있는 경우에는 위에서 구한 H₀을 무시하고 다시 새로운 서브스트링을 가지고 한글 디코딩 알고리즘을 수행한다(315).

<59> 하지만 새로운 서브스트링을 만들 수 없는 경우라면, H₀값을 해당 플로우테이블의 상위 1byte를 저장할 수 있는 부분에 저장한다(360). 즉 한글 디코딩을 수행하기 위해 수신한 패킷을 모두 검사했다는 결론에 도달 했지만 HTTP reassembly 특성에 의해 인코딩된 상위 1바이트의 H₀가 존재한다는 것이다(310). 그러므로 뒤이어 올 패킷과 H₀를 하나의 패킷으로 간주하고 처리할 수 있도록 저장해 두어야 한다(360).

도면의 간단한 설명

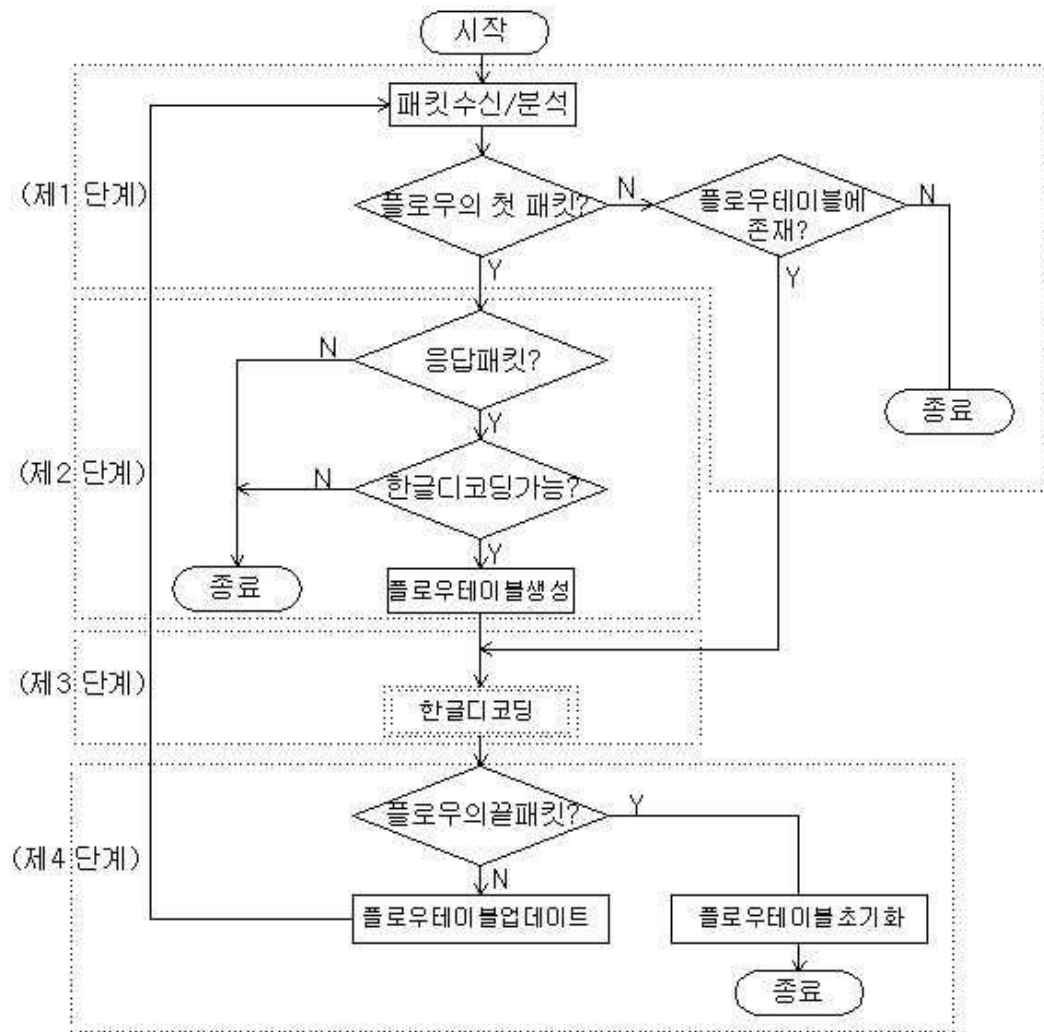
<60> 도 1은 본 발명에 따른 한글 디코딩 방법의 일예를 보여주는 전체 흐름도이다.

<61> 도 2는 본 발명의 제2단계(적합성분석단계)의 일예를 보여주는 세부 흐름도이다.

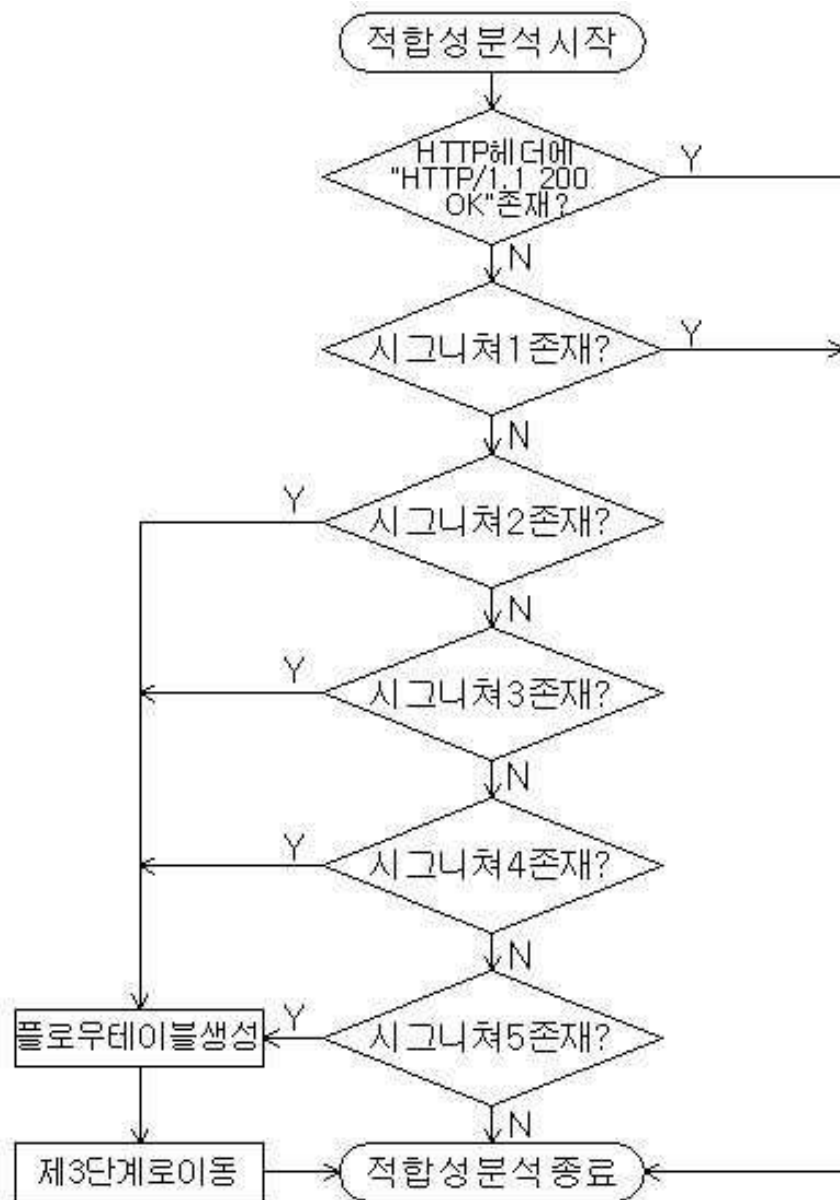
<62> 도 3은 본 발명의 제3단계(디코딩단계) 및 제4단계(플로우테이블관리단계)의 일예를 보여주는 세부 흐름도이다.

도면

도면1



도면2



도면3

