(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2016년06월27일
(11) 등록번호 10-1630383
(24) 등록일자 2016년06월08일

(51) 국제특허분류(Int. Cl.)
G06K 9/62 (2006.01) *G06K 9/34* (2006.01)
(52) CPC특허분류
G06K 9/6211 (2013.01)
G06K 9/344 (2013.01)
(21) 출원번호 10-2015-0046640
(22) 출원일자 2015년04월02일
심사청구일자 2015년04월02일
(56) 선행기술조사문헌
KR1020100007420 A
KR1020120063879 A
KR1020140128153 A
KR1020130081140 A

(73) 특허권자
한국과학기술원
대전광역시 유성구 대학로 291(구성동)
(72) 발명자
한동수
대전광역시 유성구 대학로 291 한국과학기술원
최병권
대전광역시 유성구 대학로 291 한국과학기술원
(뒷면에 계속)
(74) 대리인
김성호

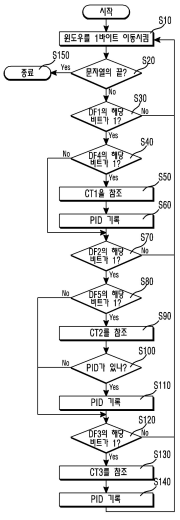
전체 청구항 수 : 총 14 항 심사관 : 남윤권

(54) 발명의 명칭 **다중 패턴 매칭 알고리즘 및 이를 이용한 처리장치**

(57) 요 약

실시 형태에 따른 다중 패턴 매칭 알고리즘은, 무빙 윈도우를 문자열의 처음부터 1바이트씩 이동시키는 이동단계; 상기 무빙 윈도우의 현재 위치에서 2바이트 길이만큼의 문자열을 정수 값으로 변환하여, 패턴의 길이가 2바이트 이상인 패턴들에 대한 제1 다이렉트 필터(DF1)에서의 관련 위치의 비트가 1로 세팅되어 있는지 확인하는 DF1확인단계; 상기 DF1확인단계에 따라 1로 세팅된 경우, 다른 다이렉트 필터(DF)로 이동하는 DF이동단계; 마지막으로 확인한 다이렉트 필터(DF)의 관련 위치의 비트가 0인 경우, 상기 무빙 윈도우를 다시 1바이트 이동시키는 재이동단계; 및 상기 무빙 윈도우가 상기 문자열의 끝에 위치하는지 확인하고, 상기 무빙 윈도우가 상기 문자열의 끝에 위치하는 경우, 알고리즘을 종료하는 종료단계;를 포함한다.

대 표 도 - 도1



(72) 발명자

남재현

대전 유성구 대학로 291

잠세드 무함마드

대전광역시 유성구 대학로 291 한국과학기술원

박경수

대전 유성구 대학로 291

이 발명을 지원한 국가연구개발사업

과제고유번호 1491105001

부처명 미래창조과학부

연구관리전문기관 IITP

연구사업명 정보통신.방송 연구개발사업

연구과제명 NfV 개념의 멀티서비스 맞춤형 스위칭 시스템 및 운영체제 개발

기 여 율 1/1

주관기관 (주)파이오링크

연구기간 2014.04.01 ~ 2015.02.28

명 세 서

청구범위

청구항 1

무빙 윈도우를 문자열의 처음부터 1바이트씩 이동시키는 이동단계;

상기 무빙 윈도우의 현재 위치에서 2바이트 길이만큼의 문자열을 정수 값으로 변환하여, 패턴의 길이가 2바이트 이상인 패턴들에 대한 제1 다이렉트 필터(DF1)에서의 관련 위치의 비트가 1로 세팅되어 있는지 확인하는 DF1확인단계;

상기 DF1확인단계에 따라 1로 세팅된 경우, 다른 다이렉트 필터(DF)로 이동하는 DF이동단계;

마지막으로 확인한 다이렉트 필터(DF)의 관련 위치의 비트가 0인 경우, 상기 무빙 윈도우를 다시 1바이트 이동시키는 재이동단계; 및

상기 무빙 윈도우가 상기 문자열의 끝에 위치하는지 확인하고, 상기 무빙 윈도우가 상기 문자열의 끝에 위치하는 경우, 알고리즘을 종료하는 종료단계;를 포함하는, 다중 패턴 매칭 알고리즘.

청구항 2

제1항에 있어서,

상기 DF이동단계 이후에는,

패턴의 길이가 2바이트 이상 4바이트 미만인 패턴들에 대한 제4 다이렉트 필터(DF4)에서의 관련 위치의 비트가 1로 세팅되어 있는지 확인하는 DF4확인단계;를 더 포함하는, 다중 패턴 매칭 알고리즘.

청구항 3

제2항에 있어서,

상기 DF4확인단계 이후에는,

상기 제4 다이렉트 필터(DF4)의 관련 위치의 비트가 1로 세팅된 경우, 2바이트 이상 4바이트 미만인 패턴들에 대한 패턴 아이디가 저장된 제1 컴팩트 테이블(CT1)을 참조하여 상기 무빙 윈도우에 위치한 문자열에 해당하는 패턴 아이디(PID)를 기록하는 PID기록단계;를 더 포함하는, 다중 패턴 매칭 알고리즘.

청구항 4

제1항에 있어서,

상기 DF이동단계 이후에는,

상기 무빙 윈도우의 상기 현재 위치로부터 2바이트 이동시키고, 이동한 위치에서 2바이트 길이만큼의 문자열을 정수 값으로 변환하여, 패턴의 길이가 4바이트 이상인 패턴들에 대한 제2 다이렉트 필터(DF2)에서의 관련 위치의 비트가 1로 세팅되어 있는지 확인하는 DF2확인단계;를 더 포함하는, 다중 패턴 매칭 알고리즘.

청구항 5

제4항에 있어서,

상기 DF2확인단계 이후에는,

상기 제2 다이렉트 필터(DF2)의 관련 위치의 비트가 1로 세팅된 경우, 패턴의 길이가 4바이트 이상 8바이트 미만인 패턴들에 대한 제5 다이렉트 필터(DF5)에서의 관련 위치의 비트가 1로 세팅되어 있는지 확인하는 DF5확인단계;를 더 포함하는, 다중 패턴 매칭 알고리즘.

청구항 6

제5항에 있어서,

상기 DF5확인단계 이후에는,

상기 제5 다이렉트 필터(DF5)의 관련 위치의 비트가 1로 세팅된 경우, 4바이트 이상 8바이트 미만인 패턴들에 대한 패턴 아이디가 저장된 제2 컴팩트 테이블(CT2)을 참조하여 상기 무빙 윈도우에 위치한 문자열에 해당하는 패턴 아이디가 있는지 확인하고, 상기 문자열에 해당하는 패턴 아이디가 있는 경우, 상기 패턴 아이디(PID)를 기록하는 PID기록단계;를 더 포함하는, 다중 패턴 매칭 알고리즘.

청구항 7

제1항에 있어서,

상기 DF이동단계 이후에는,

상기 무빙 윈도우의 상기 현재 위치로부터 6바이트 이동시키고, 이동한 위치에서 2바이트 길이만큼의 문자열을 정수 값으로 변환하여, 패턴의 길이가 8바이트 이상인 패턴들에 대한 제3 다이렉트 필터(DF3)에서의 관련 위치의 비트가 1로 세팅되어 있는지 확인하는 DF3확인단계;를 더 포함하는, 다중 패턴 매칭 알고리즘.

청구항 8

제7항에 있어서,

상기 DF3확인단계 이후에는,

상기 제3 다이렉트 필터(DF3)의 관련 위치의 비트가 1로 세팅된 경우, 8바이트 이상인 패턴들에 대한 패턴 아이디가 저장된 제3 컴팩트 테이블(CT3)을 참조하여 상기 무빙 윈도우에 위치한 문자열에 해당하는 패턴 아이디(PID)를 기록하는 PID기록단계;를 더 포함하는, 다중 패턴 매칭 알고리즘.

청구항 9

제1항 내지 제8항 중 어느 한 항에 있어서,

상기 알고리즘은 네트워크 침입 탐지 시스템(Network Instrusion Detection System, NIDS)에 사용되는, 다중 패턴 매칭 알고리즘.

청구항 10

무빙 윈도우를 문자열의 처음부터 1바이트씩 이동시키는 이동단계;

상기 무빙 윈도우의 현재 위치에서 2바이트 길이만큼의 문자열을 정수 값으로 변환하여, 패턴의 길이가 2바이트 이상인 패턴들에 대한 제1 다이렉트 필터(DF1)에서의 관련 위치의 비트가 1로 세팅되어 있는지 확인하는 DF1확인단계;

상기 DF1확인단계에 따라 1로 세팅된 경우, 다른 다이렉트 필터(DF)로 이동하는 DF이동단계;

마지막으로 확인한 다이렉트 필터(DF)의 관련 위치의 비트가 0인 경우, 상기 무빙 윈도우를 다시 1바이트 이동시키는 재이동단계; 및

상기 무빙 윈도우가 상기 문자열의 끝에 위치하는지 확인하고, 상기 무빙 윈도우가 상기 문자열의 끝에 위치하는 경우, 알고리즘을 종료하는 종료단계;를 실행시키기 위한 매체에 저장된 프로그램.

청구항 11

복수의 비트(bit)를 갖는 비트 어레이(bit array)이고, 각 비트 자신이 인덱스(index)에 해당하는 연속된 두 개의 아스키 코드가 특정 패턴의 일부에 속해 있는지 없는지를 확인하고, 하나 이상의 다이렉트 필터로 이루어지며, 상기 다이렉트 필터 각각은 패턴의 길이에 따라 상기 패턴의 2^n(n=0, 1, 2, 3, …)번째의 2바이트에 대한 정보를 갖는 다이렉트 필터(DF); 및

문자열 내에 존재하는 패턴들의 패턴 아이디(Pattern ID)를 기록하며, 어떤 패턴이 상기 문자열에 존재하는지를 확인하기 위한 구조체이고, 상기 패턴의 길이에 따라 패턴 그룹을 형성하여, 상기 패턴 그룹별로 각각 상기 패

턴 아이디가 저장된 하나 이상의 컴팩트 테이블(CT);을 포함하는, 다중 패턴 매칭 처리장치.

청구항 12

제11항에 있어서,

상기 다이렉트 필터(DF)는,

모든 패턴들의 맨 앞 2바이트에 대한 정보를 포함하는 제1 다이렉트 필터(DF1), 길이가 4바이트 이상인 패턴들의 두 번째 2바이트에 대한 정보를 포함하는 제2 다이렉트 필터(DF2), 길이가 8바이트 이상인 패턴들의 네 번째 2바이트에 대한 정보를 포함하는 제3 다이렉트 필터(DF3) 및 길이가 2바이트 이상 4바이트 미만인 패턴들의 맨 앞 2바이트에 대한 정보를 포함하는 제4 다이렉트 필터(DF4)를 포함하는, 다중 패턴 매칭 처리장치.

청구항 13

제12항에 있어서,

상기 다이렉트 필터(DF)는,

길이가 4바이트 이상 8바이트 미만인 패턴들의 두 번째 2바이트에 대한 정보를 포함하는 제5 다이렉트 필터(DF5)를 더 포함하는, 다중 패턴 매칭 처리장치.

청구항 14

제11항에 있어서,

상기 컴팩트 테이블(CT)은,

길이가 2바이트 이상 4바이트 미만인 패턴들의 패턴 아이디를 포함하는 제1 컴팩트 테이블(CT1), 길이가 4바이트 이상 8바이트 미만인 패턴들의 패턴 아이디를 포함하는 제2 컴팩트 테이블(CT2) 및 길이가 8바이트 이상인 패턴들의 패턴 아이디를 포함하는 제3 컴팩트 테이블(CT3)을 포함하는, 다중 패턴 매칭 처리장치.

발명의 설명

기 술 분 야

[0001] 본 발명은 다중 패턴 매칭 알고리즘 및 이를 이용한 처리장치에 관한 것으로, 보다 구체적으로 다이렉트 필터와 컴팩트 테이블을 이용한 다중 패턴 매칭 알고리즘 및 이를 이용한 처리장치에 관한 것이다.

배 경 기 술

[0002] 다중 패턴 매칭(Multi-Pattern Matching)은 문자열에서 1개 이상의 패턴들의 존재여부를 찾는 문제에 대한 것이다. 종래에는 다중 패턴 매칭 문제를 해결하기 위해 도 12와 같이, 각각의 패턴에 대하여 문자열을 한번씩 검색하여 존재여부를 확인하는 방법이 사용되었다. 하지만, 이러한 방법은 존재 여부를 확인하고자 하는 패턴의 개수만큼 문자열을 탐색해야 하기 때문에 성능이 느려지는 문제가 있었다.

[0003] 따라서, 이러한 문제를 해결하기 위해 패턴의 개수와는 무관하게 단 한번의 문자열 탐색으로 모든 패턴의 존재여부를 확인할 수 있는 다중 패턴 매칭 알고리즘이 연구되었다.

[0004] 일반적으로 단일 패턴 매칭 알고리즘은 다중 패턴 매칭 문제를 해결하는데 O(m+zn)의 시간 복잡도를 갖는다(m: 모든 패턴들의 길이 합, z: 패턴 수, n: 문자열의 길이). 이에 반해, 종래의 알고리즘 중 하나인 아호 코라식 알고리즘(Aho-Corasick algorithm)은 O(m+n+k)의 시간 복잡도를 갖는다(k: 문자열 내에 패턴이 발생한 횟수).

[0005] 도 13을 참조하면, 아호 코라식 알고리즘에는 패턴들이 들어있는 키워드 트리(Keyword Tree)에 실패 링크(Failure link)와 출력 링크(Output link)를 추가한 구조체가 사용된다. 이를 이용하여 아호 코라식 알고리즘은 단 한번의 문자열 탐색으로 키워드 트리에 있는 모든 패턴들의 존재 여부를 판단할 수 있었다.

[0006] 하지만, 아호 코라식 알고리즘은 패턴의 개수의 증가에 따라 아호 코라식에서 탐색에 사용하는 트리(Tree)의 크기가 빠르게 커진다는 문제가 있었다. 따라서, 트리 구조체의 특성 때문에 아호 코라식을 이용하여 탐색을 하는 과정에서 많은 캐시 미스(cache miss)가 발생하는 문제가 있었다. 일반적으로, 많은 캐시 미스는 성능 저하로 직결된다.

[0007] 따라서, 캐시 미스의 발생을 줄일 수 있는 다중 패턴 매칭 알고리즘 및 이를 이용한 처리장치의 연구가 필요하게 되었다.

선행기술문헌

특허문헌

[0008] (특허문헌 0001) 한국공개특허공보 제10-2014-0128153호(공개일: 2014.11.06)

발명의 내용

해결하려는 과제

[0009] 실시 형태는 다이렉트 필터와 컴팩트 테이블을 이용하여 종래의 문제를 해결할 수 있는 다중 패턴 매칭 알고리즘 및 이를 이용한 처리장치를 제공하는 것을 목적으로 한다.

[0010] 실시 형태는 네트워크 침입 탐지 시스템에 사용할 수 있는 다중 패턴 매칭 알고리즘 및 이를 이용한 처리장치를 제공하는 것을 목적으로 한다.

과제의 해결 수단

[0011] 실시 형태에 따른 다중 패턴 매칭 알고리즘은, 무빙 윈도우를 문자열의 처음부터 1바이트씩 이동시키는 이동단계; 상기 무빙 윈도우의 현재 위치에서 2바이트 길이만큼의 문자열을 정수 값으로 변환하여, 패턴의 길이가 2바이트 이상인 패턴들에 대한 제1 다이렉트 필터(DF1)에서의 관련 위치의 비트가 1로 세팅되어 있는지 확인하는 DF1확인단계; 상기 DF1확인단계에 따라 1로 세팅된 경우, 다른 다이렉트 필터(DF)로 이동하는 DF이동단계; 마지막으로 확인한 다이렉트 필터(DF)의 관련 위치의 비트가 0인 경우, 상기 무빙 윈도우를 다시 1바이트 이동시키는 재이동단계; 및 상기 무빙 윈도우가 상기 문자열의 끝에 위치하는지 확인하고, 상기 무빙 윈도우가 상기 문자열의 끝에 위치하는 경우, 알고리즘을 종료하는 종료단계;를 포함한다.

[0012] 여기서, 상기 DF이동단계 이후에는, 패턴의 길이가 2바이트 이상 4바이트 미만인 패턴들에 대한 제4 다이렉트 필터(DF4)에서의 관련 위치의 비트가 1로 세팅되어 있는지 확인하는 DF4확인단계;를 더 포함할 수 있다.

[0013] 여기서, 상기 DF4확인단계 이후에는, 상기 제4 다이렉트 필터(DF4)의 관련 위치의 비트가 1로 세팅된 경우, 2바이트 이상 4바이트 미만인 패턴들에 대한 패턴 아이디가 저장된 제1 컴팩트 테이블(CT1)을 참조하여 상기 무빙 윈도우에 위치한 문자열에 해당하는 패턴 아이디(PID)를 기록하는 PID기록단계;를 더 포함할 수 있다.

[0014] 여기서, 상기 DF이동단계 이후에는, 상기 무빙 윈도우의 상기 현재 위치로부터 2바이트 이동시키고, 이동한 위치에서 2바이트 길이만큼의 문자열을 정수 값으로 변환하여, 패턴의 길이가 4바이트 이상인 패턴들에 대한 제2 다이렉트 필터(DF2)에서의 관련 위치의 비트가 1로 세팅되어 있는지 확인하는 DF2확인단계;를 더 포함할 수 있다.

[0015] 여기서, 상기 DF2확인단계 이후에는, 상기 제2 다이렉트 필터(DF2)의 관련 위치의 비트가 1로 세팅된 경우, 패턴의 길이가 4바이트 이상 8바이트 미만인 패턴들에 대한 제5 다이렉트 필터(DF5)에서의 관련 위치의 비트가 1로 세팅되어 있는지 확인하는 DF5확인단계;를 더 포함할 수 있다.

[0016] 여기서, 상기 DF5확인단계 이후에는, 상기 제5 다이렉트 필터(DF5)의 관련 위치의 비트가 1로 세팅된 경우, 4바이트 이상 8바이트 미만인 패턴들에 대한 패턴 아이디가 저장된 제2 컴팩트 테이블(CT2)을 참조하여 상기 무빙 윈도우에 위치한 문자열에 해당하는 패턴 아이디가 있는지 확인하고, 상기 문자열에 해당하는 패턴 아이디가 있는 경우, 상기 패턴 아이디(PID)를 기록하는 PID기록단계;를 더 포함할 수 있다.

[0017] 여기서, 상기 DF이동단계 이후에는, 상기 무빙 윈도우의 상기 현재 위치로부터 6바이트 이동시키고, 이동한 위치에서 2바이트 길이만큼의 문자열을 정수 값으로 변환하여, 패턴의 길이가 8바이트 이상인 패턴들에 대한 제3 다이렉트 필터(DF3)에서의 관련 위치의 비트가 1로 세팅되어 있는지 확인하는 DF3확인단계;를 더 포함할 수 있다.

[0018] 여기서, 상기 DF3확인단계 이후에는, 상기 제3 다이렉트 필터(DF3)의 관련 위치의 비트가 1로 세팅된 경우, 8바이트 이상인 패턴들에 대한 패턴 아이디가 저장된 제3 컴팩트 테이블(CT3)을 참조하여 상기 무빙 윈도우에 위치

한 문자열에 해당하는 패턴 아이디(PID)를 기록하는 PID기록단계;를 더 포함할 수 있다.

[0019] 여기서, 상기 알고리즘은 네트워크 침입 탐지 시스템(Network Instrusion Detection System, NIDS)에 사용될 수 있다.

[0020] 실시 형태는 무빙 윈도우를 문자열의 처음부터 1바이트씩 이동시키는 이동단계; 상기 무빙 윈도우의 현재 위치에서 2바이트 길이만큼의 문자열을 정수 값으로 변환하여, 패턴의 길이가 2바이트 이상인 패턴들에 대한 제1 다이렉트 필터(DF1)에서의 관련 위치의 비트가 1로 세팅되어 있는지 확인하는 DF1확인단계; 상기 DF1확인단계에 따라 1로 세팅된 경우, 다른 다이렉트 필터(DF)로 이동하는 DF이동단계; 마지막으로 확인한 다이렉트 필터(DF)의 관련 위치의 비트가 0인 경우, 상기 무빙 윈도우를 다시 1바이트 이동시키는 재이동단계; 및 상기 무빙 윈도우가 상기 문자열의 끝에 위치하는지 확인하고, 상기 무빙 윈도우가 상기 문자열의 끝에 위치하는 경우, 알고리즘을 종료하는 종료단계;를 실행시키기 위한 매체에 저장된 프로그램을 포함한다.

[0021] 한편, 실시 형태에 따른 다중 패턴 매칭 처리장치는, 복수의 비트(bit)를 갖는 비트 어레이(bit array)이고, 각 비트 자신이 인덱스(index)에 해당하는 연속된 두 개의 아스키 코드가 특정 패턴의 일부에 속해 있는지 없는지를 확인하고, 하나 이상의 다이렉트 필터로 이루어지며, 상기 다이렉트 필터 각각은 패턴의 길이에 따라 상기 패턴의 2^n(n=0, 1, 2, 3, …)번째의 2바이트에 대한 정보를 갖는 다이렉트 필터(DF); 및 문자열 내에 존재하는 패턴들의 패턴 아이디(Pattern ID)를 기록하며, 어떤 패턴이 상기 문자열에 존재하는지를 확인하기 위한 구조체이고, 상기 패턴의 길이에 따라 패턴 그룹을 형성하여, 상기 패턴 그룹별로 각각 상기 패턴 아이디가 저장된 하나 이상의 컴팩트 테이블(CT);을 포함한다.

[0022] 여기서, 상기 다이렉트 필터(DF)는, 모든 패턴들의 맨 앞 2바이트에 대한 정보를 포함하는 제1 다이렉트 필터(DF1), 길이가 4바이트 이상인 패턴들의 두 번째 2바이트에 대한 정보를 포함하는 제2 다이렉트 필터(DF2), 길이가 8바이트 이상인 패턴들의 네 번째 2바이트에 대한 정보를 포함하는 제3 다이렉트 필터(DF3) 및 길이가 2바이트 이상 4바이트 미만인 패턴들의 맨 앞 2바이트에 대한 정보를 포함하는 제4 다이렉트 필터(DF4)를 포함할 수 있다.

[0023] 여기서, 상기 다이렉트 필터(DF)는, 길이가 4바이트 이상 8바이트 미만인 패턴들의 두 번째 2바이트에 대한 정보를 포함하는 제5 다이렉트 필터(DF5)를 더 포함할 수 있다.

[0024] 여기서, 상기 컴팩트 테이블(CT)은, 길이가 2바이트 이상 4바이트 미만인 패턴들의 패턴 아이디를 포함하는 제1 컴팩트 테이블(CT1), 길이가 4바이트 이상 8바이트 미만인 패턴 아이디를 패턴 아이디를 포함하는 제2 컴팩트 테이블(CT2) 및 길이가 8바이트 이상인 패턴들의 패턴 아이디를 포함하는 제3 컴팩트 테이블(CT3)을 포함할 수 있다.

발명의 효과

[0025] 실시 형태에 따른 다중 패턴 매칭 알고리즘 및 이를 이용한 처리장치는 다이렉트 필터와 컴팩트 테이블을 이용하기 때문에 종래보다 적은 캐시 미스를 유발하여 높은 성능을 낼 수 있다.

[0026] 실시 형태에 따른 다중 패턴 매칭 알고리즘 및 이를 이용한 처리장치는 패턴의 일부만을 가지고 탐색하기 때문에, 종래보다 더 적은 메모리 공간을 필요로 할 수 있다.

[0027] 실시 형태에 따른 다중 패턴 매칭 알고리즘 및 이를 이용한 처리장치는 네트워크 침입 탐지 시스템(Network Instrusion Detection System, NIDS) 등과 같은 어플리케이션에 사용될 수 있다.

도면의 간단한 설명

[0028] 도 1은 실시 형태에 따른 다중 패턴 매칭 알고리즘의 순서도이다.

도 2는 실시 형태에 따른 다이렉트 필터를 설명하기 위한 도면이다.

도 3은 패턴의 구성을 도시한 일 예이다.

도 4는 도 3에 대한 제4 다이렉트 필터와 제1 컴팩트 테이블이다.

도 5는 도 3에 대한 제2 컴팩트 테이블이다.

도 6은 도 3에 대한 제3 컴팩트 테이블이다.

도 7은 실시 형태에 따른 다중 패턴 매칭 알고리즘의 동작을 설명하기 위한 패턴 집합의 일 예이다.

도 8은 도 7에 대한 제1 내지 제5 다이렉트 필터이다.

도 9는 도 7에 대한 제1 내지 제3 컴팩트 테이블이다.

도 10은 도 7에서 2바이트 패턴의 존재가 확인되는 과정을 설명하기 위한 도면이다.

도 11은 도 7에서 4바이트 또는 8바이트 패턴의 존재가 확인되는 과정을 설명하기 위한 도면이다.

도 12는 패턴들과 문자열의 일 예이다.

도 13은 패턴 his, hers, she, he, her에 대한 키워드 트리의 예제이다.

발명을 실시하기 위한 구체적인 내용

[0029] 도면에서 각층의 두께나 크기는 설명의 편의 및 명확성을 위하여 과장되거나 생략되거나 또는 개략적으로 도시되었다. 또한, 각 구성요소의 크기는 실제크기를 전적으로 반영하는 것은 아니다.

[0030] 본 발명에 따른 실시 형태의 설명에 있어서, 어느 한 element가 다른 element의 " 상(위) 또는 하(아래)(on or under)"에 형성되는 것으로 기재되는 경우에 있어, 상(위) 또는 하(아래)(on or under)는 두 개의 element가 서로 직접(directly)접촉되거나 하나 이상의 다른 element가 상기 두 element사이에 배치되어(indirectly) 형성되는 것을 모두 포함한다. 또한 “상(위) 또는 하(아래)(on or under)”으로 표현되는 경우 하나의 element를 기준으로 위쪽 방향뿐만 아니라 아래쪽 방향의 의미도 포함할 수 있다.

[0031] 이하에서는, 도면을 참조하여 실시 형태에 따른 다중 패턴 매칭 알고리즘 및 이를 이용한 처리장치를 설명하도록 한다.

[0032] **<실시 형태>**

[0033] 도 1은 실시 형태에 따른 다중 패턴 매칭 알고리즘의 순서도이다.

[0034] 도 1을 참조하면, 실시 형태에 따른 다중 패턴 매칭 알고리즘은, 먼저, 무빙 윈도우를 문자열의 처음부터 1바이트(Byte)씩 이동시킨다(S10). 일 예로, 무빙 윈도우를 문자열의 처음부터 1바이트씩 옆(오른쪽)으로 이동시킨다.

[0035] 이때, 이동된 무빙 윈도우가 문자열의 위치를 확인하고(S20), 이동된 무빙 윈도우가 문자열의 끝에 위치하는 경우, 알고리즘을 종료한다(S150).

[0036] 반면, 이동된 무빙 윈도우가 문자열의 끝에 위치하지 않는 경우, 무빙 윈도우의 현재 위치에서 2바이트 길이만큼의 문자열을 정수(integer) 값으로 변환하여, 패턴의 길이가 2바이트 이상인 패턴들에 대한 제1 다이렉트 필터(Direct Filter, DF1)에서의 관련 위치의 비트(bit)가 1로 세팅(set)되어 있는지 확인한다(S30).

[0037] 이때, 제1 다이렉트 필터(DF1)의 관련 위치의 비트가 0인 경우, S10단계로 이동하여 무빙 윈도우를 다시 1바이트 이동시킬 수 있다.

[0038] 반면, 제1 다이렉트 필터(DF1)의 관련 위치의 비트가 1로 세팅된 경우, 같은 위치에서 제4 다이렉트 필터(DF4)에서의 관련 위치의 비트가 1로 세팅되어 있는지 확인한다(S40).

[0039] 이때, 제4 다이렉트 필터(DF4)의 관련 위치의 비트가 0인 경우, 후술하게 될 S70단계로 이동할 수 있다.

[0040] 반면, 제4 다이렉트 필터(DF4)의 관련 위치의 비트가 1로 세팅된 경우, 제1 컴팩트 테이블(CT1)을 참조한다(S50). 구체적으로, 제1 컴팩트 테이블(CT1)의 인덱스(index)를 계산한다.

[0041] 무빙 윈도우에 위치한 문자열에 해당하는 패턴의 패턴 아이디(pattern id, PID)를 기록한다(S60).

[0042] 무빙 윈도우의 현재 위치로부터 2바이트 이동시키고, 이동한 위치에서 2바이트 길이만큼의 문자열을 정수 값으로 변환하여, 패턴의 길이가 4바이트 이상인 패턴들에 대한 제2 다이렉트 필터(DF2)에서의 관련 위치의 비트가 1로 세팅되어 있는지 확인한다(S70).

[0043] 이때, 제2 다이렉트 필터(DF2)의 관련 위치의 비트가 0인 경우, S10단계로 이동하여 무빙 윈도우를 다시 1바이

트 이동시킬 수 있다.

[0044] 반면, 제2 다이렉트 필터(DF2)의 관련 위치의 비트가 1로 세팅된 경우, 같은 위치에서 패턴의 길이가 4바이트 이상 8바이트 미만인 패턴들에 대한 제5 다이렉트 필터(DF5)에서의 관련 위치의 비트가 1로 세팅되어 있는지 확인한다(S80).

[0045] 이때, 제5 다이렉트 필터(DF5)의 관련 위치의 비트가 0인 경우, 후술하게 될 S120단계로 이동할 수 있다.

[0046] 반면, 제5 다이렉트 필터(DF5)의 관련 위치의 비트가 1로 세팅된 경우, 패턴의 길이가 4바이트 이상 8바이트 미만인 패턴들에 대한 제2 컴팩트 테이블(CT2)을 참조한다(S90). 구체적으로, 제2 컴팩트 테이블(CT2)의 인덱스를 계산한다.

[0047] 무빙 윈도우에 위치한 문자열에 해당하는 패턴 아이디가 있는지 확인하고(S100), 무빙 윈도우에 위치한 문자열에 해당하는 패턴 아이디가 있는 경우, 그 패턴 아이디들을 모두 기록한다(S110).

[0048] 무빙 윈도우의 현재 위치로부터 6바이트 이동시키고, 이동한 위치에서 2바이트 길이만큼의 문자열을 정수 값으로 변환하여, 패턴의 길이가 8바이트 이상인 패털들에 대한 제3 다이렉트 필터(DF3)에서의 관련 위치의 비트가 1로 세팅되어 있는지 확인한다(S120).

[0049] 이때, 제3 다이렉트 필터(DF3)의 관련 위치의 비트가 0인 경우, S10단계로 이동하여 무빙 윈도우를 다시 1바이트 이동시킬 수 있다.

[0050] 반면, 제3 다이렉트 필터(DF3)의 관련 위치의 비트가 1로 세팅된 경우, 8바이트 이상인 패턴들에 대한 패턴 아이디가 저장된 제3 컴팩트 테이블(CT3)을 참조한다(S130). 구체적으로, 제3 컴팩트 테이블(CT3)의 인덱스를 계산한다.

[0051] 무빙 윈도우에 위치한 문자열에 해당하는 패턴 아이디들을 기록한다(S140). 여기서, 제3 컴팩트 테이블(CT3)에서의 해당 위치의 패턴 아이디들을 기록하게 된 경우, S10단계로 이동하여 무빙 윈도우를 다시 1바이트 이동시킬 수 있다.

[0052] 이하에서는 도 1에 도시된 다중 패턴 매칭 알고리즘을 구체적으로 설명하도록 한다.

[0053] 실시 형태에 따른 다중 패턴 매칭 알고리즘은 패턴의 길이를 기준으로 하여 패턴들을 총 3가지 그룹으로 나눌 수 있다.

[0054] 먼저, 패턴의 길이가 2바이트 이상 4바이트 미만인 패턴 그룹으로 나눌 수 있고, 패턴의 길이가 4바이트 이상 8바이트 미만인 패턴 그룹으로 나눌 수 있고, 마지막으로, 패턴의 길이가 8바이트 이상인 패턴 그룹으로 나눌 수 있다.

[0055] 각 패턴 그룹에 대하여, 탐색 과정 중에 사용되는 컴팩트 테이블들은 그 구조가 조금씩 다르다.

[0056] 도 2는 실시 형태에 따른 다이렉트 필터를 설명하기 위한 도면이다.

[0057] 도 2를 참조하면, 실시 형태에 따른 다이렉트 필터(Direct Filter, DF)는 65536개의 비트(bit)(10)를 가지는 비트 어레이(bit array)일 수 있다. 비트 어레이 각각의 인덱스(index)(30)는 두 개의 연속된 아스키(ASCII) 코드일 수 있다. 여기서, 다이렉트 필터(DF)는 말 그대로 필터의 역할을 한다. 따라서, 후술할 컴팩트 테이블(Compact Table, CT)에 접근하기 전에 미리 다이렉트 필터(DF)로 한 번 걸러낼 수 있다. 따라서, 다이렉트 필터(DF)를 통과한 무빙 윈도우들만이 컴팩트 테이블에 접근하게 된다.

[0058] 다이렉트 필터(DF)는 하나 이상일 수 있다. 일 예로, 실시 형태에서는 후술할 총 5개의 다이렉트 필터(DF1~DF5)가 사용될 수 있다. 구체적으로, 제1 다이렉트 필터(DF1)는 패턴의 길이가 2바이트 이상인 패턴들에 대한 다이렉트 필터이고, 제2 다이렉트 필터(DF2)는 패턴의 길이가 4바이트 이상인 패턴들에 대한 다이렉트 필터이고, 제3 다이렉트 필터(DF3)는 패턴의 길이가 8바이트 이상인 패턴들에 대한 다이렉트 필터이고, 제4 다이렉트 필터(DF4)는 패턴의 길이가 2바이트 이상 4바이트 미만인 패턴들에 대한 다이렉트 필터이고, 제5 다이렉트 필터(DF5)는 패턴의 길이가 4바이트 이상 8바이트 미만인 패턴들에 대한 다이렉트 필터이다.

[0059] 여기서, 각 비트는 자신의 인덱스에 해당하는 연속된 두 개의 아스키 코드가 특정 패턴의 일부에 속해 있는지 없는지 그 여부를 의미한다. 즉, 비트가 1이면 아스키 코드가 특정 패턴의 일부에 속해 있는 것을 의미하고, 비트가 0이면 아스키 코드가 특정 패턴의 일부에 없는 것을 의미한다. 예를 들어, “AA”라는 패턴만이 존재한다

면, AA에 해당하는 비트는 1로 세팅되고, 그 외의 모든 비트들은 모두 0으로 리셋된다.

[0060] 다이렉트 필터(DF) 각각은 생성되는 과정이 다를 수 있다.

[0061] 구체적으로, 제1 다이렉트 필터(DF1)는 2바이트 이상 길이의 패턴들의 맨 앞의 2바이트에 대한 정보를 가지고 있다. 예를 들어, “UP”, “ATTACK”, “BOMB” 및 “EXPLOSION” 이라는 패턴이 있다고 가정하면 해당 다이렉트 필터에는 “UP”, “AT”, “BO” 및 “EX”에 해당하는 비트들만 1로 세팅되고, 나머지 모든 비트들은 0으로 리셋된다.

[0062] 제2 다이렉트 필터(DF2)는 4바이트 이상 길이의 패턴들의 두 번째 2바이트에 대한 정보를 가지고 있다. 예를 들어, “UP”, “ATTACK” 및 “BOMB”, “EXPLOSION” 이라는 패턴이 있다고 가정하면 해당 다이렉트 필터에는 “TA”, “MB” 및 “PL”에 해당하는 비트들만 1로 세팅되고, 나머지 모든 비트들은 0으로 리셋된다.

[0063] 제3 다이렉트 필터(DF3)는 8바이트 이상 길이의 패턴들의 네 번째 2바이트에 대한 정보를 가지고 있다. 예를 들어, “UP”, “ATTACK” 및 “BOMB”, “EXPLOSION” 이라는 패턴이 있다고 가정하면 해당 다이렉트 필터에는 “IO”에 해당하는 비트들만 1로 세팅되고, 나머지 모든 비트들은 0으로 리셋된다.

[0064] 제4 다이렉트 필터(DF4)는 2바이트 이상 4바이트 미만 길이의 패턴들의 맨 앞의 2바이트에 대한 정보를 가지고 있다. 예를 들어, “UP”, “ATTACK”, “BOMB” 및 “EXPLOSION” 이라는 패턴이 있다고 가정하면 해당 다이렉트 필터에는 “UP”에 해당하는 비트들만 1로 세팅되고, 나머지 모든 비트들은 0으로 리셋된다.

[0065] 제5 다이렉트 필터(DF5)는 4바이트 이상 8바이트 미만 길이의 패턴들의 맨 앞의 2바이트에 대한 정보를 가지고 있다. 예를 들어, “UP”, “ATTACK”, “BOMB” 및 “EXPLOSION” 이라는 패턴이 있다고 가정하면 해당 다이렉트 필터에는 “TA” 및 "MB”에 해당하는 비트들만 1로 세팅되고, 나머지 모든 비트들은 0으로 리셋된다.

[0066] 여기서, 제4 다이렉트 필터(DF4)는 첫 번째 컴팩트 테이블의 인덱스를 계산하는데 사용된다. 또한, 제5 다이렉트 필터(DF5)는 알고리즘의 성능을 높이는데 사용된다. 즉, 제5 다이렉트 필터(DF5)가 없어도 실시 형태에 따른 다중 패턴 매칭 알고리즘은 동작할 수 있다.

[0067] 일반적으로 침입탐지 시스템(Intrusion Detection System, IDS)에서는 관리상의 편의를 위하여 패턴마다 아이디(identification, ID)를 부여한다. 그리고 이를 패턴 아이디(Pattern ID, PID)(70)라고 부른다. 컴팩트 테이블(Compact Table, CT)은 패턴들의 아이디인 패턴 아이디(70)를 저장하고 있는 구조체이다.

[0068] 다이렉트 필터(DF)를 통해서는 패턴의 존재여부만을 알 수 있기 때문에, 실제로 어떤 패턴이 문자열에 들어 있는지를 알기 위해서는 해당 패턴의 패턴 아이디(70)를 알아야 한다.

[0069] 컴팩트 테이블(CT)은 문자열 내에 존재하는 패턴들의 패턴 아이디(70)를 기록하기 위한, 즉, 어떤 패턴이 문자열에 존재하는지를 알기 위한 구조체이다.

[0070] 컴팩트 테이블(CT)은 하나 이상일 수 있다. 일 예로, 실시 형태에 따른 다중 패턴 매칭 알고리즘에는 총 3종류의 컴팩트 테이블이 사용될 수 있으며, 그 구조 또한 서로 다르다.

[0071] 먼저, 패턴의 길이가 2바이트 이상 4바이트 미만인 패턴 그룹으로 나눌 수 있고, 패턴의 길이가 4바이트 이상 8바이트 미만인 패턴 그룹으로 나눌 수 있고, 마지막으로, 패턴의 길이가 8바이트 이상인 패턴 그룹으로 나눌 수 있다.

[0072] 구체적으로, 제1 컴팩트 테이블(CT1)은 패턴의 길이가 2바이트 이상 4바이트 미만인 패턴들에 대한 컴팩트 테이블이다. 제1 컴팩트 테이블(CT1)은 길이가 2바이트 이상 4바이트 미만인 패턴들에 대한 패턴 아이디(70)만을 갖고 있고, 제2 컴팩트 테이블(CT2)은 길이가 4바이트 이상 8바이트 미만인 패턴들에 대한 패턴 아이디(70)만을 갖고 있고, 제3 컴팩트 테이블(CT3)은 길이가 8바이트 이상인 패턴들에 대한 패턴 아이디만을 갖고 있다.

[0073] 이하에서는, 예를 들어 설명하도록 한다.

[0074] 도 3은 패턴의 구성을 도시한 일 예이고, 도 4는 도 3에 대한 제4 다이렉트 필터(DF4)이다.

[0075] 예를 들어, 패턴의 구성이 도 3과 같다고 가정한다면, 제4 다이렉트 필터(DF4)와 제1 컴팩트 테이블(CT1)은 도 4와 같을 수 있다. 구체적으로, 제1 컴팩트 테이블(CT1)은 제4 다이렉트 필터(DF4)에 있는 비트의 개수를 셈으

로써 구할 수 있다.

[0076] 도 5는 도 3에 대한 제2 컴팩트 테이블(CT2)이다.

[0077] 제2 컴팩트 테이블(CT2)은 도 5와 같을 수 있다. 구체적으로, 제2 컴팩트 테이블(CT2)은 4바이트 이상 8바이트 미만인 패턴들에 대해서는 앞 4바이트만을 문자열과 비교하여 그 존재를 확인한다. 즉, 7바이트 패턴의 경우 7바이트 전체가 아닌 맨 앞 4바이트만 문자열에 존재하여도 그 패턴이 존재한다고 판단하는 것이다. 이렇게 하는 이유는 거짓 양성(False positive)를 높이는 대신 성능을 높이고자 하기 위함이다.

[0078] 도 5를 보면, 4바이트 이상 8바이트 미만의 패턴에는 “ATTACK”과 “BOMB”만이 존재한다. 그래서 제2 컴팩트 테이블(CT2)에서 “ATTA”와 “BOMO”에 해당하는 버켓(50)에만 패턴 아이디(70)가 달리고, 그외의 버켓(50)에는 패턴 아이디(70)가 달리지 않는다.

[0079] 각 버켓(50)의 오른쪽에 있는 비트(15)는 해당 4바이트 패턴이 8바이트 이상의 패턴의 일부인지 아닌지를 표현하는 비트를 의미한다. 예를 들어, 위에서 예를 든 패턴 집합에는 8바이트 이상 길이의 패턴은 “EXPLOSION” 하나 밖에 없다. 따라서, 컴팩트 테이블(CT2)에서 맨 아래 “EXPL”에 해당하는 버켓(50)에만 비트(15)가 1로 세팅되고, 그 이외의 모든 버켓(50)에는 비트(15)가 0으로 리셋되어 있다. 비트(15)는 제3 컴팩트 테이블(CT3)를 검사할지 말지를 결정하는데 사용될 수 있다.

[0080]

[0081] 도 6은 도 3에 대한 제3 컴팩트 테이블(CT3)이다.

[0082] 제3 컴팩트 테이블(CT3)은 도 6과 같을 수 있다. 구체적으로, 제3 컴팩트 테이블(CT3)에는 길이가 8바이트 이상인 패턴들에 대한 패턴 아이디(70)들만 있으므로, 제3 컴팩트 테이블(CT3)에는 패턴 “EXPLOSION”에 대한 패턴 아이디(70) ‘1004’만이 존재한다.

[0083] 버켓(50)의 “IO”는 원래 패턴 “EXPLOSION”의 4번째 2바이트로부터 나온 것이다.

[0084] “IO”에 해당하는 버켓(50)의 비트(15)는 제2 컴팩트 테이블(CT2)의 인덱스인데, 패턴 “EXPLOSION”의 앞 4바이트인 “EXPL”의 제2 컴팩트 테이블(CT2)의 인덱스(35)가 11이기 때문에 “IO”에 해당하는 버켓(50)의 비트(15)가 11이 된다. 이는 중복되는 패턴을 처리하기 위해 존재하는 것이다.

[0085] 도 7은 실시 형태에 따른 다중 패턴 매칭 알고리즘의 동작을 설명하기 위한 패턴 집합의 일 예이고, 도 8은 도 7에 대한 제1 내지 제5 다이렉트 필터(DF1~DF5)이다.

[0086] 도 8을 참조하면, 제1 다이렉트 필터(DF1)에는 모든 패턴들의 맨 앞 2바이트에 대한 정보가 들어있고, 제2 다이렉트 필터(DF2)에는 길이가 4바이트 이상인 패턴들의 두 번째 2바이트에 대한 정보가 들어있고, 제3 다이렉트 필터(DF3)에는 길이가 8바이트 이상인 패턴들의 네 번째 2바이트에 대한 정보가 들어있고, 제4 다이렉트 필터(DF4)에는 길이가 2바이트 이상 4바이트 미만인 패턴들의 첫 번째 2바이트에 대한 정보가 들어있고, 제5 다이렉트 필터(DF5)에는 길이가 4바이트 이상 8바이트 미만인 패턴들의 두 번째 2바이트에 대한 정보가 들어있다.

[0087] 도 9는 도 7에 대한 제1 내지 제3 컴팩트 테이블(CT1~CT3)이다.

[0088] 도 9를 참조하면, 제1 컴팩트 테이블(CT1)에는 길이가 2바이트 이상 4바이트 미만인 패턴들의 패턴 아이디(70)가 담겨있고, 제2 컴팩트 테이블(CT2)에는 길이가 4바이트 이상 8바이트 미만인 패턴들의 패턴 아이디(70)가 담겨있고, 제3 컴팩트 테이블(CT3)에는 길이가 8바이트 이상인 패턴들의 패턴 아이디(70)가 담겨있다.

[0089] 먼저, 도 8의 <DF4>와 도 9의 <CT1>을 참조하면, 제1 컴팩트 테이블(CT1)의 인덱스(35)는 제4 다이렉트 필터(DF4)에서 해당 비트가 몇 번째인지를 셈으로써 계산될 수 있다. 구체적으로, 제4 다이렉트 필터(DF4)에서 인덱스(30) ‘ZZ’에 해당하는 비트(10) 1은 첫 번째로 등장하는 1임을 알 수 있다. 그렇기 때문에, 제1 컴팩트 다이렉트(CT1)에서의 인덱스(35)는 첫 번째, 즉, 0이다.

[0090] 도 9의 오른쪽에 도시된 <CT2>을 참조하면, 도 7에는 4바이트 이상 8바이트 미만의 패턴은 ‘AABB’로 단 하나 밖에 없기 때문에, 제2 컴팩트 테이블(CT2)에는 하나의 패턴 아이디(70)만 들어간다. 또한, 도 7에는 ‘AAAA’, ‘AABB’, ‘CCCC’, ‘JJJJ’로 시작하는 8바이트 이상 길이의 패턴들이 존재하기 때문에, 해당 버켓(50)의 비트(15)는 1로 세팅되고, 그 이외의 버켓(50)의 비트(15)는 0으로 리셋된다. 제2 컴팩트 테이블(CT2)의 인덱스(35)는 제1 다이렉트 필터(DF1)와 제2 다이렉트 필터(DF2)의 정보를 통해 계산될 수 있다. 구체적으로, 제1 다

이렉트 필터(DF1)의 비트(10)가 1인 인덱스(30)와 제2 다이렉트 필터(DF2)의 비트(10)가 1인 인덱스(30)의 정보를 통해 순서대로 나열('AAAA' ~' ZZJJ')될 수 있다.

[0091] 도 9의 <CT3>을 참조하면, 도 7의 중복되는 패턴들에 대해서는 해당 패턴들의 패턴 아이디(70)를 어레이(array) 형태로 보관한다. 구체적으로, 패턴' AABBCCDD' (1003)와 'AABBDDDD' (1007)는 첫 번째, 두 번째, 네 번째 2바이트의 정보가 일치하기 때문에, 제3 컴팩트 테이블(CT3)의 한 버켓(50)에 두 패턴의 패턴 아이디(70)를 도 9의 <CT3>에서와 같이, 어레이 형태로 보관한다. 여기서, 도 9의 <CT3>의 인덱스(35)는 도 8의 <DF3>의 순서를 셈으로써 계산되고, 제3 컴팩트 테이블(CT3)의 비트(15)는 제2 컴팩트 테이블(CT2)의 인덱스(35)로 찾아낼 수 있다.

[0092] 도 10은 도 7에서 2바이트 패턴의 존재가 확인되는 과정을 설명하기 위한 도면이다. 여기서, 도 10의 왼쪽에 도시된 패킷의 페이로드(Payload)는 무빙 윈도우가 처음부터 한 바이트씩 이동하다가 'ZZ' 에 도달한 상태를 나타낸다. 따라서, 'ZZ' 의 패턴 아이디(70)는 '1001' 로 기록되어야 한다.

[0093] 도 10을 참조하여 2바이트 패턴의 존재가 확인되는 과정을 설명하면, 먼저, ①페이로드의 'ZZ' 를 데시멀(decimal) 값 또는 정수 값으로 바꾸어 제1 다이렉트 필터(DF1)의 인덱스(30)를 계산하고, ②제1 다이렉트 필터(DF1)의 해당 인덱스(30)의 비트(10)를 참조하고, ③해당 비트(10)가 1로 세팅되어 있기 때문에 제1 다이렉트 필터(DF1)를 통과하고, ④제4 다이렉트 필터(DF4)에서의 해당 비트(10)의 1이 제4 다이렉트 필터(DF4)에서 몇 번째로 등장하는 1인지를 확인하여 제1 컴팩트 테이블(CT1)의 인덱스(35)를 계산하고, ⑤제1 컴팩트 테이블(CT1)을 참조하여 패턴 아이디(70) '1001' 를 확인하고 기록한다.

[0094] 도 11은 도 7에서 4바이트 또는 8바이트 패턴의 존재가 확인되는 과정을 설명하기 위한 도면이다. 여기서, 도 11의 오른쪽 위에 도시된 패킷의 페이로드에는 'AABB' 와 'AABB**DD' 가 있기 때문에 패턴 아이디(70)는 '1002', '100'3', '1007' 이 기록되어야 한다. 이하에서는 단계별로 '1002', '1003', '1007' 가 패턴 아이디(70)로 기록되는 과정을 설명한다.

[0095] 도 11를 참조하여 4바이트 또는 8바이트 패턴의 존재가 확인되는 과정을 설명하면, 먼저, ① 페이로드의 현재 위치에 해당하는 'AA' 를 데시멀 값 또는 정수 값으로 바꾸어 제1 다이렉트 필터(DF1)의 인덱스(30)를 계산하고, ②제1 다이렉트 필터(DF1)의 해당 인덱스(30)의 비트(10)를 참조하고, 해당 비트(10)가 1이기 때문에 제1 다이렉트 필터(DF1)를 통과하고, ③페이로드의 무빙 윈도우를 2바이트 이동시켜 이동한 위치에서 2바이트 길이에 해당하는 'BB' 의 데시멀 값 또는 정수 값을 계산하여 제2 다이렉트 필터(DF2)의 인덱스(30)를 구하고, ④제2 다이렉트 필터(DF2)의 해당 비트(10)가 1이기 때문에 제2 다이렉트 필터(DF2)를 통과하고, ⑤제1 다이렉트 필터(DF1)에서의 인덱스(30) 'AA' 에 해당하는 비트(10) 1이 제1 다이렉트 필터(DF1)에서 몇 번째 1인지 확인하고(첫 번째 1), ⑥제2 다이렉트 필터(DF2)에서의 인덱스(30) 'BB' 에 해당하는 비트(10) 1이 몇 번째 1인지를 확인하고(두 번째 1), ⑦앞서 ⑤, ⑥에서 구한 값(첫 번째 1, 두 번째 1)들을 이용하여 제2 컴팩트 테이블(CT2)의 인덱스(30)를 계산하고(0 x 4 +1 =1, 여기서, 0은 제1 다이렉트 필터(DF1)의 첫 번째 1이라는 것을 의미하고, 4는 제2 다이렉트 필터(DF2)에 있는 모든 1의 개수를 의미하고, 1은 제2 다이렉트 필터(DF2)의 두 번째 1이라는 것을 의미한다.), ⑧ 앞서 계산 값이 1이기 때문에, 제2 컴팩트 테이블(CT2)의 인덱스(35)가 1인 버켓(50)에 접근하여 패턴 아이디(70)를 기록하고, ⑨제2 컴팩트 테이블(CT2)의 인덱스(35)가 1인 버켓(50)의 비트(15)가 1로 세팅되어 있기 때문에, 8바이트 패턴에 대해서도 탐색을 해야 함을 알 수 있다. 그래서 페이로드의 무빙 윈도우를 4바이트 더 이동시켜 이동한 위치에 2바이트 길이에 해당하는 'DD' 의 데시멀 값 또는 정수 값을 계산하여 제3 다이렉트 필터(DF3)의 인덱스(30)를 구하고, ⑩제3 다이렉트 필터(DF3)의 해당 비트(10)가 1이기 때문에 제3 다이렉트 필터(DF3)를 통과하고, ⑪제3 다이렉트 필터(DF3)에서의 인덱스(30) 'DD' 에 해당하는 비트(10) 1이 제3 다이렉트 필터(DF3)에서 몇 번째 1인지를 확인하고(두 번째 1), ⑫제3 컴팩트 테이블(CT3)의 버켓(50) 'DD' 의 비트(15)를 참조하고, ⑬비트(15) 안에 ⑦에서 계산한 계산 값 1이 어디 있는지를 확인하고, ⑭계산 값 1에 해당하는 비트(15)가 있으므로, 해당 버켓(50) 'DD' 의 비트(15) 1에 있는 모든 패턴 아이디(70)들을 기록한다.

[0096] 실시 형태에 따른 다중 패턴 매칭 알고리즘은 네트워크 침입 탐지 시스템(Network Instrusion Detection System, NIDS) 등에 사용될 수 있다. 네트워크 침입 탐지 시스템에서 문자열을 검색하는 과정에서 2바이트 길이의 무빙 윈도우가 처음부터 한 바이트씩 이동하며 탐색을 하는데, 대부분의 무빙 윈도우가 제1 다이렉트 필터

(DF1)에서 걸러지기 때문에(다이렉트 필터(DF)의 대부분의 비트(10)는 0으로 리셋되어 있기 때문에), 굉장히 적은 숫자의 무빙 윈도우들만이 컴팩트 테이블(CT)에 접근한다. 그래서 대부분의 문자열 탐색이 제1 다이렉트 필터(DF1) 하나만으로도 가능하게 된다. 다이렉트 필터의 크기는 CPU의 L1 캐시(cache)에도 들어갈 정도로 작기 때문에 문자열을 탐색하는 과정에서 기존 알고리즘 대비 더 적은 캐시 미스를 유발하며, 결과적으로 더 높은 성능을 낼 수 있게 한다. 또한, 다이렉트 필터나 컴팩트 테이블에는 패턴들의 아스키 코드가 직접 저장되는 것이 아니고, 저장되는 내용 또한 패턴의 전부가 아닌 일부만이기 때문에, 기존 알고리즘보다 더 적은 메모리 공간만을 필요로 한다.

[0097] 이와 같이, 실시 형태에 따른 다중 패턴 매칭 알고리즘은, 패턴의 전체를 가지고 문자열을 탐색하는 아호 코라식 알고리즘과는 다르게, 패턴의 일부만을 가지고 탐색하기 때문에 더 적은 메모리 공간을 필요로 하는 이점이 있다.

[0098] 또한, 실시 형태에 따른 다중 패턴 매칭 알고리즘은, 문자열의 처음부터 끝까지 한 바이트씩 옮겨가며 탐색을 하고 이렇게 옮겨지는 포인트를 윈도우라고 하였을 때, 대부분의 윈도우는 크기가 매우 작은 다이렉트 필터에서 걸러진다. 그렇기 때문에, 기존 아호 코라식 알고리즘에 비해 적은 캐시 미스(cache miss)를 유발하여 높은 성능을 낼 수 있는 이점이 있다.

[0099] 이상 첨부된 도면을 참조하여 본 발명의 실시 형태를 설명하였으나 이는 단지 예시일 뿐 본 발명을 한정하는 것이 아니며, 본 발명이 속하는 분야의 통상의 지식을 가진 자라면 본 실시 형태의 본질적인 특성을 벗어나지 않는 범위에서 이상에 예시되지 않은 여러 가지의 변형과 응용이 가능함을 알 수 있을 것이다. 예를 들어, 실시 형태에 구체적으로 나타난 각 구성 요소는 변형하여 실시할 수 있는 것이다. 그리고 이러한 변형과 응용에 관계된 차이점들은 첨부된 청구 범위에서 규정하는 본 발명의 범위에 포함되는 것으로 해석되어야 할 것이다.

부호의 설명

[0100] 10, 15: 비트

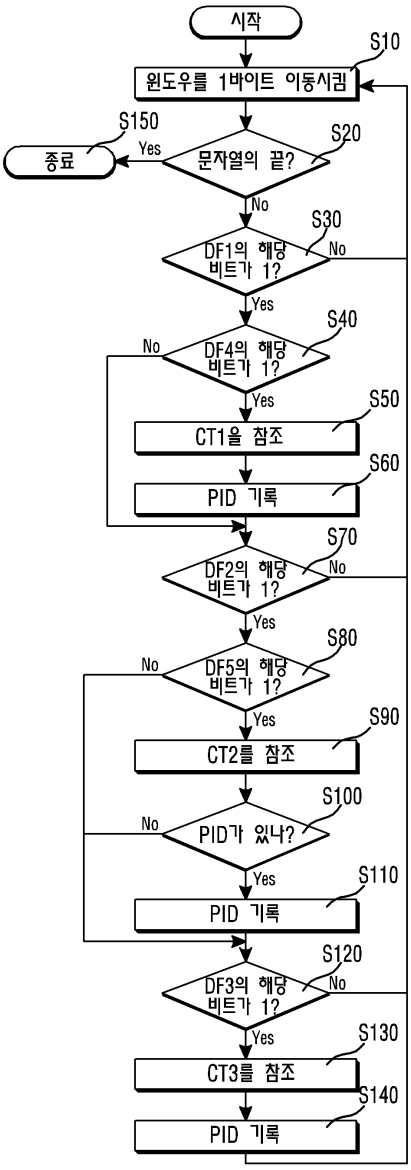
30, 35: 인덱스

50: 버켓

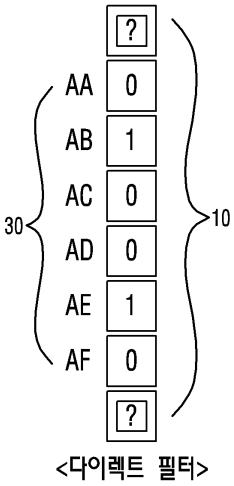
70: 패턴 아이디

도면

도면1


시작
S10
윈도우를 1바이트 이동시킴
S150
종료
Yes
S20
문자열의 끝?
No
S30
DF1의 해당 비트가 1?
No
Yes
S40
No
DF4의 해당 비트가 1?
Yes
S50
CT1을 참조
S60
PID 기록
S70
DF2의 해당 비트가 1?
No
Yes
S80
No
DF5의 해당 비트가 1?
Yes
S90
CT2를 참조
S100
No
PID가 있나?
Yes
S110
PID 기록
S120
DF3의 해당 비트가 1?
No
Yes
S130
CT3를 참조
S140
PID 기록

도면2

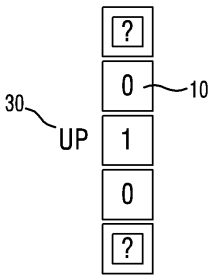


<다이렉트 필터>

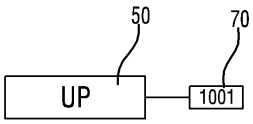
도면3

<PID>	<패턴>	<길이>
1001	"UP"	2Byte
1002	"ATTACK"	6Byte
1003	"BOMB"	4Byte
1004	"EXPLOSION"	9Byte

도면4

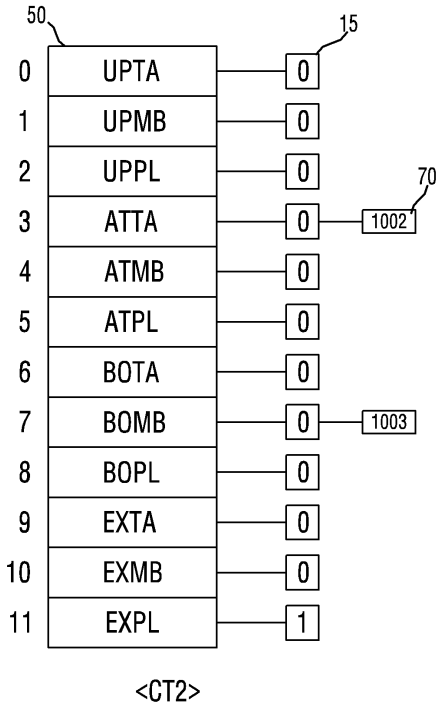


<DF4>



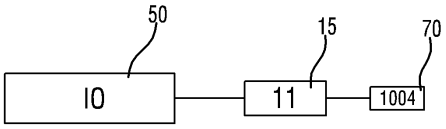
<CT1>

도면5



<CT2>

도면6

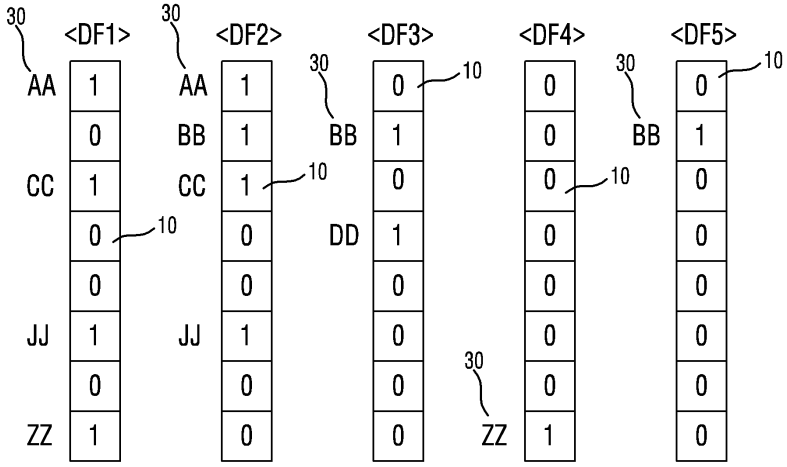


<CT3>

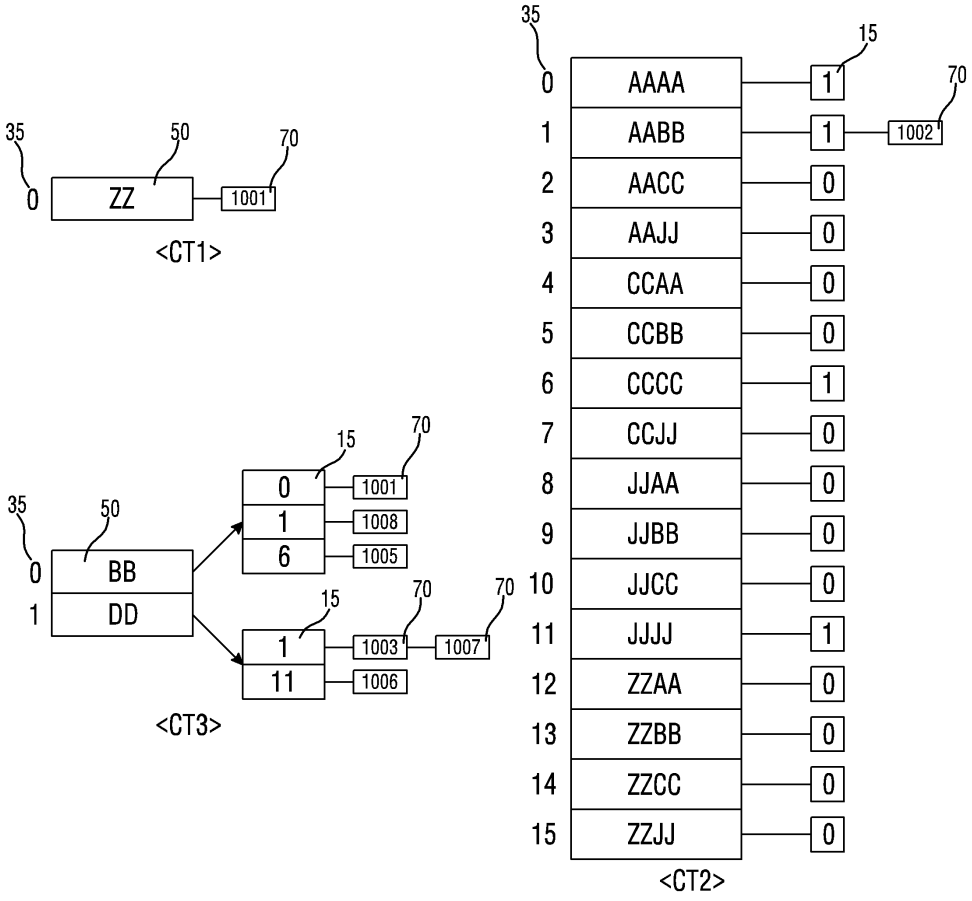
도면7

<PID>	<패턴>	<길이>
1001	"ZZ"	2Byte
1002	"AABB"	4Byte
1003	"AABBCCDD"	8Byte
1004	"AAAABBBB"	8Byte
1005	"CCCCBBBB"	8Byte
1006	"JJJJKKDD"	8Byte
1007	"AABBDDDD"	8Byte
1008	"AABBBBBB"	8Byte

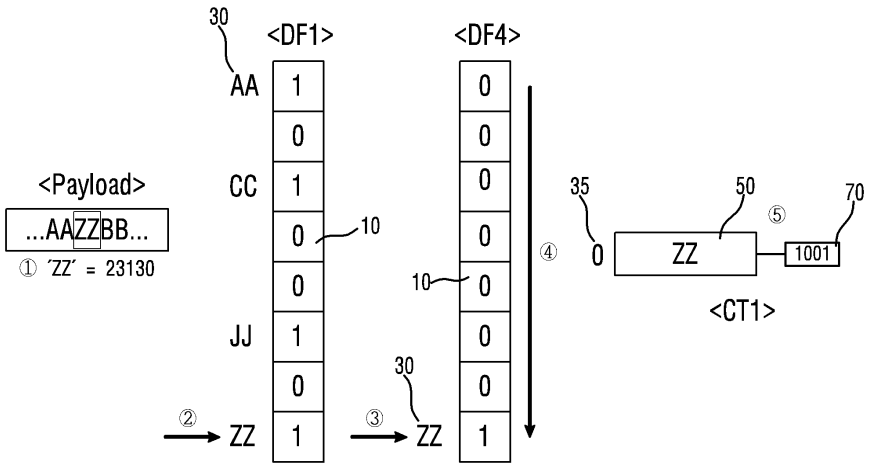
도면8


30
<DF1>
AA
CC
JJ
ZZ
10
30
<DF2>
AA
BB
CC
JJ
10
30
<DF3>
BB
DD
10
30
<DF4>
ZZ
10
30
<DF5>
BB
10

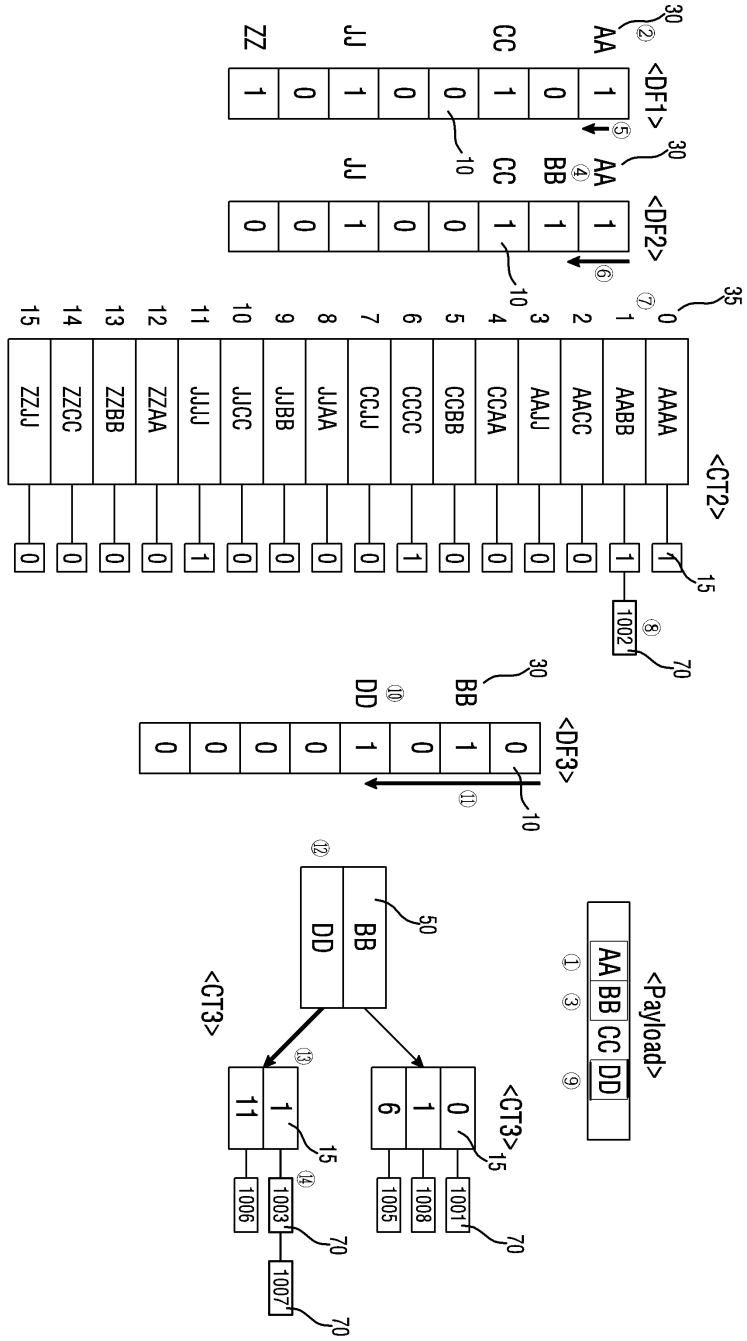
도면9


35
50
70
0
ZZ
1001
<CT1>
35
50
0
BB
1
DD
15
70
0
1
6
1001
1008
1005
15
70
70
1
11
1003
1007
1006
<CT3>
35
0
AAAA
1
AABB
2
AACC
3
AAJJ
4
CCAA
5
CCBB
6
CCCC
7
CCJJ
8
JJAA
9
JJBB
10
JJCC
11
JJJJ
12
ZZAA
13
ZZBB
14
ZZCC
15
ZZJJ
15
70
1002
<CT2>

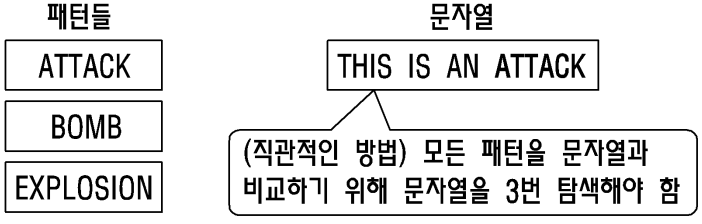
도면10


<Payload>
...AAZZBB...
① 'ZZ' = 23130
30
<DF1>
AA
CC
JJ
ZZ
②
10
<DF4>
③
30
ZZ
④
35
0
50
ZZ
⑤
70
1001
<CT1>

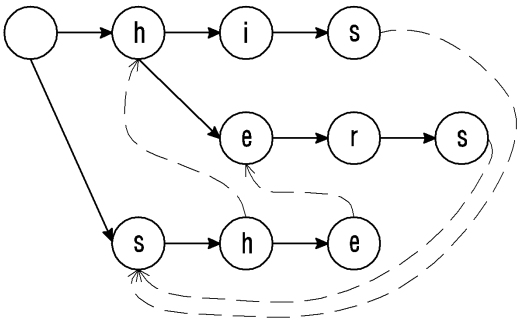
도면11


ZZ
JJ
CC
AA
<DF1>
<DF2>
BB
DD
<DF3>
<CT2>
<CT3>
<Payload>
AA BB CC DD
AAAA
AABB
AACC
AAJJ
CCAA
CCBB
CCCC
CCJJ
JJAA
JJBB
JJCC
JJJJ
ZZAA
ZZBB
ZZCC
ZZJJ
1002
1003
1004
1005
1006
1007
1008
1001
30
35
10
15
70
50

도면12


패턴들
ATTACK
BOMB
EXPLOSION
문자열
THIS IS AN ATTACK
(직관적인 방법) 모든 패턴을 문자열과
비교하기 위해 문자열을 3번 탐색해야 함

도면13



【심사관 직권보정사항】

【직권보정 1】

【보정항목】 명세서

【보정세부항목】 특허문헌

【변경전】

10-2014-0128163

【변경후】

10-2014-0128153

【직권보정 2】

【보정항목】 청구범위

【보정세부항목】 청구항 제14항 네 번째 줄

【변경전】

패턴 아이디를 패턴 아이디를

【변경후】

패턴들의 패턴 아이디를