독립 부분 매칭에 의한 행렬 기반 고성능 패턴 매칭 방법에 관한 연구

정회원 정 우 석*, 종신회원 권 택 근**

The Study on matrix based high performance pattern matching by independence partial match

Woo-Sug Jung* *Regular Member*, Taeck-Geun Kwon** *Lifelong Member*

요 약

본 논문에서는 수 Gbps 네트워크 트래픽에서 실시간 침입 탐지를 위한 패턴 매칭 방법인 MDPI를 제안한다. MDPI는 패킷 전달 순서가 유지되지 않는 경우 버퍼링, 재배열 및 재조립에서 발생하는 오버헤드 문제를 해결하기 위해 독립 부분 매칭에 의한 행렬 기반의 패턴 매칭 방법이다. MDPI는 SNORT 룰셋(Rule Set)의 평균 길이인 17바이트의 경우 w=4 바이트에서는 61%, w=8 바이트인 경우는 50%의 TCAM 메모리 효율이 증가 되었다. 또한 MDPI는 10.941Gbps 패턴 검사 속도와 5.79 LC/Char 하드웨어 자원을 소모함으로써 하드웨어 복잡성 대비 성능 측면에서 최적화된 결과를 얻었다. 따라서 본 논문에서는 하드웨어 비용 절감에 의해 가격 효율적인 고성능 침입 탐지 기술을 제안한다.

Key Words : NIDS, Packet Inspection, Pattern Matching, Partial Match, Matrix

ABSTRACT

In this paper, we propose a matrix based real-time pattern matching method, called MDPI, for real-time intrusion detection on several Gbps network traffic. Particularly, in order to minimize a kind of overhead caused by buffering, reordering, and reassembling under the circumstance where the incoming packet sequence is disrupted, MDPI adopts independent partial matching in the case dealing with pattern matching matrix. Consequently, we achieved the performance improvement of the amount of 61% and 50% with respect to TCAM method efficiency through several experiments where the average length of the Snort rule set was maintained as 9 bytes, and w=4 bytes and w=8bytes were assigned, respectively, Moreover, we observed the pattern scan speed of MDPI was 10.941Gbps and the consumption of hardware resource was 5.79LC/Char in the pattern classification of MDPI. This means that MDPI provides the optimal performance compared to hardware complexity. Therefore, by decreasing the hardware cost came from the increased TCAM memory efficiency, MDPI is proven the cost effective high performance intrusion detection technique.

Ⅰ. 서 론

최근 인터넷 대중화는 인터넷 트래픽의 급격한 증가뿐 아니라 네트워크를 통한 직접적인 침입에서

※ 본 연구는 지식경제부의 IT 기술 혁신 사업의 일환으로 수행하였음[2006-S-601-01, u-City 적용 센서 네트워크 시스템 개발]
※ 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음 (NIPA-2009-(C1090-0902 -0016))
* 한국전자통신연구원 USN응용기술연구팀(wsjung@etri.re.kr), ** 충남대학교 전기 정보통신공학부(tgkwon@cnu.ac.kr)
논문번호 : KICS2009-07-286, 접수일자 : 2009년 7월 8일, 최종논문접수일자 : 2009년 9월 15일

간접적인 침입형태의 변화를 가져왔다. 간접적인 침입은 웜 바이러스나 백도어 프로그램을 패킷에 은닉하여 시스템에 유입시킨 후, 정보 유출 및 시스템을 파괴하는 침입 형태를 의미한다. 이러한 네트워크를 통한 침입으로부터 자원의 무결성, 기밀성 및 가용성을 저해하는 일련의 행동들과 보안 정책을 위반하는 행위를 실시간으로 탐지하고 보고하는 시스템을 NIDS(Network Intrusion Detection System)라 한다. 일반적으로, NIDS는 실시간으로 침입 탐지를 수행하기 위해 패턴 매칭와 통계적인 분석 방법이 사용된다. 본 논문에서는 수 Gbps 성능을 가지는 네트워크에서 실시간 침입 탐지를 위한 패턴 매칭 방법에 대해 제안한다. 패턴 매칭은 기 정의된 룰셋을 이용하여 네트워크 트래픽으로부터 시그너처를 추출한다. 시그너처 기반 NIDS의 성능은 시그너처와 패킷을 비교하기 위해 사용되는 스트링 매칭 알고리즘의 속도에 의해 결정된다[13]. Snort는 전체 성능의 31%와 웹 관련 트래픽의 80%가 스트링 매칭에 관련된 것이다[5],[13]. Snort는 패킷 분석을 통한 네트워크 침입 탐지를 위한 무료 오픈 소서 코드이다. Snort 2.0 성능은 구현되는 스트링 매칭 알고리즘에 따라 최대 500%까지 향상시킬 수 있다[13],[14]. 실시간 침입탐지를 위해서는 알고리즘 기반의 소프트웨어 NIDS와 하드웨어 NIDS 방법이 있다. 105개 룰을 가지는 소프트웨어 NIDS인 Hogwash는 100Mbps 이더넷 정합을 가지는 733 MHz 셀러론 PC 환경에서 25.59Mbps에서 50.15 Mbps의 침입 탐지 성능을 제공한다[7]. TCP/IP 패킷의 최소 길이가 40바이트라고 가정할 때 실시간 패턴 매칭을 위해 네트워크 대역폭이 2.5Gbps에서는 7.81Mpps, 10Gbps에서는 31.25Mpps, 40Gbps에서는 125Mpps의 침입 탐지 검색 속도가 필요하다. 그러나 Hogwash는 0.079Mpps에서 0.156Mpps의 침입 탐지 속도를 지원한다. 소프트웨어 기반의 NIDS는 수 Gbps 네트워크에서 실시간 침입 탐지 기능을 제공하기에 적합하지 않다. 즉, 수 Gbps 네트워크에서 실시간 침입 탐지를 검출하기 위해서는 하드웨어 기반의 NIDS가 필요하다. 본 논문에서 제안한 패턴 매칭 방법은 다중 패킷에서 발생하는 재배열 및 재조립에 의한 하드웨어 오버헤드를 최소화시킴으로써 가격 효율적인 하드웨어 기반의 NIDS를 위한 독립 부분 매치에 의한 행렬 기반의 실시간 패턴 매칭 방법인 MDPI(Matrix based Deep Packet Inspection)를 제안한다. MDPI는 패킷에 은닉되어 유입되는 공격 패턴인 시그너처 기반의 실시간 침입 탐지 과정을 수행하며, 고속 패턴 검색을 위해 TCAM을 사용한다. 제2장에서는 TCAM의 개요와 네트워크에서 발생하는 트래픽 분석을 통해 하드웨어 기반 NIDS에서 발생하는 오버헤드에 대해 설명한다. 제3장에서는 MDPI에 의한 침입 탐지 과정 및 오버헤드 최소화 방법 그리고 성능 분석에 대해 기술한다. 제4장에서는 MDPI 기능 구현에 대해 설명하고 제5장은 MDPI의 성능에 대해 기술하며, 결론 및 향후 연구 과제는 제6장에서 설명한다.

Ⅱ. 관련 연구

2.1 네트워크 트래픽

네트워크를 통해 전송되는 데이터는 전송 편의를 위해 일정한 크기를 가지는 패킷 단위로 나뉘어 전송된다. 단일 패킷(Single Packet)은 데이터가 한 개 패킷에 전송되는 패킷을 의미하며, 2개 이상의 패킷에 전송되는 경우를 다중 패킷(Multiple Packet)이라고 한다. 다중 패킷은 전달되는 순서가 유지되는 경우를 순차 패킷(Sequence Packet)이라고 하며, 패킷은 네트워크의 상황에 따라 전달 순서가 유지 되지 않는 경우 비순차 패킷(Out-Of-Sequence Packet)이라고 한다. 비순차 패킷은 네트워크에서 불필요한 재전송을 유발하여 네트워크의 전송 속도 및 성능을 저하 시키며, 재조립을 위한 버퍼 및 부가적인 프로세서 성능을 요구하게 된다. Bennett[1], Paxon[2], Jaiswal[8] 및 Wang[11]에 의하면, 비순차 패킷은 전체 네트워크 패킷의 2-5%를 차지한다. Wang에 의하며, 비순차 패킷은 전체 사이트의 5.79%에서만 발생하지만, 발생 빈도는 전체 사이트의 20%에 집중된다. 비순차 패킷은 전송 경로의 변경에 의해 발생하는 재배열 패킷과 패킷 손실에 의해 발생하는 재전송 패킷으로 구분된다. 재배열 패킷의 90%는 5.1ms의 시간 간격과 95.3%는 2개 이하의 패킷 간격을 가진다. 이에 반해, 재전송 패킷의 3.5%만이 5.1ms의 시간 간격을 가지며, 1초 이상의 시간 간격을 가지는 패킷도 존재한다. 재전송 패킷의 78.8%는 3개 이상의 패킷 간격을 가지며 최대 55개 패킷 간격을 가진다[11]. 일반적으로 비순차 패킷에서 패턴 매칭을 위해서는 패킷 재조립 과정이 수반되어야 한다. 패킷 재조립을 위해서는 패킷 순서를 재배열하기 위한 부가적인 버퍼 및 고성능 프로세서가 요구된다. 따라서 재전송 패킷이 재배열 패킷에 비해 더 큰 버퍼와 고성능 프로세서가 요구된다. 본 논문에서는 단일 패킷 및 비순차 패킷을 포

함하는 다중 패킷에서 패턴 매칭을 위해 소요되는 하드웨어 자원을 최소화하기 위해 패킷 재조립 과정을 배제하는 수 Gbps 네트워크에서 실시간 패턴 매칭이 가능한 MDPI를 제안한다.

2.2 소프트웨어 기반 패턴 매칭

패턴 매칭 방법은 단일 패턴 매칭과 다중 패턴 매칭으로 구분된다. 단일 패턴 매칭은 입력되는 패킷와 한 개의 패턴이 비교되며, 다중 패턴을 추출하기 위해서 각각의 패턴에 대해 한 번씩 비교된다. 단일 패턴 매칭은 Knuth-Morris-Pratt, Boyer-Moore 등이 있으며 Brute Force 알고리즘을 기반으로 하고 있다. Brute Force 알고리즘은 n 바이트 패킷으로부터 m 바이트 패턴을 추출하기 위해서는 패턴 추출 시간은 최대 O(n+m)이다. 만일 k개 패턴을 검출하기 위해서는 패턴 추출 시간은 O(k(n+m))이 된다. 다중 패턴 매칭은 입력되는 패킷으로부터 패턴들의 집합에 해당하는 패턴들이 존재 유무를 검사한다. 한 개의 큰 패턴들은 여러 개의 작은 패턴으로 구성하고 유한 오토마타(Finite Automata)에 의해 패턴 추출 과정을 수행한다. 대표적인 패턴 매칭 방법은 Aho-Corasick 알고리즘이다. 이러한 소프트웨어 기반 패턴 매칭 방법은 범용 하드웨어 플랫폼을 이용하여 소프트웨어기반으로 패턴 추출 과정을 수행하기 때문에 병렬 처리에 제약이 발생하므로 수 Gbps의 네트워크에서는 적용하기 어렵다.

2.3 하드웨어 기반 패턴 매칭

소프트웨어 기반 패턴 매칭 방법은 수 Gbps 네트워크에서 실시간 패턴 추출이 어렵기 때문에 하드웨어 기반의 패턴 매칭이 요구된다. 하드웨어 기반의 패턴 매칭은 Bloom 필터, TCAM, CAM 등을 이용한다.

2.3.1 Bloom 필터

Dharmapurikar[10]는 병렬 Bloom 필터를 사용하는 다중 패턴 매칭 방법을 제안하였다. 병렬 Bloom 필터는 해당하는 패턴에 대한 Bloom 필터를 구성해야 한다. 그러나 이러한 병렬 Bloom 필터는 다양한 패턴 길이로 인해 하드웨어로 기능 구현 가능한 패턴의 제약을 받는다. 일반적으로 병렬 Bloom 필터는 수 백개 검사 가능한 패턴을 하드웨어로 구현할 수 있다.

2.3.2 TCAM

TCAM(Ternary Content Address Memory)은 한 클럭에 패턴 매칭이 가능한 고속 메모리이다. TCAM은 '0'와 '1' 이외에 'don't care' 상태를 이용한 패턴 매칭 과정을 지원한다. 'Don't care'상태는 패턴 매칭을 위한 테이블을 최소화할 수 있어 동일한 메모리 크기에 많은 패턴 매칭이 가능하도록 한다. TCAM에서 패턴 매칭을 위한 최소 단위를 "엔트리"라고 한다.

F. Yu[12]는 TCAM을 이용하여 PHL(Partial Hit List) 방법을 제안하고 있다. PHL은 단일 패킷에서 패턴 매칭 과정을 수행한다. Sung은 다중 패킷에서 패턴 매칭과 검색 속도 증가를 위해 m-byte jumping 윈도우 방법을 제안하고 있다. F. Yu 또는 Sung은 패킷의 순서가 유지될 경우에는 효과적이지만, 순차적인 패턴 검사에 의해 비순차 패킷의 경우에는 TCAM 엔트리가 증가한다.

Ⅲ. 제안한 패턴 매칭 방법

본 논문은 고속 네트워크에서 실시간 패턴 매칭 과정을 수행하기 위해 TCAM을 이용하는 독립 부분 매칭에 의한 행렬 기반 패턴 매칭 방법인 MDPI를 제안한다.

3.1 용어 정의

전달하고자 하는 스트링 $R=r_0,r_1,....r_{m-1}$은 m개의 서브스트링 $r=b^0,b^1,...,b^{n-1}$, $0\le n\le$m-1로 분리되어 네트워크를 통해 전달된다. 여기서 b는 바이트를 의미한다. 만일 검출하고자 하는 텍스트 $T=t_0,t_1,t_2,\ ...,t_{j-1}$가 한 개 r에서 검출되었다면 단일 서브스트링 매칭라고 하며, 두 개 이상의 r에서 T가 검출되었다면 다중 서브스트링 매칭라고 한다. 여기서 t는 서브스트링에서 검출되는 서브텍스트를 의미한다. 본 논문에서는 T가 2개 서브텍스트 $t_0,...,t_{k-1}$와 $t_k,...,t_{j-1}$로 구성되고 서브텍스트가 포함된 각 서브스트링을 r_i와 r_{i+1}라고 한다. $t_0,...,t_{k-1}=b_i^s,...,b_i^{s+k-1}$이며, $t_k,...,t_{j-1}=b_{i+1}^0,...,b_{i+1}^{j-k-1}$ 이므로 $r_i=b_i^0,...,b_i^s,...,b_i^{s+k-1}$와 $r_{i+1}=b_{i+1}^0,...,b_{i+1}^{j-k-1},...,b_{i+1}^{m-1}$이다. 다중 서브스트링 매칭에서 r_i와 r_{i+1}이 검출되는 순서가 유지되면 순차 서브스트링 매칭라고 하며, 순서가 유지되지 않는 경우를 비순차 서브스트링 매칭이라고 한다. r_i가 검출되고 r_{i+1}이 x개 r이후에 검출 되었을 때 x개 r을 서브스트링 간격 L_s라고 한다. 그림 1는 스트링 R와 텍스트 T의 조각화 과정을 나타낸 것이다.

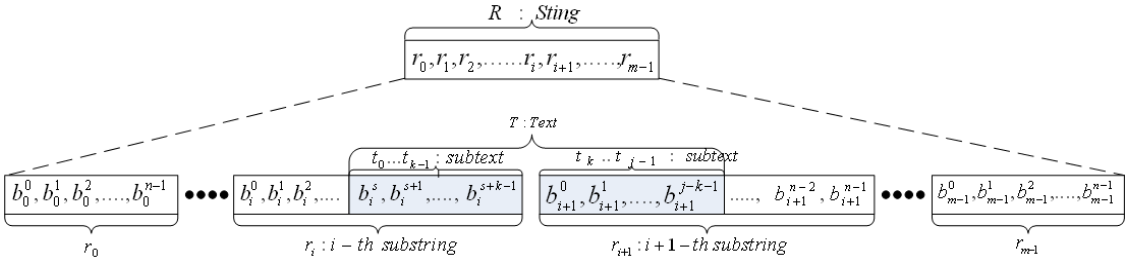


그림 1. 스트링과 텍스트의 상관 관계

3.2 문제점 정의

일반적으로 $L_s \neq 0$인 r_i와 r_{i+1}에서 T를 검출하기 위해서는 인입되는 r들의 버퍼링, 재배열 및 재조립 과정을 수반한다. 만일 L_s=x라면, 이러한 재배열 및 재조립 과정은 r*x크기의 부가 메모리가 필요할 뿐 아니라 r을 재조립하기 위한 부가적인 프로세서 성능이 요구된다. 본 논문에서는 $L_s \neq 0$인 r_i와 r_{i+1}에서 T를 검출하기 위해 요구되는 재배열 및 재조립 과정을 배제함으로써 부가 메모리 및 고성능 프로세서를 요구하지 않는 독립 부분 매칭 방법을 제안한다. MDPI는 인입되는 r로 부터 실시간으로 T를 검출하기 위해 고속 패턴 검사 메모리로 TCAM을 이용한다. 그림 2은 T="ABCDE"를 검출하기 위해 요구되는 TCAM 엔트리 및 상태를 예시하였다. TCAM 엔트리들은 상태 천이에 필요한 조건들로 표시하였다. 순차 서브스트링 매칭에서 T를 검출하기 위해서는 S_0~S_3상태가 요구된다. 그러나 비순차 서브스트링 매칭에서 T를 검출하기 위해서는 r_i와 r_{i+1}의 입력되는 순서가 유지되지 않기 때문에 추가적으로 S_4~S_7상태가 요구된다. 즉 동일한 T를 검출하기 위해 2배의 TCAM 크기를 요구한다. 본 논문에서는 $L_s \neq 0$인 서브스트링 매칭에서 발생하는 하드웨어 오버헤드를 최소화하기 위한 독립 부분 매칭에 의한 행렬 기반의 텍스트 검출 방법인 MDPI을 제안한다.

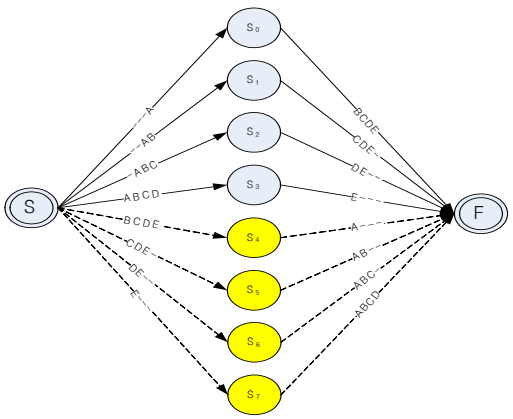


그림 2. T="ABCDE" 검출 상태도

3.3 MDPI

MDPI는 S_0: 행렬 구성 단계, S_1: 독립 부분 매칭 단계 및 S_2: 거리 검사 과정을 통해 부분 매칭에 의해 텍스트를 검출한다. 각 단계는 하드웨어로 구현하여 파이프라인 형태로 병렬 처리된다.

3.3.1 S_0 : 행렬 구성

독립 부분 매칭이란 r_i와 r_{i+1}의 인입 순서와 무관하게 t들을 검출 한 후, t가 검출된 위치 정보를 이용하여 T 검출 유무를 판단하는 방법이다. 독립 부분 매칭은 t의 검출 순서와 무관하게 T를 검출하기 때문에 T검출을 위한 TCAM 메모리 효율성 저하를 최소화할 수 있다. 본 논문에서는 독립 부분 매칭을 위해 SEM (Subtext Entries Matrix)와 MSM (Matching Status Matrix) 행렬을 구성한다. SEM은 T를 검출하기 위한 t로 구성되며, MSM은 SEM에서 검출된 t의 위치 정보를 저장한다.

3.3.1.1 SEM 구성

SEM을 구성하는 t는 w바이트 크기를 가지며, w바이트는 TCAM의 폭(Width)와 동일하다. 이러한 t는 TCAM 엔트리들로 대치될 수 있으며, p_i^j로 표기한다. i는 SEM의 행 번호를 의미하며, j는 열 번호를 의미한다. 예를 들면 p_2^3는 2번째 행, 3번째 열에 위치하는 SEM 원소를 의미한다. p_i^1는 SEM의 첫 번째 열의 원소들을 의미하며, i=1,2,...,w-1이다. p_1^j은 SEM의 첫 번째 행을 구성하는 모든 원소를 의미하며, j=1,2,..., ceil(l/w)+1이다. 여기서 l은 T의 길이를 의미한다. 그림 3은 SEM을 도시한 것이다.

$$SEM=\begin{bmatrix} p_1^1\, p_1^2 \cdots p_1^j \\ p_2^1\, p_2^2 \cdots p_2^j \\ \cdots\ \cdots\ \cdots\ \cdots \\ p_i^1\, p_i^2 \cdots p_i^j \end{bmatrix}$$

그림 3. SEM(Subtext Entries Matrix)

그림 4는 w=4이고 l=5인 T="ABCDE" 검출을 위 한 2x4 크기를 가지는 SEM을 예시한 것이다. l=5이므로 i=4이고 j=2이다. T를 검출하기 위한 필요 조건은 SEM에서 각 행을 구성하는 t가 모두 검출되는 것이다. 예를 들면 p_1^1= "---A", p_1^2="BCDE"가 모두 검출될 경우 T="ABCDE"를 구성하는 T를 검출하기 위한 t들이 모두 검출되었음을 알 수 있다. 이때 t가 검출될 때 발생하는 위치 정보들은 MSM에 저장된다.

$$SEM=\begin{bmatrix} ---A & BCDE \\ --AB & CDE- \\ -ABC & DE-- \\ ABCD & E--- \end{bmatrix}$$

그림 4. T="ABCDE"검출을 위한 SEM

3.3.1.2 MSM 구성

MSM은 p_i^j의 위치 정보이며 s_i^j로 표시한다. p_i^j와 s_i^j는 1:1 대응 관계를 가진다. 그림 5은 MSM을 도시한 것이며, MSM을 구성하는 원소 s_i^j는 다섯 개 위치 정보로 구성된다. 첫번째 위치 정보 $s_i^j(g)$는 스트링에서 T가 은닉된 r_i와 r_{i+1}의 위치이며 패킷의 시퀴언스(Sequence) 번호로 대치할 수 있다.

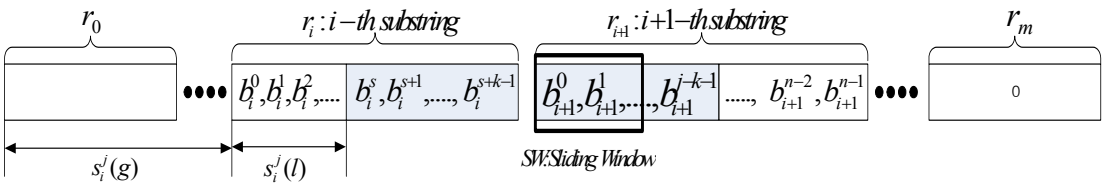
$$MSM=\begin{bmatrix} s_1^1\, s_1^2 \cdots s_1^j \\ s_2^1\, s_2^2 \cdots s_2^j \\ \cdots\ \cdots\ \cdots\ \cdots \\ s_i^1\, s_i^2 \cdots s_i^j \end{bmatrix}$$

그림 5. MSM(Matching Status Matrix)

두 번째 위치 정보인 $s_i^j(l)$는 r_i와 r_{i+1}내에서 t가 검출되는 위치를 의미한다. 본 논문에서 제안한 패턴 검출 방법은 바이트 크기를 가지는 SW(Sliding Window)를 우측으로 한 바이트씩 이동하면서 t를 검출한다 즉, $s_i^j(l)$는 r_i와 r_{i+1}에서 t가 검출될 때 SW가 우측으로 이동한 횟수를 의미한다. 그림 6은 스트링 및 서브스트링에서 $s_i^j(g)$와 $s_i^j(l)$의 상관관계를 도시한 것이다. s_i^j의 세 번째 위치정보는 $s_i^j(h)$이다. $s_i^j(h)$=1은 r_i와 r_{i+1}에서 t가 검출되었음을 의미한다. 네 번째 위치 정보 $s_i^j(b)$는 t가 r_{i+1}에서 $s_i^j(l) \le w$ 위치에서 검출되면 $s_i^j(b)$=1이 된다, 마지막 위치 정보 $s_i^j(c)$는 t가 r_l-$s_i^j(l) \le w$위치에서 검출 되었을 경우 $s_i^j(c)$=1로 설정된다. r_l은 r의 길이를 의미한다. $s_i^j(b)$와 $s_i^j(c)$는 S_2: 거리 검사 과정의 속도를 향상시키기 위한 위치 정보이다. SEM은 실시간으로 패턴 검사를 위해 TCAM상에 구현된다. 그러나 위치 정보 저장 빈도는 패턴 검사 빈도에 비해 현저히 낮으므로 SRAM 상에 구현된다.

3.3.2 S_1 : 독립 부분 매칭

S_1:독립 부분 매칭 단계는 인입되는 r에서 독립 부분 매칭에 의해 t 검출 과정과 S_2:거리 검사 단계로 천이하기 위한 ASPM(All Subtext Partial Matching)을 생성하는 과정으로 구분된다. ASPM 생성 단계는 SEM의 행을 구성하는 p_i^j가 모두 검출되었음을 의미하는 ASPM 조건을 충족 여부를 확인한다. ASPM이 1로 설정되었다는 것은 T검출 유무를 판단하기 위한 모든 t들이 검출되었음을 의미한다. MDPI는 ASPM이 1로 설정되면 S_2: 거리 검사 단계를 수행한다. t검사 과정은 r_i와 r_{i+1}에서 t를 검출하고 관련 s_i^j를 저장한다. t검사 과정은 $L_s \neq 0$



$$s_i^j(h)=\begin{cases} 0: not-match \\ 1: match \end{cases} \quad s_i^j(b)=\begin{cases} 0: not-match\ at\ last\ part\ of\ substring \\ 1: match\ at\ last\ part\ of\ substring \end{cases} \quad s_i^j(c)=\begin{cases} 0: not-match\ at\ b\ egining\ part\ of\ substring \\ 1: match\ at\ \ begining\ part\ of\ sub\ string \end{cases}$$

그림 6. 위치 상태 정보

인 다중 서브스트링에서 T검출을 위해 독립 부분 매칭에 의해 r의 인입 순서와 무관하게 t가 검출되는 순서대로 s_i^j를 저장한다. S_1: 독립 부분 매칭에서 t 검사 과정과 ASPM 생성 과정은 병렬 수행한다. 그림 7은 생성 조건을 도시한 것이다.

$$
\begin{aligned}
ASPM = &(s_1^1(h) \otimes s_1^{21}(h) \otimes s_1^3(h) \otimes \ldots\ldots\ldots \otimes s_1^j(h)) \\
&\oplus (s_2^1(h) \otimes s_2^2(h) \otimes s_2^3(h) \otimes \ldots\ldots\ldots \otimes s_2^j(h)) \\
&\oplus (s_3^1(h) \otimes s_3^1(h) \otimes s_3^2(h) \otimes \ldots\ldots\ldots \otimes s_3^j(h)) \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
&\oplus (s_i^1(h) \otimes s_i^2(h) \otimes s_i^3(h) \otimes \ldots\ldots\ldots \otimes s_i^j(h))
\end{aligned}
$$

그림 7. ASPM 생성 조건

3.3.3 S_2 : 거리 검사

S_2:거리 검사는 MSM에 저장된 s_i^j를 이용하여 검출된 t들의 거리를 검사하여 T 검출 유무를 판단하는 과정이다. t들의 거리란 스트링에서 검출된 t가 존재하는 위치들의 거리를 의미하며, 이는 검출된 t들의 $s_i^{x+1}(g) - s_i^x(g)$와 $s_i^{x+1}(l) - s_i^x(l)$을 의미한다. $s_i^x(g)$와 $s_i^x(l)$은 MSM의 i번째 행의 x번째 열에 위치하는 위치정보이다. 여기서 i는 고정 값을 가지며 $1 \le x \le j-1$이다. $s_i^{x+1}(g) - s_i^x(g)$는 스트링에서 t들이 은닉된 r_i와 r_{i+1}의 위치를 계산하기 위한 것이다. 만일 $s_i^{x+1}(g) - s_i^x(g)$=0라면, 한 개 r에서 t들이 검출되었다는 것을 의미한다. 만일 $s_i^{x+1}(g)$-$s_i^x(g)$=1라면, t들이 연속된 r_i와 r_{i+1}에서 검출된 경우이다. 즉, 검출된 t는 L_s=0인 다중 서브스트링에서 검출되었음을 알 수 있다. 만일 $s_i^{x+1}(g)$-$s_i^x(g)$ >a라면, t들이 L_s=a인 다중 서브스트링에서 검출되었음을 의미한다. 여기서 a=1,2,3,...이다. $s_i^{x+1}(l)$-$s_i^x(l)$는 r에서 t들의 거리를 계산하기 위한 것이다. 만일 $s_i^{x+1}(l) - s_i^x(l)$=w라면, 연속된 t가 검출되었음을 의미한다. $s_i^{x+1}(l) - s_i^x(l)$=w+b의 경우는 검출된 t들은 b바이트 거리를 가진다. 만일 $s_i^{x+1}(l)$-$s_i^x(l)$<w라면, 이것은 검출된 t의 데이터가 중첩된 경우로 T가 검출되지 않았음을 의미한다. 또한 $s_i^{x+1}(g) - s_i^x(g)$=1이며 $s_i^{x+1}(b)$와 $s_i^x(b)$가 1로 설정된 경우는 r_i와 r_{i+1}에서 t가 검출되었음을 의미한다.

그림 8은 비순차 서브스트링에서 T="ABCDE"를

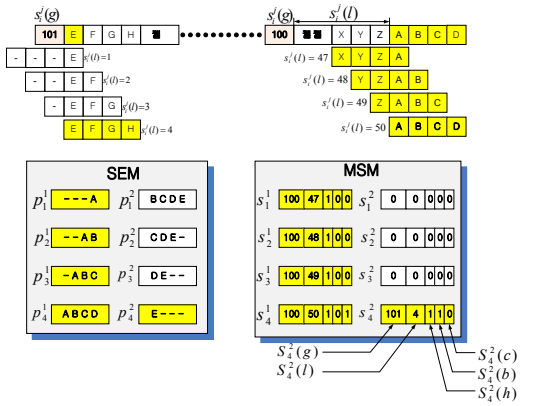


그림 8. T="ABCDE" 검출을 위한 MDPI 동작 흐름

검출하는 과정을 예시한 것이며, w=4이고 r_l=53으로 가정한다. "-"은 don't care 상태를 나타낸다. MDPI는 패턴 매칭을 위해 S_0:행렬 구성 단계에서 T="ABCDE"가 다중 서브스트링에서 발생할 수 있는 조합들을 SEM에 구성한다. 즉, p_1^1="---A"이며, p_2^1은 "--AB"가 된다. SEM 구성이 완료되면, r에서 t를 검출하기 위해 S_1:부분 매칭 과정을 수행한다. $s_i^j(g)$=101인 r이 인입될 경우 t검사 과정에 의해 SW는 우측으로 한 바이트씩 이동하면서 SEM와 상호 비교 과정을 수행한다. SW가 우측으로 4회 이동하였을 때 p_4^2="---E"가 검출되며 관련 위치 정보는 MSM의 s_4^2에 저장된다. $s_4^2(g)$=101, $s_4^2(l)$=4 이고 $s_4^2(h)$=1이다. $s_4^2(l) \le$w이므로, $s_4^2(b)$=1이 된다. 이후 SW가 $s_i^j(g)$=101에서 t검사 과정을 종료하고 n개 서브스트링 이후 $s_i^j(g)$=100을 가지는 r이 입력된다. SW는 우측으로 한 바이트씩 이동하면서 SEM와 상호 비교 과정을 수행한다. SW는 47 - 50회 이동하였을 때, p_1^1, p_2^1, p_3^1 및 p_4^1를 검출하며 관련 정보를 s_1^1, s_2^1, s_3^1 및 s_4^1 에 저장한다. 특히 p_4^1은 r_l-$s_i^j(l) \le$w의 위치에서 검출되어 $s_4^1(c)$=1로 설정된다. ASPM생성 과정은 $s_4^1(h)$와 $s_4^2(h)$이 1로 설정되어 있으므로 SEM의 4번째 행을 구성하는 t들이 모두 검출 되었으므로 ASPM 조건이 만족된다. ASPM 조건이 만족되었으므로 S_2: 거리 검사 단계는 s_4^1와 s_4^2의 위치 정보를 이용하여 T검출 유무를 판단한다. S_2: 거리 검사 단계는 $s_4^2(g)$-$s_4^1(g)$ =1이며, $s_4^1(c)$=$s_4^2(b)$=1이므로, r_i와 r_{i+1}에 나뉘어 전송된 T="ABCDE"가 검출되었음을 알 수 있다.

Ⅳ. 기능 구현

본 논문에서 제안한 독립 부분 매칭에 의한 행렬 기반의 고성능 침입 탐지 기술은 하드웨어 기반의 고성능 침입 탐지 기술이다. 그림 9는 MDPI를 구현한 내부 블록도를 도시한 것이다. 본 논문에서는 MDPI를 구현하기 위해 TCAM와 FPGA(Field Programmable Gate Array)를 사용한다. FPGA는 AND, OR 및 LUT로 구성되는 LC(Logic Cell)을 게이트 어레이 형태로 구성되어 사용자 의도에 따라 재구성이 가능한 IC를 지칭한다. SEM은 실시간 패턴 매칭을 위해 TCAM에 구현하며, MSM은 SRAM에 구현하였다. 본 논문에서는 w=8바이트로 설정하여 구현하였다. ASPM 블록은 ASPM 조건 만족 여부를 확인하는 블록으로 $s_j^i(h)$에 대한 AND/OR 게이트 조합으로 간단하게 구현할 수 있다. MatchInfoRead와 MatchInfoWrite 블록은 SRAM에 대한 s_i^j를 읽기/쓰기 동작을 수행한다. 만일 t가 검출되었다면, MatchInfoWrite 블록은 검출된 t에 대한 s_j^i를 SRAM에 저장한다. 만일 ASPM 신호가 생성되었다면, MatchInfoRead 블록은 거리 검사 수행을 위해 필요한 s_j^i를 읽어 온다. 거리 검사 블럭은 앞선 언급한 S_2:검사 상태를 수행하기 위한 것이다. DistanceCheck 블럭은 MemoryInfoRead 블록을 통해 수신한 위치 정보를 이용하여 T검출 여부를 판단한다. DistanceCheck 블록은 T가 검출되었다고 판단되면 이를 관리자에게 경보 형태로 통지한다. 본 논문에서는 Altera사의 Quartus II Ver 8.1을 이용하여 AHDL로 기능 구현하였다. 또한 FPGA는 Altera사의 Startix II를 적용하였다. MDPI는 l=57 바이트 텍스트 검출을 위해 330개 LC (Logic Cell) 가 사용되었다. MDPI는 5.79 LC/Char이 소요된다. 또한 구현된 MDPI는 l=57 바이트 텍스트 검출위해 14개 클럭을 소요하므로 최대 10.941Gbps의 패턴 검사 성능을 가진다.

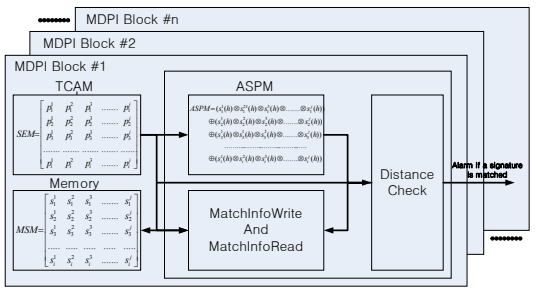


그림 9. MDPI 내부 블록도

Ⅴ. 성능 비교

MDPI는 $L_s \neq 0$인 비순차 서브스트링에서 T검출을 위한 재배열 과정을 배제함으로써 TCAM 메모리 효율성을 증대시키고 재배열 및 재조립에 의한 부가적인 메모리 및 프로세서 오버헤드를 최소화할 수 있다. 본 논문에서는 T 길이에 따른 TCAM 메모리 효율성에 대해 기존 방법과 성능 비교한다.

5.1 TCAM 메모리 효율

일반적으로 단일 서브스트링에서 T를 검출하기 위해서는 $e = ceil(l/w)$개 TCAM 엔트리가 필요하다. 여기서 e는 단일 서브스트링에서 T를 검출하기 위해 요구되는 TCAM 엔트리 수를 의미한다. 순차 서브 스트링에서 T를 검출하기 위해서는 $e \times w$개가 요구되므로, $e_s = ceil(l/w) \times w$이다. e_s는 순차 서브스트링에서 T를 검출하기 위해 요구되는 TCAM 엔트리 수를 의미한다. 비순차 서브 스트링은 인입되는 순서가 유지되지 않으므로, 순서대로 t를 검사하는 TCAM 기반의 패턴 검사 방법들은 $l+w-1$개 e가 요구된다. 즉, 비순차 서브스트링에서 T를 검출하기 위해서는 e_o=$ceil(l/w)$x$(l+(w-1))$이다. 여기서 e_o는 비순차 서브스트링에서 T를 검출하기 위해 요구되는 TCAM 엔트리 수이다. 본 논문에서 제안한 MDPI는 독립 부분 매칭을 이용하여 패턴 검사를 수행하기 때문에 비순차 서브스트링을 위해 부가적인 TCAM 엔트리를 요구하지 않는다. 본 논문에서 제안한 패턴 검사 방법인 MDPI에 의해 T를 검출하기 위해서는 e_s개 TCAM 엔트리가 필요하다. 이것은 MDPI가 인입되는 r 및 검출되는 t의 순서와 무관하게 T를 검출하기 때문이다. 따라서 MDPI를 적용함으로써 얻을 수 있는 TCAM 엔트리 이득은 아래와 같다.

$$\triangle e = e_o - e_s = ceil(i/w) \times (l-1) \qquad (1)$$

(식 2)는 MDPI가 적용되었을 경우 TCAM 메모리 효율 m_e를 도시한 것이다. m_e는 MDPI를 사용하여 텍스트 검출을 위해 필요한 TCAM 엔트리 이득을 퍼센트로 나타낸 것이다.

$$m_e = \frac{\triangle e}{e_o} \times 100 \qquad (2)$$

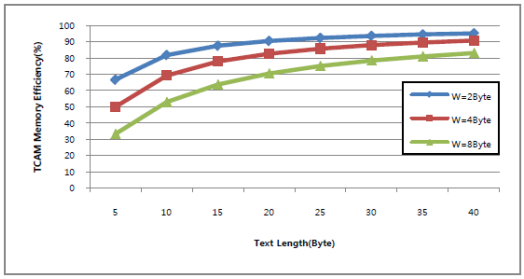


그림 10. TCAM 메모리 이득

MDPI는 텍스트 길이가 동일한 경우 m_e가 작을수록 높으며, w가 동일한 경우는 텍스트 길이가 길수록 m_e가 높다. 그림 10은 길이 l에 따른 m_e를 도시한 것이다. 오픈 소서 코드 기반의 침입 탐지 시스템인 Snort의 평균 텍스트 길이는 9바이트이다. MDPI는 l=9바이트 일 경우 w=2바이트 이면 m_e=80%, w=4바이트일 경우는 m_e=66.667% 이며, w=8바이트에서는 m_e=50%이다. 즉, w=8인 TCAM을 이용하여 9바이트 텍스트 검출을 위해서는 TCAM 기반의 순차적인 패턴 매칭 방법을 사용하는 것에 비해 MDPI를 사용할 경우 50% TCAM 엔트리 이득을 얻을 수 있다. MDPI는 l의 길이가 길거나, w의 크기가 T 검출을 위한 TCAM 엔트리 최적화에 효과적이다.

5.2 하드웨어 복잡성 및 패턴 매칭 성능

그림 11는 기존 하드웨어 기반의 패턴 매칭 방법과 본 논문에서 제안한 하드웨어 기반의 패턴 매칭 방법의 복잡도 및 패턴 매칭 처리 속도를 상호 비교한 것이다. 하드웨어 복잡성은 기능 구현에 소요되는 하드웨어 자원의 양와 성능으로 결정된다. FPGA로 기능 구현한 경우는 소모되는 LC이 증가할수록 하드웨어 복잡성은 증가한다. MDPI는 5.79 LC/Char와 10.941Gbps 패턴 매칭 속도를 제공한다. 하드웨어 복잡성 측면에서 살펴 보면 Franklin[6]이 제안한 방법이 3.17LC/Char이며, MDPI보다 우수하다. 그러나 Franklin은 패턴 매칭 속도 측면에서는 1.008Gbps로 MDPI의 10.941Gbps에 미치지 못한다. 패턴 매칭 성능 측면에서는 Sourdis[9]가 제안한 패턴 매칭 방법이 19.408Gbps로 패턴 매칭 방법중 가장 우수하나, 19.048LC/Char로 MDPI의 5.79LC/Char 보다 하드웨어 복잡성이 높다. 본 논문에서 제안하는 MDPI는 Franklin 보다 하드웨어 복잡성이 높고 성능 측면에서는 Sourdis보다 낮지만 수 Gbps 전송 속도를 가지는 네트워크에서는 하드

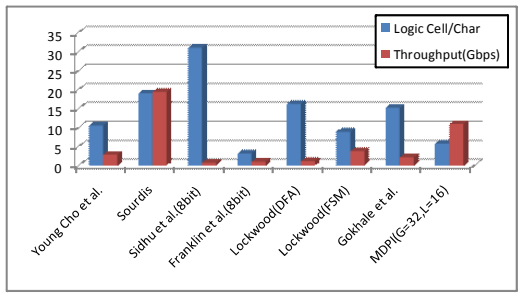


그림 11. 하드웨어 복잡성 및 성능 비교

웨어 복잡성 대비 성능 측면에서 가장 최적화된 해결 방안을 제시함을 알 수 있다.

Ⅵ. 결론 및 향후 연구 방향

본 논문에서는 수 Gbps 네트워크 트래픽의 페이로드를 검색하여 실시간 패턴 매칭을 지원하는 독립 부분 매칭에 의한 행렬 기반의 패턴 매칭 방법인 MDPI에 대해 논하였다. MDPI는 인입되는 패킷의 순서와 무관하게 TCAM을 이용하여 패턴 매칭 과정을 수행한다. MDPI는 $L_s \neq 0$인 다중 서브스트링에서 재배열 및 재조립에서 발생하는 하드웨어 오버헤드를 최소화함으로써 가격 효율적인 하드웨어 기반의 NIDS를 구성할 수 있다. MDPI는 Snort 룰셋의 평균 길이인 9 바이트의 경우 w=4바이트에서 61%, w=8바이트는 50%의 TCAM 메모리 효율이 증가했다. 또한 MDPI는 10.941Gbps 패턴 검사 성능과 5.79 LC/Char 하드웨어 복잡성을 제공함으로써, 수 Gbps 전송 속도를 가지는 네트워크에서 하드웨어 복잡성 대비 성능 측면에서 우수한 실시간 고속 패턴 매칭 방법 이다.

MDPI는 SW가 1바이트씩 이동하면서 패턴 매칭 과정을 수행함으로써 TCAM 검색 속도에 의해 성능이 결정된다. 이와 관련하여 SW가 한번에 m 바이트씩 서브스트링 검색을 통한 성능 향상에 관한 연구가 요구된다.

참 고 문 헌

〔1〕 J. C. Bennet, C. Partidge, N. Shectman, “Packet Reordering is not pathological network behavior”, *IEEE/ACM Transaction on Networking*, Vol. 7, Issue 3, pp 789-798, 1999.

〔2〕 V. Paxon, “End-to-end Internet Packet Dynamics”,

IEEE/ACM Transaction on Networking, Vol. 7, Issue 3, pp. 277-293, 1999.

〔3〕 R. Sidhu, V. K. Prasanna, "Fast regular expression matching using", *FCCM 2001*, 2001

〔4〕 J. W. Lockwood, "An open platform for development of network processing modules in reconfigurable hardware", *IEC DesignCon 2001*, 2001.

〔5〕 M. Fisk, G. Varghese, "An analysis of fast string matching applied to content-based forwarding and intrusion detection", *Technical Report CS2001-0670, Univ. of California San Diego*, 2002.

〔6〕 R. Franklin, D. Carver, B. Huchings, "Assisting network intrusion detection with reconfigurable hardware", *FCCM 2002*, 2002.

〔7〕 Y. H. Cho, S. Navab, W. H. Magione-smith, "Specialized hardware for deep network packet filtering", *FPL 2002, LNCS 2438*, pp. 452-461, 2002

〔8〕 S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, D. Towsley, "Measurement and Classification of Out-of-Sequence Packets in a Tier-1 IP Backbone", *Sprint ATL Technical Report,* 2002.

〔9〕 I. Sourdis, D. Pnevmatikatos, "Fast, large-scale string match for a 10Gbps FPGA-based network intrusion detection system", *FPL 2003*, 2003.

〔10〕 S. Dharmapurikar, "Implementation of a Deep Packet Inspection Circuit using Parallel Bloom Filters in Reconfigurable Hardware", *In Hot Interconnects*, 2003

〔11〕 Y. Wang, G. Lu, X. Li, "A Study of Internet Packet Reordering", *ICOIN 2004 ,LNCS 3090*, pp. 350-359, 2004.

〔12〕 F. Yu, R. H. Katz, T. V. Laskhman, "Gigabit Rate Packet Pattern Matching with TCAM", *UCB technical report*, UCB/CSD-04-1341, 2004

〔13〕 R. Liu, C. Kao, H. Wu, M. Shin, N. Huang, "FTSE: The FNP-Like TCAM searching engine", ISCC 2005, 2005

〔14〕 N. Desai, "Increasing performance in high speed NIDS", *http://www.linuxsecurity.com/article*

〔15〕 Snort, www.snort.org

정 우 석 (Woo-Sug Jung) 정회원

1994년 2월 명지대학교 전자공학과 석사

2004년 2월 충남대학교 컴퓨터공학과 박사 수료

1994년～현재 한국전자통신연구원 책임연구원

<관심분야> 임베디드 시스템, 무선 통신, 멀티미디어시스템, 인터넷 보안등

권 택 근 (Taeck-Geun Kwon) 종신회원

1988년 2월 서울대학교 컴퓨터공학과 학사

1990년 2월 서울대학교 컴퓨터공학과 석사

1996년 2월 서울대학교 컴퓨터공학과 박사

1992년 LG전자 정보통신 연구소 연구원

1998년～현재 충남대학교 전기 정보통신공학부 컴퓨터전공 교수

<관심분야> 네트워크 프로세서, 초고속 인터넷, 통신시스템, 인터넷 보안 등