Journal of The Korea Society of Computer and Information
Vol. 19, No. 12, December 2014
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2014.19.12.133

파이프라인형 DPI 시스템에서 효율적인 소비전력 감소를 위한 동작주파수 설계방법

김 한 수*

Adaptive Frequency Scaling for Efficient Power Management in Pipelined Deep Packet Inspection Systems

Han-Soo Kim*

요 약

여러 네트워크 보안기술 중 가장 효과적이고 신뢰할 수 있는 기술인 DPI 시스템에 쓰이는 파이프라인형 AC-DFA 구조에서, 효율적으로 전력 소모를 줄이는 방법을 제안하였다. 이는 메모리 접근 횟수가 전력 소모에 가장 큰 영향을 끼친다는 것과, 파이프라인형 AC-DFA의 스테이지 사용 횟수가 뒤쪽 스테이지로 갈수록 급격하게 감소한다는 관찰결과에 따른 것이다. 이에, 사용되지 않는 스테이지의 동작 클럭을 감소시켜 불필요하게 소모되는 전력을 줄이는 시스템을 구현하였다. 제안하는 방법을 적용한 DPI 시스템에 여러 종류의 문자열이 입력될 때의 전력 소모를 측정한 결과, 기존의 DPI 시스템에 비해 약 25 %의 전력 절감 효과를 가져왔다. 제안한 방법은 파이프라인형 DPI 구조 및 다중 패턴 문자열 검색의 어떤 응용에도 손쉽게 적용될 수 있을 것이다.

▸ Keywords : DPI, 다중 패턴 문자열 검색, AC-DFA, 주파수 스케일링, 전력 소모

Abstract

An efficient method for reducing power consumption in pipelined deep packet inspection systems is proposed. It is based on the observation that the number of memory accesses is dominant for the power consumption and the number of accesses drops drastically as the input goes through stages of the pipelined AC-DFA. A DPI system is implemented where the operating frequency of the stages that are not frequently used in the pipeline is reduced to eliminate the waste of power consumption. The power consumption of the proposed DPI system is measured upon various input character set and up to 25% of

•저 자 : 김한수
•투고일 : 2014. 8. 27, 심사일 : 2014. 9. 17, 게재확정일 : 2014. 11. 12.
* 국립과학수사연구원 디지털분석과(Digital Technology and Biometry Division, National Forensic Service)

reduction of total power consumption is obtained, compared to those of the recent DPI systems. The method can be easily applied to other pipelined architecture and string searching applications.

▸ Keywords : DPI, string searching, AC-DFA, frequency scaling, power consumption

I. 서 론

우리는 네트워크의 시대에 살고 있다. 인터넷을 비롯한 네트워크의 의존도가 높아지고, 점점 더 중요하고 복잡한 작업들이 컴퓨터와 이들의 연결 집합체인 네트워크를 통해 이루어지고 있다[1]. 인터넷과 함께하는 네트워크 기술의 발달은 은행, 산업, 군사시설에서 가정환경에 이르기까지 우리에게 많은 편리함과 풍요로움을 가져다주었다. 하지만 바이러스, 악성코드 및 DDoS공격 등 많은 위험요소 또한 함께 늘어나게 되었다. 이러한 위험에 대응하기 위하여, 백신 또는 방화벽과 같은 여러 보안기술 및 제품들이 등장하게 되었다[2]. 이들 중에서, DPI(deep packet inspection, 심층 패킷 검사. Snort[3] 등)가 바이러스, 악성코드 및 DDoS공격 등의 위험요소를 제거하는 기술 중 가장 효과적이고 신뢰할 수 있는 기술로 자리잡게 되었다. DPI는 다양한 방면에서 활용되어, 네트워크 세상에서 없어서는 안될 요소가 되어가고 있다. 네트워크 보안 문제, ISP의 효율적인 QoS 제공 등에서 traffic shaping 등의 특정 행위 간섭, 통제, 모니터링에 이르기까지[10], 그 활용도가 높아짐에 따라 다양한 방면에의 응용이 대두되고 있다.

DPI 또는 그 시스템은 다중 문자열 검색에 기반을 두고 있다. 다중 문자열 검색이란, 입력되는 문자열을 검색하여 미리 정의한 문자열 패턴들이 존재하는지를 검사하는 기술이다[4]. 많은 수의 다중 문자열 검색 기술은 아호-코라식(Aho-Corasick, AC) 알고리즘을 사용하는데, 이는 검색 시스템을 결정 유한 오토마타(deterministic finite automaton, DFA)[5]로 정의하게 된다. 검색의 속도, 정확성 및 문자열 패턴을 저장하는데 필요한 공간의 확보와 더불어, 검색을 수행하는데 소모되는 전력 또한 DPI 시스템의 중요한 문제가 되고 있다. DPI 시스템의 소비전력에서 가장 많은 부분을 차지하는 것은 다중 문자열 검색 엔진이며, 따라서 다중 문자열 검색과정에서의 전력소비를 줄이기 위한 연구들이 계속되어 왔다.

한편, 파이프라인이 도입된 검색이 대표적인 문자열 검색 기법이 됨에 따라, 파이프라인형 다중 문자열 검색엔진의 전력절감에 대한 연구들 또한 많이 행해지고 있다. 트래픽의 집약성(locality)을 이용한 캐싱 기법[6, 7]이 제안되었지만, 그 성능은 입력 문자열의 특성에 따라 심하게 변한다. 적응적 동작주파수 기법[8]이 구현되었지만, 이는 프로세서 전체의 동작주파수를 조절함으로써 전력소비 절감에 비해 큰 성능 저하를 초래하게 된다.

이러한 문제들을 바탕으로 하여, DPI시스템에서 파이프라인형 다중 문자열 검색엔진의 소비전력을 효율적으로 절감하는 방법을 제안하였다. 파이프라인 구조에서 사용되지 않은 스테이지의 동작 주파수를 줄임으로써 해당 스테이지의 전력소비를 효과적으로 감소시켰고, 동작중인 스테이지 바로 다음 스테이지의 동작 주파수를 미리 증가시킴으로써 문자열 검색의 처리지연 및 클럭 스큐 등을 방지하였다. 실험을 통하여, 기존의 DPI 시스템보다 최대 25 %의 전력소비를 절감할 수 있음을 확인하였다.

이후 본 논문의 구성은 다음과 같다. 2장에서는 다중 문자열 검색 등 관련 연구를 고찰하고, 3장에서는 전력 소비를 줄이는 방법을 제안하며, 4장에서는 이를 실험을 통하여 확인하고, 5장에서 결론을 맺는다.

II. 관련 연구

1. DPI의 개요

DPI란 일반적으로 패킷의 콘텐츠가 담긴 심층부분까지 검사할 수 있는 기술로 통용된다. 대규모 네트워크 환경에서 짧은 시간안에 다양한 애플리케이션을 식별할 수 있다는 DPI의 특징은 광범위한 네트워크를 운영하는 인터넷제공사업자(Internet Service Provider, ISP)에게 매력적인 기술로 다가온다. 실제로 네트워크에 전송되는 애플리케이션이 복잡,

다양해지고 데이터 트래픽이 급격하게 증가하면서 이에 따른 혼잡제어 등 더 정교한 트래픽 관리를 위해 ISP의 DPI에 대한 관심은 증가하고 있다[9]. 하지만 패킷 전체를 도청 또는 감청하는 정도의 보안 검사를 시행하기 때문에 사용자의 사생활 보호 등에 대한 문제와도 민감하게 맞닿아 있다. 현재 많은 이슈가 되고 있는 DPI의 대표적인 활용 가능성은 다음과 같다[10].

- 네트워크 보안(network security) : 가장 아래 부분까지 세밀하게 검사할수 있는 DPI의 능력은 바이러스, 악성코드 및 DDoS 공격을 효과적으로 막아낼 수 있다.
- 접속 제어(connection control) : 가입자와 미가입자의 구분, 정부 정책 주도 아래의 접속 정책 등을 효율적으로 반영할 수 있다.
- 품질 보장(QoS) : ISP로 하여금 비정상적인 데이터의 흐름(P2P 서비스 등)을 제어할 수 있게 하여, 모든 사용자들에게 균일한 품질을 보장할 수 있다.
- 맞춤 서비스(tailored service) : ISP로 하여금 사용자의 과금체계 및 가입조건에 따라 알맞은 서비스를 제공할 수 있게 한다.
- 저작권 관리(DRM management) : 음악, 영화 등 저작권에 위배되는 데이터의 불법 유포 등을 효과적으로 관리할 수 있게 한다.
- 통신량 조절(traffic shaping): 통신량 조절은 특정한 네트워크 및 가입자에게 통신량 또는 대역폭을 임의로 할당하는 것을 말한다. ISP는 DPI를 사용하여 이를 절대적으로 조절할 수 있으므로, 이에 따른 망 중립성의 훼손에 대한 논란이 제기되고 있다.
- 행위 분석(behavioral targeting, BT): ISP 또는 이에 준하는 권한을 가진 곳에서, 사용자의 행동을 DPI를 이용하여 분석함으로써 이에 맞는 광고 등의 서비스를 제공할 수 있다.

2. 파이프라인형 AC-DFA

기존의 AC-DFA(Aho-Corasick deterministic finite automaton)는 모두 n개의 문자를 갖는 패턴들의 집합을 O(n) state 만에 DFA로 변환한다. 이는 상태 천이도(state transition table)의 형식으로 저장되어, 입력 문자열을 한 클럭에 한 문자씩 비교하게 된다. 각각의 입력 문자는 한 번에 하나의 패턴 문자와 비교되어, 그 결과로 하나의 상태 천이를 갖는다[4].

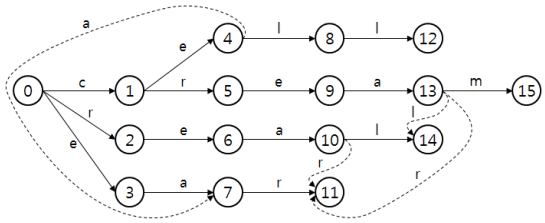

그림 1. AC-DFA의 형태
Fig. 1. a form of an AC-DFA

그림 1은 4개의 패턴 "cell", "cream", "real" 및 "ear"로 이루어진 AC-DFA의 구성을 나타낸다. AC-DFA의 구성은 root에서부터 시작하고, 각각의 패턴들은 한 문자당 하나의 state를 생성하며 root에서부터 패턴 끝까지 진행한다[4]. 이러한 전이를 *진행 전이*(*goto transition*)라 한다[5]. 그림 1의 예에서, 패턴 "cell"은 state 1, 4, 8, 12를 생성한다. 입력 문자열이 root에서부터 진행하며 비교를 수행하다가 상이한 문자가 나타나면(mismatch) 다른 방향으로의 전이가 추가되는데, 이를 *실패 전이*(*failure transition*)라 한다. 그림 1에서 state 4, 10, 13을 제외한 모든 state는 root state를 실패 전이의 종착지로 한다. 한편, 몇몇 패턴들은 특정 문자를 다른 패턴들과 공유하는데, 이럴 때에는 root state가 아닌 곳으로 실패 전이를 수행하게 된다. 그림 1에서 패턴 "real"과 "ear"는 문자열 "ea"를 공유하므로, state 10에서 입력 문자가 "r"인 경우 실패 전이는 root state 대신 state 11로 이동한다. 진행 전이는 *전방 전이*(*forward transition*)이라고도 하고, 실패 전이는 *교차 전이*(*cross transition*)이라고도 한다. 그림 1에서 전방 전이는 실선으로, 교차 전이는 점선으로 나타내었다(root state로의 교차 전이는 나타내지 않음).

한편, 압축된 AC-DFA[11]는 본래의 AC-DFA의 state를 압축하여, 하나의 전이에 여러 개의 문자가 대응되는 구조이다. 본래의 AC-DFA는 검색 패턴에서 한 번에 여러 개의 문자를 하나의 전이에 대응시켜 AC-DFA를 구성함으로써 압축될 수 있는데, 이 때 하나의 전이에 대응되는 문자의 개수를 stride라 한다.

그림 2는 그림 1과 같은 본래의 AC-DFA를 stride=2를 이용하여 압축한 AC-DFA이다. 입력 문자열에서 한 번에 두 개의 문자를 가져와서 AC-DFA의 두 개의 문자와 비교한다. 이 두 개의 문자가 서로 일치하면(match), 마찬가지로 전방 전이가 발생하여 다음 두 개의 문자 비교를 수행하게 된다. 두 개의 문자 중 하나라도 일치하지 않으면(mismatch) 교차 전이가 발생하며 처음 한 문자를 제외하고 다음 한 문자가 추

가된(one character offset) 두 개의 입력 문자로 비교 작업을 진행하게 된다.

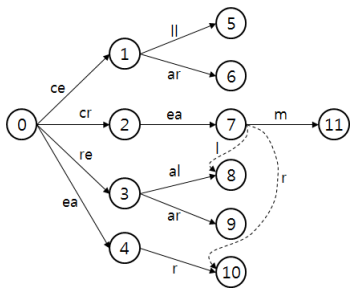

그림 2. 압축된 AC-DFA
Fig. 2. a compressed AC-DFA

그림 2에서 입력 문자열이 "crear"라고 가정하면, 처음 두 개의 문자 "cr"을 AC-DFA의 "ce", "cr", "re" 및 "ea"와 순서대로 비교하게 된다. 이 때는 node 2로 전이가 이루어진다. 그 다음, 입력 문자 "ea"를 가져와서 AC-DFA의 "ea"와 비교하게 되고, 서로 일치하므로 node 7로 전이된다. 마지막으로 입력 문자 "r"과 AC-DFA의 "m" 및 "r"을 비교한다. 이 때 교차 전이에 대한 일치가 일어나므로, node 10으로 이동한다. 입력 문자가 "dreal"일 경우, 처음 두 개의 입력 문자 "dr"을 가져와서 "ce", "cr", "re" 및 "ea"와 순서대로 비교하게 된다. 일치하는 문자가 없으므로(mismatch), 한 문자만큼 이동된 두 개의 입력 문자 "re"를 가져와서 "ce", "cr", "re" 및 "ea"와 순서대로 비교한다. 이러한 방식으로, 놓치는 문자 없이 여러 개의 문자에 대한 모든 문자열의 비교가 가능하게 된다.

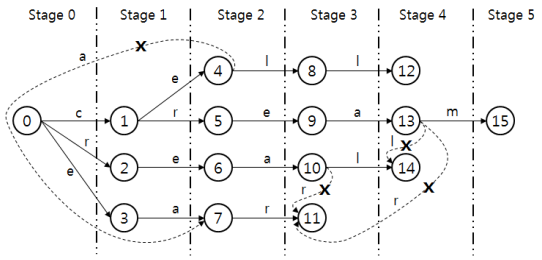

그림 3. 파이프라인형 AC-DFA
Fig. 3. a pipelined AC-DFA

그림 3은 그림 1에서 나타난 AC-DFA의 각각의 level을 파이프라인의 각 스테이지에 대응한 것이다. 동작 클럭마다 새로운 입력 문자를 계속 가져와서 root에서 비교를 수행함으로써, 교차 전이를 효과적으로 제거하게 되었다[12, 4]. 입력 문자열을 "crear"라 가정하면, 클럭1에서 첫 번째 입력 문자 "c"를 root로 가져와서 "c", "r" 및 "e"와 순서대로 비교하게 된다. 일치하는 문자의 방향으로 전이가 일어나고, 클럭2에서 두 번째 입력 문자 "r"을 node 1로 가져온다.

이와 동시에, 입력 문자 "r"을 root로 가져와서 비교하게 되는 것이다. 즉, 클럭2일 때 스테이지 1에서 입력 문자 "r"과 node 1의 "e" 및 "r"을 비교할 동안, 스테이지 0에서 입력 문자 "r"과 root의 "c", "r" 및 "e"를 동시에 비교하게 된다.

클럭3에서는 스테이지 0에서 입력 문자 "e"와 root의 "c", "r" 및 "e", 스테이지 1에서 입력 문자 "e"와 node 2의 "e", 스테이지 2에서 입력 문자 "e"와 node 5의 "e"를 동시에 비교한다. 클럭4에서는 스테이지 0에서 입력 문자 "a"와 root의 "c", "r" 및 "e", 스테이지 1에서 입력 문자 "a"와 node 3의 "a", 스테이지 2에서 입력 문자 "a"와 node 6의 "a", 스테이지 3에서 입력 문자 "a"와 node 9의 "a"를 동시에 비교한다.

한편, 클럭5에서는 스테이지 0에서 입력 문자 "r"과 root의 "c", "r" 및 "e", 스테이지 2에서 입력 문자 "r"과 node 7의 "r", 스테이지 3에서 입력 문자 "r"과 node 10의 "l" 및 "r", 스테이지 4에서 입력 문자 "r"과 node 13의 "m" 및 "r"을 비교하게 된다. 스테이지 2에서 입력 문자 "r"과 node 7의 "r"을 비교한 결과로 인해 node 11에 도달하게 되므로, node 10과 node 13에서의 교차 전이인 "r"은 불필요해지는 것이다.

이러한 방식을 사용하여 입력 문자열 "crear", "rear", "ear", "ar", and "r"을 순차적으로 계속 입력하는 효과를 가져오고, node 11은 결국 입력 문자열 "ear"을 처리할 때 도달하게 된다. 따라서 언급한 것과 같이 node 10 및 13에서 출발하는 교차 전이는 모두 불필요해지고, 이러한 교차 전이를 저장하는데 필요한 엄청난 메모리 공간을 효과적으로 절약하게 된다.

3. 전자회로에서의 전력 제어 기술

전원관리 및 전력소모 절감 기술은 휴대폰, 노트북 등의 휴대 기기 사용이 보편화되고 다기능화, 고성능화함에 따라 지속적으로 발전해 왔다. 특히 반도체 소자의 선폭이 나노미터급으로 초미세화 됨에 따라 누설 전류가 급증하고 칩의 처리 성능을 높이기 위해 클럭 주파수를 높이면서 스위칭 전류 소모도 증가하므로, 이러한 동적/정적 전력소모 증가를 억제시킬 수 있는 기술이 SoC 설계에 점진적으로 적용되고 있다. 전자회로에서의 전력 소모에는 크게 두 가지 종류가 있는데, 그것은 동적전력 및 정적전력이다[13]. 그 중 동적전력이 전

자회로 소비전력의 많은 부분을 차지하므로, 식 1을 이용한 동적소비 전력의 절감 기술이 비약적으로 발달해 왔다.

$$P = \frac{1}{2} CV^2 f\alpha \quad (1)$$

식 1에서, P는 동적 소비전력, C는 정전용량, V는 입력전압, f는 동작주파수, α는 변환 상수(switching activity)이다. 동적소비전력의 절감은 이러한 요소들의 하나 또는 그 이상을 줄임으로써 이루어진다.

많은 연구들 가운데, 클럭 게이팅[14]이 다른 성능 저하를 최소화시키며 전력을 효율적으로 절약하는 기술 중의 하나로 자리매김하였다. 전자회로의 동작 주파수를 줄임으로서 전체 동적 소비전력을 절약하는 방법인데, 클럭 스큐 및 이에 따른 동작지연 등이 문제가 되고 있다. 인가되는 동작주파수보다 실제 동작주파수가 낮아지는 경우, 클럭 스큐가 발생하며 이에 따른 동작 지연 또한 나타나게 된다. 이러한 지연을 방지하기 위하여 클럭 버퍼 또는 중계기의 설치가 필요하게 되는데, 이 또한 추가적인 전력 낭비를 유발하게 된다[13, 14].

III. 제안하는 방식 및 알고리즘

1. 고찰 및 정의

파이프라인형 AC-DFA 구조를 적용하여 패턴 비교를 수행할 때, 파이프라인의 동작 횟수는 각 스테이지마다 매우 다르게 나타난다[6, 7]. 입력 문자열이 파이프라인에서 처리될 때, 파이프라인의 각 스테이지는 첫 번째 스테이지부터 차례대로 동작하게 된다. 각 스테이지에 저장된 패턴과 입력 문자열을 비교하는 과정에서 불일치(mismatch)가 발생하면, 그 이후의 스테이지는 비교 동작을 수행하지 않게 된다. 따라서 첫 번째 스테이지는 항상 비교 동작을 수행하지만, 끝 부분의 스테이지들은 거의 동작하지 않는 경우가 발생하게 되는 것이다[7].

그림 4는 파이프라인형 AC-DFA의 각 스테이지별 동작 횟수를 나타낸다. 입력 문자열은 무작위로 생성된 25,000,000 바이트(Byte)의 문자열을 사용하였다. AC-DFA의 파이프라인 스테이지의 수는 150개이다(그림4에는 48개까지만 표시). 가로축은 AC-DFA의 파이프라인 스테이지, 세로축은 로그 스케일로 나타낸 각 AC-DFA 파이프라인의 스테이지별 동작 횟수이다. 스테이지가 증가함에 따라, 동작 횟수는 엄청나게 줄어드는 것을 확인할 수 있다(스테이지 39부터는 동작 횟수가 0이 됨). 즉, 뒷부분의 스테이지로 갈수록 동작 횟수는 거의 존재하지 않게 되는데, 이렇게 동작하지 않는 스테이지들의 불필요한 전력 소모를 줄여 전체 전력소모를 절감하는 것이 제안하는 시스템의 주된 방식이다.

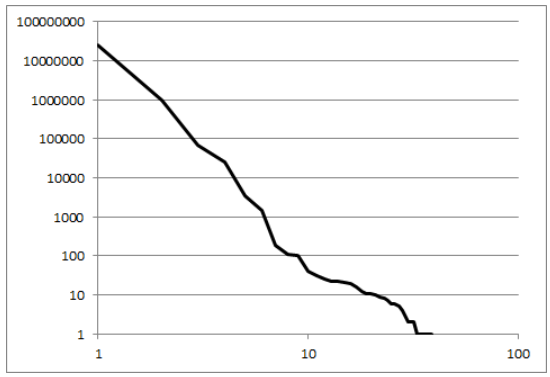

그림 4. 파이프라인형 AC-DFA의 스테이지별 동작 횟수
Fig. 4. The number of accesses to each stage of the pipelined AC-DFA

제안하는 방식의 효율적인 기술을 위하여, 아래와 같은 용어를 정의하였다.

Nc : 입력 문자열의 문자 수
Na : 파이프라인형 AC-DFA의 스테이지별 동작 횟수
Ns : 최대 패턴의 길이
Nm : 시스템이 파이프라인형 AC-DFA에 접근하는 횟수의 최대값
Pa : 활성 상태의 파이프라인 스테이지의 소비전력
Pr : 대기 상태의 파이프라인 스테이지의 소비전력
Ps : 휴식 상태의 파이프라인 스테이지의 소비전력
Pconv : 기존의 DPI 시스템의 전체 소비전력
Pprop : 제안하는 DPI 시스템의 전체 소비전력

입력 문자열의 문자 수 (*Nc*)는 패턴의 존재 유무를 검사할 입력 문자열의 크기(Byte)를 나타낸다. *Na*는 문자열 비교를 위한 파이프라인형 AC-DFA의 각 스테이지별 동작 횟수의 총합을 나타낸다. *Ns*는 가장 긴 패턴의 길이이며, 이는 AC-DFA의 높이와 같다. *Nm*은 패턴 비교 과정에서 파이프라인형 AC-DFA의 각 스테이지에 접근하는 횟수의 최대값이며, Nc × Ns와 같다. *Pconv*는 기존의 DPI 시스템이 Nc의 입력 문자열을 처리하는 동안 소비하는 전체 소비전력이며, *Pprop*는 제안하는 DPI 시스템이 Nc의 입력 문자열을 처리하는 동안 소비하는 전체 소비전력이다.

파이프라인형 AC-DFA의 스테이지가 패턴 비교를 수행하기 위해서는, 입력 문자열에서 비교할 문자를 가져와서 AC-DFA에 저장된 패턴들과 비교를 수행하게 된다. 이러한 패턴 비교를 수행하는 스테이지를 *활성 상태*라 한다. 패턴 비교를 수행하기 위해 기다리고 있는 스테이지를 *대기 상태*라 하며, 대기 상태에서 동작 주파수가 감소된 스테이지를 *휴식 상태*라 한다. 휴식 상태에서 대기 상태로 전환되는 스테이지를 *turn on*되었다고 하며, 대기 상태에서 휴식 상태로 전환되는 스테이지를 *turn off*되었다고 한다.

*Pa*는 활성 상태의 스테이지가 한 클럭당 소비하는 전력을 나타낸다. *Pr*은 대기 상태의 스테이지가 한 클럭당 소비하는 전력을 나타낸다. *Ps*는 휴식 상태의 스테이지가 한 클럭당 소비하는 전력을 나타낸다(파이프라인의 모든 스테이지는 동일한 하드웨어 구조를 가지고, 각 스테이지의 Pa, Pr 및 Ps는 서로 동일하다고 가정한다).

2. 사용하지 않는 스테이지의 동작 주파수 감소

전자회로에서 사용되지 않는 특정 블럭의 전력을 절약하는 데 있어 동작주파수를 조절하는 것이 가장 효과적이므로[7], 동작 횟수가 적은 파이프라인 스테이지의 동작 주파수를 조절하는 방법을 제안하는 DPI 시스템에 적용하였다. 사용되지 않는 파이프라인 스테이지가 많을수록, 더 많은 전력 절감 효과를 기대할 수 있다.

기존의 DPI 시스템의 전체 소비전력은 식 2로 표현할 수 있다.

$$P_{conv} = N_a \times P_a + (N_m - N_a) \times P_r \qquad (2)$$

또한, 동작 횟수가 적은 파이프라인 스테이지의 동작 주파수를 감소시킨 후의 전체 소비전력은 식 3으로 표현할 수 있다.

$$P_{prop} = N_a \times P_a + (N_m - N_a) \times P_s \qquad (3)$$

Pr, Ps의 정의 및 식 1에 의하면 Pr은 Ps보다 크므로, 동작 횟수가 적은 파이프라인 스테이지의 동작 주파수를 조절함으로써 (Nm - Na) × (Pr - Ps)만큼의 소비전력을 절약할 수 있다. 제안하는 알고리즘에 의하면, 사용되지 않는 파이프라인 스테이지가 많을수록 더 많은 전력을 절약하게 된다.

그림 3의 파이프라인형 AC-DFA를 예로 하고 입력 문자열이 "crearemo"라고 가정하면, 스테이지 0, 1, 2, 3, 4 및 5의 동작 횟수는 순서대로 8, 5, 4, 2, 1 및 0이 된다. 따라서 , Nc = 8, Ns = 6, Na = 8 + 5 + 4 + 2 + 1 = 20, Nm = 48이 됨을 알 수 있다(본래의 동작 주파수는 200 MHz, 감소된 동작 주파수는 50 MHz로 가정하였다).

가장 많이 사용되고 있는 소모전력 예측기[15]에 의하면, Pa, Pr 및 Ps는 각각 1.9 W, 0.9 W 및 0.6 W가 된다. 이 예측값을 대입하면, 그림 3의 Pconv는 63.2 W, Pprop는 54.8 W가 된다. 따라서, 제안하는 방식에 의하면 기존의 시스템보다 약 13.3 %의 소비전력 감소를 가져오게 되는 것이다.

3. 클럭 스큐 및 동작 지연의 방지

동작 주파수를 조절하는 데서 발생하는 클럭 스큐 및 동작 지연을 방지하기 위해서, 클럭 버퍼 또는 중계기의 설치 대신 활성 상태 바로 다음의 스테이지를 미리 활성 상태로 만든다. 즉, 대기 상태에서 활성 상태로 전환하는데 걸리는 클럭 스큐 및 동작 지연을 미리 발생시켜, 정상적인 활성 상태에서의 패턴 비교가 이루어지도록 하는 것이다. 따라서, 식 3은 식 4로 수정된다.

$$P_{prop} = N_a \times P_a + N_c \times P_r + (N_m - N_a - N_c) \times P_s \qquad (4)$$

그림 3의 예에서, 식 4와 같이 제안하는 DPI 시스템의 수정된 소비전력 Pprop은 57.2 W가 되며, 이는 기존의 시스템보다 약 9.5 %의 소비전력 감소를 가져오게 된다.

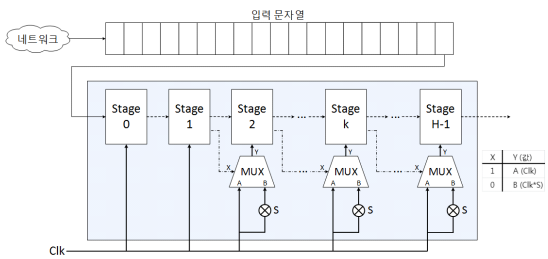

그림 5 제안하는 DPI 시스템의 구조
Fig. 5. The proposed DPI architecture

그림 5는 제안하는 DPI 시스템에서 다중 문자열 검색 엔진 부분의 개괄적 구조이다. 파이프라인의 스테이지가 문자열 비교를 수행할 때, 그 다음 스테이지를 turn on하는 신호를 보낸다. 이러한 신호를 *toggle signal*이라 한다(그림 5의 쇄선). 비교한 문자가 일치할 경우, 다음 스테이지에게 다음의 문자열 비교를 수행할 것을 알리는 신호를 보낸다. 이 신호를 *match notification*이라 한다(그림 5의 점선). 한편, *H*는 파이프라인 스테이지의 개수를 나타내고, *Clk*는 본래의 동작 주파수이며, *S*는 동작 주파수 감소를 위한 *scaling constant*

이다.

입력 문자열은 "crearemo"라고 가정하였다. 그림 4에서와 같이 파이프라인 스테이지의 수는 6(H = 6)이며, 초기 상태에서 스테이지 1에서 4까지에서 나오는 toggle signal은 모두 0 (x = 0) 이므로 스테이지 2에서 스테이지 5까지의 동작 주파수는 Clk × S가 된다. 감소된 동작 주파수를 0으로 설정하지 않은 이유는, 패턴 비교를 위해 0에서 본래의 동작 주파수까지 값을 증가시킬 경우 보다 많은 클럭 스큐 및 동작 지연을 초래하기 때문이다[6, 7]. 본 실험에서 사용된 scaling constant S는 1/4이지만, 이는 실제 하드웨어 시스템의 사양 또는 구현 방식에 따라 달라질 수 있다.

클럭 1에서, 스테이지 0은 입력 문자열의 첫 번째 문자 "c"를 가져와서 저장된 패턴 "c", "r" 및 "e"와 비교한다. 일치하는 문자가 있으므로, 스테이지 0은 match notification을 스테이지 1로 보내서 다음 패턴 비교를 수행하게 한다. 클럭 1에서, 스테이지 1은 입력 문자열의 두 번째 문자 "r"을 가져와서 저장된 패턴 "e" 및 "r" 과 비교한다. 패턴 비교를 수행하기 전에, 스테이지 1은 toggle signal (x = 1) 을 스테이지 2로 보내 스테이지 2를 대기 상태로 만든다. 즉, 스테이지 2를 turn on시켜 동작 주파수를 Clk/4에서 Clk가 되게 한다. 여기에서도 일치하는 문자가 있으므로, 스테이지 1은 같은 방식으로 match notification을 스테이지 2로 보낸다. 스테이지 2는 스테이지 3을 turn on시키며 입력 문자열의 세 번째 문자 "e"를 가져와서 저장된 패턴 "e"와 비교하게 된다. 즉, 스테이지가 패턴 비교를 수행하기 전에 다음 스테이지를 turn on시키는 것이다. 패턴 비교를 수행하는 스테이지는 패턴 비교를 수행하기 전에 이미 정상적인 동작 주파수(Clk)로 동작하고 있기 때문에, 주파수의 변화로 인한 클럭 스큐나 동작 지연이 발생하지 않게 된다.

스테이지 k($1 \leq k \leq H-2$)에서 불일치가 발생하게 되면, 스테이지 k+1 에게 toggle signal (x = 0) 을 보내 스테이지 k+1의 동작 주파수를 Clk/4 로 만들어 turn off시킨다. 클럭 5에서, 스테이지 0은 입력 문자열의 여섯 번째 문자 "e"를 가져와서 저장된 패턴 "c", "r" 및 "e"와 비교한다. 일치하는 문자가 있으므로, 스테이지 0은 match notification을 스테이지 1로 보내서 다음 패턴 비교를 수행하게 한다. 클럭 6에서, 스테이지 1은 스테이지 2를 turn on시키며 입력 문자열의 일곱 번째 문자 "m"을 가져와서 저장된 패턴 "a" 와 비교한다. "m"과 "a"는 불일치하므로, 스테이지 1은 toggle signal (x = 0) 을 보내 스테이지 2를 turn off시키며 동작 주파수를 Clk/4로 만든다.

IV. 실험 결과

실험을 통하여 제안하는 DPI 시스템과 기존의 DPI 시스템의 성능을 비교하였고, 그 결과를 표 1 및 표 2에 나타내었다.

표 1. 입력 문자열이 무작위로 생성된 문자열일 때
Table 1. The results on the randomly generated text

방식		A	B	C
스테이지의 평균 개수	활성상태	1.04	1.04	1.04
	대기상태	46.96	0	1.00
	휴식상태	0	46.96	45.96
전력 소모(W)		5058.6	3965.4	3774.7
전력소모 절감율(%)		-	**21.61**	**25.38**

표 2. 입력 문자열이 "영어 동화"일 때
Table 2. The results on the "English Fairy Tale"

방식		A	B	C
스테이지의 평균 개수	활성 상태	3.12	3.12	3.12
	대기 상태	44.88	0	1.02
	휴식 상태	0	44.88	43.86
전력 소모(W)		73.0	59.1	56.3
전력소모 절감율(%)		-	**19.10**	**22.92**

동작주파수는 200 MHz를 사용하였고, 감소된 주파수 및 미리 turn on되는 스테이지의 성능 측정을 위하여 두 가지 형태의 시스템을 실험에 사용하였다. 대기 상태의 동작 주파수를 감소시키지 않는 기존의 시스템을 방식 A, 대기 상태에서 감소된 주파수가 75 MHz인 경우(S = 3/8)를 방식 B, 대기 상태에서 감소된 주파수가 50 MHz (S = 1/4)이며 미리 turn on되는 스테이지를 1개로 설정한 경우를 방식 C로 하였다. (scaling constant의 설정은 전력 소모에 관한 기존 연구[13, 14]를 바탕으로, 전력 낭비를 최소화하며 시스템 전체의 효율을 최대화할 것으로 추정되는 값으로 설정한 것이며, 실제 전력 낭비 및 시스템 전체의 효율은 구현한 시스템의 하드웨어 사양 또는 입력 문자열의 형태에 따라 달라질 수 있다.)

실험에 사용된 패턴은 잘 알려져 있는 DPI 시스템 중의 하나인 Snort[3]에서 제공하는 패턴이다. 이 패턴의 content field의 문자열을 이용하여 AC-DFA를 구현하였고, 패턴의 개수는 86,248개이고, 가장 긴 패턴은 150바이트(Byte)이므로 구현된 AC-DFA의 레벨은 150이 된다. 이러한 AC-DFA와 DPI 시스템은 Microsoft사의 Visual C++

2008을 이용하여 구현하였으며, 소비전력을 정확하게 계측하고 내부오차의 고려를 최소화하기 위하여, 가장 많이 사용되고 있는 DPI 시스템의 사양을 차용하였다. 목표 하드웨어는 Xilinx사의 FPGA용 기기인 Vertex-4 XC4VLX100[15], Samsung사의 DDR3 DRAM[16] 2개로 설정하였다. 전력 소모의 측정은 Xilinx사의 Power Estimator (XPE) 11.1[15]를 사용하였다. 또한 시스템의 구성은 그림 5를 기본 구조로 하며, 구현 및 실험에는 Samsung사의 Series 5 530U3B, CPU는 Intel사의 Core i5 2467M이 사용되었다.

AC-DFA에서 교차 전이의 목적지가 되는 스테이지의 레벨만이 압축된 AC-DFA의 고려대상이 되므로, 구현한 AC-DFA의 48번째 레벨(레벨 47)까지만을 파이프라인의 스테이지에 저장하였다. 또한, 해싱 알고리즘을 전혀 사용하지 않았다고 가정하여, 하나의 패턴 비교를 수행할 때마다 매번 패턴이 저장된 메모리에 접근하는 것으로 설계하였다.

입력 문자열로는 두 가지를 사용하였다. 첫 번째는 무작위로 생성된 25,000,000 개의 문자열(25 MBytes), 두 번째는 341,362 개의 영문자(341,362 Bytes)로 구성된 "영어 동화"[17]이다.

표 1 및 표 2에서와 같이, 제안하는 시스템은 기존의 시스템과 비교했을 때 약 19 %에서 25 %에 해당하는 전력 소모를 줄일 수 있었다.

V. 결 론

우리가 살고 있는 네트워크 환경을 안전하고 깨끗하게 유지하기 위하여, 신뢰성 있고 효율적인 네트워크 보안기술의 확립과 더불어 전력소비를 줄이는 것 또한 그 중요성을 더해가고 있다. 따라서 휴대용 단말기의 설계에서 세계적인 에너지 및 환경 문제에 이르기까지, 여러 조건에 따른 수많은 형태의 전력 절감 기술이 대두되고 있다.

본 논문에서는 파이프라인형 AC-DFA 구조의 DPI 시스템에서 전력 소비를 효율적으로 줄이는 방법을 제안하였다. 사용되지 않는 파이프라인 스테이지의 동작 주파수를 조절함으로써, 불필요하게 낭비되는 전력 소모를 줄여 시스템 전체의 전력을 효과적으로 절약할 수 있었다. 이러한 방법을 적용하여 구현한 DPI 시스템은 기존의 DPI 시스템에 비해 최대 25 %의 전력을 절약할 수 있음을 실험을 통하여 확인하였다.

본 논문에서 제안한 방법은 적은 비용과 손쉬운 구현으로, 파이프라인형 검색 구조를 가지는 어느 분야에도 적용할 수 있을 것이다. 현재 디지털 포렌식 및 빅 데이터(Big Data)로의 응용을 진행 중에 있다.

참고문헌

[1] J. Wang, H. Kwon, Y. Jung, H. Kwak, and K. Chung, "A high performance IPS based on signature hashing," in Proceedings of Korea Computer Congress 2007, Vol. 34, No. 1(D), pp. 489-494.

[2] H. Kim, Y. Kim, and J. W. Jang, "Reduced power consumption via fewer memory accesses for deep packet inspection," Journal of Information Science and Engineering, Vol. 30, No. 3, May 2014, pp. 819-833.

[3] "Snort: network intrusion detection system," Sourcefire Inc., *http://www.snort.org*

[4] W. Jiang, Y. E. Yang, and V. K. Prasanna, "Scalable multi-pipeline architecture for high performance multi-pattern string matching," in Proceedings of IEEE International Conference on Parallel and Distributed Processing Symposium, 2010, pp. 1-12.

[5] A. V. Aho and M. J. Corasick, "Efficient string matching: an aid to bibliographic search," Communications of the ACM, Vol. 18, 1975, pp. 333-340.

[6] T. Nishimura, S. Fukamachi, and T. Shinohara, "Speed-up of Aho-Corasick pattern matching machines by rearranging states," in Proceedings of the 8th IEEE International Symposium on String Processing and Information Retrieval, 2001, pp. 175-185.

[7] W. Jiang and V. K. Prasanna, "Reducing dynamic power dissipation in pipelined forwarding engines," in Proceedings of the 27th IEEE International Conference on Computer Design, 2009, pp. 144-149.

[8] A. Kennedy, X. Wang, Z. Liu, and B. Liu, "Low power architecture for high speed packet classification," in Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems,

2008, pp. 131-140.
[9] Y. Kang, "The Internet traffic management and DPI," Broadcasting and Communications Policy, Vol. 25, No. 8, 2013, pp. 23-48.
[10] M. Kassner, "Deep packet inspection: what you need to know," TechRepublic, *http://www.techrepublic.com/blog/data-center/deep-packet-inspection-what-you-need-to-know/*
[11] M. Alicherry, M. Muthuprasanna, and V. Kumar, "High speed pattern matching for network IDS/IPS," in Proceedings of IEEE International Conference on Network Protocols, 2006, pp. 187-196.
[12] D. Pao, W. Lin, and B. Liu, "Pipelined architecture for multistring matching," Computer Architecture Letters, Vol. 7, 2008, pp. 33-36.
[13] M. Pedram and J. Rabeay, "*Power Aware Design Methodologies,*" Kluwer Academic Publishers, 2002, pp. 171-174 and pp. 386-412.
[14] S. B. Hyun, S. W. Kang and N. W. Eum, "Low power SoC technology wireless terminals," Electronics and Telecommunications Trends, Vol. 23, 2008, pp. 92-101.
[15] "Xilinx power estimator (XPE)," Xilinx Inc., *http://www.xilinx.com*
[16] "Samsung consumer DRAM," Samsung Electronics Co., Ltd., http://www.samsung.com/*global/business/semiconductor/product/consumer-dram/catalogue*
[17] "English fairy tales," Joseph Jacobs Page, PSU's Electronic Classics Site, http://www.hn.psu.*edu/faculty/jmanis/jimspdf.htm*

저 자 소 개

김 한 수
2002: 서강대학교 전자공학과 공학사
2004: 서강대학교 전자공학과 공학석사
2014: 서강대학교 전자공학과 공학박사
현 재: 국립과학수사연구원 디지털분석과
관심분야: 침입탐지, 네트워크보안, 문자열검색, 디지털포렌식, 영상분석, 문서감정
email : kutestar@korea.kr