논문 15-40-06-07 The Journal of Korean Institute of Communications and Information Sciences '15-06 Vol.40 No.06
http://dx.doi.org/10.7840/kics.2015.40.6.1032

효율적인 서비스 기능 체이닝을 위한 최적의 플로우 분배 알고리즘

김 명 수◆, 이 기 원*, 주 석 진*, 백 상 헌°, 김 영 화**

Optimal Flow Distribution Algorithm for Efficient Service Function Chaining

Myeongsu Kim◆, Giwon Lee*, Sukjin Choo*, Sangheon Pack°, Younghwa Kim**

요 약

서비스 기능 체이닝(SFC: Service function chaining)은 다수의 서비스 기능들을 순차적으로 구성하는 기술이다. 서비스 기능 체이닝에서는 확장성과 fault-tolerant를 위해 다수의 서비스 기능 인스턴스가 필요하며, 네트워크에 인입된 플로우는 다수의 서비스 기능 인스턴스로 적절하게 분배되어야 한다. 따라서 본 논문에서는 각 서비스 기능인스턴스들의 자원을 고려하면서 종단 간 지연시간(latency)을 최소화 할 수 있는 플로우 분배 문제를 정의한다. 또한 GT-ITM 토폴로지 생성 툴을 사용하여 보다 현실적인 네트워크 토폴로지 상에서 시뮬레이션을 수행하고, 그 결과 최적의 플로우 분배 기법이 전체 지연시간을 줄일 수 있음을 확인하였다.

Key Words : Flow distribution, integer linear programming, network function virtualization, service function chaining

ABSTRACT

Service function chaining(SFC) defines the creation of network services that consist of an ordered set of service function. A multiple service function instances should be deployed across networks for scalable and fault-tolerant SFC services. Therefore, an incoming flows should be distributed to multiple service function instances appropriately. In this paper, we formulate the flow distribution problem in SFC aiming at minimizing the end-to-end flow latency under resource constraints. Then, we evaluate its optimal solution in a realistic network topology generated by the GT-ITM topology generator. Simulation results reveal that the optimal solution can reduce the total flow latency significantly.

Ⅰ. 서 론

최근 네트워크, 서버, 저장 장치, 응용프로그램들이 가상화(virtualization), 네트워크 오버레이(network overlay), 오케스트레이션(orchestration) 등의 기술 발전으로 인해 새로운 변화를 겪고 있다. 하지만 네트워

※ 본 연구결과의 일부는 International Conference on Future Internet Technologies (CFI) 2015 국제학술대회에서 발표되었음[1]
※ 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 스마트 네트워킹 핵심 기술 개발 사업 (14-000-05-001) 및 대학 ICT연구센터 육성지원사업의 연구결과로 수행되었음 (IITP-2015-H8501-15-1007)
◆ First Author : Department of Electrical Engineering, Korea University, myeongsukim@korea.ac.kr, 학생회원
° Corresponding Author : Department of Electrical Engineering, Korea University, shpack@korea.ac.kr, 종신회원
* Department of Electrical Engineering, Korea University, {goodkw, choo0128}@korea.ac.kr
** Electronics and Telecommunications Research Institute (ETRI), yhwkim@etri.re.kr
논문번호 : KICS2015-05-146, Received May 12, 2015; Revised June 2, 2015; Accepted June 2, 2015

크 주소 변환(NAT: Network address translation), 침입 방지 시스템(IPS: Intrusion prevention system), 중복 제거(RE: Redundancy elimination)와 같이 네트워크에 필수적인 기능들은 여전히 해결해야 할 많은 이슈가 있다. 따라서 네트워크 운영자는 네트워크에 필수적인 기능들을 관리하기 위해 높은 CAPEX 및 OPEX가 든다. 이를 해결하기 위해 네트워크의 기능을 소프트웨어적으로 가상화 하여 다수의 서비스 기능들로 구성하는 네트워크 기능 가상화(NFV: Network function virtualization) 기술이 등장하였으며, 서비스 기능들을 순차적으로 처리하는 기술인 서비스 기능 체이닝 기술이 각광 받고 있다[2-3]. 또한 서비스 기능 체이닝을 지원하는 기술로서 소프트웨어 정의 네트워킹(SDN: Software defined networking) 기술이 소개되었다[4-7]. 즉, 소프트웨어 정의 네트워킹의 중앙 집중화된 컨트롤러 구조와 프로그래밍 능력을 통해 네트워크 상황에 맞춰 동적으로 네트워크를 관리하는 것이 가능하며 다양한 네트워크 서비스의 적용 및 확대가 가능하다.

서비스 기능 체이닝 기술과 관련하여 국제 표준화 기구인 IETF에서는 서비스 기능 체이닝 워킹 그룹[8]을 구성하여 서비스 기능 체이닝과 관련된 구조 및 프로토콜을 정의한다. IETF 인터넷 표준화 문서[9-10]의 정의에 따르면, 서비스 기능 체이닝은 다수의 서비스 기능(SF: Service function)들을 하나의 논리적인 체인으로 구성한 뒤 순차적으로 처리하는 기술이다. 그림 1은 2개의 서비스 기능 체인(SF1-SF2-SF3-SF4, SF1-SF2-SF5)이 존재하는 서비스 기능 체이닝의 예제를 나타낸다. 그림 1에서는 네트워크에 플로우가 인입되면 먼저 분류자(classifier)에 의해 플로우가 분류되며, 분류된 플로우는 정해진 서비스 기능 체인에 따라 순차적으로 전달되고 처리된다. 서비스 기능 전달자(SFF: Service function forwarder)는 플로우를 해당하는 서비스 기능에게 전달해주는 기능을 수행한다.

서비스 기능 체이닝에서는 확장성과 fault-tolerant를 위해 동일한 기능을 수행하는 다수의 서비스 기능 인스턴스들을 생성하고 배치시키는데 많은 이슈가 있다. 이와 관련하여 IETF 인터넷 표준화 문서 중 자원 관리(resource management)[11]문서 에서는 서비스 기능 인스턴스의 fault, 경로 최적화, 플로우 최적화, 부하 분산 등의 유즈케이스를 다루고 있다.

본 논문에서는 네트워크에 플로우가 인입되면 해당 플로우를 다수의 서비스 기능 인스턴스들에게 적절하게 분배시키는 플로우 최적화에 초점을 맞춘다. 서비스 기능 체이닝에서 플로우를 분배하는 기존의 연구

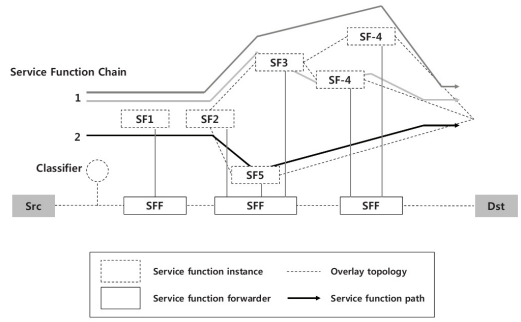

그림 1. 서비스 기능 체이닝 예제
Fig. 1. Service function chaining example

에는 균등 플로우 분배 기법과 네트워크 인지 플로우 분배 기법 2가지가 있다[12]. 첫 번째 균등 플로우 분배 기법(uniform flow distribution)에서는 네트워크에 인입되는 플로우를 균등하게 서비스 기능 인스턴스들에게 분배한다. 예를 들어, 서비스 기능 체인이 IPS-RE의 순서로 구성되어 있고, 각 서비스 기능 IPS와 RE에는 3개와 4개의 서비스 기능 인스턴스가 존재한다고 가정한다. 만약 네트워크에 N개의 플로우가 인입되면 각 IPS 서비스 기능 인스턴스는 균등하게 N/3개의 플로우를 전달받고 처리한다. 처리를 마친 플로우는 다음 RE 서비스 기능 인스턴스로 균등하게 분배되며, 이를 통해 각 IPS 서비스 기능 인스턴스는 N/12개의 플로우를 각 RE 서비스 기능 인스턴스에게 전달한다. 따라서 각 RE 서비스 기능 인스턴스는 N/4개의 플로우를 전달 받는다. 하지만 균등 플로우 분배 기법은 서비스 기능 인스턴스의 자원과 서비스 기능 인스

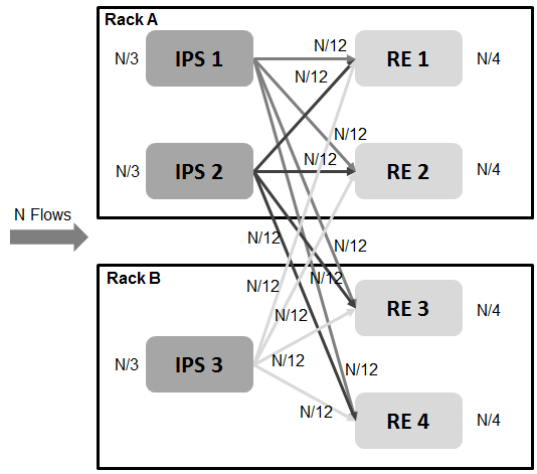

그림 2. 균등 플로우 분배 예제
Fig. 2. Uniform flow distribution example

턴스간 링크의 대역폭 혹은 지연시간을 고려하지 않고 플로우를 분배시키는 한계점이 있다.

기존의 균등 플로우 분배 기법의 한계점을 해결하기 위해 서비스 기능 인스턴스간 링크의 지연시간을 고려하는 네트워크 인지 플로우 분배 기법(network-aware flow distribution)이 제안되었다. 그림 2와 그림 3은 균등 플로우 분배 기법과 네트워크 인지 플로우 분배 기법의 비교를 보여준다. 먼저 서비스 기능 체인이 IPS-RE의 순서로 구성되어 있고, 각 서비스 기능 IPS와 RE에는 3개와 4개의 서비스 기능 인스턴스가 존재한다고 가정한다. 또한 IPS1, IPS2, RE1, RE2 서비스 기능 인스턴스는 랙 A에 배치되어 있고, 나머지 IPS3, RE3, RE4 서비스 기능 인스턴스는 랙 B에 배치되어 있다고 가정한다. 먼저 균등 분배 기법에서는 만약 네트워크에 N개의 플로우가 인입되면 균등 플로우 분배 기법에서는 각 IPS 서비스 기능 인스턴스가 균등하게 N/3개의 플로우를 전달받고 처리하며, 처리를 마친 플로우는 N/12개 만큼 균등하게 다음 RE 서비스 기능 인스턴스에게 분배된다. 따라서 각 RE 서비스 기능 인스턴스는 N/4개의 플로우를 전달 받는다.

반면 그림 3의 네트워크 인지 플로우 분배 기법에서는 랙 간에 이동하는 플로우의 양을 최소화 시킬 수 있도록 플로우를 분배한다. 즉, 동일한 랙에 배치된 서비스 기능 인스턴스간 플로우를 분배시키는 경우 발생하는 비용을 0으로 가정하고, 다른 랙에 배치된 서비스 기능 인스턴스간 플로우를 분배시키는 경우 발생하는 비용을 1로 가정한다. 이와 같이 플로우 분배 시 발생하는 비용을 고려하여 종단 간 지연시간을

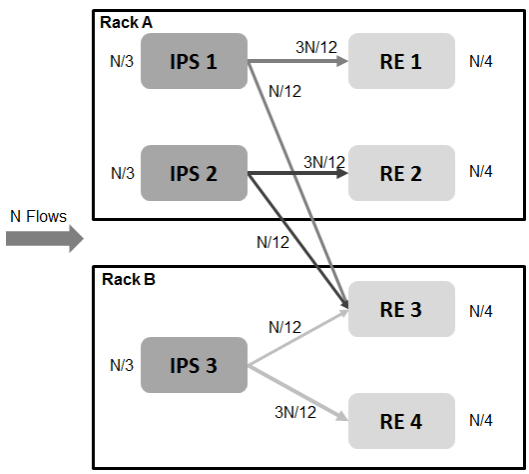

그림 3. 네트워크 인지 플로우 분배 예제
Fig. 3. Network-aware flow distribution example

최소화하는 문제를 LP(linear programming)로 정의하였다. 그 결과 그림 3의 네트워크 인지 플로우 분배 기법 예제와 같이 IPS1 서비스 기능 인스턴스에서는 N/3개의 플로우를 전달 받아 처리하고, 랙 간에 이동하는 플로우의 양을 최소화 시킬 수 있도록 동일한 랙에 배치된 RE1 서비스 기능 인스턴스에게 3N/12개의 플로우를 분배한다. 또한 다른 랙에 배치된 RE3 서비스 기능 인스턴스에게는 N/12개의 플로우를 분배한다. 따라서 각 RE 서비스 기능 인스턴스는 그림 2의 균등 플로우 분배 기법과 동일하게 N/4개의 플로우를 전달 받는다. 하지만 균등 플로우 분배 기법에서는 랙 A와 랙 B 사이에서 이동하는 플로우의 양은 4N/12개이지만, 네트워크 인지 플로우 분배 기법에서는 랙 A와 랙 B 사이에서 이동하는 플로우의 양은 2N/12개이다. 이를 통해 네트워크 인지 플로우 분배 기법에서는 랙 간에 이동하는 플로우의 양을 최소화 시킬 수 있다. 하지만 실제 네트워크에는 서비스 기능 인스턴스들이 다양한 물리적인 노드에 배치가 되며 서비스 기능 인스턴스간 다양한 지연시간이 존재한다. 따라서 실질적인 지연시간을 고려한 최적의 플로우 분배 기법이 필요하다.

본 논문에서는 서비스 기능 체이닝에서 각 서비스 기능 인스턴스들의 자원을 고려하면서 종단 간 지연시간을 최소화할 수 있는 플로우 분배 문제를 ILP(integer linear programming)로 정의한다. 또한 기존의 네트워크 인지 플로우 분배 기법과 다르게 본 논문에서는 운영자의 네트워크에 초점을 맞추어 다수의 데이터 센터를 가정한다. 더불어 제안한 최적의 플로우 분배 기법의 성능 평가를 위해 GT-ITM[13] 토폴로지 생성 툴을 사용하여 보다 현실적인 네트워크 토폴로지 상에서 시뮬레이션을 수행하고, 그 결과 제안하는 최적의 플로우 분배 기법이 기존 기법들에 비해 전체 지연시간을 감소시킨 것을 확인하였다.

본 논문의 구성은 다음과 같다. 다음 2장에서는 최적의 플로우 분배를 위한 문제를 정의하고, 3장에서는 시뮬레이션 결과를 살펴보고, 마지막으로 4장에서는 결론을 맺는다.

Ⅱ. 문제 정의

본 장에서는 최적의 플로우 분배를 위한 문제를 정의한다. 먼저 플로우 분배를 위한 시스템 모델은 그림 4와 같다. 시스템 모델에서는 서비스 기능 체이닝을 위한 서비스 기능 s가 존재한다고 가정하며, 서비스

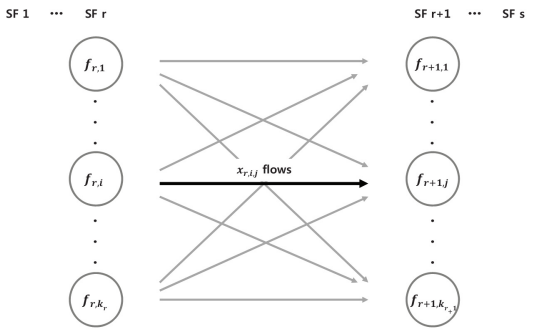

그림 4. 시스템 모델
Fig. 4. System model

기능 인덱스의 집합을 $S=\{1,2,\cdots,s\}$로 정의한다. 또한 r번째 서비스 기능 ($r\in S$)은 k_r개의 인스턴스를 갖으며, $f_{r,1}, f_{r,2}, \cdots, f_{r,k_r}$과 같이 정의한다. 그리고 F_r은 r번째 서비스 기능 인스턴스 인덱스의 집합을 의미하며 $F_r=\{1,2,\cdots,k_r\}$과 같이 정의한다. 또한 각 서비스 기능 인스턴스 $f_{r,i}(i\in F_r)$에 대해서 $r+1$번째 서비스 기능 인스턴스 $f_{r+1,j}(j\in F_{r+1})$로 분배되는 플로우 수를 $x_{r,i,j}$로 정의하며, $x_{r,i}$는 $f_{r,i}$ 서비스 기능 인스턴스에서 처리될 수 있는 플로우 수를 나타낸다. 그리고 각 서비스 기능 인스턴스가 갖는 자원의 양을 $C_{r,i}$ 로 정의한다. 그 다음 $f_{r,i}$와 $f_{r+1,j}$는 다수의 물리적인 연결을 통해 서로 연결되어 있다고 가정하며, $L_{r,i,j,m}$을 $f_{r,i}$와 $f_{r+1,j}$ 사이에서 m번째 물리적인 링크의 지연시간으로 정의한다. $f_{r,i}$와 $f_{r+1,j}$ 사이에서 링크의 지연시간 총합은 $L_{r,i,j}$로 정의하며 이는 $\sum_m L_{r,i,j,m}$을 통해 계산된다. 링크의 지연시간은 플로우의 사이즈와 링크의 대역폭을 통해 결정되며 서비스 기능 인스턴스에서의 큐잉 지연시간은 총 지연시간에 큰 영향을 미치지 않기 때문에 고려하지 않는다[14]. 플로우 분배 문제는 전체 링크의 지연시간 합을 최소화 하도록 다음과 같이 정의할 수 있다.

$$\min_{x_{r,i,j}}\left(\sum_{r,i,j} x_{r,i,j} * L_{r,i,j}\right) \tag{1}$$

또한 제약 조건은 다음과 같이 정의한다.

$$\forall r\in S, \forall i\in F_r: x_{r,i} \le C_{r,i} \tag{a}$$

$$\forall r\in S: \sum_{i,j} x_{r,i,j} = N \tag{b}$$

$$\forall r\in S, \forall i\in F_r: \sum_{j} x_{r,i,j} = x_{r,i} \tag{c}$$

$$\forall r\in S, \forall j\in F_{r+1}: \sum_{i} x_{r,i,j} = x_{r+1,j} \tag{d}$$

$$\forall r\in S, \forall i\in F_r, \forall j\in F_{r+1}: x_{r,i,j}\in\{1,2,\cdots,N\} \tag{e}$$

제약 조건 (a)에서는 각 서비스 기능 인스턴스가 갖는 자원 $C_{r,i}$에 따른 $x_{r,i}$의 제한을 나타내며, (b)에서는 주어진 네트워크에 N개의 플로우가 인입되면 모든 서비스 기능으로 분배되는 플로우 양의 총합도 N으로 동일함을 나타낸다. (c)에서는 $f_{r,i}$에서 처리되는 플로우는 F_{r+1}의 서비스 기능 인스턴스로 모두 분배됨을 나타내며, (d)에서는 r번째 서비스 기능에서 분배되는 플로우가 서비스 기능 인스턴스 $f_{r+1,j}$로 모두 전달됨을 나타낸다. 마지막 (e)에서는 $x_{r,i,j}$는 서비스 기능 인스턴스 $f_{r,i}$에서 $f_{r+1,j}$로 분배되는 플로우 양을 의미하며 정수값을 갖는다. 이와 같이 서비스 기능 인스턴스 사이 링크의 지연시간과 서비스 기능 인스턴스의 자원을 동시에 고려하여 최적의 플로우 분배 양을 구할 수 있다.

Ⅲ. 시뮬레이션 결과

서비스 기능 체이닝을 실질적인 환경에서 실험하기에는 많은 어려움이 있기 때문에 대부분의 기존 연구[15-17]에서는 시뮬레이션 툴을 이용하여 성능 분석을 수행한다. 또한 아마존 EC2와 같은 환경에서 시뮬레이션을 수행한 연구[18]도 있지만 가상의 서비스 기능 인스턴스를 배치하고 임의로 플로우를 분배시키는 실험을 수행하기에는 어려움이 존재한다. 따라서 본 장에서는 GT-ITM 토폴로지 생성 툴과 Matlab을 사용하여 최적의 플로우 분배 문제를 시뮬레이션 한 결과를 살펴본다. 최적의 플로우 분배 문제를 시뮬레이션 하기 위해 GT-ITM 토폴로지 생성 툴을 사용하여 실제 네트워크 환경의 토폴로지를 생성하였다. 그림 5는 GT-ITM 토폴로지 생성 툴을 사용하여 생성한 transit-stub[19] 네트워크 모델을 나타낸다. transit-stub 네트워크 모델에서는 1개의 transit domain과 4개의 stub domain이 존재한다. 또한 transit-stub 네트워크 모델에서 transit domain은 stub domain들과 서로 연결되어 있는 구조이며, 각 domain에는 4개의 노드가

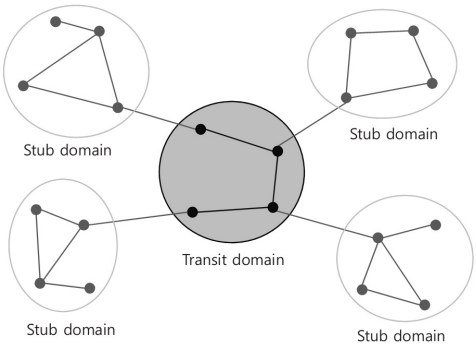

그림 5. Transit-Stub 모델
Fig. 5. Transit-Stub model

존재한다고 가정하였다. 서비스 기능 체이닝의 경우 3개의 서비스 기능(즉, $s = 3$)이 있으며, 각 서비스 기능 인스턴스도 3개(즉, $k_r = 3$)가 존재한다고 가정하였다. 또한 서비스 기능 인스턴스, 출발지 노드, 목적지 노드의 경우 초기에 transit domain과 stub domain에 랜덤으로 배치되며, 모든 인스턴스 혹은 노드들 간 링크의 지연시간을 구하기 위해 Euclidean distance를 사용하였다[20].

본 논문에서는 최적의 플로우 분배 기법(Optimal 기법), 균등 플로우 분배 기법(Uniform 기법), 네트워크 인지 플로우 분배 기법(Aware 기법)의 전체 지연시간을 비교한다. Aware 기법에서는 서비스 기능 인스턴스가 동일한 transit domain 혹은 stub domain에 존재하는 경우 링크의 지연시간을 0으로 설정하고, 다른 domain에 존재하는 경우 링크의 지연시간을 1로 설정하였다. 전체 플로우 N의 개수는 720개로 가정하였다.

3.1 $C_{r,i}$의 영향

그림 6에서는 서비스 기능 인스턴스들의 자원에 따른 전체 지연시간(total flow latency)를 나타내며, 모든 서비스 기능 인스턴스들은 동일한 캐패시티를 갖는다고 가정하였다. Uniform 기법은 서비스 기능 인스턴스의 캐패시티를 고려하지 않기 때문에 가장 높은 전체 지연시간을 갖고, Aware 기법은 Uniform 기법보다 낮은 전체 지연시간을 갖는다. 이는 Aware 기법이 서비스 기능 인스턴스 배치에 따른 링크 지연시간을 고려하여 플로우를 분배시키기 때문이다. 하지만 Aware 기법의 경우 서비스 기능 인스턴스들의 캐패시티를 고려하지 않았기 때문에 자원 $C_{r,i}$에 영향을 받지 않는다. 반면 Optimal 기법은 서비스 기능 인스

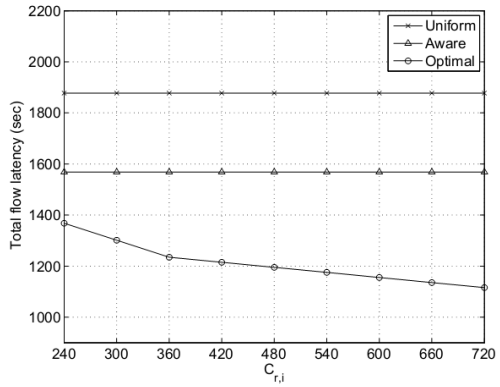

그림 6. $C_{r,i}$의 영향
Fig 6. Effect of $C_{r,i}$

표 1. 플로우 드롭 비율 (%)
Table 1. Flow dropping rate (%)

σ	Uniform	Aware	Optimal
10	3.84	3.84	0
20	8.73	8.73	0
30	11.96	11.96	0

턴스의 캐패시티가 증가할수록 전체 플로우 지연시간이 감소한다. 이는 서비스 기능 인스턴스간 작은 링크 지연시간을 갖고, 캐패시티가 많은 서비스 기능 인스턴스에게 더 많은 플로우를 분배시키기 때문이다. 또한 Optimal 기법의 경우 Uniform 기법에 비해 27.14 - 40.56%의 전체 지연시간 감소를 보이며, Aware 기법에 비해 12.77 - 28.84%의 전체 지연시간 감소를 보인다.

보다 현실적인 서비스 기능 체이닝 환경에서는 다수의 서비스 기능 인스턴스들이 다른 캐패시티를 갖기 때문에, 캐패시티를 초과하는 플로우가 인입되는 경우 플로우가 드롭(drop) 된다. 따라서 서비스 기능 인스턴스들이 평균 240과 표준 편차 σ에 따른 캐패시티를 가질 때, 플로우가 드롭 되는 비율을 계산하였다. 표 1은 각 기법에 따른 플로우 드롭 비율을 나타낸다. 표준 편차 σ가 커짐에 따라 Uniform 기법과 Aware 기법의 플로우 드롭 비율은 증가한다. 이는 Uniform과 Aware 두 기법 모두 서비스 기능 인스턴스의 캐패시티를 고려하지 않았기 때문이다. 또한 동일한 양의 플로우가 각 서비스 기능 인스턴스에게 분배되기 때문에 Uniform과 Aware 기법이 동일한 드롭 비율을 갖는다. 반면, Optimal 기법의 경우 서비스 기능 인스턴스의 캐패시티를 고려하여 적응적으로 플로우를 분배하기 때문에 플로우 드롭이 발생하지 않는다.

3.2 k_r의 영향

그림 7은 서비스 기능 인스턴스의 개수 k_r에 따른 전체 지연시간을 나타내며, 이 때 모든 서비스 기능들은 동일한 인스턴스의 개수를 갖는다고 가정한다. 각 서비스 기능 인스턴스는 모두 540의 동일한 캐패시티를 갖는다고 가정한다. 그 결과 모든 기법이 서비스 기능 인스턴스의 개수가 증가함에 따라 전체 지연시간이 감소한다. 이는 플로우가 더 많은 인스턴스로 분배되기 때문이다. 반면 Optimal 기법의 경우 서비스 기능 인스턴스의 개수가 증가함에 따라 가장 큰 전체 지연시간의 감소가 발생한다. 또한 서비스 기능 인스턴스의 개수가 2개에서 4개로 증가할 때 Optimal 기법의 전체 지연시간은 43.39% 감소하고, Uniform 기법과 Aware 기법은 각각 20.06%, 30.45% 감소한다. 따라서 Optimal 기법은 큰 스케일을 갖는 서비스 기능 체이닝 환경에서 더욱 효과적으로 플로우 분배가 가능함을 확인하였다.

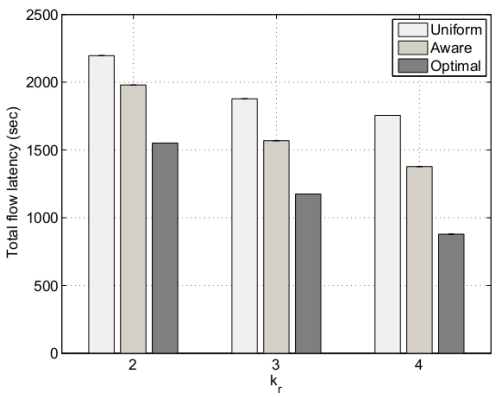

그림 7. k_r의 영향
Fig. 7. Effect of k_r

Ⅳ. 결 론

본 논문에서는 서비스 기능 체이닝에서 서비스 기능 인스턴스들의 자원을 고려함과 동시에 종단 간 지연시간을 최소화 할 수 있는 최적의 플로우 분배 문제를 ILP(integer linear programming)로 정의하였으며, 시뮬레이션 결과를 통해 최적의 플로우 분배 기법이 다른 기존의 기법에 비해 전체 지연시간이 감소했음을 확인할 수 있다. 또한 서비스 기능의 자원이 증가하고 서비스 기능 인스턴스의 개수가 많아지는 경우 제안하는 기법이 기존 기법에 비해 더욱 효율적인 것을 확인할 수 있다. 향후 연구에는 가상화된 서비스 기능과 물리적인 서비스 기능이 존재하는 경우에서 플로우 분배 문제에 대해서 연구할 예정이다.

References

[1] G. Lee, M. Kim, S. Choo, S. Pack, and Y. Kim, “Optimal flow distribution in service function chaining,” in *Proc. Int. Conf. Future Internet Technologies (CFI)*, Jun. 2015.

[2] P. Quinn and J. Guichard, “Service function chaining: Creating a service plane via network service headers,” *J. Computer*, vol. 47, no. 11, pp. 38-44, Nov. 2014.

[3] W. John, K. Pentikousis, G. Agapiou, E. Jacob, M. Kind, A. Manzalini, F. Risso, D. Staessens, R. Steinert, and C. Meirosu, “Research directions in network service chaining,” in *Proc. IEEE SDN4FNS*, pp. 1-7, Trento, Nov. 2013.

[4] S. Mehraghdam, M. Keller, and H. Karl, “Specifying and placing chains of virtual network functions,” in *Proc. IEEE CloudNet,* pp. 7-13, Luxembourg, Oct. 2014.

[5] G. Lee, I. Jang, W. Kim, S. Joo, M. Kim, S. Pack, and C. Kang, “SDN-based middlebox management framework in integrated wired and wireless networks,” *J. KICS*, vol. 39B, no. 6, pp. 379-386, Jun. 2014.

[6] J. Jo, S. Lee, J. Kong, and J. Kim, “A centralized network policy controller for SDN-based service overlay networking,” *J. KICS*, vol. 38B, no. 4, pp. 266-278, Apr. 2013.

[7] H. Kim and H. Kim, “Control algorithm for virtual machine-level fairness in virtualized cloud data center,” *J. KICS*, vol. 38C, no. 6, pp. 512-520, Jun. 2013.

[8] IETF Service Function Chaining (SFC) Working Group (WG), Retrieved May 1, 2015, from https://datatracker.ietf.org/wg/sfc/charter/

[9] J. Guichard and C. Pignataro, “Service function chaining (SFC) architecture,” Internet-Draft draft-sfc-architecture-05, Feb. 2015.

[10] W. Liu, H. Li, O. Huang, M. Boucadair, N. Leymann, Z. Cao, Q. Sun, and C. Pham, *Service function chaining (SFC) general use*

cases, Internet-Draft draft-liu-sfc-use-cases-08, Sept. 2014.

[11] S. Lee, S. Pack, M. Shin, and E. Paik, *Resource management for dynamic service chain adaptation*, internet-draft draft-lee-nfvrg-resource-management-service-chain-00, Oct. 2014.

[12] A. Gember, A. Krishnamurthy, S. S. John, R. Grandl, X. Gao, A. Anand, T. Benson, A. Akella, and V. Sekar, *Stratos: A network-aware orchestration layer for middleboxes in the cloud*, arXiv preprint arXiv:1305.0209, 2013.

[13] K. Calvert and E. Zegura, *GT internetwork topology models (GT-ITM)*, Retrieved May 1, 2015, from http://www.cc.gatech.edu/fac/Ellen.Zegura/gt-itm.

[14] G. Wang and T. S. E. Ng, "The impact of virtualization on network performance of amazone EC2 data center," in *Proc. IEEE INFOCOM 2010*, pp. 1-9, San Diego, CA, Mar. 2010.

[15] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, "Optical service chaining for network function virtualization," *IEEE Commun. Mag.*, vol. 53, no. 4, pp. 152-158, Apr. 2015.

[16] S. Mehraghdam, M. Keller, and H. karl, "Specifying and placing chains of virtual network functions," in *Proc. IEEE Cloud Netw. (CloudNet)*, pp. 7-13, Luxembourg, Jun. 2014.

[17] M. Bagaa, T. Taleb, and A. Ksentini, "Service-aware network function placement for efficient traffic handling in carrier cloud," in *Proc. IEEE Wirel. Commun. Netw. Conf. (WCNC) 2014*, pp. 2402-2407, Istanbul, Apr. 2014.

[18] C. Pham, H. D. Tran, S. I. Moon, K. Thar, and C. S. Hong, "A general and practical consolidation framework in CloudNFV," in *Proc. IEEE Int. Conf. Inf. Netw. (ICOIN) 2015*, pp. 295-300, Cambodia, Jan. 2015.

[19] O. Heckmann, M. Piringer, J. Schmitt, and R. Steinmetz, "On realistic network topologies for simulation," in *Proc. ACM SIGCOMM MoMeTools Workshop*, pp. 28-32, Aug. 2003.

[20] B. Zhang, T. S. E. Ng, A. Nandi, R. Riedi, P. Druschel, and G. Wang, "Measurement-based analysis, modeling, and synthesis of the internet delay space," *IEEE/ACM Trans. Netw.*, vol. 18, no. 1, pp. 229-242, Feb. 2010.

김 명 수 (Myeongsu Kim)

2014년 2월 : 한성대학교 정보통신공학과 학사

2014년~현재 : 고려대학교 전기전자공학과 석·박사과정

<관심분야> 미래 인터넷

이 기 원 (Giwon Lee)

2009년 2월 : 고려대학교 전기전자전파공학부 학사

2011년 2월 : 고려대학교 전기전자전파공학부 석사

2011년~현재 : 고려대학교 모바일 솔루션 학과 박사과정

<관심분야> 소프트웨어 정의 네트워크, 모바일 클라우드

주 석 진 (Sukjin Choo)

2013년 8월 : 고려대학교 전기전자전파공학부 학사

2013년~현재 : 고려대학교 전기전자공학과 석 · 박사과정

<관심분야> 미래 인터넷, 모바일 데이터 오프로딩

백 상 헌 (Sangheon Pack)

2000년 2월 : 서울대학교 컴퓨터공학부 학사
2005년 2월 : 서울대학교 전기컴퓨터공학부 박사
2007년~현재 : 고려대학교 전기전자전파공학과 부교수
<관심분야> 미래 인터넷, 무선 이동 네트워크

김 영 화 (Younghwa Kim)

2005년 2월 : 충남대학교 공학박사
1998년~현재 : 한국전자통신연구원 책임연구원
2015년~현재 : 한국전자통신연구원 통신인터넷연구소 SDN 기술연구실장
<관심분야> 미래 인터넷, 소프트웨어 정의 네트워크, 네트워크기능 가상화