Journal of Knowledge Information Technology and Systems

ISSN 1975-7700

http://www.kkits.or.kr

Optimal Multimedia Data Division Method for Network User using Docker-based Virtual Machine

Sanghyun Park[1], Ho-Yong Ryu[2], Jinsul Kim[*1]

[1]*School of Electronics and Computer Engineering Chonnam National University*
[2]*Smart Network Research Department, Electronics and Telecommunications Research Institute, Daejeon, Korea*

ABSTRACT

Existing multimedia transmission method does not consider the network situation in real time such as a frequent interruption while playing back at a multimedia client. Thus, in this paper, we propose a customized split transmission method that considers the device characteristic and network environment. The user receives multimedia data based on a Docker-based system. Besides, we describe the configuration of a network environment in the Docker platform where multimedia quality changes according to the changing of network condition in real time. The virtual machine consists of a main server that manages the entire server and data servers that provide multimedia services. Moreover, we build a Docker-based virtual machine to cope with a changing network environment. The main server manages the generated data servers in real time. When a user requests multimedia contents, the main server selects an optimal server through a redirect algorithm based on the user's network and data server environment. The data servers provide multimedia quality in real time according to the user's network situation through the multimedia division transmission technique. In the experiment, we compare and analyze the data transmission rate between two methods, 1:1 multimedia transmission and server-based network transmission. Throughout comparative analysis of experiments, it showed that the split transmission method is faster than the conventional method.

KEYWORDS: Division transmission, Optima multimedia transmission, Docker, User customized, Redirect, Dynamic network

ARTICLE INFO: Received 22 February 2017, Revised 14 March 2017, Accepted 7 April 2017.

[*]Corresponding author is with the School of Electronics and Computer Engineering Chonnam National University, Gwangju, Korea
E-mail address: jsworld@jnu.ac.kr

1. 서 론

전 세계적으로 스마트 폰과 이동통신 서비스의 발달로 인하여 실시간으로 다양한 데이터를 원활하게 전송할 수 있는 기술들이 연구되고 있다[1,2]. 그중에서 가장 많은 데이터 비중을 차지하는 멀티미디어 데이터는 네트워크 기반의 원활한 서비스를 제공하기 위해서 제공 업체에서는 많은 양의 데이터 센터를 보유하거나 외부 데이터 센터를 이용하여 사용자들에게 서비스를 제공하고 있다[3]. 하지만 방대한 양의 멀티미디어 자료를 관리하기에는 많은 양의 비용과 소규모 업체에서는 불가능하다. 또한 제공하는 서버가 사용자로부터 멀리 떨어져 있거나 서버의 과부하가 발생한다면 사용자에게 원활한 서비스를 제공하지 못한다. 따라서 본 논문에서는 멀티미디어 콘텐츠를 제공하기 위해 가상머신 기반의 사용자 맞춤형 멀티미디어 분할 전송 방법을 제안하고자 한다.

2. 관련연구

멀티미디어 콘텐츠 제공방법과 관련하여 많은 연구들이 진행되었으며, 그 중에서 클라우드 기반의 캐싱과 라우팅 최적화를 제시한 논문[4]에서는 다양한 콘텐츠를 업로드 하는데 제한된 스토리지의 비용과 물리적 자원에 따른 문제점을 해결하기 위해 지연시간을 줄여 사용자가 요청한 콘텐츠를 빠르게 제공하는 방법을 제안한다. 제안된 방법은 물리적 스토리지 자원을 동적 캐싱을 이용하여 콘텐츠를 전송하는데 이는 기존의 알고리즘들 보다 적은 비용이 소모된다. 분산 클라우드 서비스 관련 논문[5]에서는 분산 네트워크 서비스의 크기와 데이터센터의 증가로 인하여, WAN 대역폭의 비용과 데이터센터 능력의 제한을 고려한 데이터 분배가 필요하기 때문에 데이터 센터 요청에 따른 최적화된 알고리즘과 접속 패턴 및 클라이언트 위치에 대한 로그 수집에 관한 방법을 제안하였다. 논문에서는 제안한 알고리즘을 이용하여 수집한 로그 데이터를 분석하고 분석한 데이터를 이용하여 클라우드 서비스에 적용한다. <그림 1>은 논문에서 제안한 데이터 순서도이다.

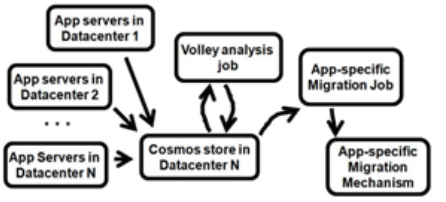

그림 1. Volley 동작 순서도
Figure 1. Using the flowchart Volley

스토리지와 멀티 캐스트의 최적화를 통한 VoD(video on demand) 서비스를 제시한 논문[6]에서는 VoD 시스템의 사용자 수와 라이브러리의 데이터양이 계속 늘어남에 따른 VoD 서비스에 확장성과 대역폭의 중요성이 부각된다고 설명하였다. 따라서 이를 해결하기 위해 PAB-MP(Prepopulation assisted batching with multicast patching) 개발을 하였으며 최종 사용자 장치에 영상 초기 세그먼트를 미리 읽어와 서버로부터의 비디오 멀티 캐스트를 빠르고 간편한 구조로 만들었다. <그림 2>는 논문에서 제안한 네트워크의 구조이다.

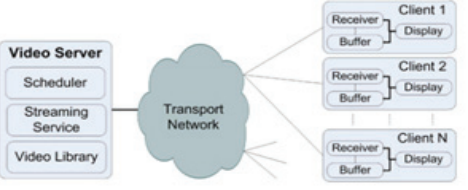

그림 2. 비디오 전송 네트워크 구조
Figure 2. Structure of Video delivery network

위 논문들의 멀티미디어 전송 방법들은 서버를

기준으로 서비스를 제공하기 때문에 아무리 좋은 서버가 구축이 되어 있어도 사용자에게 적합한 서비스가 제공되지 못한다면 소용이 없다. 본 논문에서는 각각의 서버들을 하나로 통합하여 사용자를 중심으로 최적의 서버를 선택한 후 멀티미디어 콘텐츠를 제공하는 알고리즘을 개발하고 이를 적용한 테스트 및 분석을 한다.

3. 최적의 멀티미디어 전송을 위한 서버 환경 구성

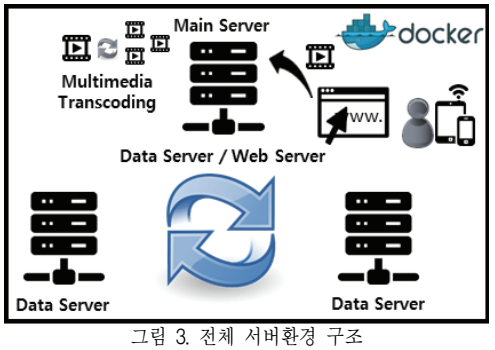

그림 3. 전체 서버환경 구조
Figure 3. Overall server environment structure

알고리즘 1. 멀티미디어 서비스 제공을 위한 서버 검색
Algorithm 1. Search server for multimedia service

```
WHEN u Request the content Data THEN // ①
  WHILE S_i != NULL THEN
      S_tmp = Server Check(S_i);
      IF PerformanceCompare(S_opi, S_tmp) THEN // ②
          S_opi = S_tmp;
          i++;
      END IF
  END WHILE
      u = ConnectionServer(S_opi); // ③
END WHEN
```

본 논문에서는 Docker[7,8]기반의 플랫폼을 이용한 네트워크 환경을 구성하였다. Docker[9]는 PaaS(Platform as a Service)[10] 형태로 제공되며, 단순 명령어를 통해 기존에 구성한 데이터 서버를 쉽게 구축할 수 있기 때문에 Docker기반의 플랫폼을 사용하였다. 웹 서버와 데이터 서버는 Node.js 언어를[11,12] 이용하여 개발을 하였다.

<그림 3>과 같이 서버는 메인 서버와 데이터 서버로 구성이 되며, 각각의 서로 다른 물리적인 서버 안에 Docker 플랫폼 기반의 서버가 구성이 된다. 알고리즘 1과 같이 사용자가 멀티미디어 콘텐츠를 제공받기 위해 웹을 이용하여 메인 서버에 접속하면 메인 서버는 자신을 포함한 모든 서버들의 CPU사용률, 메모리 사용률, 네트워크 반응 속도를 종합하여 최적의 서버에 연결을 시켜준다. 사용자는 최초 메인서버의 웹을 접속을 하지만 상황에 따라 최적의 서버로부터 웹과 멀티미디어 콘텐츠를 제공받는다. 또한 각각의 콘텐츠목록은 <그림 4>와 같이 실시간으로 공유를 하며, JSON파일[13] 형식을 이용하여 콘텐츠 목록과 콘텐츠가 존재하는 서버의 주소가 명시되어 있다.

```
{name : video1 title, StreamAdd : 'Server1 Stream Address' }, ··· {name : video2 title, StreamAdd : 'Server2 Stream Address' }
```

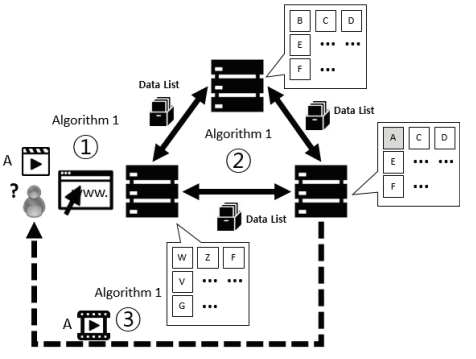

그림 4. 콘텐츠 리스트 구조 및 데이터 전송 방법
Figure 4. Contents list structure and data transmission method

콘텐츠 리스트는 각각의 서버가 서로 공유를 하며, 사용자가 콘텐츠를 요청하였을 때 우선적으로 자신이 보유하고 있는 콘텐츠를 제공하지만 보유하고 있지 않을 경우 다른 서버에 연결[14, 15] 하

여 콘텐츠를 제공한다. 또한, 사용자가 이용하는 서버에 보유하고 있지 않은 콘텐츠를 계속 요청을 하게 되면 해당하는 서버는 다른 서버로부터 사용자가 자주 요청하는 데이터를 분산 전송방식을 이용하여 전송받게 된다.

알고리즘 2는 콘텐츠 사용 빈도수에 따른 콘텐츠 알고리즘으로써 사용자가 최적의 서버에 접속하여 콘텐츠를 요청할 때 해당 서버에 콘텐츠가 없을 경우 다른 서버로부터 연결되어 콘텐츠를 제공받는다. 콘텐츠 사용 빈도가 적을 경우 다른 서버로부터 콘텐츠를 제공받을 수 있지만 최적의 서버로 선택되어 접속하게 된 사용자에게는 최적의 멀티미디어를 제공할 수 있다는 보장이 없다. 따라서 최적의 서버에 접속한 사용자에게 원활한 콘텐츠를 제공하기 위해서 알고리즘 2를 적용한다. 접속한 서버에 콘텐츠가 없을 경우 $C_T[n]$에 사용자가 요청한 콘텐츠 목록과 $C_r[n]$ 요청한 횟수를 카운트 한다. $C_r[n]$의 카운트 수가 설정한 S_c 횟수보다 많을 경우 잦은 콘텐츠 요청으로 판단을 하여 *RequestServer*($C_T[n]$) 함수를 통해 다른 서버로부터 콘텐츠를 제공받는다.

알고리즘 2. 콘텐츠 사용 빈도수에 따른 콘텐츠 요청
Algorithm 2. Request content based on frequency of usage

```
WHEN u Request the content Data THEN
  IF !ContentExistence(u) THEN
    C_T[n] = u.title;
    C_r[n]++;
  END IF
  WHILE C_r[n] != NULL THEN
    IF C_r[n] >= S_c THEN
      RequestServer(C_T[n]); //Content Down
    END IF
    n++;
  END WHILE
END WHEN
```

4. 사용자 맞춤형 멀티미디어 전송을 위한 분할 전송 방법

4.1 네트워크 상황에 따른 콘텐츠 제공

<그림 5>와 같이 사용자가 이동 중에도 원활한 멀티미디어 콘텐츠를 제공받도록 다양한 해상도에 따른 콘텐츠를 인코딩한다. 기본적으로 멀티미디어 콘텐츠를 제공할 때 제공받는 사용자 디바이스의 네트워크 환경이 원활하지 않을 경우 멀티미디어 콘텐츠가 재생 중에 끊기는 현상이 발생한다. 따라서 본 논문에서는 최소한의 속도에서도 멈춤 없이 멀티미디어 서비스를 제공받을 수 있도록 해상도를 최저 144p / 203kbps까지 제공을 한다. 또한 사용자의 네트워크 상황에 따라서 최저 해상도에서 최고 해상도까지 자동으로 해상도가 조정이 되기 때문에 사용자는 끊김 없이 멀티미디어 서비스를 이용할 수 있다. 네트워크 상황에 따른 해상도 자동변화 기술을 적용하기 위해서는 기본적으로 <그림 5>와 같이 하나의 파일을 작은 단위의 파일로 나눠야 한다.

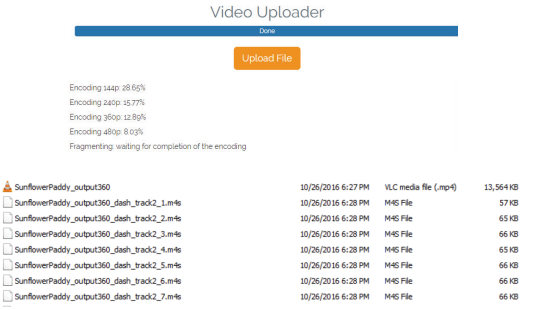

그림 5. 멀티미디어 콘텐츠 해상도에 따른 분할 코딩
Figure 5. Partition coding based on multimedia content resolution

4.2 서버 간 콘텐츠 공유 및 전송 기법

<그림 5>와 같이 멀티미디어 콘텐츠를 분할 코딩하여 저장함으로써 <그림 6>과 같이 서버 간의 분할 전송을 할 수 있다. 이전 설명과 같이 빈도수에 따른 콘텐츠를 제공하기 위해서는 빠른 데이터 전송이 필요하다.

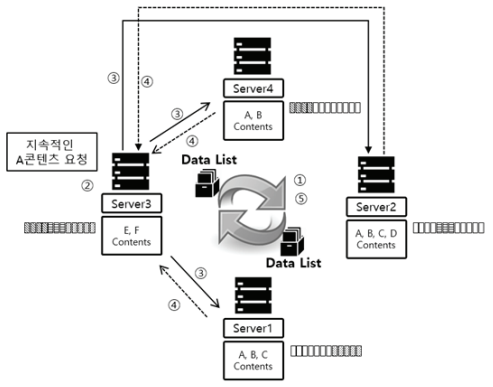

그림 6. 멀티미디어 콘텐츠의 분할전송
Figure 6. Division transmission of multimedia contents

따라서 <그림 6-①> 서버 간 멀티미디어 콘텐츠가 등록, 삭제, 변경에 따라서 콘텐츠 리스트를 실시간으로 공유한다. <그림 6-②> 서버 3의 A콘텐츠를 지속적으로 요청을 할 경우 6-③, 6-④ 각각 서버에 콘텐츠를 요청하고, 서버의 네트워크 속도에 따라서 서버 3에게 전송할 데이터가 비율적으로 할당이 되어 해당하는 콘텐츠를 분할 전송하게 된다.

알고리즘 3. 사용자 기반의 멀티미디어 데이터 분할방법
Algorithm 3. Multimedia Data Division Method based on User

```
WHEN u Server ready for data transmission THEN
  WHILE S_c(i+1) != NULL THEN
    S_f(i+1) = (1 - S_f(0)) * S_b(i+n) / (S_b(i+1) + ... + S_b(i+n))   (1)
    S_c(i+1).send();
  END WHILE
  IF S_c(0) == sort(0) THEN
    S_c(0).send();
 END IF
 IF (Complete S_c(0) data transfer) or
    (Complete Sort(0) data transfer) THEN
    Sort(0).Send();
 END IF
END WHEN
```

알고리즘 3과 같이 u 사용자가 멀티미디어 데이터를 요청하면 콘텐츠를 가지고 있는 서버 $S_c(i)$가 전송 준비를 한다. 전송 준비가 완료되면 첫 번째 콘텐츠를 가지고 있는 빠른 서버 $S_c(0)$와 사용자가 같은 지역 서버 중에서 제일 빠른 서버 $Sort(0)$가 같은 서버인지 체크한다. 만약 같은 서버이면, 서버는 사용자에게 요청한 콘텐츠를 제공하고, 서버가 서로 다르면 콘텐츠를 가지고 있는 서버 중에서 제일 빠른 서버가 데이터 앞부분을 사용자에게 스트리밍 형식으로 서비스를 제공한다. 다음 나머지 서버들은 속도와 크기에 비례해서 사용자에게 전송할 데이터를 비율적으로 분할한다. S_f는 비율에 따라서 나눠지는 파일 크기이며, 1은 전체 파일을 의미한다. $S_f(0)$은 사용자에게 서비스를 먼저 제공해야 되기 때문에 앞부분을 사용자에게 직접 연결하여 제공한다. $S_b(i+n)$은 사용자에게 다이렉트로 전송하는 서버를 제외하고 식 1과 같이 $S_c(i)$에 속하는 서버들의 데이터를 속도 비율 S_b를 기반으로 데이터를 분할한다. 나머지 파일 크기는 1-$S_f(0)$이다. 각각 서버는 할당된 멀티미디어 데이터를 제일 빠른 $Sort(0)$ 서버에 데이터를 전송하며, <그림 6>의 서버 3과 같다. $S_c(0)$ 서버의 멀티미디어 데이터가 전송이 끝나거나 제일 빠른 $Sort(0)$ 서버에 데이터 전송이 완료되면 사용자는 제일 빠른 서버에 연결되어 사용자가 요청한 멀티미디어 콘텐츠를 이어서 제공받는다.

5. 멀티미디어 전송 실험 및 결과

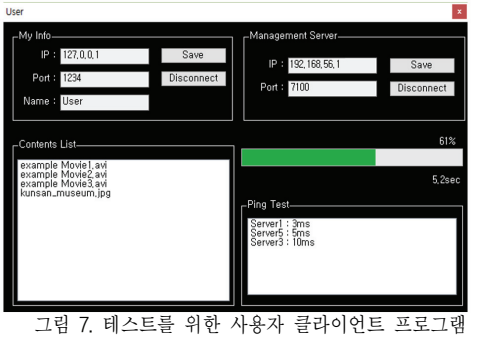

그림 7. 테스트를 위한 사용자 클라이언트 프로그램
Figure 7. User client program for testing

표 1. 테스트를 위한 서버 제한 속도
Table 1. Server speed limit for testing

	서버1	서버2	서버3	서버4	서버5
제공 속도	6Mbps	3Mbps	2Mbps	1.5Mbps	4Mbps

본 논문에서는 개발한 알고리즘을 이용하여 실제 데이터 분할전송 테스트를 진행하였다. 정확한 수치 측정을 위해서 별도의 클라이언트 프로그램을 개발하여 측정을 하였다. <그림 7>은 사용자 입장에서 데이터를 제공받는 프로그램으로 메인 서버와 데이터 서버는 개발한 Docker 기반의 플랫폼을 사용하였으며, 클라이언트 테스트 프로그램은 윈도우 환경 기반의 C#언어를 이용하여 개발하였다.

<표 1>은 각각의 서버에 설정한 제한 속도를 말한다. 예를 들어 전체 파일 크기가 360mb이면, 서버 1의 속도를 이용하여 데이터를 분할 할 경우 약 130mb의 데이터가 할당된다. 따라서 데이터가 할당이 되면 파일이 시작되는 시작점과 끝점의 정보를 서버에게 전송하기 때문에 해당하는 부분의 파일들을 사용자에게 제공받는다.

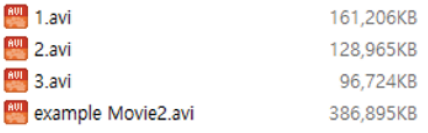

그림 8. 분할된 데이터와 결합된 데이터 파일
Figure 8. Data file combined and divided data

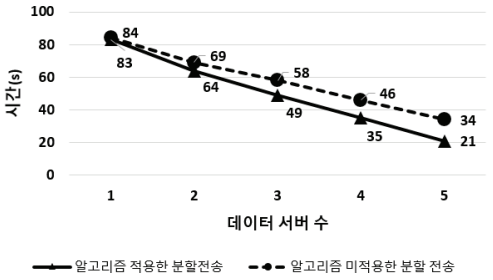

그림 9. 멀티미디어 데이터 분할 전송 테스트 결과
Figure 9. Multimedia data division transmission test result

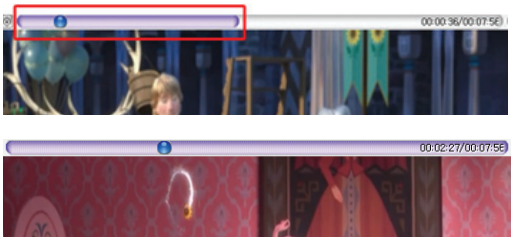

그림 10. 빠른 서버로부터 제공받은 데이터 앞부분(위), 결합된 데이터(아래)
Figure 10. The first part of the data in red rectangle provided by the fast server(Top), Combined data(Bottom)

<그림 8>은 3개의 서버들로부터 분할 전송받은 멀티미디어 데이터 1~3.avi이고, 결합된 멀티미디어 데이터 파일은 example Movie2.avi이다. <그림 9>는 멀티미디어 데이터를 분할 전송한 속도 테스트 결과 그래프이며, 분할 전송하기 때문에 1:1보다 빠른 전송속도를 보여준다. 또한 비례적으로 데이터를 분할하기 때문에 단순 분할하였을 때보다 빠른 것을 확인할 수 있었다. 전송 테스트 결과는 클라이언트 기준으로 진행하였으며, 각각의 서버로부터 분할 전송받아 파일이 생성되기까지의 시간을 측정하였다.

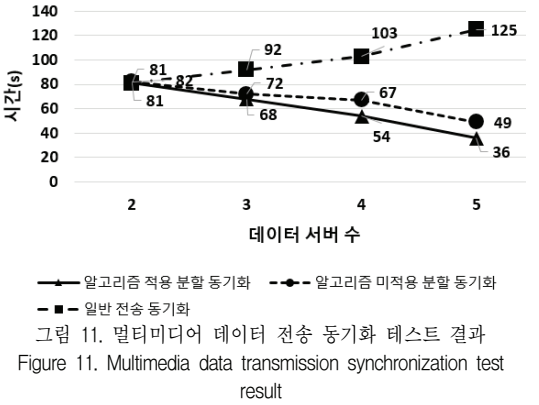

그림 11. 멀티미디어 데이터 전송 동기화 테스트 결과
Figure 11. Multimedia data transmission synchronization test result

<그림 11>은 서버 기준으로 특정 파일 하나를 모든 서버가 동기화하는 시간을 측정한 테스트 결과 그래프이다. 서버가 2대일 경우에는 1:1방식이

기 때문에 속도의 차이는 없었지만 서버의 수가 많아질수록 속도의 차이가 점점 차이가 나는 것을 확인할 수 있다. 서버가 5대일 경우 일반 전송 동기화는 1:4 방식으로 전송하며, 분할 동기화는 모든 서버가 서로 파일을 분할해서 동기화하기 때문에 서버의 수가 점점 많아질수록 동기화 속도가 빨라진다. 또한 알고리즘 적용시 서버가 제공할 수 있는 속도를 기준으로 분할 전송할 파일을 비례적으로 전송하기 때문에 서버가 제공할 수 있는 최대 속도를 사용할 수 있다. 따라서 서버에서 제공받을 수 있는 속도를 효율적으로 사용함으로써 특정 서버에 과부하를 줄일 수 있다.

6. 결 론

본 논문은 최적의 멀티미디어 전송을 위한 사용자 맞춤형 분할전송 방법을 제안하였으며 실험결과를 통해 분할 받는 서버가 많은 수록 제공하는 콘텐츠의 속도가 빨리지는 것을 확인 할 수 있었다. 테스트 결과를 통해 서버의 자원을 최대한 활용할 수 있기 때문에 특정 서버에 과부하 및 사용률을 줄일 수 있었다. 본 논문에서는 빠른 확장성을 위해 Docker 플랫폼을 이용하여 개발을 진행하였으며, 개발한 내용은 VM 및 OpenStack에서도 적용하여 사용할 수 있다. 속도 테스트를 위하여 서버는 리눅스 기반의 Docker 플랫폼을 사용하고 제공받는 클라이언트는 C#기반의 테스트 프로그램을 사용하였다. 향후 동적 VM환경에서 데이터 분할 전송뿐만 아니라 자원을 효율적으로 사용할 수 있는 연구를 지속적으로 진행할 예정이다.

References

[1] K. Konstanteli, T. Cucinotta, K. Psychas, and T. A. Varvarigou, *Elastic admission control for federated cloud services,* Cloud Computing, Transactions on IEEE, Vol. 2, No. 3, pp. 348-361, 2014.

[2] H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qiu, C. Lin, H. Zhang, and B. Li, *Livesky: Enhancing cdn with p2p*, ACM Transactions on Multimedia Computing, Communications, and Applications, Vol. 6, No. 3 pp. 16-26, 2010.

[3] H. Shen, Z. Li, Y. Lin, and J. Li, *SocialTube: P2P-assisted video sharing in online social networks*, NFOCOM, 2012 Proceedings IEEE, pp. 2886-2890, 2014.

[4] N. Carlssona, D. Eager, A. Gopinathana, and Z. Li, *Caching and optimized request routing in cloud-based content delivery systems,* Performance Evaluation, Vol. 79, pp. 38-55, 2014.

[5] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, and A. Wolman, *Volley: Automated data placement for geo-distributed cloud services*, Networked Systems Design and Implementation, pp. 17-32, 2010.

[6] C. C. Jayasundara, and M. Zukerman, *Improving scalability of VoD systems by optimal exploitation of storage and multicast*, Circuits and Systems for Video Technology, Transactions on IEEE, Vol. 24, No. 3, 2014.

[7] L. Di, and L. Zhao. *The research and implementation of cloud computing platform based on docker,* Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 2014 11th International Computer Conference on. IEEE, pp. 475-478, 2014.

[8] Preeth E N, Fr. Jaison P. Mulerickal, B.

Paul, and Y. Sastri, *Evaluation of docker containers based on hardware utilization,* In: Control Communication & Computing India (ICCC), 2015 International Conference on. IEEE, pp. 697-700, 2015.

[9] M. Miglierina, *Application deployment and management in the cloud,* Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2014 16th International Symposium on. IEEE, 2014.

[10] A. Calinciuc, C. C. Spoiala, C. O. Turcu, and C. Filote, *Openstack and docker: building a high-performance iaas platform for interactive social media applications,* Development and Application Systems (DAS), 2016 International Conference on. IEEE. pp. 287-290, 2016.

[11] M. Madsen, F. Tip, and O. Lhoták. *Static analysis of event-driven node. js JavaScript applications,* ACM SIGPLAN Notices. Vol. 50, No. 10, pp. 505-519, 2015.

[12] I. K. Chaniotis, K-I. D. Kyriakou, and N. D. Tselikas. *Is node. js a viable option for building modern web applications? A performance evaluation study,* Computing, Vol. 97, No. 10, pp. 1023-1044, 2015.

[13] Z. H. Liu, B. Hammerschmidt, and D. McMahon. *JSON data management: supporting schema-less development in RDBMS,* Proceedings of the 2014 ACM SIGMOD international conference on Management of data. ACM, pp. 1247-1258, 2014.

[14] Y-D. Lin, T-B. Shih, Y-S. Wu, and Y-C. Lai, *Secure and transparent network traffic replay, redirect, and relay in a dynamic malware analysis environment,* Security and Communication Networks, Vol. 7, No. 3, pp. 626-640, 2014.

[15] P. Cao, E. C. Badger, Z. T. Kalbarczyk, R. K. Iyer, A. Withers, and A. J. Slagell, *Towards an unified security testbed and security analytics framework,* Proceedings of the 2015 Symposium and Bootcamp on the Science of Security. ACM, No. 24, pp. 1-2, 2015.

도커 기반에 가상머신을 이용한 사용자 네트워크의 최적 멀티미디어 데이터 분할 전송 방법

박상현[1], 류호영[2], 김진술[1]

[1] *전남대학교 전자컴퓨터공학부*

[2] *한국전자통신연구원*

요 약

기존 멀티미디어 전송 방식은 실시간으로 네트워크 상황을 고려하지 않기 때문에 멀티미디어 서비스를 제공받는데 있어 잦은 멈춤 현상이 발생한다. 따라서 본 논문에서는 실시간 네트워크 상황을 고려하지 않은 기존의 멀티미디어 전송방식을 상황에 따른 가상머신 기반의 멀티미디어 전송방식으로 사용자가 제공받는 디바이스와 네트워크 환경을 고려한 맞춤형 분할전송 방법을 제안한다. 또한, 실시간으로 변화하는 네트워크 환경에 맞춰 가상머신에 따른 네트워크 환경 구성과 최적의 멀티미디어 전송을 위한 분할 전송 기법을 설명한다. 가상머신은 전체 서버를 관리하는 메인 서버와 멀티미디어 서비스를 제공하는 데이터 서버로 구성된다. 또한, 실시간으로 변화하는 네트워크 환경에 대처하기 위하여 Docker를 이용하여 구성하였으며, 생성된 다수의 데이터 서버는 메인 서버가 실시간으로 관리를 한다. 사용자가 멀티미디어 콘텐츠를 요청하면 메인서버는 사용자의 네트워크 환경과 데이터 서버 환경에 따라 리다이렉트 알고리즘을 통해 최적의 서버를 선택한다. 데이터를 제공하는 서버는 멀티미디어 분할전송기법을 통해 사용자의 네트워크 상황에 따라 적합한 멀티미디어 데이터를 실시간으로 변화하며 제공을 한다. 본 논문에서는 1:1방식의

멀티미디어 전송방식과 서버 네트워크 환경에 따른 데이터 전송방식의 전송속도를 비교 분석한다. 또한 비교 분석을 통해 기존의 방식보다 분할 전송 방식이 빠르다는 것을 실험을 통해 증명 하였다.

감사의 글

본 연구는 미래창조과학부 및 정보통신산업진흥원의 정보통신연구기반구축사업의 일환으로 수행하였음. [12221-14-1001, 차세대 네트워크·컴퓨팅 플랫폼연구 기반구축] 또한, 미래창조과학부 및 정보통신기술진흥센터의 대학ICT연구센터육성 지원사업의 연구결과로 수행되었음. (IITP-2017-2016-0-00314)

Sanghyun Park received his B.S. Degree in Computer and Information from the University of Korea Nazarene in 2010, and the M.S. degree in School of Electronics and Computer Engineering, Chonnam National University, South Korea. He worked as an engineer in System Development Team of Media Flow Company from 2010 to 2012. He is now studying Ph.D. Degree in School of Electronics and Computer Engineering, Chonnam National University. His research interests are Interactive Media, Systems Development, Embedded systems, Digital Media and Cloud computing.

E-mail address: sanghyun079@gmail.com

Ho-Yong Ryu received the B.S., M.S., and Ph.D. degrees in electronic communication engineering from the Kwangwoon University, Seoul, Korea. In 1993, 1995 and 1999 respectively. From January 1999, he is Principal Member of Researcher with eth Electronics and

Telecommunications Research Institute, he is engaged in research on the project manager of Network Software Platform Research Section. His research interests include Internet and routing mobile IP and security.

E-mail address: hyryu@etri.re.kr

Jinsul Kim received the B.S. Degree in computer science from University of Utah, Salt Lake City, Utah, USA, in 2001, and the M.S. and Ph.D degrees in digital media engineering, department of information and communications from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2005 and 2008. He worked as a researcher in IPTV Infrastructure Technology Research Laboratory, Broadcasting/ Telecommunications Convergence Research Division, Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea from 2005 to 2008. He worked as a professor in Korea Nazarene University, Chon-an, Korea from 2009 to 2011. Currently, he is a professor in Chonnam National University, Gwangju, Korea. He has been invited reviewer for IEEE Trans. Multimedia since 2008. He has been invited for TPC(Technical Program Committee), IWITMA2009/2010, and PC(Program Chair), ICCCT2011 His research interests include QoS/QoE, Measurement/ Management, IPTV, Mobile IPTV, Smart TV, Multimedia Communication and Digital Media Arts.

E-mail address: jsworld@jnu.ac.kr