creative commons
COMMONS DEED

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

공학 석사학위 논문

콘텐츠 중심 네트워킹에서의 그룹 기반 분산 협력 캐싱 방법

Group-based Distributed Cooperative Caching in Content-Centric Networking

아 주 대 학 교 대 학 원

컴 퓨 터 공 학 과

김 성 민

콘텐츠 중심 네트워킹에서의 그룹 기반 분산 협력 캐싱 방법

지도교수 노 병 희

이 논문을 공학석사 학위 논문으로 제출함

2014년 2월

아 주 대 학 교 대 학 원

컴 퓨 터 공 학 과

김 성 민

김성민의 공학 석사학위 논문을 인준함.

심사위원장 노 병 희

심 사 위 원 고 영 배 (인)

심 사 위 원 최 영 준

아 주 대 학 교 대 학 원

2013년 12월 13일

감 사 의 글

적지 않은 나이에 멀티미디어통신 연구실에 소속되어 생활한지 2년이란 시간이 지났습니다. 새로운 시각과 전문성을 얻고 배우는 계기가 되었고, 무엇보다도 사람들과의 인연이라는 가장 큰 자산을 일깨워준 소중한 시간이었습니다. 이러한 기회를 제공해주시고, 항상 아낌없는 조언과 지도를 해 주신 노병희 지도교수님께 먼저 감사의 말씀을 올립니다. 사회에 나가서도 교수님의 가르침을 항상 마음에 새기며 생활하겠습니다. 아울러 바쁘신 와중에도 논문 심사를 통해 아낌없는 조언을 해 주신 고영배 교수님, 최영준 교수님께도 감사 드립니다.

연구 분야의 큰 그림을 잡고 랩장으로써 연구실의 살림꾼 역할을 마다하지 않은 동기 광수, 연구 분야에서의 세밀함을 더하는데 도움을 준 북고 후배 보성이에게도 고맙다는 말을 전합니다. 듬직한 은호 형, 언제나 열정적인 승오, 꾸준히 노력하는 세훈. 우리 동기들 밖에 나가서도 건승하길 기원합니다. 연구실의 주축이 될 병성, 승엽, 송현, 규민에게도 고맙다는 말을 전합니다. 개성 강하고 열정적인 울 후배들, 원하는 결실을 얻길 기원합니다. Also thanks to Zerabruk, who give me the variety of viewpoint.

2년이라는 시간 동안 물심양면으로 지원과 조언을 아끼지 않으셨던 부모님, 동생 소아에게도 고맙고 사랑한다는 말을 전합니다. 그리고 힘든 시간을 이겨낼 수 있게 도와준 친구들에게도 고맙다는 말을 전하고 싶습니다.

끝으로 저와 인연이란 이름의 끈으로 연결되어 형성된 네트워크에 속한 많은 지인 및 미처 언급하지 못한 많은 분들께도 감사의 말씀을 올립니다.

요 약

오늘날 인터넷 트래픽의 양은 급격히 증가하고 있으며, 그 중심에 인터넷 비디오 트래픽으로 대변되는 콘텐츠 전송이 있다. 이러한 환경에서 트래픽의 양을 줄이기 위한 대안으로 미래 인터넷 기술 중 하나인 콘텐츠 중심 네트워킹 방식이 소개되었다. 하지만 콘텐츠 중심 네트워킹에서 사용되는 최저 사용 빈도 캐시 교체 알고리즘과 콘텐츠가 전달되는 모든 경로에 캐시가 되는 기존 방식은 한정적인 용량을 가진 라우터 캐시 환경에서 캐시 교체 작업을 빈번하게 일으켜 캐시 적중률을 저하시켜 대역폭 소비를 효과적으로 줄이지 못하는 단점을 갖고 있다. 본 논문에서는 이러한 단점을 극복하고자 콘텐츠 서버로 향하는 경로를 기준으로 그룹 단위로 라우터를 묶은 뒤, 콘텐츠를 분산 배치하고 관리하는 그룹 기반 협력 캐싱 방식을 제안한다. 이를 통해 네트워크에서 콘텐츠의 캐시 중복 저장 빈도를 줄이고, 더 많은 종류의 콘텐츠를 저장 가능하게 함으로써 캐시 적중률 향상을 비롯한 네트워크 성능 향상을 기대할 수 있다. 우선 기존 콘텐츠 중심 네트워킹 방식과 제안된 방법 간 캐시 적중률 및 평균 지연 시간, 평균 홉 수, 전체 네트워크 로드를 비교함으로써 제안한 기술의 성능 향상으로 검증하였다. 또한 다양한 종류의 콘텐츠가 제공되는 네트워크 환경에서 캐시 적중률 변화를 통한 성능 측정을 통해 제안된 기술의 우수성을 확인하였다.

목 차

1. 서 론 1

2. 연구 배경 및 관련 연구 4

2.1. 콘텐츠 중심 네트워킹 개요 4

2.2. 캐싱 개요 8

3. 그룹 기반 협력 캐싱 11

3.1. 시스템 모델 11

3.2. 그룹 구성 메커니즘 13

3.3. 콘텐츠 요청 및 전송 17

3.4. 그룹 내 콘텐츠 관리 24

4. 성능 평가 26

4.1. 시뮬레이션 환경 26

4.2. 시뮬레이션 결과 30

5. 결 론 43

참고문헌 44

그 림 목 차

그림 1. CCN 패킷 종류 및 구조 [2] 4
그림 2. CCN 이름 구조 5
그림 3. CCN 포워딩 엔진의 동작 메커니즘 7
그림 4. 그룹 구성 시 사용된 변수들 13
그림 5. 헤드 라우터 선발 알고리즘 14
그림 6. 그룹 구성 예시 16
그림 7. CCN 포워딩 엔진에서 추가적인 자료 구조 17
그림 8. 그룹 내 같은 콘텐츠 요청 및 전달 예시 19
그림 9. CCN 포워딩 엔진에서 같은 그룹 내 콘텐츠 요청 처리 과정 (Node ID: 1) 21
그림 10. 그룹 내 Data 패킷 처리 과정 알고리즘 23
그림 11. 캐시 교체나 타임아웃으로 인한 그룹 내 라우터 간 동기화 과정 25
그림 12. 시뮬레이션 토폴로지 26
그림 13. 캐시 크기 변화에 따른 캐시 적중률 변화 32
그림 14. 캐시 크기 변화에 따른 평균 지연 시간 변화 32
그림 15. 캐시 크기 변화에 따른 평균 홉 수 변화 33
그림 16. 캐시 크기 변화에 따른 전체 네트워크 로드 변화 33
그림 17. 카테고리 당 콘텐츠가 라우터에 있을 확률 (Cache=2%, α=0.7) 35
그림 18. 카테고리 당 콘텐츠가 라우터에 있을 확률 (Cache=20%, α=0.7) 35
그림 19. 카테고리 당 평균 홉 수 (Cache=2%, α=0.7) 36
그림 20. 카테고리 당 평균 홉 수 (Cache=20%, α=0.7) 36
그림 21. 콘텐츠 중요도에 따른 캐시 적중률 변화 38
그림 22. 캐시 크기 변화에 따른 콘트롤 패킷 오버헤드 변화 38
그림 23. α 값 변화에 따른 캐시 적중률 변화 40
그림 24. α 값 변화에 따른 콘트롤 패킷 오버헤드 변화 40
그림 25. 카테고리 당 콘텐츠가 라우터에 있을 확률 (Cache=2%, α=1.6) 42
그림 26. 카테고리 당 평균 홉 수 (Cache=2%, α=1.6) 42

표 목 차

표 1. 시뮬레이션 파라미터들 27

사 용 기 호

ABC	Age-Based Cooperative Caching
Betw	Betweenness-Centrality
CCN	Content-Centric Networking
CDF	Cumulative Distribution Function
CDN	Content Distribution Network
CS	Content Store
ER	Edge Router
FIB	Forwarding Information Base
FIFO	First-In First-Out
GT	Grout Table
HR	Head Router
ICN	Information-Centric Networking
LFU	Least Frequently Used
LRU	Least Recently Used
MIP	Mixed Integer Program
NDN	Named Data Networking
OSPF	Open Shortest Path First
P2P	Peer-to-Peer
PIT	Pending Interest Table
PSM	Priority Storage Management
WFSM	Weighted Fair Storage Management

1. 서 론

오늘날 인터넷 트래픽의 양은 급격히 증가하고 있으며, 2017년에는 한 달에 120 엑사바이트 (Exabytes, 10^{18} bytes)를 넘어설 것으로 예상되는데, 이는 2012년에 비해 2.8배 늘어난 수치이다 [1]. 특히 인터넷 비디오 트래픽은 5년간 29% 증가하여 전체 인터넷 트래픽 중 44%를 차지할 것으로 예상된다. 이러한 예상을 바탕으로 비추어 보았을 때, 인터넷 비디오 트래픽의 비중과 양은 꾸준히 늘어날 것임을 알 수 있다.

하지만 기존의 인터넷 환경에서는 앞서 언급한 트래픽의 급격한 증가를 대처할 수는 있으나, 근본적으로 줄이는 데에는 한계가 있다. 이러한 문제점을 해결하기 위해 콘텐츠 중심 네트워킹 (Content-Centric Networking, CCN) [2] 개념이 소개되었다. 콘텐츠 중심 네트워킹은 IP 주소가 아닌 콘텐츠 이름을 기반으로 하며, 콘텐츠가 저장된 위치보다 요청하는 콘텐츠 자체에 초점을 맞추는 현재 인터넷 패러다임의 변화를 반영하고 있다. 정보 기반 네트워크 (Information- Centric Networking, ICN)라고도 하며, 현재는 이름 기반 데이터 네트워킹 프로젝트 (Named Data Networking, NDN)에 속해있다 [3].

CCN의 핵심은 라우터 내 캐시이다. 네트워크 내 모든 라우터가 캐시를 갖고 있고, 콘텐츠 서버에서 요청된 콘텐츠가 사용자에게로 전달되는 경로에 있는 모든 라우터 캐시에 저장된다. 이러한 방식은 향후 다른 사용자가 같은 콘텐츠를 요청하는 경우, 경로 내에 있는 라우터 캐시에 요청한 콘텐츠가 저장되어 있으면 바로 보낼 수 있다는 장점을 갖는다. 그러나 라우터 내 캐시는 한정된 용량을 갖기 때문에, 캐시가 모두 차 있

는 경우 새로 들어온 콘텐츠를 저장하기 위해 여유 공간을 확보해야 하는 작업이 필요하다.

기존 CCN에서는 이를 위해 최저 사용 빈도 (LRU) 교체 알고리즘을 사용하고 있다 [2][3]. 이 방식은 라우터 내 캐시에 여유 공간이 없어 캐시 교체가 일어나는 경우, 저장되어 있는 콘텐츠 중 사용 빈도가 가장 낮은 콘텐츠를 찾아 새로 들어온 콘텐츠와 교체하는 방식이다. 라우터 내 캐시에 여유 공간이 있는 경우 콘텐츠가 전달되는 모든 경로에 저장되어 네트워크 내 캐시 효율을 높일 수 있으나, 캐시에 여유 공간이 없어 교체가 일어나는 경우 콘텐츠 중심 네트워크에서 캐시 저장 방법의 특성상 캐시 교체가 더 빈번하게 발생하게 된다. 특히 캐시 크기가 작을수록 보관할 수 있는 콘텐츠의 양이 상대적으로 줄어들어 더 많은 교체가 발생하게 되어 캐시 적중률 저하에 따른 콘텐츠가 요청된 클라이언트까지 도달하는 평균 지연 시간과 콘텐츠가 전달되는 홉 수가 증가하는 등 전체적인 네트워크 성능이 떨어지게 되는 단점을 가지고 있다.

본 논문에서는 콘텐츠가 전달되는 모든 경로에 저장되어 캐시 교체 빈도수를 늘리고 캐시 적중률 저하 등의 비효율성을 개선하고자 콘텐츠 서버로 향하는 경로를 기준으로 그룹 단위로 라우터를 묶은 뒤, 콘텐츠를 분산 배치하고 관리하는 그룹 기반 협력 캐싱 방법을 제안한다. 이를 통해 네트워크에서 콘텐츠의 캐시 중복 저장 빈도를 줄이고, 더 많은 종류의 콘텐츠를 저장 가능하게 함으로써 캐시 적중률 향상을 비롯한 네트워크 성능 향상을 기대할 수 있다. 이를 위해 우선 기존 CCN 방식과 제안된 그룹 기반 방법 간 캐시 적중률 및 평균 지연 시간, 평균 홉 수, 전체 네트워크 로드를 비교함으로써 제안한 기술의 성능 향상으로 검증하

였다. 또한 다양한 종류의 콘텐츠가 제공되는 네트워크 환경에서 캐시 적중률 변화를 통한 성능 측정을 통해 제안된 기술의 우수성을 확인하였다.

본 논문의 구성은 다음과 같다. 2장에서는 현재 연구되고 있는 캐시 교체 알고리즘 및 네트워크 내부의 콘텐츠 캐시 배치 방식에 대해 살펴본다. 3장에서는 그룹 기반 협력 캐싱과 동작 방식에 대해 알아본다. 4장에서는 실험 환경과 성능 분석이 이루어지고, 전체적인 결론을 5장에 기술한다.

2. 연구 배경 및 관련 연구

2.1. 콘텐츠 중심 네트워킹 개요

콘텐츠 중심 네트워킹 (CCN)은 콘텐츠가 위치한 주소가 아닌 콘텐츠 이름 자체를 사용하여 콘텐츠를 요청하고 이를 얻는 방식을 사용하며, 콘텐츠 이름에 초점을 맞추는 현재 인터넷 패러다임을 반영하고 있다.

CCN에서는 콘텐츠를 요청하기 위한 Interest 패킷과 실제 콘텐츠가 담겨 있는 Data 패킷을 사용하며, 그림 1에 두 가지 종류의 패킷 구조가 나타나 있다. 먼저 (a)의 Interest 패킷에서는 콘텐츠 이름 (content name)과 요청한 콘텐츠의 속성을 갖는 Selector, Interest 패킷의 중복을 방지하기 위한 무작위 값인 Nonce로 구성되어 있다. (b)의 Data 패킷에서는 콘텐츠 이름과 패킷과 콘텐츠의 보안을 위한 Signature와 Signed Info, 실제 데이터가 들어있는 Data 필드로 구성되어 있다.

(a) Interest 패킷 (b) Data 패킷

그림 1. CCN 패킷 종류 및 구조 [2]

Fig. 1. CCN packet type and structure [2]

CCN에서 콘텐츠를 요청하고 전달할 때 사용되는 콘텐츠 이름은 그림 2와 같이 계층적 구조를 가지고 있다. 가장 첫 부분에 나타나는 이름은 글로벌 라우팅 이름 (globally-routable name)으로써, 콘텐츠를 제공하는 모든 사용자에게 알려져 있는 서버의 고유한 값이며 향후 라우팅 과정에서 사용된다. 두 번째 부분은 글로벌 라우팅 이름의 하위 부분 경로와 콘텐츠 이름이 담긴 지역 이름 (organizational name)이고, 현재 콘텐츠의 버전 (versioning)과 파일 조각 번호 (segmentation)을 나타내는 부분이 뒤에 나타나 있다. 콘텐츠 이름의 각 부분은 '/' 구분자를 사용해 나누어지며, 버전과 파일 조각 번호 부분은 각각 v, s 문자가 구분자 뒤에 나타나 구분이 가능하게끔 구성되어 있다.

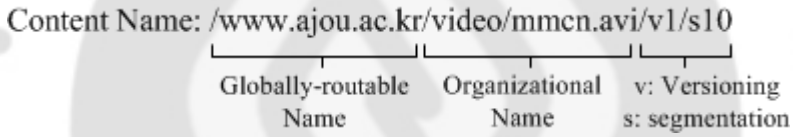


그림 2. CCN 이름 구조

Fig. 2. Structure of CCN name

CCN에서는 현재 사용 중인 IP 기반 네트워크와는 달리 패킷 입출력 인터페이스를 Face로 설정하여 관리하고 있다. 이 때 입출력 인터페이스는 자신과 연결되어 있는 다른 노드뿐만 아니라 로컬 영역 내부까지 커버한다. 또한 Interest, Data 패킷을 처리하기 위해 콘텐츠 중심 네트워킹에서는 세 가지 요소로 이루어진 포워딩 엔진을 갖고 있다. 첫 번째는 Content Store (CS)이며 Data 패킷을 저장하는 캐시 역할을 수행한다. 두 번째는 Pending Interest Table (PIT)이며 Interest 패킷이 다른 노드로 전달됐을 때 해당 Interest 패킷에 담긴 콘텐츠 이름과 패킷이 들어온 Face 번호를 저장한다. 만약 다른 노드에서 같은 콘텐츠 이름을 가진 Interest

패킷이 들어온다면 PIT에 들어온 Face 번호만 추가함으로써 불필요한 패킷 발생을 막고 Multicast 역할을 수행한다. 마지막으로는 Forwarding Information Base (FIB)이며 Interest 패킷이 도착했을 때 해당 콘텐츠가 어디에 위치해 있는지 알려주는 역할을 한다. IP 네트워크에서 라우팅 테이블과 비슷한 역할을 수행하나 다른 점이 있다면 IP 주소 대신 Face 번호를 알려준다는 점이다.

그림 3은 CCN에서 Interest 패킷과 Data 패킷의 처리 과정을 보여주는 그림이다. Interest 패킷이 도착하면 CCN 포워딩 엔진에서는 먼저 CS를 체크해 요청된 콘텐츠가 캐시에 있는지 확인한다. 만약 해당 콘텐츠가 존재한다면 패킷이 들어온 Face로 Data 패킷을 바로 전송한다. 만약 존재하지 않는다면 PIT로 넘어가 해당 콘텐츠 이름이 있는지 확인한다. 만약 존재한다면 이전에 다른 노드가 같은 콘텐츠를 요청한 상태이므로 Face 번호를 PIT에 추가하고 Interest 패킷을 폐기한다. 만약 존재하지 않는다면, FIB로 넘어가 콘텐츠가 존재하는 다른 경로를 찾아 해당 Face로 Interest 패킷을 전달하고 이를 PIT에 저장한다. Interest 패킷에 대한 응답으로 Data 패킷이 도착했을 때, CCN 포워딩 엔진에서는 PIT를 확인하여 전달할 Face 번호를 얻는다. 만약 PIT에 리스트가 존재하지 않는다면 요청되지 않은 패킷이라 판단하여 폐기한다. 이후 PIT에 해당 테이블을 지운 뒤 CS에 해당 콘텐츠를 저장하고 패킷을 전송한다.

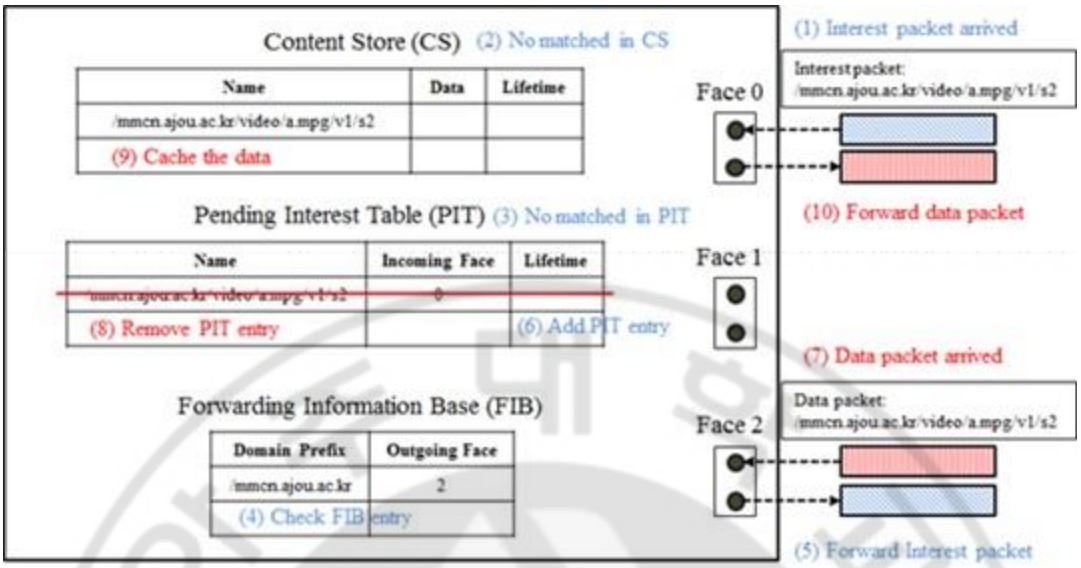


그림 3. CCN 포워딩 엔진의 동작 메커니즘

Fig. 3. Basic mechanism in CCN forwarding engine

2.2. 캐싱 개요

네트워크 분야에서 캐싱은 대역폭 소비, 서버 로드, 평균 지연 시간을 줄이기 위해 고안된 방식으로써 여러 분야에서 사용되고 있다. World-Wide Web 분야에서의 웹 캐싱 (Web Caching) [4]은 IP 계층 위에 오버레이 형식을 취하는 계층적 캐시 구조를 갖는 프록시 (proxy)를 두어 웹 콘텐츠를 캐싱하여 네트워크 성능을 높이는 방법을 취하고 있다. Peer-to-peer (P2P) 네트워크에서의 피어도 IP 계층 위에 오버레이 형식을 취하며, 다른 피어에서 받은 콘텐츠 조각들을 저장하고 있다가 해당 콘텐츠 조각을 요청한 다른 피어들에게 서비스하여 트래픽을 분산시키고 대역폭 소비를 줄이는 역할을 수행한다 [5]. 콘텐츠 전송 네트워크 (Content Distribution Network, CDN)은 네트워크에 콘텐츠를 보관하고 있는 Replica 서버를 하나 이상 두어 서비스함으로써, 네트워크 트래픽을 분산시키고, 전체적인 성능을 향상시키는 역할을 한다. 하지만 CDN 캐시도 계층적 Web Caching과 마찬가지로 캐시의 직접적인 사용이 불가능하다는 단점을 갖고 있다 [6].

CCN에서의 캐싱 분야를 살펴보면, 기본적으로 최저 사용 빈도 (LRU) 알고리즘을 기본적으로 사용하고 있으며, 또는 최소 사용 빈도수 (LFU)나 FIFO 알고리즘도 적용 가능하게 되어 있다. 하지만 콘텐츠의 중요도를 고려하지 않고, 빈도수나 사용 시간만을 고려하고 있다는 단점이 있다. 이를 보완하기 위해 [7]에서는 콘텐츠의 중요도를 기준으로 하는 Priority Storage Management (PSM) 캐시 교체 알고리즘을 제안하였다. 캐시 내 콘텐츠 중 중요도가 가장 낮은 콘텐츠와 새로 들어온 콘텐츠의 중요도를 비교해 더 높은 중요도를 갖는 콘텐츠를 캐시에 저장하는 방식이다.

하지만 이 방식은 중요도가 높은 콘텐츠들이 캐시를 독식하게 되어 좀 더 낮은 비중을 갖는 콘텐츠가 캐시에 저장되는 기회를 없애는 단점이 있다. 이를 보완하기 위해 중요도에 따라 최소한의 캐시 공간을 보장하는 Weighted Fair Storage Management (WFSM) 방식을 제안하였다. 하지만 캐시 교체 알고리즘을 개선하는 방식은 콘텐츠가 전달되는 모든 경로에 저장하여 불필요한 캐시 교체를 일으킨다는 단점을 근본적으로 해결하지 못하고 있다.

이러한 비효율성을 개선하기 위해 네트워크 구조를 고려하여 네트워크 내에서 콘텐츠의 저장 위치를 고려하는 협력 캐싱 연구도 함께 진행되고 있다. [8]에서는 콘텐츠를 요청하는 사용자와 가까이 있는 단말 라우터에 중요도가 높은 콘텐츠가 저장되는 경우 캐시 적중률을 높일 수 있다는 점을 찾고, 이를 구현하기 위해 콘텐츠의 중요도와 홉 수를 반영한 Age-based 협력 캐싱 방식 (ABC)을 제안하였다. 이 방식을 사용하여 사용자에 가까운 라우터 캐시에서 중요도가 높은 콘텐츠를 좀 더 오래 보관할 수 있게 하여 전체적인 네트워크 성능을 꾀하였다. [9]에서는 콘텐츠가 전달되는 모든 라우터에 저장되는 단점을 설명하고, 이를 해결하기 위해 네트워크 내에서 경로가 가장 많이 몰려 있는 곳에 콘텐츠를 저장하는 Betweenness Centrality (Betw) 방식을 제안하였다. [10]에서는 Mixed Integer Program (MIP) 모델을 사용하여 콘텐츠가 전달되는 경로 중 캐시 용량과 링크 값을 고려하여 가장 최적인 위치에 있는 라우터에만 캐시하는 single-path caching과 이를 네트워크 전체로 확장한 network-wide caching 방식을 제안하였다. 이 방식이 [9]에서 제안한 방식보다 캐시 적중률이 더 높다는 것을 확인하였으나, 계산양의 증가로 인해 콘텐츠를 묶음 형태로 전달하는 방식도 함께 제안되었다. [11]에서는 해당 콘텐츠의 조

각들을 Round-Robin 알고리즘을 사용하여 인접한 라우터에 저장한 뒤 이를 서비스하여 실시간 방송에서 타임시프트 기능을 지원하고 버퍼링 시간을 단축시킨 방식을 제안하였다. [12]에서는 Filtering Effect를 이용해 중간 라우터와 단말 라우터의 특성에 맞게끔 콘텐츠 교체 방법을 차등화하여 콘텐츠를 차등적으로 저장하고 교체하는 Progressive Caching 기법을 소개하였다.

3. 그룹 기반 협력 캐싱

본 절에서는 본 논문에서 제안하는 그룹 기반 협력 캐싱을 위한 그룹 구성 방법 및 Interest, Data 패킷이 그룹 내부에 도착했을 때의 처리 방법에 대해 기술하고 있다. 또한 그룹 내 캐싱되어 있는 콘텐츠를 관리하는 방법에 대해서도 설명하고 있다. 그룹 기반 협력 캐싱을 위한 시스템 모델은 3.1에, 그룹 구성 메커니즘은 3.2에, 콘텐츠 요청 및 전달, 저장은 3.3에, 라우터 그룹 내 콘텐츠 관리 기법은 3.4에 기술하고 있다.

3.1. 시스템 모델

본 논문에서 제안하는 그룹 기반 협력 캐싱은 기존의 CCN 구조를 따른다. 따라서 클라이언트에서 콘텐츠 서버로 향하는 라우팅 경로는 도메인(domain) 단위로 이루어지며, OSPF를 라우팅 알고리즘으로 사용하여 경로가 미리 설정되어 있다고 가정한다. 또한 그룹 형성 및 관리, 그룹 내 콘텐츠 요청 및 배포 등의 부가적인 작업을 위해 라우터 및 패킷 종류가 추가되었다. 먼저 추가된 라우터 종류는 다음과 같다.

- 헤드 라우터 (Head Router, HR): 그룹 내부와 외부를 연결해주는 라우터이다. 오직 그룹 내부에서 요청한 콘텐츠들만 캐시에 저장한다는 특징을 갖고 있다.

- 단말 라우터 (Edge Router, ER): 클라이언트와 연결되어 있는 라우터이다.

추가된 패킷 종류는 다음과 같다.

- *join* Interest 패킷: 그룹 구성을 위해 하위 라우터에서 상위 라우터로 보내는 패킷이다. 하위 라우터에서 받은 *join* Interest 패킷 수와 하위 라우터의 노드 번호가 들어 있다.

- *join* Data 패킷: 그룹 구성이 이루어진 뒤, 헤드 라우터에서부터 하위 라우터로 보내는 패킷이다. *join* Interest 패킷의 응답 형태로 보내어지며, 이 패킷을 받은 라우터는 자신이 가입된 그룹과 주변 노드를 알 수 있다.

- *expire* Interest 패킷: 그룹 내 라우터에서 캐시 교체나 타임아웃으로 인해 캐시에서 사라지는 경우, 이 사실을 상위 라우터로 알리는 데 사용되는 제어 패킷이다. 이 패킷을 먼저 받은 CS에서는 캐시 정보를 업데이트하고 PIT에 저장하지 않고 바로 상위 라우터로 전달한다. 이를 통해 Data 패킷을 보내지 않게 한다.

그룹 구성의 경우 도메인 단위로 구성되며, 하나의 AS 내부에 있는 라우터들 사이에서만 가능하다고 가정한다. 이는 그룹 관리 패킷이 AS 외부로 주기적으로 전송되는 경우 발생하는 외부 링크 사용 비용이나 속도 측면의 비효율성을 줄이기 위함이다 [13]. 또한 그룹 구성은 단말 라우터에서 *join* Interest 패킷을 상위 라우터로 전송하는 단계부터 시작된다. 이에 대한 자세한 설명은 3.2에 기술되어 있다.

3.2. 그룹 구성 메커니즘

AS 내부에는 그룹 관리를 위한 헤드 라우터가 존재한다. 하나의 AS에는 여러 개의 그룹이 생성될 수 있기 때문에 하나 이상의 헤드 라우터가 생성될 수 있다. 또한 그룹 구성은 콘텐츠 서버 수에 따라 달라질 수 있다. 콘텐츠 서버가 네트워크에 하나만 존재하는 경우, AS 내부에 위치한 각각의 라우터들은 오직 하나의 그룹에만 속할 수 있다. 하나 이상의 콘텐츠 서버가 네트워크에 있는 경우, 각각의 서버에 대해 그룹이 생성되기 때문에 같은 라우터라 하더라도 다른 그룹에 속할 수 있게 된다. 그룹 구성 시 사용하는 변수는 그림 4와 같으며, 헤드 라우터 선발과 그룹을 구성하는 방법은 다음과 같다.

i:	하위 라우터
j:	상위 라우터
k:	도메인
m_0^k:	도메인 k에 대한 단말 라우터의 *join* Interest 패킷 합
m_i^k:	도메인 k에 대한 하위 라우터 i의 *join* Interest 패킷 합
θ_j^k:	도메인 k에 대한 상위 라우터 j의 *join* Interest 패킷 합
$\theta_{threshold}$:	그룹 구성 계수 (θ로 줄여서 표현)

그림 4. 그룹 구성 시 사용된 변수들

Fig. 4. Variables for group establishment

먼저 도메인 k에 대한 그룹을 구성하기 위해 단말 라우터에서 상위 라우터로 *join* Interest 패킷을 전송한다. 이때 m_0^k은 1로 설정되어 있다. 이를 받은 상위 라우터 j는 연결되어 있는 복수의 하위 라우터 i가 보

낸 m_i^k값들을 θ_j^k에 추가한다. 이를 공식으로 표현하면 (1)과 같다.

$$\theta_j^k = \sum_{\forall i} m_i^k \cdot I(j,i) \quad (1)$$

여기에서 $I(j,i)$는 (2)와 같은 의미를 갖는다.

$$I(j,i) = \begin{cases} 1 & if\ join\ Interest\ packet\ is\ sent\ from\ j\ to\ i \\ 0 & Otherwise \end{cases} \quad (2)$$

$I(j,i)$가 1이면 하위 라우터 i에서 상위 라우터 j로 보낸 패킷이 있음을 의미하고, 0이면 보낸 패킷이 없음을 의미한다. 이후 헤드 라우터 선발을 위해 그림 5와 같은 알고리즘을 사용하여 비교한다.

1: **if** $\theta_j^k \leq \theta_{threshold}$
2: $m_j^k = head\ router\ for\ domain\ k$
3: **else then**
4: Send *join* Interest packet to higher router with θ_j^k
5: **end if**

그림 5. 헤드 라우터 선발 알고리즘

Fig. 5. Head Router selection algorithm

여기에서 $\theta_{threshold}$ (또는 θ)는 그룹 구성 계수로써, 그룹이 구성되는데 필요한 θ_j^k의 최소값이다. 만약 θ_j^k가 θ 이상이면 라우터 j는 헤드 라우터로 선발된다. 그렇지 않은 경우에는 θ_j^k을 *join* Interest 패킷에 m_j^k 형태로 실어 상위 라우터로 재전송한 뒤 응답을 기다린다.

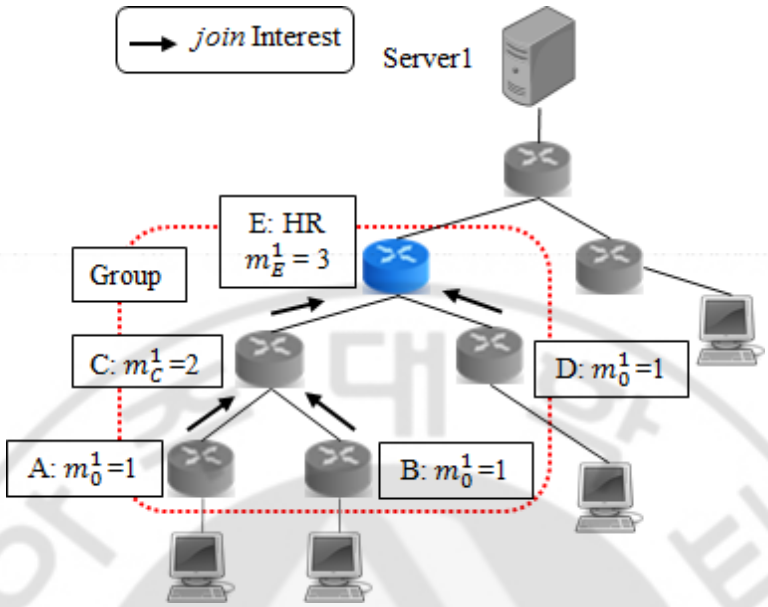
헤드 라우터 선발이 완료되면, *join* Interest 패킷에 대한 응답으로 *join*

Data 패킷을 보낸다. 이 패킷을 받은 하위 라우터들은 자신이 속한 그룹의 상위 라우터의 위치를 확인하고 그룹 형성 작업을 종료한다. *join* Data 패킷을 받지 못한 라우터들은 그룹 구성에 실패했으므로, 도메인 k에 속하지 않는 라우터로 네트워크 내에 존재하게 된다.

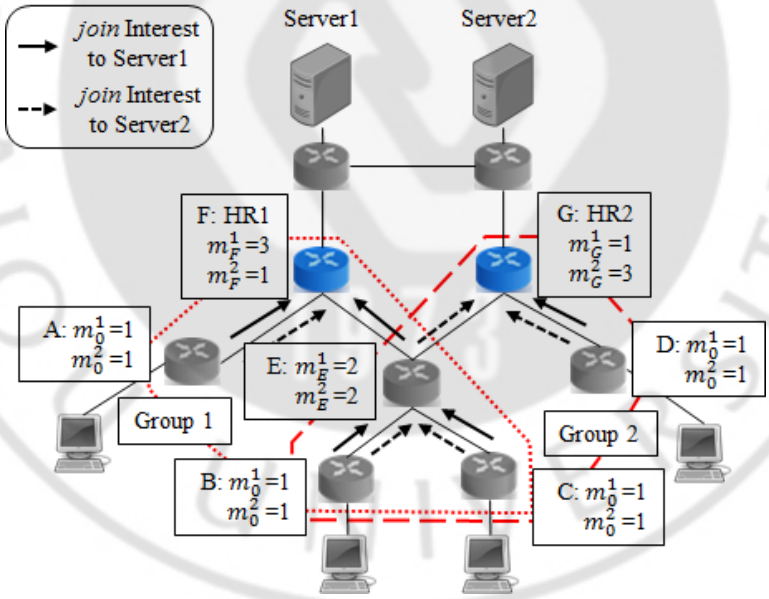
이러한 방식을 적용한 예시는 그림 6에 나타나 있다, 여기에서 사용된 θ는 3으로 설정하였고, Case 1은 1개의 콘텐츠 서버, Case 2는 2개의 콘텐츠 서버를 가진 네트워크이다 이에 대한 설명은 다음과 같다.

- Case 1: 라우터 A, B에서 *join* Interest 패킷을 Server 경로를 따라 상위 라우터로 보내고, 이를 받은 라우터 C의 θ_C^1 값은 2가 되어 그룹 구성이 불가능하게 된다. 따라서 m_C^1를 2로 설정하고 *join* Interest 패킷을 상위 라우터로 다시 보낸다. 이 패킷을 받은 라우터 E의 θ_E^1 값은 라우터 D에서 받은 1과 C에서 받은 2를 더한 값이 3이 되어 헤드 라우터가 되고, *join* Interest 패킷을 보낸 하위 라우터들과 함께 그룹을 형성한다.

- Case 2: 라우터 A, B, C, D에서 Server1과 Server2로 각각 *join* Interest 패킷을 보낸다. B, C로부터 패킷을 받은 라우터 E는 Server1, Server2의 θ_E^1, θ_E^2 값이 각각 2가 되어 그룹 구성이 불가능하고, m_E^1, m_E^2 값을 각각 2로 설정한 뒤 Server1의 경로에 위치한 상위 라우터 F, Server2의 경로에 위치한 상위 라우터 G로 각각 *join* Interest 패킷을 보낸다. 라우터 F는 Server1의 θ_F^1 값이 3이 되어 Server1에 대한 그룹의 헤드 라우터가 되고, 라우터 G는 Server2의 θ_G^2 값이 3이 되어 Server2에 대한 그룹의 헤드 라우터가 된다.



Case 1: 서버가 1 개인 네트워크에서 생성되는 1 개의 그룹 (θ = 3)



Case 2: 서버가 2 개인 네트워크에서 생성되는 2 개의 그룹 (θ = 3)

그림 6. 그룹 구성 예시

Fig. 6. Examples of group establishment

3.3. 콘텐츠 요청 및 전송

그룹 기반 협력 캐싱은 기존 CCN 방식을 따르기 때문에 Interest 패킷을 보내 콘텐츠를 요청하고 Data 패킷을 통해 얻는 pull 방식을 사용하고 있다. 다만 그룹 내부에서 사용하는 콘텐츠 전달 및 요청 방식은 추가 단계를 거치게 된다.

이를 위해 추가된 자료 구조는 그림 7과 같다. (a)에서 추가된 Node ID 필드는 콘텐츠가 실제 저장되어 있는 노드의 번호가 저장되어 있는 값이다. 만약 Node ID 필드에 들어있는 노드 번호가 현재 라우터의 노드 번호와 일치한다면 이 라우터에 해당 콘텐츠가 저장되어 있음을 의미한다. 그렇지 않다면, 그룹 내 다른 노드에 콘텐츠가 저장되어 있음을 뜻한다. 이 때 다른 노드의 위치는 (b)를 통해 알 수 있다. GT는 현재 라우터에서 1 hop 관계에 있는 그룹 내 다른 멤버 라우터의 노드 번호인 Node ID 필드와, 해당 라우터로 가기 위한 Face 번호가 있는 Face 필드, 해당 라우터의 현재 캐시 용량을 나타내는 Cache Remained 필드로 구성되어 있다.

Name	Data	Lifetime	Node ID

(a) Content Store (CS)

Node ID	Face	Cache Remained

(b) Group Table (GT)

그림 7. CCN 포워딩 엔진에서 추가적인 자료 구조

Fig. 7. Additional data structure in CCN forwarding engine

그림 8은 그룹 내에서 콘텐츠가 요청되고 전달되는 기본 과정을 보여주는 그림이다. 먼저 User A가 콘텐츠를 서버에 요청하면 (1) 서버는 해당 콘텐츠를 전송하고, 이 콘텐츠는 그룹의 헤드 라우터 C에 가장 먼저 도착한다 (2). 헤드 라우터 C는 하위 라우터 A의 캐시 용량을 확인하고 저장 여부를 결정하는데, 여기에서는 헤드 라우터에 저장하지 않고 그대로 전달하여 라우터 A에 저장한다고 가정한다 (3). 이후 User B가 같은 콘텐츠를 요청하여 해당 Interest 패킷이 헤드 라우터 C에 도착하였다 (4). 헤드 라우터는 라우터 A에 해당 콘텐츠가 있음을 알고, 라우터 A로 Interest 패킷을 보내 (5) 해당 콘텐츠를 User B로 전달한다. 이 때 전달된 콘텐츠는 라우터 B 내 캐시에 여유 공간이 있을 경우 저장된다 (6).

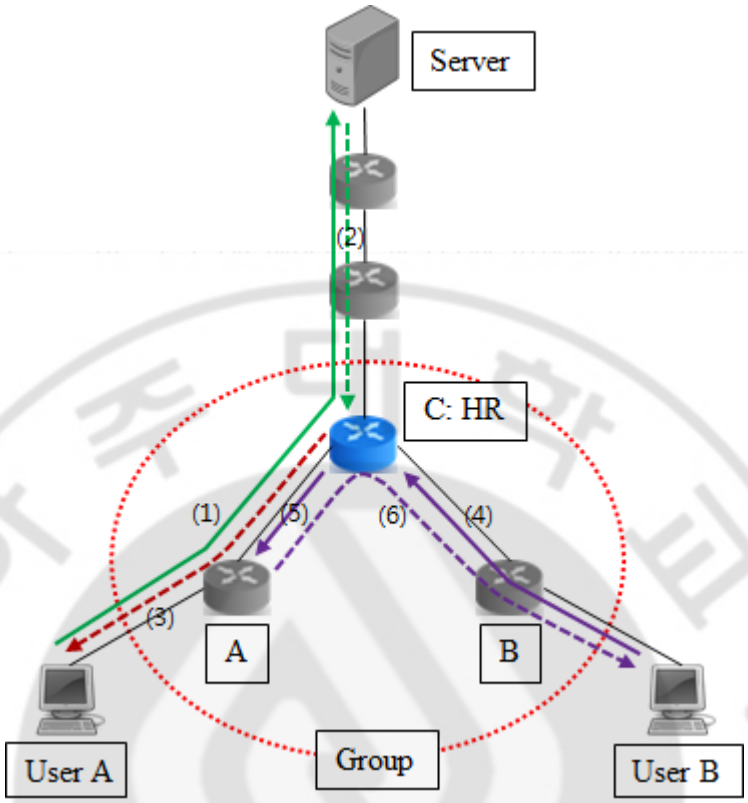


그림 8. 그룹 내 같은 콘텐츠 요청 및 전달 예시

Fig. 8. Example of same content request and forwarding in the group

앞서 설명한 것처럼 그룹에 속해 있는 라우터들은 그림 7의 CS 내 Node ID 필드와 GT를 추가적으로 사용한다. 하나의 그룹에도 속하지 못한 라우터들은 기존 CCN 방식을 따르며, 추가된 필드는 사용하지 않게 된다. 그룹 내부에 속해 있는 라우터에 콘텐츠가 요청되었을 경우, 그림 9와 같은 순서로 처리가 이루어진다.

먼저 Interest 패킷이 도착하면 CS에 해당 콘텐츠가 있는지 확인한다. 콘텐츠가 존재한다면 Node ID 필드를 참조해 콘텐츠가 저장된 위치를 파악하고, 존재하지 않는다면 기존 CCN 방식을 따른다. CS에서 확인할 때 자신의 노드 번호가 테이블에 저장되어 있다면, 이는 곧 현재 라우터 캐시에 해당 콘텐츠가 저장되어 있음을 의미하므로 콘텐츠를 바로 서비스한다. 만약 다른 노드 번호가 있다면, 해당 노드로 Interest 패킷을 보내 요청한 콘텐츠를 받도록 한다. 이러한 경우 CS는 심볼릭 링크와 같은 역할을 수행한다. Node ID 값이 자신의 노드 번호와 다르면 GT를 참조해 Face 값을 얻고, PIT에 이름과 Incoming Face를 추가한 뒤, 해당 Face로 Interest 패킷을 전송한다. 기존 CCN 방식에서는 Node ID 필드가 없기 때문에 해당 라우터에 저장된 콘텐츠만 서비스할 수 있으나, 제안된 방식에서는 그룹 내 위치를 얻어 해당 노드로 Interest 패킷을 보내 받음으로써 평균 지연시간과 홉 수를 줄이는 효과를 얻을 수 있다.

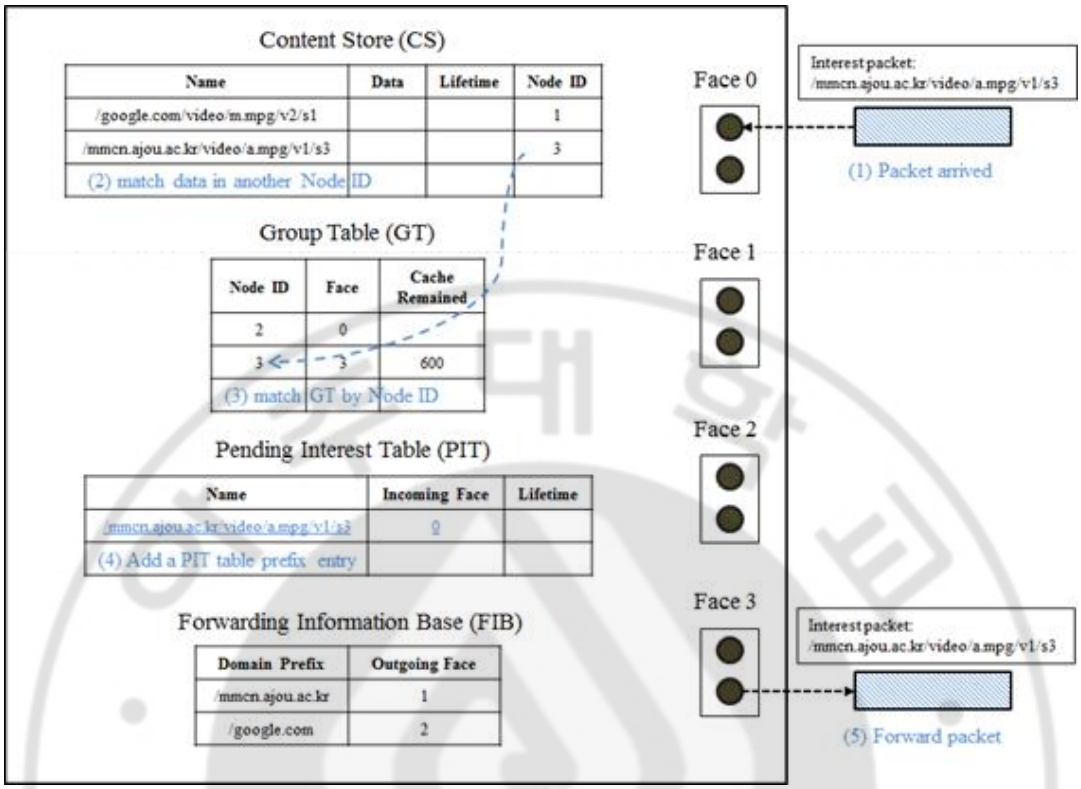


그림 9. CCN 포워딩 엔진에서 같은 그룹 내 콘텐츠 요청 처리 과정 (Node ID: 1)

Fig. 9. Content request process in CCN forwarding engine in case of same group (Node ID: 1)

그룹 내 라우터에 Data 패킷이 들어왔을 때 처리 방법은 기존 CCN과 동일하나, 콘텐츠를 CS에 바로 저장하지 않고 몇 가지 기준에 따라 저장 여부를 결정한다. 이 때 추가된 기준들은 다음과 같다.

- 요청한 콘텐츠가 CS에 이미 존재하고 있으면, 그룹 내부에서 요청해 서비스한 Data 패킷이므로 CS에 다시 저장하지 않고 바로 전송한다.

- 콘텐츠의 목적지가 현재 라우터면 캐시에 저장하고, CS의 Node ID 부분에 자신의 노드 번호를 추가한다.

- 콘텐츠의 목적지가 그룹 내 다른 라우터면 PIT에서 해당 콘텐츠 요청이 들어온 Face 값을 GT에서 찾아 노드 번호와 해당 노드의 현재 캐시 상태를 Node ID, Cache Remained 필드에서 가져온다. 이후 그림 10에 나타난 것처럼 현재 캐시 상태 값을 바탕으로 캐시 저장 유무를 판단한다. 캐시에 여유 공간이 있으면 현재 노드의 캐시에 Data 패킷을 저장하지 않고, Data 패킷의 이름과 다음 노드의 노드 번호만 저장한 뒤 해당 패킷을 전달한 뒤 다음 해당 Data 패킷의 크기만큼 Cache Size 값을 Cache Remained에서 감소시킨다. 만약 캐시에 여유 공간이 없으면 자신의 노드에 해당 Data 패킷을 저장하고 전달한다.

- 콘텐츠가 전달되는 다음 노드가 같은 그룹 노드가 아니라면 캐시에 저장하지 않고 전달한다.

전달된 콘텐츠는 단말 라우터에 캐시될 수도 있고, 하위 라우터에 캐시 공간이 부족한 경우 상위 라우터에 캐시가 이루어진다. 또한 그룹 내

부에서 요청해 서비스한 콘텐츠는 헤드 라우터 이외의 다른 라우터에 중복 저장할 수 있다. 같은 콘텐츠가 하위 라우터에서 다시 요청되는 경우, 상위 라우터에 있는 해당 콘텐츠가 하위 라우터로 한 홉씩 이동하며 캐시 교체가 발생하게 되며 단말 라우터에 가까운 위치로 이동하는 효과를 가져온다. 이러한 방식을 통해 그룹 내 콘텐츠의 중복 저장을 줄이고, 이를 다른 콘텐츠를 저장하는데 활용함으로써 그룹에 속한 클라이언트들에게 보다 높은 네트워크 성능과 향상된 QoS를 제공할 수 있다.

```
    node_id = Node ID of the router
    content = Data Packet
1:  Find GT to match Incoming Face in PIT
2:  GT[i] ← matched in GT
3:  if GT[i].cache_remained is EMPTY then
4:      Add CS (content, node_id)
5:  else then
6:      GT[i].cache_remained ← GT[i].cache_remained - content size
7:  end if
8:  Forward content
```

그림 10. 그룹 내 Data 패킷 처리 과정 알고리즘

Fig. 10. Algorithm of data packet procedure in the group

3.4. 그룹 내 콘텐츠 관리

AS 내에서 두 개 이상의 그룹에 속해 있는 라우터의 경우, 라우터 내 캐시는 라우터가 속해 있는 그룹들이 공동으로 사용하게 되며, 해당 콘텐츠는 그룹 기반 협력 캐싱에서 사용하는 콘텐츠 요청 및 전달 방식을 따르게 된다. 하지만 그룹에 속해 있지 않는 콘텐츠를 처리해야 하는 경우 해당 라우터는 기존 CCN 방식을 사용한다. 이러한 방식은 콘텐츠 서버마다 고유한 Domain Prefix 값을 갖고 있고, 이를 통해 그룹 내 콘텐츠의 구분이 가능하기 때문에 가능한 일이다.

그룹 내 라우터의 CS에 저장된 심볼릭 링크 형태 콘텐츠의 경우 기존 CCN과는 달리 타임아웃이 존재하지 않는다. 이는 하위 라우터에서 콘텐츠의 캐시 교체시기를 알 수 없기 때문이다. 그래서 타임아웃이 발생하거나 캐시에서 제거되는 경우 이를 그룹 내부에 알려주는 동기화 작업이 필요하다. 이에 대한 처리 과정은 그림 11에 나타나 있다. 라우터 B에서 타임아웃이나 캐시 교체가 일어난 경우 상위 라우터인 C로 *expire* Interest 패킷을 보낸다. 이 때 *expire* Interest 패킷은 PIT에 저장되지 않아 Interest 패킷에 대응하는 Data 패킷을 보낼 필요가 없다. 이 패킷을 받은 라우터 C는 해당 콘텐츠의 심볼릭 링크를 CS에서 삭제하고, B의 캐시 용량 변화를 GT에 기록한 뒤 상위 라우터인 D로 보낸다. D는 헤드 라우터이기 때문에 심볼릭 링크 삭제 및 C의 캐시 용량 변화를 업데이트하고 패킷을 삭제한다.

이러한 갱신 작업은 3.3에서 언급한 것과 같이 단말 라우터를 제외한 Data 패킷을 받은 라우터에서의 저장 여부는 패킷이 전달될 다음 노드의 캐시 상태를 보고 판단하기 때문에 반드시 필요한 작업이다.

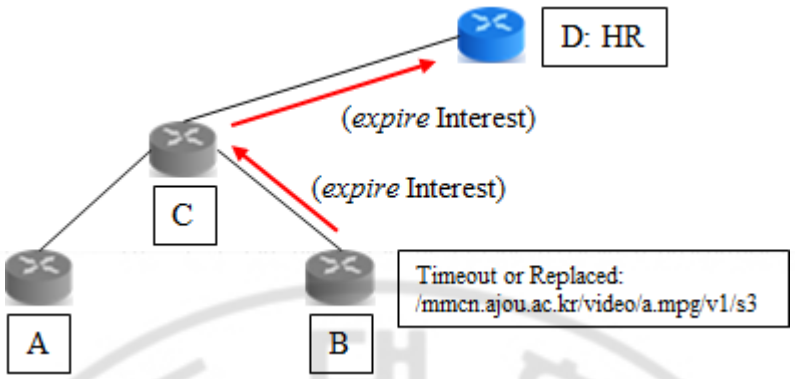


그림 11. 캐시 교체나 타임아웃으로 인한 그룹 내 라우터 간 동기화 과정

Fig. 11. Synchronization between routers in same group due to cache replacement or timeout because of cache replacement or timeout

4. 성능 평가

4.1. 시뮬레이션 환경

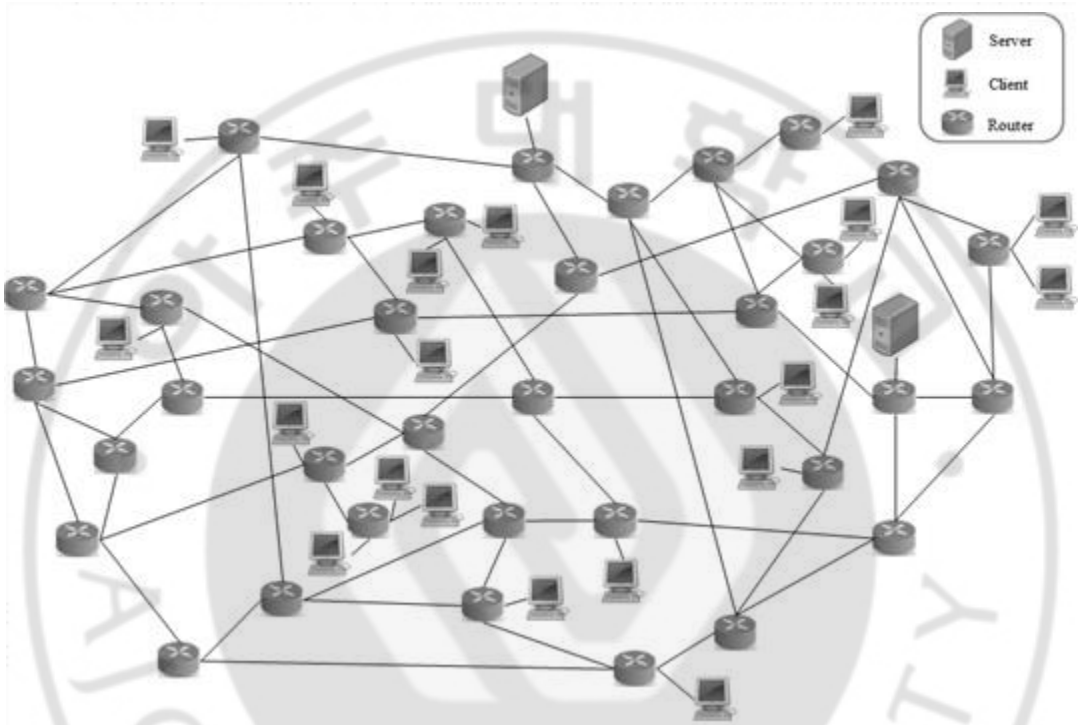


그림 12. 시뮬레이션 토폴로지

Fig. 12. Simulation Topology

네트워크 토폴로지는 그림 12와 같다. 먼저 GT-ITM을 기본으로 하여 하나의 AS 환경인 1 stub이 생성될 수 있도록 수정하여 무작위로 생성하였다 [14]. 네트워크 생성은 메시 구조를 갖는 35개의 라우터를 기본으로 20개의 클라이언트와 2개의 서버로 구성되어 있으며, 100 Mbps 이더넷으로 연결되어 있다.

표 1. 시뮬레이션 파라미터들

Table 1. Simulation Parameters

Parameter	Value
Number of Servers	2
Number of Clients	20
Number of Routers	35
Group Establishment Value: θ	1, 2, 3
Zipf-like: α	0.7 (option) 0.7 – 1.6
Number of Contents	5,000 per server
Number of Categories per server	200
Number of contents per category	25
Content request	Approx. λ = 500 request/s per client (with Poisson Distribution)
Interest packet size	215 bytes
Data packet size	4K bytes
Cache Size	2% (per content size) (option) 2 – 20%
Network Bandwidth	100 Mbps
Simulation Environment	OPNET Modeler 16.1A
Simulation Count	5 per each case

실험 환경 변수는 표 1과 같다. 우선 실험을 위해 OPNET Modeler 16.1A [15]를 사용하여 구현한 CCN 시뮬레이터를 사용하였다 [16]. 그룹 생성 계수 θ는 2와 3을 사용하여 그룹 형성을 하였고, 대조군으로 1인 기존 CCN도 함께 실험하여 비교하였다. Zipf 분포에서 α 값은 기본 값을 0.7로 하였고, 부가적으로 0.7부터 1.6까지 변화를 주었다. 콘텐츠의 수는 각 콘텐츠 서버 당 Zipf 분포 [17]를 따르는 5,000개로 하였는데 200개의 카테고리를 갖고 있고, 한 카테고리 당 25개의 콘텐츠 분포를 갖고 있다. 초당 콘텐츠 요청 횟수는 각 노드 당 Poisson 분포를 따르는 λ = 500 request/s로 하였다. 실험에 사용된 패킷 크기는 CCNx [18]를 참고하여 Interest 패킷은 215 bytes를 사용하였고, Data 패킷은 기본 청크 (chunk) 크기인 4 Kbytes로 하였다. 라우터 내 캐시 크기는 전체 콘텐츠 크기의 대비 비율을 갖고 있으며, 2%를 기본 값으로 하고 2%에서 20%까지 변화를 주었다. 실험은 5회 반복하여 실행하였다.

본 실험에서 사용된 성능 측정 결과 값에 대한 정의는 다음과 같다.

- 캐시 적중률 (Cache Hit Ratio): 네트워크 내에서 라우터 내 CS에 요청한 콘텐츠의 수 대비 실제 서비스된 횟수를 나타내는 값으로써, 다음과 같은 식으로 계산된다.

$$\text{Cache Hit Ratio} = \frac{\sum_i hit_i}{\sum_i req_i}, \forall i = 1, .., N \quad (3)$$

여기에서 req_i는 Interest 패킷이 라우터 i의 CS에 도착해 요청한 콘텐츠가 있는지 확인한 횟수의 합이고, hit_i는 요청된 콘텐츠가 실제로 라우터 i의 CS에 있어 바로 서비스 될 때의 횟수의 합이다. N은 AS 내에 있는 라우터의 총 개수이다. 본 논문에서 헤드 라우터 내 CS에 저장되어

있는 심볼릭 링크는 hit_i에서 제외하고, 실제 콘텐츠가 저장되어 있는 라우터에서만 hit_i를 계산하였다. 심볼릭 링크로 인해 전달되는 Interest 패킷은 req_i에 추가된다.

- 평균 지연 시간 (Average Latency Time): 클라이언트에서 콘텐츠를 요청하고 받을 때까지 소요된 시간의 평균값을 나타낸 값이다.
- 평균 홉 수 (Average Hop Count): Data 패킷이 클라이언트까지 도달하는데 거쳐 간 라우터 홉 수의 평균값을 나타낸 값이다.
- 전체 네트워크 로드 (Overall Network Load): 1초당 전체 네트워크에서 흘러간 데이터의 양을 측정한 값이다.
- 콘트롤 오버헤드 (Control Overhead): 전체 네트워크 로드 대비 *expire* Interest 패킷의 양의 비율을 나타낸 값이다.
- 콘텐츠가 라우터에 있을 확률 (Prob. of content in router): 클라이언트가 요청한 콘텐츠가 라우터의 캐시에 저장되어 있을 확률을 나타낸 값이다.

4.2. 시뮬레이션 결과

먼저 Zipf 분포에서 사용하는 α 값을 0.7로 고정하고, 라우터 내 캐시 크기를 2%에서 20%까지 변화시키면서 기존 CCN과 제안 기법, ABC, Betw의 성능 차이를 비교해 보았다. 제안 기법에서 그룹 구성 계수 θ는 2와 3을 사용하여 각각의 차이를 비교하였다.

그림 13은 캐시 크기 변화에 따른 네트워크 내 캐시 적중률의 변화를 보여주는 그래프이다. 제안 기술이 기존 CCN보다 더 높은 캐시 적중률을 보이고, 캐시 크기가 커질수록 더 큰 차이를 보임을 알 수 있다. 캐시 크기가 2%인 경우 기존 CCN과 θ가 2인 제안 기술의 차이는 2% 정도이나, 캐시 크기가 20%로 늘어난 경우 6% 가까이의 성능 차이를 보여준다. 또한 θ가 3인 경우 기존 CCN과의 차이는 더 커지게 되며, 캐시 크기가 20%일 때 약 10% 정도의 성능 차이를 보여준다.

그림 14는 캐시 크기 변화에 따른 평균 지연 시간의 변화를 보여주는 그래프이다. 그림 13에서와 마찬가지로 θ가 3인 경우 가장 낮은 평균 지연 시간을 보여주고 있다. 하지만 캐시 크기가 커질수록 차이가 줄어들고 있음을 알 수 있는데, 이는 캐시 크기가 늘어남에 따라 저장할 수 있는 콘텐츠의 수가 늘어나게 되고, θ 값이 커질수록 우회하여 콘텐츠를 요청하고 전달하는 횟수가 늘어나기 때문이다.

그림 15는 캐시 크기 변화에 따른 평균 홉 수의 변화를 보여주는 그래프이다. 그림 14과 마찬가지로 θ가 3인 경우 가장 낮은 평균 홉 수를 보여주고 있는데, 이는 요청된 Data 패킷이 클라이언트까지 도달하는데 소요되는 홉 수가 가장 작다는 것을 의미한다. 하지만 그룹 구성의 특성

상 θ가 클수록 우회하는 경로가 많아지는 단점을 띄고 있어 캐시 크기가 늘어남에 따라 기존 CCN이나 θ가 2인 경우보다 차이가 줄어듦을 알 수 있다.

그림 16은 전체 네트워크 로드 변화를 보여주는 그래프이다. 캐시 크기가 늘어날수록 기존 CCN과 제안 기술의 차이가 줄어듦을 알 수 있다. 3.4에서 언급한 것처럼 그룹 내 라우터 사이에는 동기화를 위한 *expire* Interest 패킷이 발생되게 되는데 θ가 커질수록 발생되는 *expire* Interest 패킷이 많아지게 되고, 그림 15에서 언급한 것처럼 평균 홉 수의 차이도 줄어들기 때문에 이와 같은 결과가 발생하게 된다.

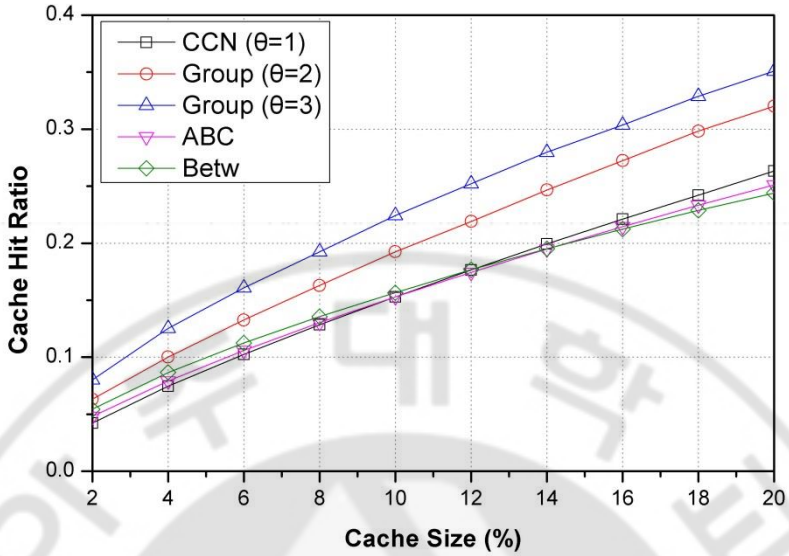


그림 13. 캐시 크기 변화에 따른 캐시 적중률 변화

Fig. 13. Cache hit ratio as cache size varies

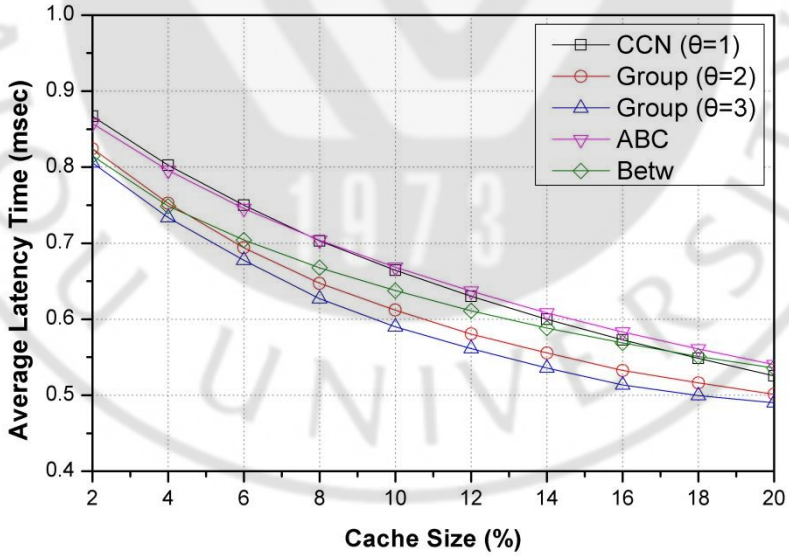


그림 14. 캐시 크기 변화에 따른 평균 지연 시간 변화

Fig. 14. Average latency time as cache size varies

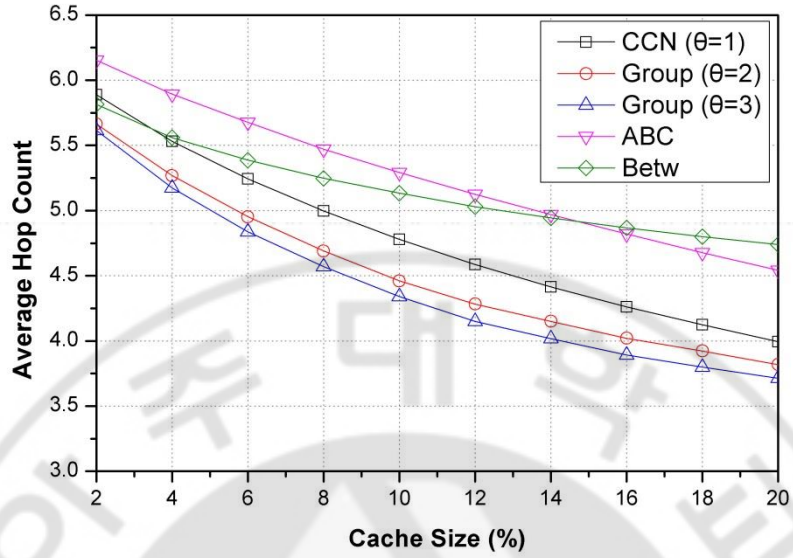


그림 15. 캐시 크기 변화에 따른 평균 홉 수 변화

Fig. 15. Average hop count as cache size varies

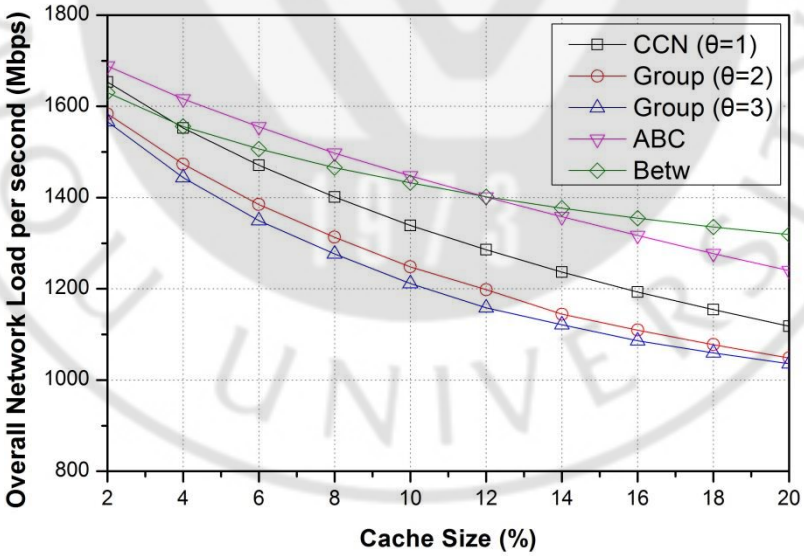


그림 16. 캐시 크기 변화에 따른 전체 네트워크 로드 변화

Fig. 16. Overall network load as cache size varies

이러한 성능 차이는 카테고리 별로 콘텐츠가 라우터에 있을 확률과 홉 수의 차이에 의해 발생한다. 그림 17은 캐시 크기가 2%일 때 카테고리 당 콘텐츠가 라우터에 있을 확률을 나타낸 그래프이고, 그림 18은 캐시 크기가 20%일 때의 그래프이다. 캐시 크기가 2%인 경우, Betw 방식이 카테고리가 1일 때의 확률이 약 0.41로 가장 높은 값을 갖고 시작하며, θ가 3일 때의 제안 기술이 약 0.33, θ가 2일 때의 제안 기술이 약 0.295 값을 갖고 시작한다. 이는 그림 13에서 캐시 크기가 2%일 때의 결과와 비슷한 값을 보임을 알 수 있다. 위와 같은 결과를 이를 CDF로 표현하면 제안 기술이 더 많은 카테고리를 캐시에 저장하고 있음을 알 수 있다. 캐시 크기가 20%인 경우, θ가 3일 때의 제안 기술이 카테고리가 1일 때의 확률이 약 0.86으로 다른 기술에 비해 더 높은 확률을 갖고 있으며, θ가 2일 때의 제안 기술이 0.83, 기존 CCN이 약 0.80, Betw 방식이 약 0.77, ABC 방식이 약 0.70으로 그림 13에서 캐시 크기가 20%일 때의 결과와 유사한 추세를 보인다. CDF로 보면 Betw는 상위 카테고리에 캐시 저장이 치중되어 있음을 알 수 있고, 제안 기술이 기존 기술에 비해 더 넓은 영역의 카테고리를 캐시에 저장함을 알 수 있다.

그림 19와 그림 20은 캐시 크기가 각각 2%, 20%일 때의 카테고리 당 콘텐츠의 평균 전달 홉 수를 나타낸 그래프이다. 그림 17과 그림 18과 유사한 결과를 가지며, 제안 기술이 기존 기술에 비해 더 낮은 평균 홉 수 값을 갖는다. 이는 라우터를 그룹 단위로 형성한 뒤 콘텐츠를 분산 저장 및 관리하기 때문에 상대적으로 그룹 내에서 요청한 콘텐츠를 찾을 확률이 높아지기 때문이다.

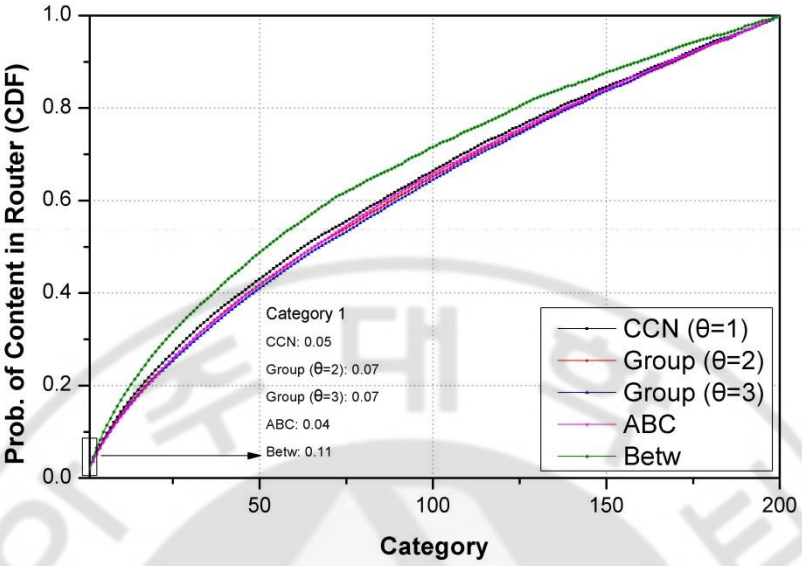


그림 17. 카테고리 당 콘텐츠가 라우터에 있을 확률 (Cache=2%, α=0.7)

Fig. 17. Probability of content in router per category (Cache=2%, α=0.7)

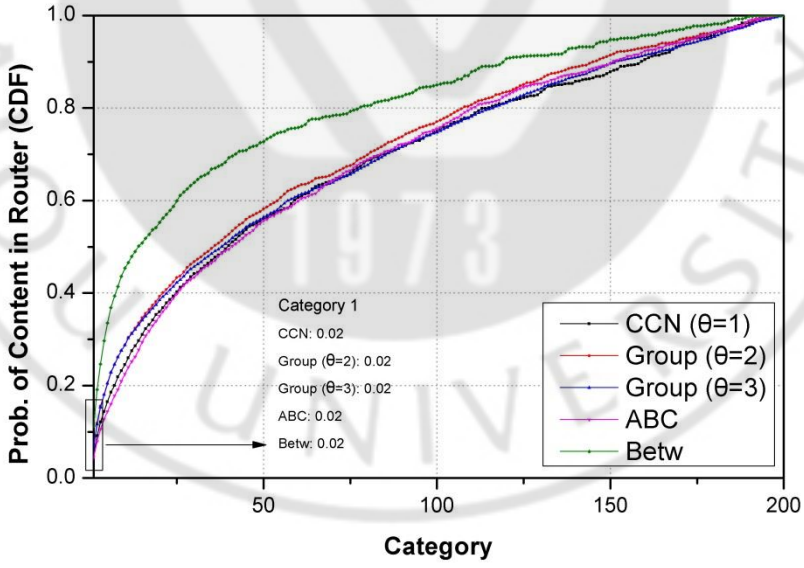


그림 18. 카테고리 당 콘텐츠가 라우터에 있을 확률 (Cache=20%, α=0.7)

Fig. 18. Probability of content in router per category (Cache=20%, α=0.7)

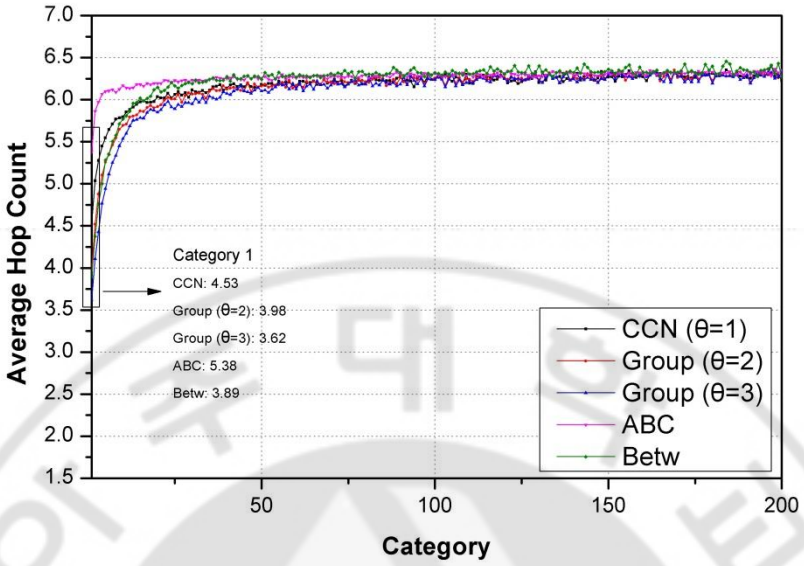


그림 19. 카테고리 당 평균 홉 수 (Cache=2%, α=0.7)

Fig. 19. Average hop count per category (Cache=2%, α=0.7)

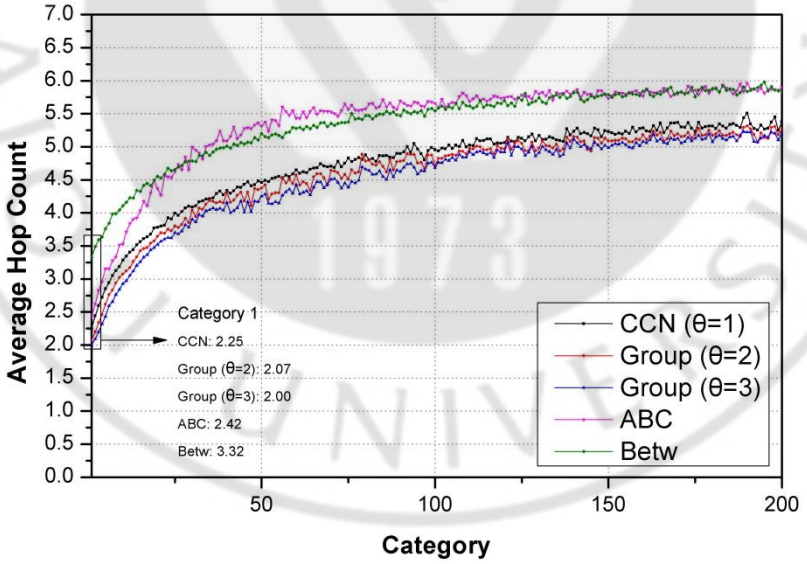


그림 20. 카테고리 당 평균 홉 수 (Cache=20%, α=0.7)

Fig. 20. Average hop count per category (Cache=20%, α=0.7)

그림 21은 콘텐츠 중요도에 따른 캐시 적중률의 변화를 보여주는 그래프이다. 콘텐츠 중요도 상위 2.5%와 상위 20%인 경우 모두 제안 기술이 기존 CCN보다 더 높은 캐시 적중률을 갖고 있다. 이는 제안 기술의 특성상 인접한 라우터를 그룹으로 묶어 관리하기 때문에 그룹 내 저장 가능한 콘텐츠의 수를 늘려 전체적인 캐시 적중률을 상승시키는 효과를 기대할 수 있다.

그림 22은 *expire* Interest 패킷 발생이 전체 네트워크에서 차지하는 비중을 알아보기 위한 그래프이다. 캐시 크기가 늘어남에 따라 오버헤드가 점점 줄어들어 캐시 크기가 14%에 도달하면 오버헤드가 거의 일정해 짐을 알 수 있다. 이는 그룹 내 캐시 교체나 타임아웃 빈도수가 거의 같아졌음을 의미한다. 또한 θ 값이 클수록 더 높은 오버헤드를 갖고 있음을 알 수 있는데, 그룹 구성 규모가 커져서 보다 많은 노드를 거치기 때문이다.

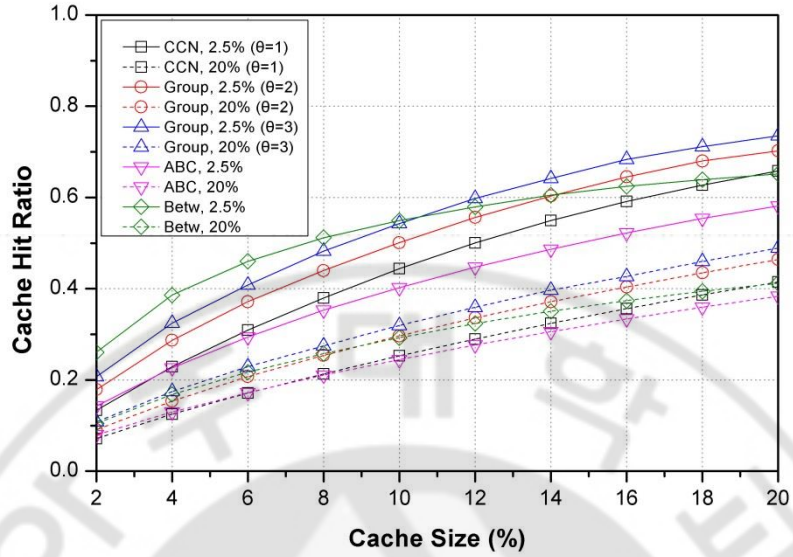


그림 21. 콘텐츠 중요도에 따른 캐시 적중률 변화

Fig. 21. Cache hit ratio based on content popularity

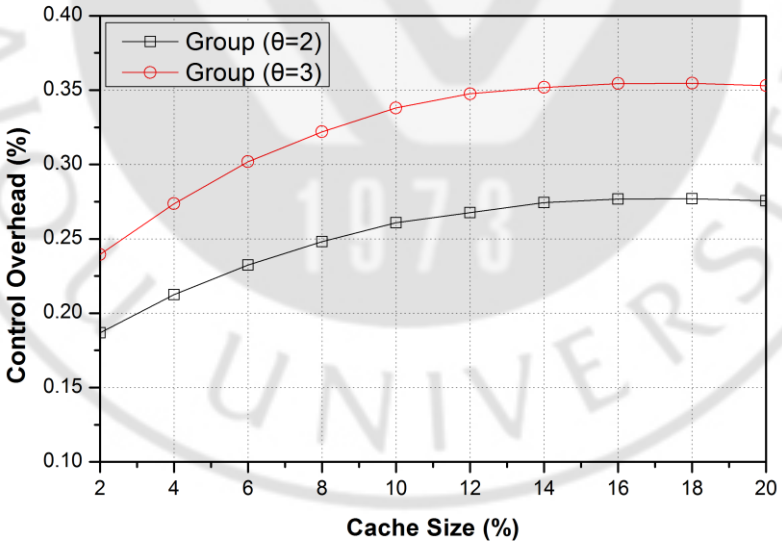


그림 22. 캐시 크기 변화에 따른 콘트롤 패킷 오버헤드 변화

Fig. 22. Control packet overhead as cache size varies

다음으로 Zipf 분포에서 사용하는 α 값에 변화를 주어 실험을 수행하였다. 기존에 사용한 α = 0.7은 일반적인 Web 콘텐츠를 나타낼 때 사용한 값이고 [17], α = 1.6은 Web 미디어 콘텐츠를 나타낼 때 사용하는 값이다 [19]. 또한 α 값이 커질수록 인기도가 높은 콘텐츠의 요청 횟수가 높아지는 경향을 보이고 있다. 따라서 본 절에서는 α 값을 0.7에서 1.6까지 0.1 단위로 변화를 주어 기존 CCN과 제안 기술, ABC, Betw에서의 캐시 적중률을 비교하였다.

그림 23는 α 값 변화에 따른 캐시 적중률의 차이를 보여주는 그래프이다. α 값이 커질수록 캐시 적중률이 상승하게 되며, 제안 기술의 경우 θ가 2와 3일 때의 차이가 점점 줄어들어 α가 1.6일 때 거의 같아짐을 알 수 있다. 이는 Zipf 분포의 특성을 따르기 때문이다.

그림 24은 α 값 변화에 따른 콘트롤 오버헤드를 보여주는 그래프이다. α 값이 상승함에 따라 전체 네트워크에 걸리는 오버헤드가 늘어나나 α가 1.3 이상으로 커지면 반대로 감소함을 알 수 있다. 이를 통해 α가 1.3 이상이면 캐시에 콘텐츠가 교체되는 횟수가 줄어들어 전체적인 성능 향상을 이끌어 냄을 알 수 있다.

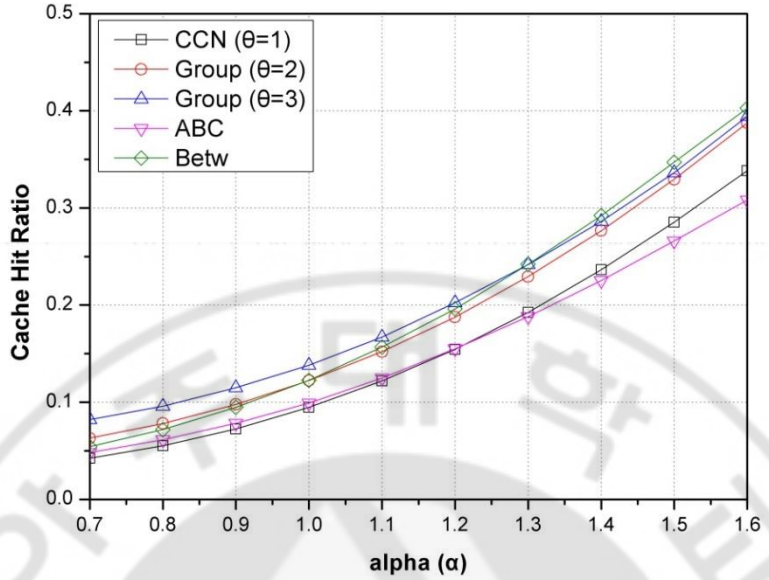


그림 23. α 값 변화에 따른 캐시 적중률 변화

Fig. 23. Cache hit ratio as α value varies

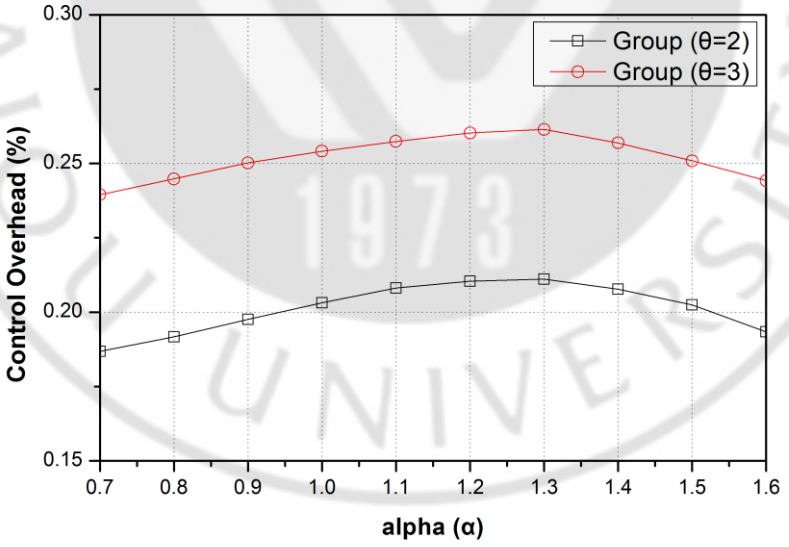


그림 24. α 값 변화에 따른 콘트롤 패킷 오버헤드 변화

Fig. 24. Control packet overhead as cache size varies as α value varies

그림 25는 캐시 크기가 2%, α = 1.6일 때 카테고리 당 콘텐츠가 라우터에 있을 확률을 나타낸 그래프이다. α = 1.6인 경우 카테고리가 1일 때 Betw약 0.20으로 가장 높은 확률을 갖고, θ가 3일 때의 제안 기술이 약 0.90, θ가 2일 때의 제안 기술이 약 0.86, 기존 기술이 약 0.80, 가 약 0.68의 값을 가진다. 이를 CDF로 표현해 200개의 카테고리까지의 추세를 살펴보면, Betw가 가장 급격한 변화를 보임을 알 수 있고, α 값이 1.6인 특성상 상위 카테고리에 더 많은 요청이 몰리기 때문에 가장 높은 캐시 적중률을 갖는 방식임을 알 수 있다. 또한 그림 17과 비교했을 때 카테고리 당 콘텐츠의 존재 확률이 변화하게 되고, 이에 따라 그림 23과 같은 결과를 도출해 낼 수 있음을 알 수 있다.

그림 26은 캐시 크기가 2%, α = 1.6일 때 카테고리 당 콘텐츠의 평균전달 홉 수를 나타낸 그래프이다. 제안 기술이 기존 기술에 비해 더 낮은 평균 홉 수를 가짐을 알 수 있으며, 그림 19와 그림 26의 결과를 종합해 봤을 때 제안 기술이 기존 기술에 비해 콘텐츠의 분산 저장 및 관리가 가능하여 더 낮은 평균 홉 수를 가짐을 알 수 있다.

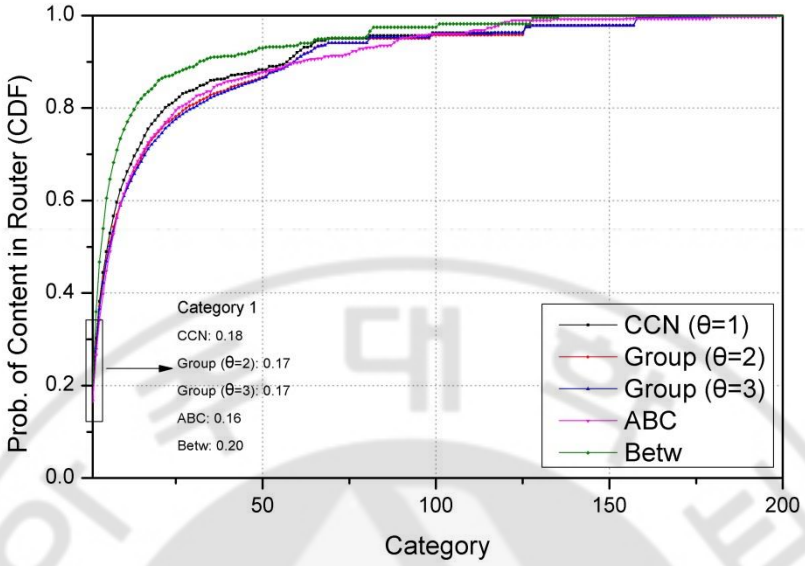


그림 25. 카테고리 당 콘텐츠가 라우터에 있을 확률 (Cache=2%, α=1.6)

Fig. 25. Probability of content in router per category (Cache=2%, α=1.6)

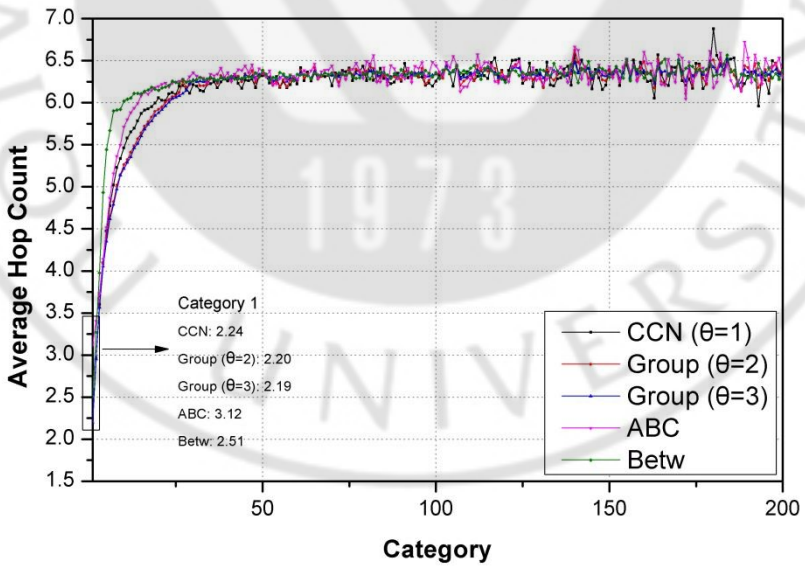


그림 26. 카테고리 당 평균 홉 수 (Cache=2%, α=1.6)

Fig. 26. Average hop count per category (Cache=2%, α=1.6)

5. 결 론

오늘날 증가하는 인터넷 트래픽의 40% 이상을 인터넷 비디오 트래픽이 차지하고 있으며, 유무선 환경의 발달로 인해 증가폭은 점점 늘어날 것으로 예상된다. 하지만 기존 IP 네트워크 기술로는 이러한 트래픽의 증가를 경감시키는데 한계가 있으므로, 트래픽을 근본적으로 줄이는 대안 기술로 콘텐츠 중심 네트워킹 개념이 제안되었다.

본 논문을 통해 인접한 라우터를 그룹으로 묶어 관리함으로써 캐시 용량을 늘리는 효과와 함께 그룹 내 저장 가능한 콘텐츠의 수를 늘리는 그룹 기반 협력 캐싱을 제안하였다. 또한 실험을 통해 기존의 최저 사용빈도 교체 알고리즘에 비해 전체 캐시 적중률, 평균 지연시간, 평균 홉수의 개선이 이루어져 전체적인 네트워크 성능 향상이 있음을 알 수 있었다. 그룹 생성 계수 θ가 클수록 보다 나은 성능 향상을 기대할 수 있으나, 콘트롤 오버헤드가 상승하는 단점도 알 수 있었다. 마지막으로 상대적으로 중요도가 높은 콘텐츠뿐만 아니라 낮은 콘텐츠의 캐시 적중률도 함께 높임으로써 다양한 종류의 콘텐츠에 대해 향상된 QoS를 보장할 수 있을 것으로 기대된다.

참고문헌

[1] Cisco Systems White Paper. (2013, May). Cisco Visual Networking Index: Forecast and Methodology, 2012-2017. Cisco Systems, CA. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, “Networking named content,” in *Proc. ACM CoNEXT*, 2009, pp.1–12.

[3] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. Thornton, D. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos, et al.. (2010, Oct.). Named Data Networking (NDN) Project. PARC, CA. Available: https://www.parc.com/content/attachments/named-data-networking.pdf

[4] A. Wolman, M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. Levy. (1999, Dec.). On the scale and performance of cooperative web proxy caching. *ACM SIGOPS Operating Systems Review*. *33(5)*, pp.16–31.

[5] E. Cohen and S. Shenker, "Replication strategies in unstructured peer-to-peer networks," in *Proc. ACM SIGCOMM*, 2002, pp.177-190.

[6] G. Zhang, Y. Li, and T. Lin. (2013, Dec.). Caching in information centric networking: A survey. *Computer Networks*. *57(16)*, pp.3128-3141.

[7] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino. (2013, Apr.). Evaluating per-application storage management in content-centric networks. *Computer Communications*. *36(7)*, pp.750-757.

[8] Z. Ming, M. Xu, and D. Wang, "Age-based cooperative caching in Information-Centric Networks," in *Proc. INFOCOM WKSHPS*, 2012, pp.268-273.

[9] W. K. Chai, D. He, I. Psaras, and G. Pavlou. (2013, Apr.). Cache “less for more” in information-centric networks (extended version). *Computer Communications*. *36(7)*, pp.758-770.

[10] Y. Kim and I. Yeom. (2013, Sep.). Performance analysis of in-network caching for content-centric networking. *Computer Networks*. *57(13)*, pp.2465-2482.

[11] Z. Li and G. Simon, "Time-Shifted TV in Content Centric Networks: The Case for Cooperative In-Network Caching," in *Proc. IEEE ICC*, 2011, pp.1-6.

[12] J. M. Wang and B. Bensaou, “Progressive caching in CCN,” in *Proc. IEEE GLOBECOM*, 2012, pp.2727-2732.

[13] G. Tyson, S. Kaune, S. Miles, Y. El-khatib, A. Mauthe, and A. Taweel, "A Trace-Driven Analysis of Caching in Content-Centric Networks," in *Proc. ICCCN*, 2012, pp.1-7.

[14] E. W. Zegura. 1996. GT-ITM: Georgia Tech internetwork topology models. Available: http://www.cc.gatech.edu/projects/gtitm/

[15] OPNET Technologies, Inc. Available: http://www.opnet.com

[16] S. Kim, K. Kim, S. Choi, B. Kim, and B. Roh, “An implementation of Content-Centric Network, using OPNET Modeler,” presented at OPNETWORK 2012, Washington D.C., USA, Aug. 13-17, 2012.

[17] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications," in *Proc. IEEE INFOCOM*, 1999, pp.126-134.

[18] CCNx Project™ (version 0.8.1). (2013, Aug.). Available: http://www.ccnx.org

[19] L. Cherkasova and M. Gupta, "Characterizing locality, evolution, and life span of accesses in enterprise media server workloads," in *Proc. ACM NOSSDAV*, 2002, pp.33-42.

ABSTRACT

These days, the Internet traffic sores and the biggest proportion is due to content transmission of Internet video traffic. In order to lessen the amount of traffic, Content-Centric Networking (CCN), which is one of future internet technology, is introduced. However, the process of CCN, which is LRU algorithm for cache replacement and store the content all the routers through the pass, occurs frequent cache replacement. This problem degrades cache hit ratio and cannot reduce network traffic effectively because of limited cache size per each router. In this paper, adjacent routers can be made a group based on the route to server in order to overcome the problem of current CCN. The proposed scheme, group-based cooperative caching, distributes the contents and manages it per group. Also, the number of redundant caching can be reduced and more variety of contents in the network has a chance to save the cache. Therefore, improved network performance would be expected because of higher cache hit ratio. First, performance factors, which are cache hit ratio, average latency time, average hop count, and overall network load is compared between current CCN and proposed scheme. After evaluation of the performance mentioned above, we verify better performance of proposed scheme to evaluate several kinds of characteristic of the contents.