콘텐츠 중심 네트워킹에서 콘텐츠-이름 유사도에 따른 연관 콘텐츠 다중 포워딩 기법 연구

이두호○ 홍충선*
경희대학교 컴퓨터공학과
{ dooholee, cshong }@khu.ac.kr

Multi-Contents Forwarding Scheme According to Contents-Name Similarity in Content-Centric Networking

Dooho Lee○ ChoongSeon Hong*
Department of Computer Engineering, Kyunghee University

요 약

콘텐츠 중심 네트워킹에서는 사용자가 콘텐츠를 요청하면 그에 해당하는 콘텐츠를 서버로부터 전달 받아와서 중간 라우터들이 저장하고 전송하여 똑같은 콘텐츠를 다른 이용자가 요청하는 경우 가까운 라우터에서 빠르게 콘텐츠를 전송받는다. 사용자가 포털 사이트에서 제공하는 연관 검색어 기능을 이용하여 콘텐츠를 요청하게 된다고 가정하자. 이 경우 먼저 요청 했던 콘텐츠와 달리 유사하지만 다른 콘텐츠를 요청하는 것이기 때문에 서버로부터 새로 콘텐츠를 전송 받아와야 한다. 본 논문에서는 라우터가 콘텐츠를 전송 받을 때 유사한 콘텐츠 또한 전송받아 이후에 사용자가 유사한 콘텐츠를 요청할 때 빠르게 사용자에게 전송하는 기법을 제안 한다.

1. 서 론

현재 인터넷 트래픽은 예전에는 상상할 수 없었던 만큼 빠르게 늘어가고 있다. 그 중에서도 동영상 콘텐츠에 대한 트래픽이 가장 큰 부분을 차지하고 있으며 증가폭도 가장 크다. 그 중 유투브의 동영상 트래픽이 전 세계에서 제일 높은 동영상 트래픽 양을 차지하고 있다[1]. 이와 같은 이유로 급격히 증가하는 동영상 트래픽을 관리하기 위한 연구들이 활발하게 진행되고 있다. 본 논문에서는 이러한 연구들 중에서도 가장 활발히 연구되는 CCN(Contents Centric Networking)을 다루고자 한다[2]. 이 CCN기술을 활용하여 넘치는 동영상 트래픽을 줄이고 수많은 이용자들에게 보다 빠르고 효율적으로 동영상 콘텐츠를 제공 할 수 있는 방법이 필요하다.

그리고 유투브에서 동영상 서비스를 이용하다 보면 연관된 동영상도 같이 제공해준다. 인기 동영상 순위에도 인기 있는 동영상과 관련하여 연관된 동영상이 같이 올라가 있는 경우가 많다. 그리고 연관 검색어에 대한 적합도가 상당히 높아 사용자들이 연관된 동영상을 이용하기 유용하다[3]. 이와 같은 연관된 유사한 콘텐츠를 효율적으로 포워딩하는 방법이 필요하다.

2. 관련 연구

2.1 CCN (Content Centric Networking) [2]

CCN은 IP 주소가 아닌 콘텐츠의 이름으로 원하는 콘텐츠를 가져와 라우터 내에 캐시를 이용해 저장해 두었다가 후에 같은 콘텐츠에 대한 요청이 왔을 때 캐시에 두었던 콘텐츠를 분배한다. CCN에서 유저는 Interest 패킷을 브로드캐스팅 해서 가장 먼저 Data 패킷을 응답한 라우터나 서버를 주 경로로 지정하여 콘텐츠를 전송 받아온다. 즉 하나의 요청에 하나의 단일 경로가 설정된다. 더불어 브로드캐스팅 된 Interest 패킷을 받은 다른 서버나 라우터가 보낸 Data 패킷은 유저가 폐기시킨다.

2.2 Levenshtein Distance Algorithm [4]

Levenshtein Distance 알고리즘은 한 문자열을 다른 문자열로 바꿀 때 몇 번의 변경이 필요한지 측정하는 방식이다. Levenshtein Distance를 구하는 알고리즘은 다음과 같다.

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j)+1 \\ \mathrm{lev}_{a,b}(i,j-1)+1 \\ \mathrm{lev}_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} \end{cases} \quad \begin{array}{l} \text{if } \min(i,j)=0, \\ \\ \text{otherwise.} \end{array}$$

그림 1. Levenshtein Distance 알고리즘

예를 들어 새끼 고양이를 뜻하는 kitten을 앉았다는 sitting으로 변경하기 위해서는 첫 문자 k를 s로 바꾸고

본 연구는 미래창조과학부가 지원한 2014년 정보통신/방송(ICT) 연구개발사업의 연구결과로 수행되었음.

*Dr. CS Hong is the corresponding author

끝에서 2번째 문자인 e를 i로 바꾸고 마지막으로 g를 마지막에 추가한다. 이와 같은 과정을 거치면 총 변경된 횟수는 3이다. 이 값을 구하기 위해서 2차원 배열을 이용하여 Levenshtein distance를 구한다. 첫 번째 행에 문자열 하나를, 첫 번째 열에 다른 문자를 순차적인 숫자로 변환하여 입력하고, 먼저 (s, k)는 같은 문자가 아니므로 (s, k) 의 왼쪽, 위, 그리고 왼쪽/위 대각선의 값에 1을 더한 값 중에 최소 값을 입력한다. (s, k)의 왼쪽과 위쪽의 값은 2, 왼쪽/위 대각선은 1이 됨으로 1을 입력한다. (s, i) 역시 두 문자가 같지 않으므로, (2, 3, 2) 중에 최소 값인 2가 된다. (i, i)의 경우, 두 문자가 동일하게 되므로 왼쪽/위 대각선의 값을 입력하면 된다. 여기서는 1이 된다. 동일한 방법으로 나머지 값들을 입력하면 다음과 같이 배열이 완성 되고 마지막 값이 Levenshtein Distance 값이 된다.

		k	i	t	t	e	n
	0	1	2	3	4	5	6
s	1	1	2	3	4	5	6
i	2	2	1	2	3	4	5
t	3	3	2	1	2	3	4
t	4	4	3	2	1	2	3
i	5	5	4	3	2	2	3
j	6	6	5	4	3	3	2
g	7	7	6	5	4	4	3

그림 2. Levenshtein distance 계산

3. 본론

3.1 Levenshtein distance 알고리즘 적용

Levenshtein distance 알고리즘으로 콘텐츠들의 이름을 비교하여 이름 간의 유사도를 판단한다. 가령 사용자가 요청한 콘텐츠의 이름이 “이승철 말리꽃”이고, 요청을 받은 라우터 내에 “이승철 말리꽃”과 “이승철 인연”, “이승환 왼손잡이”, “이선희 J에게”라는 콘텐츠들이 저장되어 있다고 하자. 알고리즘에 의해 Levenshtein distance 값이 “이승철 인연”은 2가 되고 “이승환 왼손잡이”는 5가 되고, “이선희 J에게”는 5가 된다. 그러므로 “이승철 인연”이 가장 유사한 콘텐츠라고 할 수 있다[4].

3.2 CST (Contents Smilarity Table)

유사한 콘텐츠 선별을 위해 라우터 내에 저장되어 있는 콘텐츠들을 요청된 콘텐츠와 Levenshtein distance 알고리즘을 적용한 값을 CST에 적용한다.

A's CST	LD
B	8
C	3
...	...
K	10
L	14

CST : Contents Smilarity Table
LD : Levenshtein Distance

그림 3. CST 구조 예시

그림 3에서는 A콘텐츠를 요청했을 때 생기는 A콘텐츠의 CST 테이블을 예시로 보여준다. LD는 Levenshtein distance 값이며 A콘텐츠와 각 콘텐츠들의 이름의 유사도를 측정한 값이다. 이 값이 작을수록 A콘텐츠와 유사한 콘텐츠이고 라우터는 사용자의 요청에 대해 A콘텐츠를 보내면서 이 값들을 계산한다. 값들이 모두 계산되고 Levenshtein distance 값이 제일 작은 콘텐츠를 중간 경로에 있는 라우터들로 포워딩한다.

3.3 포워딩 알고리즘

사용자가 A콘텐츠를 요청했다고 가정하자. 만약 종단 라우터가 A콘텐츠를 가지고 있지 않다면 상위라우터에게 A콘텐츠에 대한 요청을 보낸다. 요청은 받은 상위라우터가 A콘텐츠를 가지고 있다고 가정하자. 상위라우터는 저장하고 있는 콘텐츠들 중에서 A콘텐츠와 유사한 콘텐츠가 있는지 검사를 하고 A콘텐츠에 대한 CST(Contents Smilarity Table)을 만들어서 Levenshtein distance 알고리즘을 활용해 콘텐츠들 간에 유사도를 계산해 CST의 빈칸을 채운다. CST에서 가장 유사하다고 판정된 콘텐츠를 A콘텐츠와 함께 종단 라우터로 포워딩한다.

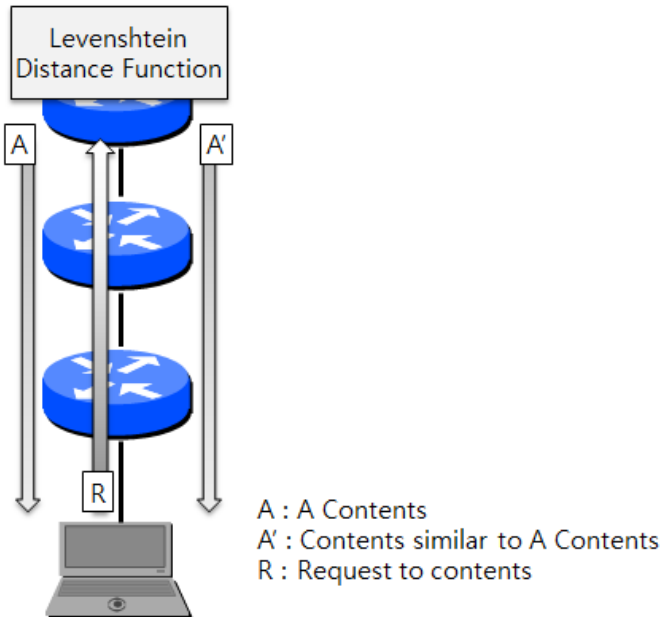

그림 4. 포워딩 방식

다음은 levenshtein distance 알고리즘을 이용해 가장 유사한 콘텐츠를 찾는 함수이다.

Algorithm 1 Levenshtein Distance Function

```
1: CST_init();
2: update_CST_list();
3: while(not-null next_contents) then
4:     LD_value = measuring_LD(next_contents);
5:     put_CST(next_contents, LD_value);
6: end while
7: return MIN_LD(CST);
```

위 알고리즘은 먼저 CST 테이블을 초기화하고, 새로 추가되거나 삭제된 콘텐츠에 대해 변경사항을 테이블에 적용하고 각 콘텐츠에 대해서 levenshtein distance를 계산해서 값을 채워 넣는다. 그리고 가장 작은 값을 가지는 콘텐츠를 반환 한다.

다음 알고리즘은 라우터에서 콘텐츠 요청을 받은 경우 라우터의 동작을 나타낸다.

Algorithm 2 Receive Request Operation

```
1:  receive_request(contents_name);
2:  if end_router() then
3:    if exist_contents(contents_name) then
4:      send_data(contents_name);
5:    end if
6:    else then
7:      forwarding_request(contents_name);
8:    end else
9:    send_similar_contents_request();
10: end if
11: else then
12:   if exist_contents(contents_name) then
13:     send_data(contents_name);
14:     similar_contents = levenshtein_distance_
          function(contents_name);
15:     send_data(similar_contents);
16:   end if
17:   else then
18:     forwarding_request(contents_name);
19:   end else
20: end else
21: close();
```

콘텐츠 요청을 받은 라우터는 종단 라우터냐 중간 라우터냐에 따라 동작이 달라진다. 종단 라우터인 경우는 요청 받은 콘텐츠가 존재하면 콘텐츠를 사용자에게 보내고 없으면 콘텐츠 요청을 상위 라우터에게 보낸다. 그리고 유사콘텐츠요청을 상위 라우터에게 보낸다.

종단 라우터가 아닌 경우 요청된 콘텐츠를 보유하고 있다면 그 콘텐츠를 보내주고 levenshtein distance 함수를 통해 유사한 콘텐츠를 선정 후 유사한 콘텐츠 또한 보내준다. 만약 콘텐츠를 보유하고 있지 않다면 상위 라우터에게 콘텐츠 요청을 보내게 된다.

4. 성능 평가

성능평가를 위해 클라이언트에서 서버사이에 5홉의 거리로만 구성된 네트워크가 있다고 가정하고 클라이언트에서 10개의 연관된 콘텐츠를 요청하면서 기존 CCN에서의 요청과 본 논문에서 제안하는 유사 콘텐츠 요청 기법을 비교한다.

그림 5는 콘텐츠를 요청할 때 마다 발생하는 요청메시지와 응답메시지의 수를 합해 카운트한 것이다. 기존 CCN에서는 유사한 콘텐츠를 요청해도 늘 새로운 콘텐츠이기 때문에 서버에서 새로 받아와야 하기 때문에 한 콘텐츠 당 5개의 요청과 5개의 응답이 생겨 10개의 메시지

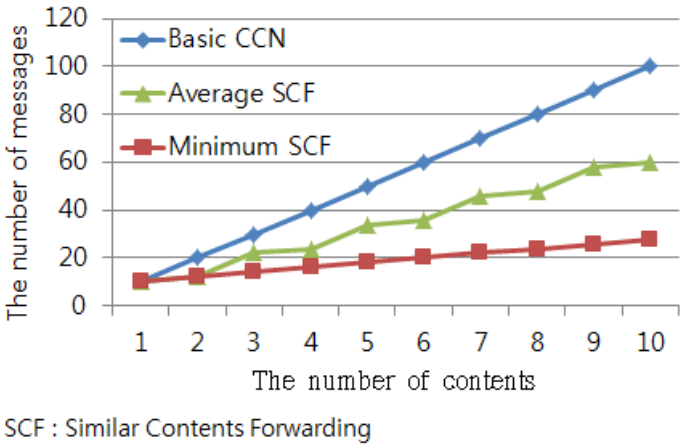

그림 5. 콘텐츠 당 발생 메시지 수

가 생성된다. 유사한 콘텐츠 10개를 요청하게 되면 총 100개의 메시지가 생성된다. SCF를 적용했을 때 최소의 경우는 처음 10개의 메시지가 필요하고 그 다음부터 종단에서 저장된 콘텐츠를 보내주기 때문에 콘텐츠 당 2개 메시지씩 필요해서 총 28개가 필요하고, 적합한 콘텐츠의 평균수 가 10개 중 5개의 콘텐츠라고 가정하면 5개 콘텐츠는 적합하지 않아 50개의 메시지가 발생하고 나머지 적합한 5개의 콘텐츠는 10개, 총 60개의 메시지가 발생하게 될 것이다.

5. 결론

본 논문에서는 유투브 같은 동영상 서비스에서 요청된 콘텐츠와 별개로 유사 콘텐츠를 멀티 포워딩하여 후에 요청되는 경우에 생기는 오버헤드를 줄이는 기법을 제안하였다. 성능 평가 결과로, SCF를 적용한 CCN이 그렇지 않은 CCN보다 메시지 수가 확연히 적다. 유사 콘텐츠의 적합도가 높을수록 메시지 수가 더 적어진다. 향후 계속되는 연구에서는 시뮬레이션을 통해 좀 더 정확한 성능평가를 보일 것이며 유사 콘텐츠의 적합도를 더 높이기 위한 연구를 할 것이다. 또 콘텐츠를 미리 저장하므로 라우터 캐쉬 용량 문제가 야기될 수 있으므로 SCF에 맞는 캐쉬 정책을 수립해야 할 것이다.

6. 참고 문헌

[1] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013-2018

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, R. L. Braynard, "Networking Named Content", CoNEXT 2009, Rome, Dec, 2009.

[3] 박수연, "검색 포털들의 검색어 추천 서비스 분석 평가: 네이버와 구글의 연관 검색어 서비스를 중심으로", 한국정보관리학회, 정보관리학회지 30(2), 2013.6, 297-315 (19 pages)

[4] Levenshtein distance [Online]. Available: http://en.wikipedia.org/wiki/Levenshtein_distance