대형 멀티미디어 파일의 익명성 지원을 위한 수정 GNUnet

이명훈*, 박병연**, 조인준*

*배재대학교, **한국과학기술정보연구원

GNUnet improvenemt for anonymity supporing in large multimedia file

Myoung-hoon Lee, Byung-yeon Park, In-June Jo,

*Dept. of Computer Engineering, Paichai Univisity. **KISTI

e-mail : lopez@mail.pcu.ac.kr, bypark@kisti.re.kr, injune@mail.pcu.ac.kr

요 약

GNUnet은 파일의 익명성 지원을 위해 파일을 1KByte 블록코기로 분리하는 인코딩 방법, 인코딩된 블록을 비구조적 분산기법을 통해 피어들에게 분산시키는 방법, 분산된 블록을 검색하여 원본 데이터로 복구하는 방법을 제안하였다. 하지만 인코딩 및 블록 분산 방안은 약 600~700MB의 대용량 멀티미디어 파일을 처리할 경우 첫째, 추가적으로 R블럭과 I블럭을 과도하게 요구하게 되어 약 4%의 저장공간을 낭비하는 문제를 지니게 되었다. 둘째, 비구조적 브로드캐스팅 분산방에 의하여 네트워크 부하가 과도한 단점을 내포하였다. 셋째, 이용자의 데이터 검색방법에 있어 파일명을 키워드로 생성하는 방법을 제안했기 때문에 키워드 검색의 한계점이 들어났다.

본 논문에서는 이러한 문제점 해결을 위하여 인코딩 블록의 크기를 가변적으로 결정하는 방법과 구조적 위상을 통해 블록을 분산시키는 방안 그리고 데이터 컨텐츠를 이용한 키워드 생성방안을 제안하였다. 따라서 제안 인코딩 방법은 기존 4%에 해당되는 추가 블록생성을 1% 이내로 최소화하였고, 다면체의 구조적 위상에 인코딩된 블록을 전송하기 때문에 네트워크 부하를 감소시켰다. 그리고, 데이터의 컨텐츠를 키워드로 작성한 컨텐츠 기반 키워드 추출방법을 제안하였다.

ABSTRACT

The GNUnet proposed a file encoding method by 1KB block size to support anonimity of files, decentralizes encoded block to peers through unstructed mode and original data decoding method a block searching of encoded blocks. but, the encoding and block decentralizing method with 600~700MB large multimedia file appered two problems. First problem, it need addition R block and I block, which make about 4% of storage resource. Second problem, unstructed model added network load by broadcasting decentralizing method. Third probloem, The critical point of keyword search function.

This paper suggest variable encoding block size and structured model by block decentralizing solution. Suggested encoding method reduced block request supplementary block generation from 4% to 1%, network load by proposal structured model sending answer through dedicated peer to decentralize block and we defined content-based keyword and identifier of sharing file.

키워드

P2P, 익명성, GNUnet, CAN, 위상

I. 서론

음반 데이터 공유를 위하여 설계도니 냅스터는 저작권 보호법에 의하여 서비스 중지 판정을 받게 된다. 따라서 자유로운 데이터 공유를 위한 익명성 지원방법이 제안이 되었고, 가장 연구가 활발하게 진행되는 것이 GNUnet이다. GNUnet은 자유로운 정보공유를 서비스하기 위하여 익명성 지원방법을 제안하였다. 익명성이란 파일을 공유하는 출판자와 파일을 저장하는 저장자, 파일을 요구하는 요구자에 대한 비밀을 보장하는 것이다.

GNUnet은 익명성 제공을 위하여 3가지 관점에서 접근을 시도하였다. 첫째, 파일명 및 파일내용의 익명성 지원을 위하여 해쉬와 암호화기법을 적용한 블록을 생성한다. 둘째, 저장자의 익명성 지원을 위하여 생성된 블록은 주변의 다른 피어에게 블록 저장을 요구한다. 세 번째, 검색의 편리성을 제공하기 위하여 파일명을 키워드화한 루트블록을 생성한다. 결론적으로 GNUnet은 익명성 제공을 위하여 파일 인코딩 방법과 인코딩된 블록의 전송방법 그리고 블록 검색방법을 제안하였다. 그러나 GNUnet에서 제안하고 있는 인코딩 방법은 대용량의 멀티미디어 파일에는 비합리적이고, 인코딩 블록을 비구조적인 방식으로 분산하기 때문에 과도한 네트워크 트래픽의 원인이 된다. 또한 키워드 검색방법은 동일한 파일명으로 질의를 해야만 파일을 받을 수 있었다.

본 논문은 GNUnet의 불필요한 저장공간 및 과도한 네트워크 트래픽 그리고 블록 검색의 어려움등을 해결하기 위하여 인코딩 블록 크기를 "$2^{10} \leq 2^n \leq 2^{20}$" 범위에서 결정하는 방법을 제안하였고, 블록 분산방법으로 CAN에서 제안한 다면체형태의 구조적 모델을 새롭게 적용하였다. 그리고 컨텐츠 기반 키워드 추출 및 파일 식별자를 생성하는 키워드 검색 지원 방법을 제안하였다. 이를 위하여 II장에서는 GNUnet의 인코딩 방법 및 파일 분산방법 그리고 구조적 위상을 제안한 CAN과 검색을 위한 역분산 해쉬 테이블에 대하여 설명하였다. III장에서는 이 논문에서 제시하는 성능개선 방법을 설명한다. IV장에서는 제안 프로토콜을 검토 및 고찰하였고, 마지막으로 결론을 맺었다.

II. 관련 연구

1. GNUnet 파일 인코딩 방법[1][2]

GNUnet[1][2]은 익명성 지원을 위하여 파일을 1Kbyte의 고정길이 블록으로 분리하고, 분리된 블록을 분산하는 방법을 제안하였다. 제안된 인코딩 방법은 D(data)블록, I(Indirection)블록, R(root)블록의 3가지 블록을 생성하고, 생성된 블록은 네트워크의 다른 피어에게 전송된다. D블록은 다른 피어에게 분산되기 때문에 D블록을 지시하기 위한 I블록을 추가적으로 생성되어야 하고, 이용자 검색의 편리성 제공을 위하여 R블록을 제공한다. 그러나, 원본 파일의 추가 블록인 I, R은 4%에 해당되는 추가적인 저장공간을 요구한다.

2. GNUnet 인코딩 파일분산 및 탐색방법[4]

인코딩된 블록은 출판자의 익명성 제공을 위하여 다른 피어에게 블록을 저장한다. GNUnet에서 제안한 파일 분산 방법은 피어의 라우팅 테이블을 기준으로 랜덤하게 2개의 피어를 선택하여 블록을 전송하고, 참고문헌[6]의 방식을 적용하여 TTL 값을 생성한다. GNUnet은 TTL값이 커질수록 저장되는 피어의 수가 2^n으로 증가된다. 이러한 비 구조적방법의 블록 분배방법은 블록 저장에 있어 특정한 규칙이 없기 때문에 브로드캐스트방법에 의하여 블록을 탐색하여야 하고, 네트워크의 트래픽을 증가시킨다.

3. CAN 파일 정보 분산 및 탐색 방법[5]

Chord 시스템은 P2P 응용을 위한 분산처리 자원 탐색 프로토콜이다. CAN 시스템은 피어의 위치를 다면체의 구조적 모델로 구성하고, 피어 식별자를 중심으로 파일정보를 제공한다. 이러한 서비스를 제공하기 위하여 피어식별자와 파일정보는 SHA-1 알고리즘과 같은 해쉬함수를 이용하여 해쉬되고, 파일정보의 해쉬 값을 피어 식별자와 비교하여 근접한 피어를 결정한다(파일정보 ≤ 피어식별자). 그러나 새로운 피어의 추가 및 탈퇴의 경우 피어어의 정보를 유지하기 위하여 모든 피어에게 추가 및 탈퇴 메시지를 전송한다. 시험결과 피어가 10000대를 넘을 경우 발생하는 네트워크 트래픽에 의하여 시스템은 재성능을 발휘하지 못하였다.

4. 역분산 해쉬 테이블[6]

역분산 해쉬 테이블 기반의 P2P 시스템은 공유파일 데이터는 {keyword, list of values} 형태로 표현된다. 'keyword'는 공유 파일을 지칭하는 키워드들로 구성된다. 'list of values' 는 키워드에 해당하는 공유 파일을 실제 소유하고 있는 피어의 위치정보이다. 그러나 역분산 해쉬테이블은 공통키워드를 처리할 경우 많은 수의 키워드를 생성하기 때문에 저장공간의 낭비가 생겼고, 다중 키워드 질의에 따른 네트워크 트래픽 증가에 대한 문제점이 있다.

III. 제안 시스템 구조

GNUnet은 파일의 인코딩 과정에서 1KByte라는 단위로 블록을 분리한다. 이 블록은 저용량의 파일의 경우 문제가 없지만 대용량 멀티미디어 파일의 경우 I블록과 R블록 생성에 따른 저장 공간의 낭비가 발생한다. 동영상 멀티미디어 파일의 기본 크기인 700Mbyte를 예로 들어 보겠다. 만약

700Mbyte 파일 저장을 요구할 경우 716,800개의 D블록으로 파일을 분리한다. 그리고, D블록을 간접적으로 지시하는 I블록은 29868개가 생성된다. 이와 같이 분리된 블록은 비구조적 위상에서 2개의 이웃 피어를 선택하여 블록을 전송하고, TTL 값에 의하여 2n개의 피어에게 전송한다. 따라서 I, R블록 및 비구조적 방법의 파일 분리 방법에 따른 추가된 저장공간이 요구되었고, 비구조적 방법으로 분산된 블록의 탐색에서 브로드캐스트방법을 사용하기 때문에 과도한 네트워크 트래픽이 발생하였다. 이와 같이 분산된 블록의 검색은 키워드에 의하여 가능하였다. 그러나 파일명을 키워드화 하기 때문에 다양한 검색이 불가능한 문제점이 있었다.

본 논문에서는 저장 공간의 낭비와 트래픽 발생 그리고 검색의 효율성을 제공하기 위하여 새로운 인코딩 방법과 인코딩된 블록을 구조적 위상으로 구성된 피어에게 분산하는 방법 그리고 컨텐츠 기반의 키워드 추출방안을 제안하였다.

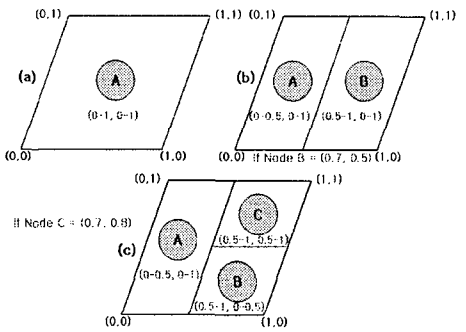
3.1. 확장 CAN 오버레이 네트워크 위상

인코딩된 블록의 분산 및 분산된 블록의 검색을 위하여 CAN 메커니즘의 개념을 도입하였다. CAN 메커니즘의 오버레이 네트워크는 그림 1과 같이 {x, y}의 공간 좌표를 갖는 기본 CAN 메커니즘의 2-차원 다면체 공간을 활용한다.

각 피어는 랜덤 함수를 통해 생성한 노드 식별자, 이웃 피어의 관리 공간영역 및 라우팅 테이블, 키워드 검색을 위한 KID와 CKD를 소유하고 있다.

새로운 피어의 네트워크 가입은 PLS의 접속을 통해 이미 네트워크에 가입되어 있는 활성 피어의 IP주소를 얻게 되며 활성피어와 피어삽입 메커니즘을 수행한다.

확장 CAN에서 피어 및 데이터는 두개의 해쉬 함수를 이용하여 키{x, y} 형태로 표현되고, 각각의 피어는 임의의 한 점 {x, y}을 선정하여 그 점을 담당하고 있는 피어의 담당 구역을 반으로 나누어 담당하게 된다.



[그림 1] 확장 CAN 오버레이 네트워크 위상

그림 1.(b)는 두개의 피어로 구성된 확장 CAN 오버레이 네트워크의 구성을 보이고 있다. 만약 그림 1.(b)와 같이 구성된 네트워크에서 새로운 피어 C가 네트워크에 참여하고자 자신의 노드 식별자를 두개의 해쉬 함수를 사용하여 공간좌표 {0.7, 0.8}를 생성했다면, 피어 B는 이전의 자신의 담당구역 {0.5-1, 0-1}을 반으로 나누어 피어 C에게는 {0.5-1, 0.5-1}를 담당하게 하고 피어 B는 {0.5-1, 0-0.5} 공간 영역을 담당한다.

3.2. 제안 시스템 인코딩 방법

제안 인코딩 방법은 추가적으로 생성되는 I, R 블록의 추가되는 저장공간을 최소화하기 위하여 블록의 크기를 기존 1Kbyte에서 "1Kbyte ≦ 2m ≦ 1Mbyte(210 ≦ 2m ≦ 220)"의 크기로 제안하였다. 네트워크에서 전송되는 파일의 크기는 1Kbyte를 기준으로 전송되기 때문에 단편화 최적화를 위한 2m으로 결정하였고, 최대 1Mbyte의 크기는 사용자가 공유하는 저장 공간의 크기를 고려하여 결정하였다.

[표 1] 용어 설명

기호	설명
F	원본 파일의 크기
BX	블록 크기와 가장 유사한 값
BC	결정된 인코딩 블록 크기
Bn	BC의 크기로 분리된 블록
n, m	대입되는 순차 값
P(XX)	피어를 식별하기 위한 피어 식별자
B(XX)	파일을 해쉬한 블록의 해쉬 값

3.1.1 인코딩 블록 크기 결정

GNUnet의 인코딩 방법은 1Kbyte의 고정길이로 인코딩 블록 생성 방법을 제안하였다. 그러나 1Kbyte의 블록은 4%의 추가 블록을 생성하였고, 추가적인 저장공간을 요구하였다. 따라서 제안 시스템에서는 인코딩 블록의 크기를 1Kbyte ≦ 2^m ≦ 1Mbyte의 범위에서 결정하였고, 결과적으로 1% 이내의 추가블록을 생성하였다. 다음은 인코딩 블록의 크기 결정 방법이다.

step1) GNUnet에서 제안된 I블록은 25개의 블록을 지시한다. 따라서 식 (1)과 같이 파일(F)을 25^m으로 나누고, 나눈 값이 "1Kbyte ≦ BX ≦ 1Mbyte" 의 범위에 결과 값을 구한다.

$$BX = F \div \sum_{n=1}^{25^n} \quad (1)$$

step2) 제안 시스템은 단편화 최적화를 위하여 2^m의 크기로 구성된다. step1에서 결정된 블

록의 크기는 2^m의 결과 값이 아니기 때문에 식 (2)의 조건을 만족하는 BC의 값을 구한다. BC의 범위는 1Kbyte ≦ BC ≦ 1Mbyte이다.

$$BC = BX \leq \sum_{m=1}^{2^m} \quad (2)$$

step3) 결정된 인코딩 블록의 크기 BC에 따라 파일을 분리한다.

$$F = \{ B_1, B_2, \cdots B_n \} \quad (3)$$

3.1.2 D블록 생성

결정된 BC의 크기에 따라 분리된 블록은 저장자의 익명성 보호를 위하여 블록의 데이터는 암호화하고, 파일명은 160비트 RIPE-MD로 해쉬한다. 저장자의 익명성 제공을 위하여 블록의 데이터는 암호화하고, 파일명은 160bit로 해쉬한 값이다. 다음은 D블록 생성 방법이다.

step1) 파일이 분리된 각 블록 B_i를 160비트 RIPE-MD[8]로 해쉬한다.
$\{ H(B_1), H(B_2), \cdots H(B_n) \}$ (4)

step2) 해쉬 값을 키로 하여 분리된 블록의 데이터를 암호화한다.
$E_{H(D1)}(B_1), E_{H(D2)}(B_2) \cdots E_{H(Dn)}(B_n)$ (5)

step3) 생성된 블록을 160비트 RIPE-MD로 해쉬하여 D블록의 파일명을 결정한다.
$H(E_{H(D1)}(D_1)), \cdots H(E_{H(Dn)}(Dn))$ (6)

3.1.3 I블록 생성

생성된 D블록은 다른 피어에게 분산되어 저장된다. 따라서 분산된 블록을 지시하기 위한 식별자가 필요하다. I블록은 분산된 D블록을 지시하는 블록이다. 다음은 I블록 생성 방법이다.

step1) 각 D블록에 대하여 식 (4)의 평문 해쉬 값과 식 (6)의 암호화된 해쉬값을 저장한다. 총 25개의 D블록의 정보하고, 이 정보는 D블록의 파일명과 암호키 값을 의미한다. 다음은 생성된 I블록의 예이다.
$(H(D_1),H(E_{H(D1)}(D_1))), \cdots (H(D_n),H(E_{H(Dn)}(D_n)))$ (7)

step2) CRC32 값을 계산한다. (4Byte)

step3) 생성하고자 하는 I블록과 연결된 모든 해쉬 값을 160비트로 해쉬한 20바이트의 슈퍼해쉬 값을 계산한다.

step4) D블록 또는 I블록을 지시하는 I블록을 생성한다.
$(H(D_1),H(E_{H(D1)}(D_1))),(H(D_{25}),H(E_{H(D25)}(D_{25}))(1000) + ,CRC(4) +$ 슈퍼해쉬$(20)4$ (8)

step5) I블록을 160bit RIPE-MD로 해쉬하여 파일명을 생성한다.

3.1.4 R블록 생성

D블록과 I블록은 파일을 저장하기 위하여 생성된 블록이다. 따라서 사용자 관점에서는 키워드에 의한 검색을 요구한다. R블록은 사용자의 키워드 검색을 제공하기 위하여 구성된 블록이다.

step1) 기밀 키워드 K_i를 선택하여 160비트 RIPE-MD로 해쉬하여 다음의 해쉬값을 계산한다.
$H(K_i), H(H(K_i))$ (9)

step2) H(Ki)의 값을 키로 하여 루트 I블록의 파일명을 암호화한다.
$E_{H(Ki)}(H(R))$ (10)

step3) $H(H(K_i))$을 파일명으로 하고 루트 I블록의 파일명을 암호화한 최종 R블록을 생성한다.

3.1.5 저장된 블록 탐색 방법

제안한 인코딩 블록 방법에 따라 파일은 분리되고, 분리된 파일은 다른 피어에게 분산된다. 분산된 블록을 탐색하기 위해서는 다음과 같은 절차를 수행한다.

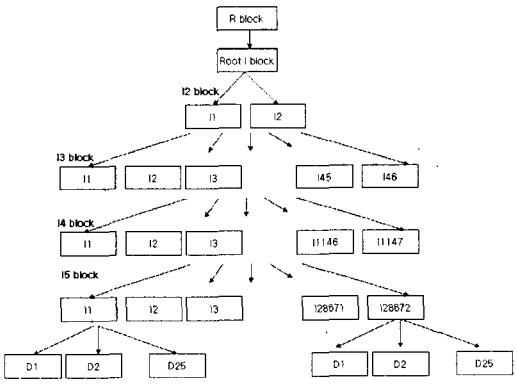
step1) 이용자는 원하는 파일의 키워드를 입력하고, 160bit RIPE-MD로 해쉬한다. 해쉬 값은 식 (9)와 같다.

step2) 파일명이 H(H(Ki))의 값이 데이터를 탐색하여 전송받고, 전송받은 데이터를 H(Ki)를 키로하여 복호화한다.

step3) 복호화한 결과 루트 I블록은 I블록 또는 D블록의 키와 블록의 해쉬값을 지시하고 있고, 지시하고 있는 해쉬 값을 탐색하여 전송받는다. 전송받은 블록은 키에 의하여 복호화하고, 이 과정을 D블록을 모두 받을때까지 진행한다.

step4) D블록을 모두 전송받아 복호화한 다음 블록을로 순서대로 연결하여 최종 파일을 생성한다.

제안 인코딩 방법에 의하여 블리된 블록의 구조는 그림 2의 트리구조로 구성된다.



[그림 5] 제안 시스템 인코딩된 블록

3.2 PLS

PLS는 P2P 오버레이 네트워크에 참여하고 있는 활성 피어들에 관한 정보를 담고 있는 초기 접속 서버이다. 중앙집중식 P2P에서의 중앙서버는 전체 네트워크를 구성하고 피어의 접속 목록, 피어의 공유파일 목록 관리, 질의 처리 등에 직접적인 간섭을 한다. 반면 비중앙집중식 P2P의 PLS는 현재 네트워크에 참여하고 있는 몇 개의 활성 피어의 IP주소 목록만을 유지하여 새로운 피어가 네트워크에 접속을 원할 때 IP주소 목록을 전송하여 네트워크 참여를 돕는다.

컨텐츠 기반 확장 CAN에서는 키워드 검색 지원을 위해 PLS의 기능을 다음과 같이 정의하였다.

3.2.1. 활성 피어 접속 목록 유지

PLS는 새로운 피어의 참여를 위해 현재 네트워크에 참여하고 있는 몇 개의 활성 피어의 IP주소를 유지한다. 이때 피어의 IP주소 목록은 KID와 CKD를 갱신하는 피어 중 컴퓨팅 능력이 높은 피어로 선정하여 IP주소 목록을 유지한다.

3.2.2. KID 생성 및 배포

KID는 키워드 인덱스 사전으로 피어들이 공유하는 파일들의 키워드와 파일 식별자를 맵핑한 {file-id, list of keyword} 형태의 인덱스 사전이다. KID는 피어가 공유 파일을 검색하기 위해 키워드 형태의 질의문을 작성했을 때 키워드들을 KID에서 검색하여 파일 식별자를 찾는 역할을 한다.

KID 생성 및 갱신, 배포는 PLS에서 담당하며 그 절차는 다음과 같다.

Step1) 피어는 파일 공유를 위해 컨텐츠 기반의 KI(Keyword Index)를 {file-id, list of keyword} 형태로 생성한다. 키워드 KI의 생성과정은 3.3 절차에 따른다.

Step2) 피어가 PLS 에 최초 접속 했을 때 자신이 소유한 KI를 전송한다. PLS는 KID와 비교하여 피어가 전송한 KI 목록 중 새로운 파일 식별자가 발견되면 KID 해당 목록을 추가한다.

Step3) PLS는 KI를 전송한 피어에게 갱신된 KID를 전송한다. KID는 CKD와 함께 피어의 키워드 검색 시 파일 식별자를 찾는 역할을 한다. 또한 피어의 KID 갱신은 피어의 KI가 수정되거나 일정한 시간 간격에 따라 PLS와 주기적으로 이루어진다.

3.2.3. CKD 생성 및 배포

CKD는 공통 키워드 사전으로 키워드 검색 시 공통키워드 처리를 위해 사용된다. CKD는 KID를 기반으로 {Common keyword, Hit} 형태로 생성된다. 'Common keyword'는 공통 키워드, 'Hit'는 KID를 기반으로 한 빈도수를 의미한다.

CKD 생성 및 갱신 및 배포는 PLS에서 담당하며 그 절차는 다음과 같다.

Step1) 피어에 의해 PLS의 KID가 갱신 되었을 때 KID에 갱신된 각각의 키워드를 CKD의 'Common keyword' 에서 검색한다.

Step2) 만약 이전에 존재하는 키워드일 경우 CKD의 'Hit'를 1증가 시키고, 존재하지 않는다면 CKD에 키워드를 등록한다. 이렇게 CKD에 등록된 키워드 중 'Hit'수가 높은 키워드는 공통 키워드로 선정이 된다.

Step3) CKD의 배포는 KID의 배포와 마찬가지로 피어의 초기 접속 또는 일정한 시간 간격에 따라 PLS와 주기적으로 이루어진다.

3.3 인코딩된 블록 분산 방법

출판자의 익명성 제공을 위하여 인코딩된 블록은 다른 피어에게 전송한다. GNUnet 시스템은 비구조적 방법으로 전송하기 때문에 블록 저장소는 2^n으로 증가되고, 저장된 블록을 탐색하기 위하여 브로드캐스트 방식을 사용한다. 그러나 브로드캐스트방식은 이웃한 모든 피어에게 트래픽을 발생시키기 때문에 네트워크를 매우 혼잡하게 만드는 결과가 발생된다. 따라서 제안 시스템은 다면체의 구조적 모델을 도입하여 블록의 분산 및 탐색에서 유니캐스트 방법으로 질의하기 때문에 네트워크의 트래픽을 최소화하는 방안을 제안하였다.

3.3.1. 피어의 참여

컨텐츠 기반 CAN 메커니즘의 오버레이 네트워크는 2-차원의 다면체 공간에 피어들이 구성되며 그 절차는 다음과 같다.

Step1) CAN 오버레이 네트워크에 참여를 원하는 피어는 랜덤 함수를 사용하여 'node-id'를 생성하고 3.3의 절차에 따라 KI를 생성한다.

Step2) 새로운 피어는 네트워크에 참여를 시도하기 위해 현재의 활성 피어들에 관한 정보를 담고 있는 PLS에 연결을 요청한다.

Step3) 피어의 연결을 수락한 PLS는 피어에게 노드 식별자와 KI를 요구한다. 이는 기존의 KID, CKD에 현재 접속을 요구하는 피어의 정보를 갱신하기 위함이다.

Step4) 피어는 PLS의 요구에 따라 'node-id' 와 KI를 전송하며 추가적으로 노드 식별자를 두개의 해쉬 함수를 사용하여 공간좌표 노드 식별자{x, y}를 생성하여 전송한다.

Step5) PLS는 새로운 피어의 네트워크 참여를 위해 활성피어 중 '노드 식별자' 기반의 공간좌표 노드 식별자{x, y}와 가장 유사한 피어를 선정하여 IP주소를 피어에게 전송한다. 이는 피어의 초기 접속 시 공간할당을 위한 라우팅 수를 줄이기 위함이다. 또한 전송받은 KI를 기반으로 3.2의 KID 및 CKD 의 메커니즘에 따라 각 키워드 사전들의 갱신을 수행한 후 피어에게 전송한다.

Step6) 새로운 피어는 PLS에서 전송받은 정보를 바탕으로 활성 피어의 IP주소로 노드 식별자, 공간좌표 노드 식별자{x, y}를 전송하여 네트워크 참여를 요구한다.

Step7) 새로운 피어로부터 네트워크 참여를 요청받은 활성 피어가 공간좌표 노드 식별자{x, y}를 관리하고 있다면, 자신의 공간 영역의 반을 나누어 새로운 피어에게 담당하게 한다. 그렇지 않을 경우 공간좌표 노드 식별자{x, y}를 관리하는 피어를 3.4 라우팅 메커니즘에 네트워크 참여 요구 메시지를 라우팅 한다.

Step8) 새로운 피어에게 공간영역을 할당해준 활성 피어는 기존의 라우팅 테이블과 이웃 피어의 존 정보에 새로운 피어가 포함되도록 수정하고, 이를 새로운 피어와 이웃 피어들에게 갱신하도록 요구한다.

3.6. 공유데이터 삽입

컨텐츠 기반 CAN 메커니즘에서의 공유 파일 삽입은 피어의 초기 네트워크 접속 시 또는 공유 파일의 갱신이 발생했을 때 수행되며 절차는 다음과 같다.

Step1) 피어가 공유 파일을 오버레이 네트워크에 삽입하기 위해 공유파일의 파일 식별자를 생성한다. 컨텐츠 기반 CAN 메커니즘에서는 파일 식별자가 공유 파일의 컨텐츠 기반 키워드의 조합으로 이루어지며 해쉬된 키워드 값은 피어의 키워드 검색에 활용된다. 따라서 3.3 컨텐츠 기반의 파일 식별자 생성 절차에 KI를 생성한다.

Step2) 피어는 PLS 접속하여 KI를 전송한다. 3.2의 절차에 따라 피어의 KI를 기반으로 PLS의 KID와 CKD 갱신하고 이를 피어에게 전송한다.

Step3) 오버레이 네트워크의 파일 식별자를 담당하는 피어에게 {file-id, value}를 삽입한다. 컨텐츠 기반 CAN 메커니즘은 분산 해쉬 테이블 기반의 2차원 다면체 구조를 갖는다. 따라서 파일 식별자를 두개의 해쉬 함수를 사용하여 공유파일의 키를 생성하고, 키를 담당하는 피어에게 {key, value}쌍을 삽입한다. 공유 파일의 {key, value} 삽입과정은 키를 기반으로 3.4의 라우팅 메커니즘을 수행한다.

3.4. 컨텐츠기반 키워드 및 파일 식별자 생성

본 절에서는 공유파일에 대한 명확한 키워드 선정을 위해 컨텐츠 내용 기반의 키워드를 선정하고, 키워드의 조합을 통한 파일 식별자를 생성한다. 이를 통해 각 피어는 KI를 생성하여 파일 식별자는 네트워크의 피어들에게 분산하며 KI는 PLS에 전송하여 KID와 CKD의 생성을 돕는다. 이를 위해 본 논문에서는 오디오 파일을 대상으로 하였으며 ID3 태그 정보를 활용하였다.

[표 2] ID3 태그 헤더

위 치 (Bytes)	길 이 (Bytes)	설 명
00 ~ 02	03	TAG ID
03 ~ 32	30	Title(노래 제목)
33 ~ 62	30	Artist(연주자,가수)
63 ~ 92	30	Album(앨범명)
93 ~ 96	04	Year(발매년도)
97 ~ 126	30	Comment
127	01	Genre(장르)

ID3는 MPEG Layer 3(MP3) 오디오 파일에서 오디오 파일의 내용(곡명이나 저작자, 음악 채널 등)을 확인하기 위한 정보를 담고 있는 표준으로 사용되고 있는 태그 형식이다. ID3v1 태그는 MP3파일의 앞부분에 128bytes로 표 2과 같이 구성되어 있다.

오디오 파일의 키워드 및 파일 식별자 생성은 ID3v1 태그를 이용하며 KI 생성 절차는 다음과 같다.

Step1) 오디오 파일의 키워드는 ID3v1 태그를 활용하여 Title, Artist 정보를 추출한다.

Step2) 추출된 Title, Artist 정보를 공백단위로 분리한다. 이는 공백 단위의 분리된 정보들을 키워드로 사용하기 위함이며 본 논문에서는 Title의 공백은 10, Artist의 공백단위는 5로 제안하였다.

Step3) 공백단위로 분리된 키워드를 SHA-1 해쉬 함수를 사용하여 해쉬된 키워드 값으로 생성한다.

keyword1 = SHA-1(Title1)

keyword2 = SHA-1(Title2)

.....

keyword10 = SHA-1(Title10)

keyword11 = SHA-1(Artist1)

.....

keyword15 = SHA-1(Artist5)

list of keyword = (keyword1,

keyword2,

....., keyword15)

Step4) 세 번째 단계에서 생성된 'list of keyword'를 조합하여 해쉬된 'file-id'를 생성한다.

$$\text{file-id} = \text{SHA-1}\{\sum_{n=1}^{10}\text{SHA-1(Titlen)} + \sum_{n=1}^{5}\text{SHA-1(Artistn)}\}$$

Step5) 상기의 단계를 통해 생성된 'list of keyword'와 'file-id'를 바탕으로 표 3와 같이 KI를 생성한다.

[표 3] 피어의 키워드 인덱스

file-id	list of keywords
file-id1	keyword1, ··· keyword15
:	:
file-idn	keyword1, ··· keyword15

컨텐츠 기반의 키워드 생성방법의 가장 큰 목적은 키워드 선정 방법을 명확히 하고 공통키워드수를 줄이기 위함이다. 기존의 키워드 선정 방법은 사용자의 임의대로 공유 파일에 대한 키워드들을 선정하기 때문에 동일한 공유 파일임에도 불구하고 키워드들이 서로 일치하지 않고, 공통키워드가 다수 발생하는 문제가 있다. 본 논문에서 제안하는 컨텐츠 기반의 키워드 생성방법은 공유파일 컨텐츠의 메타정보에 따라 키워드가 생성되기 때문에 동일한 파일은 항상 같은 키워드들로 구성이 되어 키워드 선정의 명확성과 공통키워드 수의 감소를 갖게 된다.

3.7. 공유데이터 검색

컨텐츠 기반 CAN 메커니즘에서의 공유 파일의 검색은 키워드를 이용한 검색방법과 파일 식별자의 정확한 일치를 통한 검색방법으로 나누어진다.

3.7.1. 다중 키워드 검색

컨텐츠 기반 CAN 메커니즘에서는 공유 파일에 대해 다중 키워드검색이 가능하다. 다중 키워드 검색방법은 피어가 PLS로부터 전송받은 KID와 CKD를 이용하며 다중키워드 검색 메커니즘의 절차는 다음과 같다.

Step1) 사용자는 요청하고자 하는 공유파일을 유추할 수 있는 키워드들을 선정한다.

Step2) 선정된 키워드들을 불리언 연산자가 포함된 검색 메시지를 작성하여 질의를 요청한다. 불리언 연산자란 사용자가 검색어간의 관계성을 정의하는데 사용되며 본 제안 방안에서는 OR, AND, NOT의 세 가지 연산자가 사용된다.

Step3) 불리언 연산자가 포함된 검색메시지 질의가 요청되면 사용자의 요청과 가장 유사한 파일 식별자 목록을 추출하기위해 다음의 네 단계를 수행한다.

Step3.1) 다중키워드 검색 질의문에 포함된 각각의 키워드를 불리언 연산자 단위로 분리하고 각각을 SHA-1 해쉬 함수를 사용하여 해쉬한다. 그리고 불리언 연산자가 포함된 해쉬된 다중 키워드 형태의 검색질의를 생성한다.

Step3.2) 각각의 해쉬된 형태의 다중 키워드 중 CKD를 검색하여 공통키워드를 추출하고, 공통키워드를 제외한 형태로 검색질의를 재 작성한다. 공통키워드를 제외한 질의문의 작성은 키워드를 기반으로 KID에서 파일 식별자를 추출 할 때의 처리수를 줄이기 위함이다.

Step3.3) 재 작성된 검색질의를 바탕으로 불리언 연산자 룰에 따라 KID를 검색하여 일치하는 {file-id, list of keyword}들을 추출한다.

Step3.4) 최종적으로 제외 되었던 공통키워드를 포함하여 KID를 통해 추출되었던 {file-id, list of keyword}와 비교하여 최종 파일 식별자 목록을 추출한다.

Step4) 상기의 절차를 통해 추출한 공유 파일의 파일 식별자를 두개의 해쉬 함수를 사용하여 공유파일의 키를 생성한다. 그리고 공유 파일의 키를 기반으로 3.4의 라우팅 메커니즘을 수행하여 해당키를 담당하는 피어에게 검색 질의문을 라우팅한다.

Step5) 검색 질의문을 전송 받은 담당 피어는 공유 파일 해쉬 테이블의 {key, value}에서 키에 해당하는 실제 파일을 소유한 피어의 'value'(IP주소)를 전송한다.

Step6) 검색 요청피어는 실제 파일을 소유한 피어에게 파일의 송신을 요청한다. 파일의 송수신은 HTTP 프로토콜을 이용하여 1:1로 수행된다.

3.7.2. 파일 식별자를 통한 검색

파일 식별자를 통한 검색은 사용자가 요청하고자 하는 공유 파일의 파일 식별자를 정확히 알고 있거나 완전한 키워드 목록을 알고 있을 경우 사용되며 검색 메커니즘의 절차는 다음과 같다.

Step1) 사용자가 요청하고자하는 공유파일의 완전한 키워드 목록을 알고 있을 경우 더블쿼테이션마크(" ")안에 키워드를 나열하여 검색 메시지를 작성한다. 만약 요청하고자하는 공유 파일의 파일 식별자를 알고 있다면 쿼테이션마크(' ')안에 파일 식별자를 입력하여 검색 메시지를 작성하여 질의를 요청한다.

Step2) 더블쿼테이션마크를 사용한 검색 메시지는 완전한 키워드 목록을 알고 있은 경우라 판단하여 3.3 컨텐츠 기반 파일 식별자 생성 메커니즘에 따라 각각의 키워드들을 SHA-1 해쉬 함수를 사용하여 이를 조합한 파일 식별자를 생성한다. 쿼테이션마크를 사용한 검색 메시지는 파일 식별자를 이미 알고 있으므로 두 번째 단계는 생략한다.

Step3) 상기 절차를 통해 획득한 파일 식별자를 기반으로 3.7.1의 다중키워드 검색의 Step4 절차 이하를 수행한다.

Ⅳ. 제안 프로토콜 검토 및 고찰

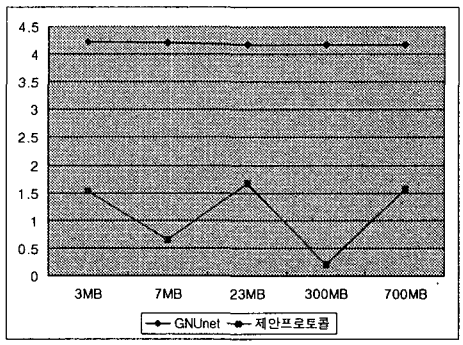
제안한 알고리즘의 타당성 검토를 위하여 멀티미디어 데이터의 용량별 인코딩 블록 수와 분리된 블록의 분산과 탐색에 발생되는 트래픽량에 대하여 검토 및 고찰하였다. 제안 시스템은 파일을 인코딩하여 블록을 분리하고, 분리된 블록은 다른 피어에게 전송한다. 따라서 블록 분산방법은 시스템에 큰 영향을 제공한다. 이를 위하여 저장공간의 낭비를 줄일 수 있는 최적의 인코딩 방법과 효율적으로 블록을 분산하고, 분산된 블록을 하나의 파일로 처리하기 위하여 링형의 구조적 위상을 제안하였다.

제안 프로토콜은 3가지 응용분야 문제점을 해결에 대한 검토 및 고찰이다. 즉 I, R블록에 의한 부가적인 저장공간의 낭비의 최소화와 인코딩된 블록의 저장 및 검색 그리고, 컨텐츠 키워드 추출을 위한 대한 검토 및 고찰이다. 따라서 이러한 3가지 측면에서 검토 및 고찰은 다음과 같다.

[표 4] 파일용량별 블록 수

파일용량	GNUnet		제안 프로토콜	
	D블록	I,R블록	D블록	I,R블록
3M	3072	130	24	2
7M	7168	301	14	2
28M	28672	1197	448	20
300M	307200	12802	600	26
700M	716800	29869	11200	468

첫째, 파일 크기의 대형화에 따라 블록을 지시하는 I, R블록의 저장공간 낭비이다. (그림 3)은 멀티미디어 데이터에 추가되는 블록에 대한 내용이다. 현재 GNUnet에서 제안된 인코딩 방식을 적용했을 경우 "4%"의 저장공간 낭비가 발생하였다. 그러나 제안 프로토콜을 적용할 경우 "1%" 이내인것을 확인할 수 있다. 따라서 제안 프로토콜을 GNUnet에 적용할 경우 간접 지시자 블록을 저장공간 낭비를 줄일 수 있다.



[그림 6] 간접 지시자 생성 비율

둘째, 분리된 블록에 대하여 저장 및 검색 기능을 제공하기 위하여 효율적인 위상을 제안하였다. 기존 위상의 방식은 근접 라우팅 테이블에서 랜덤하게 피어를 선정하여 블록을 전송하기 때문에 분산된 블록을 하나의 파일로 처리하기 위해서는 브로드캐스트 방법으로 인코딩 블록을 요청해야 된다. 따라서 브로드캐스트를 피어가 수행했을 경우 검색 블록수가 많을 경우 GNUnet 시스템은 매우 혼잡하게 되었고 피어들은 자신의 시스템이 느려지는 현상을 체험하게 되었다. 이러한 혼잡현상을 최소화하기 위하여 제안 프로토콜에서는 다면체의 구조적 위상을 제안하였다. 다면체의 구조적 위상은 블록의 저장 및 검색에서 특정 피어의 피어 식별자와 검색하고자 하는 블록 식별자를 비교하여 블록 식별자보다 같거나 큰 피어 식별자를 만났을때 저장 및 검색을 수행한다. 따라서 기존의 브로드캐스트 방식이 아닌 특정피어를 지정하는 방식이다. 이 방식을 적용했을 경우 다수의 피어가 검색을 수행해도 특정피어에 대하여만 트래픽이 발생하고, 요청을 받지 않은 피어들은 서비스 대기상태에 놓인다. 제안 위상방법을 GNUnet 시스템에 제공할 경우 네트워크의 혼잡사태를 최소화할 수 있다.

셋째, 명확한 키워드 선정을 위한 컨텐츠 기반의 키워드 및 파일 식별자 생성에 의하여 기존의 키워드 검색을 위한 역 분산 해쉬 테이블 기반의 메커니즘에서는 공유 파일의 키워드 선정 방법이 모호하여 해당 파일을 유추할 수 있는 몇 개의 단어들로 키워드를 선정 하였다. 이러한 키워드의 선정 방법은 동일파일에 대해 사용자에 따라 서로 다른 키워드 선정이 이루어지므로 키워드 및 공통 키워드의 증가를 가중시킨다. 따라서 본 논문에서는 공유파일에 대한 명확한 키워드 선정을 위해 컨텐츠 내용 기반의 키워드를 선정하고, 키워드의 조합을 통한 파일 식별자를 생성한다. 따라서 공유 파일 컨텐츠의 메타 정보에 따라 키워드가 생성되기 때문에 동일한 파일은 항상 같은 키워드들로 구성이 되어 키워드 선정의 명확성과 공통키워드 수의 감소를 갖게 된다.

결론으로 제안 프로토콜은 파일을 블록단위로 분리하여 논리적인 네트워크에 분산과 검색을 수행할 발생하는 저장공간 낭비 및 네트워크 트래픽의 최소화방안과 효율적인 블록 검색방법을 제시하였다. 따라서 GNUnet에 적용할 경우 저장공간 최소화 및 네트워크 트래픽을 최소화 그리고, 사용자의 편리성을 제공할 수 있다. 그러나 I,R 블록 도청에 위험성과 구조적 위상의 성능 및 안정성에 대한 정확한 방안이 제시되어 있지 않기 때문에 구체적인 구조적 위상의 연구개발과 컨텐츠 기반 확장에 따른 KID 및 CKD의 인덱스 비용을 줄이기 위한 임계값 선정과 공유 파일의 인기도를 키워드 검색에 포함시키는 방안에 대한 연구가 진행되어야 한다.

V. 결론 및 향후과제

GNUnet은 익명성 제공을 위하여 파일 인코딩 방법과 인코딩된 블록을 다른 피어에게 분산하는 방법 그리고, 분산된 블록을 검색하는 방법을 제안하였다. 그러나 파일을 1KByte로 인코딩하는 방법은 대용량의 멀티미디어 파일에는 적합하지 않았다. 700MByte의 대용량 멀티미디어의 경우 1KByte로 인코딩 할 경우 716800개의 D블록을 생성하였고, 추가적인 블록인 I,R은 29868개를 생성하였다. 이것은 원본파일의 4%에 해당되는 부가적인 저장 공간을 요구하였고, 분리된 블록의 분산방법은 비구조적인 모델에 의하여 구성된 피어에게 전송하기 때문에 브로드캐스트 방식을 도입하였다. 따라서 브로드캐스트 방식에 의하여 과도한 네트워크 트레픽이 발생하게 되었다. 그리고, 분산된 블록의 검색을 위하여 파일명을 키워드로 생성하였기 때문에 하나의 키워드만이 존재하였다.

본 논문은 GNUnet의 파일 인코딩 방법과 인코딩 블록을 분산하는 방법 그리고 컨텐츠 키워드 생성방법을 제안하였다. 인코딩 방법은 $2^{10} \leq 2^n \leq 2^{20}$의 범위의 블록크기를 결정하여 인코딩 하였다. 기존에 인코딩 블록을 지시하기 위한 부가적인 공간으로 4%를 소비했지만, 제안 프로토콜을 도입할 경우 1% 이내로 저장공간 낭비를 최소화 하였다. 그리고, 분리된 파일의 분산방법으로써 CAN에서 제시한 구조화된 모델을 도입하였다. 피어들이 GNUnet 시스템에 참여할 경우 피어 ID를 부여받게 되고, 이를 근거로 다면체 형태의 구조에 참여하게 된다. 참여한 피어는 '피어 ID'를 근거로 '파일ID'가 같거나 작은 파일을 저장하게 된다. 따라서, 네트워크에 분산되어 있는 피어들은 다면체 형태를 구성하기 때문에 파일의 저장및 검색 능력이 기존 비 구조적방법과 비교할 경우 빠른 검색과 네트워크 트래픽 최소화의 장점을 갖고있다. 마지막으로 사용자 검색의 효율성을 제공하기 위하여 공통키워드 처리를 위해 PLS를 확장하여 KID와 CKD 관리를 담당하는 방법을 제안하였다. 결과적으로 제안하는 시스템은 GNUnet의 성능향상을 위하여 저장공간과 네트워크 트래픽을 최소화하는 방안 그리고 컨텐츠 기반 키워드 생성방법을 제안하였다.

본 논문의 향후과제로는 블록 분산방법에서 피어의 추가 및 탈퇴시 발생하는 피어들의 정보 테이블 갱신 방법에 대한 연구가 필요하다.

참고문헌

[1] Dennis Kugle, " An Analysis of GNUnet and the Implications for Anonymous, Censorship-Resistant Networks", PET, 2003.

[2] Krista Bennett, Christian Grothoff, Tzvetan Horozov, Ioana Patrascu, "Efficient Sharing

of Encrypted Data", ACISP 2002.
[3] Krista Bennett, Christian Grothoff, "practical anonymous networking", PET,2003.
[4] Christian Grothoff, Krista Grothoff, Tzvetan Horozov, J. T. Lindgren, "An Encoding for Censorship-Resistant Sharing"
[5] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content Addressable Network," In Proc. of ACM SIGCOMM 2001, San Diego, CA, 2001.
[6] P. Reynolds, A. Vahdat, "Efficient Peer-to-Peer Keyword Searching," In Proc. of Middleware 2003, Rio de Janeiro, Brazil, 2003.
[7] Jianning Yang, "APTPFS : Anonymous Peer-to-Peer File Sharing" April, 2005.
[8] David R. Karger, Matthias Ruhl, "Diminished Chord: A Protocol for Heterogeneous Subgroup Formation in Peer-to-Peer Networks", 2003.
[9] A. Keromytis, N.Provos, "The Use of HMAC-RIPEMD-160-96 within ESP and AH", rfc 2857, June, 2000.
[10] I. Stoica, R. Morris, D.Liben-Nowell, D.R.Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications," In Proc. of SIGCOMM 2001, San Diego, CA, 2001.
[11] Lintao Liu, Kyung Dong Ryu, Kang-Won Lee, "Keyword Fusion to Support Efficient Keyword-based Search in Peer-to-Peer File Sharing," In 4th Init'l Workshop on Global and P2P Computing, Chicago IL, April 2004.
[12] 앤디 오람, "Peer-to-Peer 차세대 인터넷 P2P", pp169~177, 2001.