Journal of the Korean Data & Information Science Society
2011, **22**(2), 269–285

한국데이터정보과학회지

이동 애드-혹 망을 위한 클러스터 기반 퍼지 협력 캐싱 방법의 설계 및 평가

이은주[1] · 배인한[2]

[12]대구가톨릭대학교 컴퓨터정보통신공학부

접수 2011년 2월 5일, 수정 2011년 3월 17일, 게재확정 2011년 3월 22일

요 약

이동 애드-혹 망에서 자주 액세스 되는 데이터의 캐싱은 데이터 액세스 성능 및 가용성을 향상시킬 수 있는 잠재적 기술이다. 다수의 클라이언트들 간의 캐시된 데이터의 공유와 협동을 허용하는 협력 캐싱은 캐싱 기술의 잠재력을 더 향상시킬 수 있다. 이 논문에서, 우리는 이동 애드-혹 망을 위한 클러스터 기반 퍼지 협력 캐싱 방법을 제안한다. 그리고 제안된 방법의 성능은 분석적 모델을 통해 평가하고, 그것의 성능을 다른 모바일 협력 캐싱 방법과 비교한다.

주요용어: 이동 애드-혹 망, 클러스터, 퍼지논리, 협력캐싱.

1. 서론

인터넷 그리고 인프라구조 기반 무선망과 모바일 애드-혹 망 (MANETs; mobile ad-hoc networks)의 통합은 유비쿼터스 네트워킹의 주요한 부분이 되고 있고, 제조업체와 네트워크 회사의 연구자들로부터 주목을 끌어 왔다.

제한된 대역폭과 빈번한 링크 율 변화에 기인한 MANETs의 유일한 특성은 연구 과제를 유발시킨다. 제한된 대역폭은 유선 링크에 의해 제공되는 대역폭에 비해 아주 낮다. 더욱이, 무선 링크는 장애물, 날씨, 다른 무선 장치들로부터 간섭에 의해 쉽게 저하되고, 시간에 따라 변하는 링크 율을 발생시킨다. 이것은 기지국 또는 액세스 포인트와 같은 집중화된 인프라구조의 부재에 기인한다. 따라서 각 모바일 장치들은 다른 모바일 장치들을 위한 데이터 패킷을 전송하는 라우터로 활동한다. 즉, 이동 호스트들 (MHs; mobile hosts)는 멀티 홉 전송을 통해 서로 통신한다. 다른 장애 특성은 MANETs가 전력 공급, 저장 공간, 컴퓨팅 용량에서 제한된 자원을 갖는다는 것이다. 과거 몇 년 동안, MANETs에 대한 대부분의 연구들은 계속 변하는 위상에서 MHs 간의 연결성을 향상시키기 위하여 매체 접근 제어 프로토콜과 라우팅 프로토콜에 집중되었다 (Denko 등, 2009).

MANETs가 제공하는 폭 넓은 기회에도 불구하고, 사회로부터 신뢰를 얻기 전에 해결되어야 할 연구 문제들이 아직 존재한다. 첫째, 멀티 홉 통신을 통한 원격 정보국을 액세스하는 것은 긴 질의 지연에 이르고 더 높은 에너지 소비를 발생시킨다. 둘째, 많은 클라이언트들이 자주 데이터베이스를 액세스할 때, 서버에서 높은 부하를 일으키고 서버 응답 시간을 증가시킨다. 셋째, 망 전체를 가로지르는 멀티 홉 통신에 다수의 노드들이 있다면, 망 용량은 급격히 떨어질 것이다 (Li 등, 2001). 모든 요청이 서버로 전송

[1] (712-702) 경북 경산시 하양읍 금락로 5, 대구가톨릭대학교 일반대학원 컴퓨터정보통신공학과, 박사과정.
[2] 교신저자: (712-702) 경북 경산시 하양읍 금락로 5, 대구가톨릭대학교 컴퓨터정보통신공학부, 교수.
Email: ihbae@cu.ac.kr

되기 때문에 서버 주변 지역은 다른 지역에 비해 더 혼잡해 진다. 혼잡과 패킷 길이 제한에 기인하여 많은 패킷들이 폐기될 것 이다. 트래픽 혼잡과 과부하는 서버 또는 기지국이 병목현상 되게 만든다. 최근에, 망 분할이 발생했을 때, 사용자는 정보국으로 직접 또는 간접 링크를 가질 때 까지 요청된 정보를 얻을 수 없다. 그러므로 데이터베이스 액세스를 위하여 인터넷과 MANETs를 통합에서, 효율적인 메커니즘 설계가 요구되어진다.

캐싱은 모바일 애드-혹 망에서 모바일 클라이언트의 데이터 검색 성능을 향상시키는 핵심 기법이다. 캐싱에서, 어떤 사용자가 데이터 서버로부터 데이터 아이템들을 액세스 후, 그 데이터 아이템들은 캐시되어 진다. 반면에 복사에서, 데이터 아이템들이 미래사용을 위해 사전에 망 내의 다수의 노드들에 위치되어진다. 캐싱은 반응적이고, 복사는 선행적이다. 그리고 강인하고 신뢰성 있는 P2P (peer-to-peer) 기술의 출현은 모바일 클라이언트들이 이웃 피어들로부터 데이터 항목들을 액세스할 수 있는 협력 캐시의 실현을 가져왔다.

본 논문에서는 모바일 애드-혹 망을 위한 데이터 캐싱을 위한 클러스터 기반 퍼지 협력 캐싱 (CFCC; Cluster-Based Fuzzy Cooperative Caching) 방법을 제안한다. CFCC에서, 망 위상은 물리적 망 근접성에 기초하여 겹치지 않는 클러스터로 분할하고, 각 노드는 데이터 캐싱을 위해 노드 활동 상태와 데이터 유용성에 따른 퍼지 논리에 기초하여 CacheData 방법 또는 CachePath 방법을 적응적으로 사용한다. 각 CH (cluster head)는 그 CMs (cluster members)의 캐시 된 정보와 클러스터 간 통신을 관리한다. 어떤 노드의 정보 검색 연산은 로컬 캐시, CHs, 경로 기반 노드, roadside 노드들, 그리고 마지막으로 데이터 서버 순으로 데이터 아이템을 요청한다.

본 논문의 구성은 다음과 같다. 2절에서 MANETs에서 기존의 협력 캐싱 방법을 소개하고, 3절에서 제안하는 CFCC 방법을 설명하고, 그리고 4절에서 CFCC의 성능을 분석적 모델로 평가한다. 마지막으로 5절에서 향후 연구 과제와 결론을 기술한다.

2. 관련연구

모바일 애드-혹 망에서 데이터 액세스를 편리하게 하기 위하여, 다수의 데이터 복사 방법과 캐싱 방법들이 제안되었다. 데이터 복사는 액세스 요구를 만족시키기 위하여 데이터 객체들의 사본들을 MHs에 할당하는 문제를 해결한다. 그러한 기법들은 작동 환경에 대한 사전 지식을 일반적으로 요구하고 노드 이동성에 취약하다. 데이터 복사 방법과 달리, 캐싱 방법들은 데이터 액세스를 용이하게 하기 위하여 사전의 데이터 객체들의 분산에 의존하지 않는다. 데이터 요청을 해결하는 일반적인 방법은 먼저 로컬 캐시를 검사하고, 그리고 로컬 캐시가 실패하면 서버로 요청을 전송한다. 이 방법을 SimpleCache라 한다 (Yin과 Cao, 2006). 이 방법은 서버와 연결이 신뢰적인 한 잘 작동하고 비용도 많이 들지 않는다. 아니면, 그것은 실패한 데이터 요청 또는 요청 타임아웃을 일으킨다.

데이터 가용성을 증가시키고 증가된 데이터 접근 지연과 증가된 에너지 소비에 대한 비용을 줄이기 위하여 Hop-by-hop 캐시 해결은 전송 경로 상의 어떤 노드가 그 요청을 해결하기 위한 프락시로 서비스한다. 만일 어떤 전송 노드가 그 요청된 데이터의 유효한 사본을 캐시하고 있다면, 그것은 그 요청 노드에게 응답을 전송하고 그 데이터 요청 전송을 중지한다 (Du 등, 2009).

노드의 협력 지역은 그 노드의 r-홉 범위 내의 주변 노드들로 구성된다. 매개변수 r을 협력 지역의 반경이라 부른다. 만일 어떤 노드가 그 요청된 데이터 아이템이 그 지역에서 가용인지에 대한 정보가 없다면, 그 노드는 그 지역 내의 요청을 플로딩하여 이것을 반응적으로 발견할 수 있다. 플로딩 범위를 제한하기 위하여, 그 요청의 TTL 값은 지역 반경 r로 설정된다 (Du 등, 2009).

Cao과 Yin (2004)은 3가지 방법: CachePath, CacheData, 그리고 HybridCache를 제안하였다. CacheData에서, 중간 노드들은 데이터 센터로부터 그 데이터를 인출하는 대신 미래 요청들을 서비스

하기 위하여 데이터를 캐시 한다. CachePath에서, 모바일 노드들은 데이터 경로를 캐시하고 미래 요청들을 먼 데이터 센터 대신 그 데이터를 갖는 가까운 노드로 전송하기 위하여 그것을 사용한다. 그 성능을 더 향상시키기 위하여, CacheData와 CachePath의 장점을 취하여 그 성능을 더 향상시킬 수 있는 하이브리드 방법인 HybridCache를 설계하였다. HybridCache에서, 어떤 모바일 노드가 데이터 아이템을 전송할 때, 그것은 다소의 기준에 기초하여 데이터나 경로를 캐시한다. 그러한 기준은 데이터 아이템 크기와 그 데이터의 TTL (time-to-live)을 포함한다. 어떤 데이터 아이템 d에 대해, 데이터 또는 경로를 캐시할지를 결정하기 위하여 다음 휴리스틱이 사용되어진다. 만일 d의 크기가 작다면, 그 데이터 아이템이 캐시의 매우 작은 부분만을 요구하기 때문에 CacheData가 채택되어야 한다. 아니면, 캐시 공간을 절약하기 위하여 CachePath가 채택되어야 한다. 만일 d의 TTL이 작다면, 그 데이터 항목은 곧 무효화될 수 있기 때문에 CachePath는 좋은 선택이 아니다. CachePath 사용은 잘못된 경로 추적을 일으킬 수 있고, 그 질의를 결국 데이터 센터로 재전송 된다. 만일 d의 TTL이 크면, CachePath가 채택되어야 한다. TTL에 대한 임계값은 시스템 조정 매개변수이다.

Du와 Gupta (2005)는 MANETs에서 주문형 데이터 액세스 응용들을 위한 이상적인 협력 캐싱 방법, COOP를 제안하였다. 그 목적은 모바일 노드들의 지역 자원들을 협력하여 데이터 가용성과 액세스 효율성을 향상시키는 것이다. 캐싱 노드들의 협력은 두 부분이 있다. 첫째, 어떤 캐싱 노드는 다른 노드들로부터의 데이터 요청에 응답할 수 있다. 둘째, 어떤 캐싱 노드는 자신의 필요에 의해서 뿐만 아니라 다른 노드의 필요에 기초하여 데이터를 저장한다.

Denko 등 (2009)은 협력 캐싱 방법과 선인출 방법의 성능을 더 향상시키기 위하여 교차 계층 설계 방법을 사용하는 클러스터 기반 협력 캐싱 방법을 제안하였다. 교차 계층 정보는 독립된 자료 구조로 관리되고 망 프로토콜 계층들 간에 공유된다.

Chand 등 (2007)은 캐시 발견 오버헤드를 줄이고 더 나은 협력 캐싱 성능을 제거하기 위하여 클러스터 협력 캐싱을 제안하였다. 그리고 Kumar (2010)는 이동 애드-혹 망을 위한 COOP, Hybrid Cache, IXP (Index Push) 그리고 DPIP (Data Pull/Index Push) 등을 포함한 다양한 캐시 관리 기법들을 검토하였다.

Bae (2009)는 이동 호스트의 이동 방향을 예측하고, 그 이동 호스트의 소속도에 기초한 퍼지 논리 제어 규칙에 따라 이웃 프락시 서버들에 대해 다양한 캐시 방법을 사용하는 퍼지 프락시 캐싱 방법을 제안하였다. 그리고 Bae (2010)는 데이터 유사도와 데이터 이용률에 기초하여 CacheData와 CachePath를 적응적으로 사용하는 퍼지 협력 캐싱 방법을 제안하였다.

3. 클러스터 기반 퍼지 협력 캐싱

이 논문에서, 우리는 다수의 데이터 서버와 모바일 노드라 부르는 다수의 MHs를 갖는 환경을 고려한다. 모든 다른 MHs에 의해 알려진 주소를 갖는 데이터 서버는 전체 MANET에 의해 필요한 데이터 아이템들을 저장할 수 있는 데이터베이스 서버 또는 MANET에서 다른 MHs를 위해 정보 서비스를 제공하는 인터넷에 대한 게이트웨이로 고려될 수 있다. 우리는 텍스트 파일과 웹 페이지와 같은 정규 데이터 아이템들이 데이터 서버에 저장된다고 가정한다.

CFCC의 시스템 구조는 그림 3.1에서 보여준다. CFCC는 하부 망 스택의 위에 있고 MANETs 환경에서 상위 계층 응용들에 캐싱과 다른 데이터 관리 서비스를 제공하는 클러스터 기반 미들웨어이다. CFCC는 크게 3가지 기본 모듈 : 클러스터링, 캐시 관리, 캐시 해결로 구성되어진다. 클러스터링 모듈은 클러스터의 생성과 관리에 책임이 있다. 캐시 관리는 MH가 데이터 항목을 그것의 로컬 캐시에 위치시킬 것인지 제거시킬 것인지를 결정한다. 그리고 캐시 해결은 MH가 사용자 또는 응용에 의해 요청된 데이터 아이템을 어디서 인출할지를 결정한다.

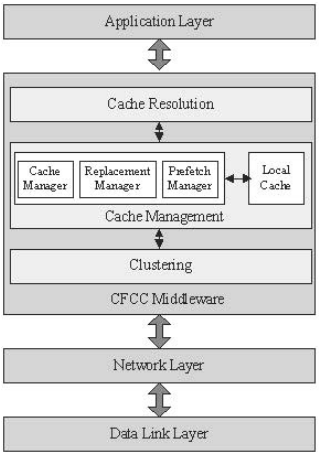

그림 3.1 CFCC 방법의 시스템 구조

3.1. 모바일 캐시 구조

어떤 캐시 모델의 설계는 캐시 데이터의 특성과 특정 캐시 환경에 기초되어야 한다. MANETs 환경에서, 캐시 된 데이터는 원래 데이터 서버들로부터 직접 인출되는 대신에 다른 MHs로부터 현재 MH로 이주될 수도 있다. 캐시 된 데이터의 특성과 모바일 환경을 처리하기 위하여, 그림 3.2와 같은 모바일 캐시 구조를 설계한다.

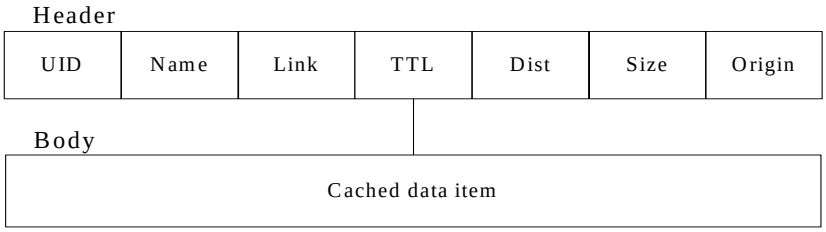

그림 3.2 모바일 캐시 구조

모바일 캐시 구조는 2 부분으로 구성되어진다. 캐시 구조의 본체는 캐시 된 데이터 아이템이다. 헤더 부분은 UID, Name, Link, TTL, Dist, Size와 Origin을 포함한다. UID 필드는 캐시 된 데이터의 URI가 해시되어질 UUID (universal unique ID)를 포함한다. Name은 캐시 된 데이터의 이름이다. 캐시된 데이터 아이템은 다른 MHs로부터 이주될 수도 있기 때문에, Link는 이 데이터를 이전에 소유하였거나 검색하였던 MH에 대한 링크를 제공하는데 필요하다. 데이터 서버는 모든 데이터 아이템에 TTL 값을 할당한다. MH는 만일 데이터의 TTL (time-to-live) 값이 유효하다면 최신 캐시 된 사본으로 고려한다. Dist는 캐시 된 데이터를 전송하는 노드에 도달하기 위한 홉의 개수이고, Size는 캐시 된 데이터의 크기를 나타낸다. 그리고 Origin은 이 현재 데이터 사본이 원래 데이터 서버로부터 인출되었는지 또는 다른 MH로부터 이주되었는지를 가리킨다.

3.2. 클러스터 생성과 관리

CFCC에서, 망 위상은 클러스터라 부르는 미리 정해진 동일한 크기의 지리적 격자들로 분할되어진다. 격자 크기는 어떤 클러스터내의 두 노드 간의 최대 거리를 결정한다. 각 클러스터 지역에서, 그것의 클러스터 영역내의 다른 노드들과 캐시 상태 정보를 관리하는 하나의 CH가 선택되어진다.

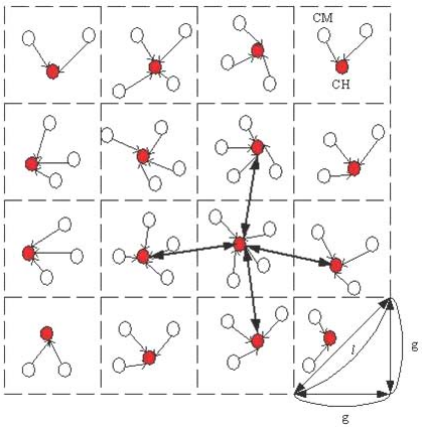

그림 3.3 MANET 클러스터링

그림 3.3과 같이 어떤 MH가 어떤 데이터 아이템이나 경로를 캐시 또는 교체하면, 그것의 캐시 상태 정보는 그 데이터의 헤더를 전송하여 그 CH와 이웃한 4개의 CH들에서 갱신되어진다. CH는 그 클러스터에 속하는 다른 클라이언트들에 의해 캐시된 데이터 아이템들에 대한 정보를 저장하기 위한 클러스터 캐시 상태 보유자로만 동작하고, 캐시 발견 동안에 부가적인 서비스를 제공한다. 이 클러스터링 방법에서, 격자 면의 길이 g가 클러스터링을 위한 핵심 요소이다. 두 MHs 간의 최대 거리 l은 $l = \sqrt{g^2 + g^2} = \sqrt{2}g$이다. 어떤 클러스터내의 호스트들 간에 한 홉 통신을 보장하기 위하여, l은 전송 범위 r 보다 작거나 같아야 한다. 즉, $\sqrt{2}g = r$이고, 따라서 $g = r/\sqrt{2}$이다.

일반적인 클러스터는 하나의 CH와 다수의 CMs로 구성된다. 하나의 CM은 단지 하나의 CH에 속한다. 어떤 새로운 MH가 어떤 클러스터에 결합될 때, GPS에 의해 얻어질 수 있는 좌표 (x, y)로 표현된 그것의 위치 정보를 갖는 방송 요청을 한 홉 이웃들에게 전송하여 CH의 존재를 검사한다. 그 요청한 MH와 동일한 클러스터에 속하는 이웃들은 클러스터 ID와 CH-ID로 응답할 것이다. 어떤 CH로부터의 첫 번째 긍정응답으로, 그 클라이언트는 캐시 정보로 캐시 헤더를 그 CH로 전송하여 그 클러스터의 CM이 된다. 그러나 만일 미리 명시된 임계 시간 안에 CH 응답이 없다면, 그 MH가 그 클러스터에 진입한 첫 번째 노드이므로 새로운 CH가 된다.

CH가 다른 클러스터로 이동이 예상되면, 그 CH는 이 클러스터를 위한 새로운 CH를 CM 중에서 무작위로 선택할 것이다. 후보 노드가 선택된 후, 모든 캐시 상태 엔트리들은 후보 노드로 이주된다. 그 다음, 새로운 CH는 CH 변경 메시지를 그 클러스터내의 모든 CMs들에게 멀티캐스트 한다. 그리고 그 CMs는 CH를 변경한다.

3.3. 캐시 관리

CFCC의 캐시 관리는 어떤 데이터 아이템을 노드의 로컬 캐시에 유지할지를 결정한다. 그 목표는 캐시 적중률을 증가시키는 것이다. 캐시 관리는 3가지 구성요소 : 캐시 관리자, 선인출 관리자, 교체 관리자로 구성되어진다.

3.3.1. 캐시 관리자

캐시 관리자는 노드 활동 상태와 데이터 유용성에 따른 퍼지 논리에 기초하여 CacheData와 CachePath를 적응적으로 사용한다. 캐시 관리자의 CachePath에서, 어떤 라우터 노드는 데이터 소스 보다 캐싱 노드에 더 가까울 때 데이터 경로를 기록한다. CacheData 방법에서, 라우터 노드가 자주 액세스 되는 데이터를 찾을 때 경로 대신에 데이터를 캐시 한다.CFCC에서, 각 MH는 3가지 활동 상태 : LA (Low Activity), MA (Medium Activity) 그리고 HA (High Activity) 중의 하나가 될 수 있다. LA 상태에 있는 MH를 LAM (Low Activity Mobile host)라 부르고, HA 상태에 있으면 HAM (High Activity Mobile host)라 부른다. LAM은 장치를 끄지는 않지만 시스템에 대한 액세스를 가끔 생성한다. 각 MH는 CAR (cache access rate)를 감시하여 자신의 상태를 탐지한다. CAR은 식 (3.1)에서 정의된 것처럼 시간 δ의 주기마다 재계산되어진다.

$$CAR(t_c, t_l) = \frac{N_a}{t_c - t_l} \tag{3.1}$$

여기서 $CAR(t_c, t_l)$은 시간 구간 (t_c, t_l)에서 CAR이고, N_a는 δ 동안에 캐시 액세스 횟수이고, t_c는 현재 시간이고, 그리고 t_l은 마지막 CAR이 계산된 시간이다.

만일 어떤 MH가 높은 SA (sharing ability)를 가진다면, 그 MH는 LAM으로 고려된다. 특별한 시점에 MH의 SA 값은 CAR에 반비례하고, CAR 값이 클수록 SA 값은 더 작아진다. 즉, SA는 1/(CAR+1)로 정의될 수 있다. 초기에 SA는 0으로 설정된다. 히스토리의 요약을 이용하여 시간 주기 동안 SA의 점차적인 변화를 알기 위하여, SA는 새로운 평가와 과거 평가 간의 가중치 합으로 계산되어진다 (Chow 등, 2004).

$$SA_t = w \times \frac{1}{CAR + 1} + (1 - w) \times SA_{t-1} \tag{3.2}$$

여기서 w는 시간 t−1의 이전 SA 값의 중요도에 대한 시간 t의 최근 SA 값의 중요도에 가중치를 주기 위한 매개변수이다.

MH의 상태는 임계값에 의해 정의가 내려진다. 만일 SA값이 임계값보다 크거나 같으면 MH는 LAM으로 고려된다. 마찬가지로, 만일 SA값이 임계값보다 작으면 MH는 HAM으로 분류된다. LAM인 MH가 데이터 아이템을 캐시하면, HAM인 이웃 MH들이 LAM 캐시에 존재하는 데이터 아이템을 이용할 수 있다.

MH의 현재 활동 상태 (AS; activity state)는 그림 3.4에서 보여진 SA_t 값에 따른 소속함수를 사용하여 3가지 기본 퍼지 집합: H (high), M (medium), L (low)로 사상된다. 소속함수는 소속 정도를 표현한 [0, 1]에 포함되는 실수값으로 사상되는 함수이다. 소속함수는 퍼지집합으로 정의된다. SA에 대한 소속함수는 SA의 퍼지집합으로 표현되며, 퍼지집합 AS (Activity State)을 나타내는 소속함수는 $\mu_{AS}(SA_t)$로 표현된다.

데이터 아이템의 크기가 작을수록 그 데이터 아이템을 위하여 필요한 캐시 공간은 더 적어진다. 어떤 데이터 아이템이 더 큰 Dist 값을 가질수록, 그 데이터 캐싱이 더 많은 유용성을 갖는다. 따라서 캐시

된 데이터 아이템 d_i에 대한 유용성, $Util_i$는 식 (3.3)을 사용하여 계산되어진다.

$$Util_i = \frac{Dist_i}{Size_i} \tag{3.3}$$

데이터 유용성 (DU; data utility)은 그림 3.5에서 보여진 $Util_i$ 값에 따른 소속함수를 사용하여 3가지 기본 퍼지 집합: H (high), M (medium), L (low)로 사상된다. 여기서 데이터 크기의 단위는 KB (kilobyte)이다. Util의 소속함수는 Util의 퍼지집합으로 표현되며, 퍼지집합 DU를 나타내는 소속함수는 $\mu_{DU}(Util_i)$로 표현된다.

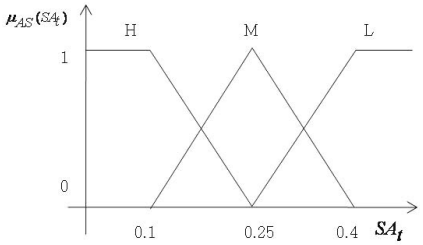

그림 3.4 MH의 활동 상태에 대한 소속함수

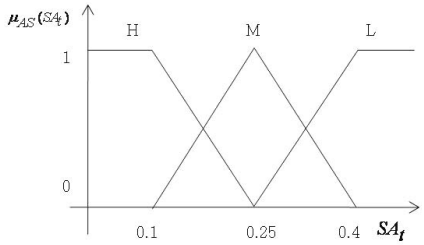

그림 3.5 데이터 유용성에 대한 소속함수

협력 캐시들의 용량을 최대화하기 위하여, CFCC는 가까운 거리 이웃 안에 중복된 캐시 감소를 시도하여 캐시 공간이 더 많은 별개의 데이터 아이템들을 수용하는데 사용될 수 있게 한다. 캐시 된 데이터 사본이 이미 이웃에서 가용인지 아닌지에 기초하여 캐시 된 데이터 사본을 분류한다. 데이터 사본이 이웃에서 비가용이면 데이터 사본이 기본 (primary)이다. 아니면, 그 데이터 사본은 보조 (secondary)이다. 기본 데이터와 보조 데이터의 캐싱 우선순위를 결정하기 위하여 범주 간 규칙과 범주 내 규칙을 사용한다.

범주 간 규칙의 개념은 어떤 우선 단계에 기본 아이템을 넣는 것이다. 즉, 보조 아이템은 기본 아이템을 수용하기 위하여 제거된다. MH의 활동 상태와 데이터 유용성을 고려한 CFCC를 위한 제어 규칙은 표 3.1에서 보여준다. 어떤 MH에 대해, 만일 활동 상태가 낮고 데이터 유용성이 높을 경우 , 데이터를 아이템을 관리하기 위하여 CacheData(i)가 사용되고, 활동 상태가 중간정도이고 데이터 유용성이 높으면, 데이터를 아이템을 관리하기 위하여 CacheData(j)가 사용된다. 또한 활동 상태가 중간정도이고 데이터 유용성이 중간정도이면, 데이터를 아이템을 관리하기 위하여 CacheData(k)가 사용되고, 활동 상태가 높고 데이터 유용성이 낮으면, 데이터를 아이템을 관리하기 위하여 CachePath가 사용된다. 여기서 i, j, k는 이웃 범위를 나타내는 매개변수들이고, i<j<k이다.

표 3.1 CFCC를 위한 제어 규칙

		DU		
		H	M	L
AS	H	CacheData(k)	CachePath	CachePath
	M	CacheData(j)	CacheData(k)	CachePath
	L	CacheData(i)	CacheData(j)	CacheData(k)
(입력변수)		DU: H (high), M (medium), L (low) AS: H (high), M (medium), L (low)		
(출력변수)		캐시방법: CacheData(i), CacheData(j), CacheData(k), CachePath		

범주내 규칙은 같은 범주내의 데이터 아이템들을 평가하기 위하여 사용되어진다. 이 목적을 위하여, 캐시 관리자는 교체 관리자를 호출한다. CFCC에서, 교체 관리자는 LPV (least profit value) 정책을 사용한다.

3.3.2. 선인출 관리자

캐시된 데이터의 TTL이 만기가 되면, 선인출 관리자가 호출된다. 선인출 관리자는 만기된 데이터의 인기도 지표 (PI; popularity index)를 평가한다. 어떤 데이터의 PI는 그 MH에서 그 데이터의 액세스 확률을 반영한다. 만기된 데이터 아이템 d_i 의 인기도 지표 PI_i 는 식 (3.4)로부터 구해진다.

$$PI_i = \frac{A_i}{\sum_{i=1}^{n} A_i} \tag{3.4}$$

여기서 A_i는 데이터 아이템 d_i에 대한 평균 액세스 율이고 n은 그 로컬 캐시에 저장된 데이터 아이템들의 전체 개수이다. 만일 그 PI_i가 임계값보다 크면, 그 데이터는 선인출 된다. 아니면 그 데이터 아이템의 공간은 해제된다.

3.3.3. 교체 관리자

제한된 캐시 크기 때문에, 새로운 데이터가 도착했을 때 캐시로부터 데이터를 제거하기 위하여 교체 관리자가 채택되어야 한다. 교체 관리자는 최소 데이터 이익에 기초한 교체 정책을 사용한다. 어떤 MH에서, 교체 관리자는 그 MH에 캐시된 모든 데이터 아이템들의 데이터 이익을 평가한다. 어떤 캐시된 데이터 아이템의 이익은 식 (3.5)로 평가되어진다.

$$Profit_i = \frac{A_i \times Dist_i \times CTTL_i}{Size_i} \tag{3.5}$$

캐시된 데이터 아이템 d_i의 현재 TTL, $CTTL_i$는 다음 수식으로 계산되어진다.

$$CTTL_i = TTL_i - (T - T_i^{initial}) \tag{3.6}$$

여기서 $T_i^{initial}$는 d_i가 데이터 서버로부터 인출되었을 때 시간이다.

캐시될 새로운 데이터 아이템을 수용하기 위하여 자유 공간이 부족할 때, 교체 관리자는 모든 캐시된 데이터 아이템들 중에서 최소 이익 값, LPV를 갖는 데이터 아이템을 제거한다.

3.4. 캐시 해결

제안하는 CFCC는 캐시 해결을 위하여 그림 3.6과 같은 혼합 방법을 사용한다. 데이터 요청이 로컬 캐시에 없으면 데이터 요청을 지역 클러스터로 전송된다. 지역 클러스터가 그 데이터를 캐시하고 있으면 그 데이터를 반환한다. 아니면, 지역 클러스터내의 CMs가 그 데이터를 캐시하고 있다면, 그 CMs 중에서 첫 번째로 검색된 CM으로 데이터 요청을 전송한다. 아니면, 지역 클러스터내의 CMs가 그 데이터의 경로를 캐시하고 있다면, 요청 데이터로의 최단 경로를 갖는 CM으로 데이터 요청을 전송한다. 그리고 데이터 요청이 지역 클러스터에 의해 해결되지 않으면, Roadside 해결 (Du 등, 2009)을 사용한다.

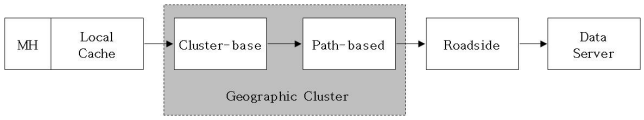

그림 3.6 CFCC 방법의 해결 프로세스

4. 분석적 평가

다음 가정 하에 CFCC의 성능을 분석을 한다.

- 로컬 캐시가 적중되면 응답 지연과 에너지 비용은 0이다. 노드 간 통신에 비교했을 때 로컬 캐시 액세스 비용은 무시될 수 있기 때문에 합리적인 가정이다.
- 만일 어떤 문서 요청이 로컬 캐시로부터 만족될 수 없다면, 응답 지연은 그 문서 요청의 운행된 홉 거리에 비례한다. 즉, 응답지연=α×운행 거리이다. 여기서 α는 상수이다. 이 분석에서, α가 분석과 성능 경과를 변경시키지 않기 때문에 우리는 그것을 무시한다.
- 만일 어떤 문서 요청이 로컬 캐시로부터 만족될 수 없다면, 우리는 문서 요청의 에너지 비용은 그 요청에 의해 발생되는 메시지 개수에 비례한다고 가정한다. 즉, 에너지 비용=β×메시지 개수이다. 응답지연에서 α와 마찬가지로, β는 결과를 변경시키지 않기 때문에 상수 β를 무시한다.

성능 분석에서 사용된 기호는 표 4.1에서 보여준다.

표 4.1 성능 분석에 사용된 기호

기호	설 명
P_d^s	SimpleCache 방법에서 요청된 데이터가 로컬 캐시에 있을 확률
P_d^h	Hop-by-hop 방법에서 요청된 데이터가 로컬 캐시에 있을 확률
P_f^h	Hop-by-hop 방법에서 요청된 데이터가 전송 노드의 로컬 캐시에 있을 확률
P_p^{hc}	HybridCache 방법에서 요청된 데이터에 대한 경로가 로컬 캐시에 있을 확률
P_p^{cf}	CFCC 방법에서 요청된 데이터에 대한 경로가 로컬 캐시에 있을 확률
L_p^{hc}	HybridCache 방법의 CachePath에서 요청 노드로부터 요청된 데이터의 사본을 갖는 노드까지의 경로 길이
P_p^x	TTL 만기 또는 경로 파손으로 로컬 경로에 따른 데이터가 유용하지 않을 확률
P_{cf}^{cd}	CFCC의 CacheData 방법에 의한 로컬 적중률
P_{pd}^h	CFCC의 선인출 관리자에 의한 선 인출된 데이터로부터의 로컬 적중률
P_{xpd}^r	무효화된 인기 있는 데이터가 요청될 확률
P_d^p	요청된 데이터가 인기 있는 데이터일 확률
R_a^c	CFCC 방법에서 모바일 노드의 데이터 액세스 요청 당 캐싱 비율
ρ	평균 노드 밀도
r	지역 기반 캐시 방법과 COOP 방법에서 요청 노드의 협력 지역의 반경
r_1	CFCC 방법에서 요청 노드의 로컬 클러스터의 반경
r_2	CFCC 방법에서 로컬 클러스터 헤드에서 이웃 클러스터까지의 반경
L_s	요청 노드와 데이터 서버간 거리

(1) SimpleCache 방법

SimpleCache 방법에 대해, 문서 요청은 로컬 캐시 실패 후 서버로 전송된다. 이 경우에, 평균 응답 지연과 평균 에너지 비용은 다음과 같다.

$$D_{SimpleCache} = L_s(1 - P_d^s) \tag{4.1}$$

$$E_{SimpleCache} = L_s(1 - P_d^s) \tag{4.2}$$

그 함수는 응답지연과 에너지 비용은 서버 거리가 증가하는 만큼 증가하고, P_d^s가 증가하는 만큼 감소한다는 것을 보여준다.

(2) Hop-by-hop 캐시 방법

Hop-by-hop 캐시 해결 방법에 대해, 각 전송 노드는 전송된 데이터 요청을 검사하고, 만일 그 데이터가 로컬 캐시에 있다면 그 요청에 응답한다. 이 방법의 평균 응답 지연은 다음과 같다.

$$D_{HopbyHop} = (1-P_d^h)\left[\frac{(1-P_f^h)-(1-P_f^h)^{L_s-1}}{P_f^h} - (L_s-1)(1-P_f^h)^{L_s-1} + L_s(1-P_f^h)^{L_s-1}\right]$$
$$= (1-P_d^h)\left[\frac{(1-P_f^h)-(1-P_f^h)^{L_s-1}}{P_f^h} + (1-P_f^h)^{L_s-1}\right] \quad (4.3)$$

평균 에너지 비용은 식 (4.4)와 같다.

$$E_{HopbyHop} = (1-P_d^h)\left[\frac{(1-P_f^h)-(1-P_f^h)^{L_s-1}}{P_f^h} + (1-P_f^h)^{L_s-1}\right] \quad (4.4)$$

Hop-by-hop 해결 방법의 평균 운행 거리는 L_s가 어느 정도 커질 지에 상관없이 $(1-P_d^h)(1-P_d^f)/P_f^h$에 의해 실질적으로 한정되어진다. L_s와 전송 노드의 개수가 증가하면 전송 경로 상에서 문서 요청을 해결할 확률도 증가한다.

(3) 지역 기반 캐시 방법

만일 그 문서가 로컬 캐시에 없다면, 그 문서를 위해 협력 지역 요청을 플로딩 한다. 지역 기반 방법의 성공 확률은 협력 지역에 문서의 존재 여부에 의존한다. r 을 협력 지역의 반경이라 하고, P_d^z 을 협력 지역 내의 어떤 노드가 국부적으로 d를 캐시할 확률이라 하자. 협력 지역으로부터 요청된 문서 d를 얻을 확률은 다음과 같다.

$$P(r) = 1-(1-P_d^z)^{\rho\pi r^2-1}$$

평균 응답 지연은 3가지 가능 상황: 요청이 로컬 캐시로부터, 협력 지역 내의 방송으로부터, 웹 서버로부터 만족될 수 있다는 것에 기초하여 계산되어진다. 이 3가지 시나리오의 확률은 P_d^l, $(1-P_d^l)P(r)$, 그리고 $(1-P_d^l)(1-P(r))$이다. 상대적으로, 그러한 상황에 대해 응답 지연은 0, r, 그리고 L_s 이다. 따라서 평균 응답 지연은 식 (4.5)와 같다.

$$D_{ZoneCache} = (1-P_d^l)[rP(r) + L_s(1-P(r))] \quad (4.5)$$

로컬 캐시 실패이고 그 문서가 지역에서 발견된다면, 에너지 비용은 $\rho\pi r^2$이 든다. 따라서 평균 에너지 비용은 식 (4.6)과 같다.

$$E_{ZoneCache} = (1-P_d^l)[\rho\pi r^2 + L_s(1-P(r)] \quad (4.6)$$

(4) HybridCache 방법

HybridCache에서, 만일 요청된 데이터의 사본이 요청한 노드의 로컬 캐시 또는 요청한 노드의 로컬 경로상의 노드의 로컬 캐시에 있다면, 그 데이터는 반환되어진다. 만일 요청한 노드가 그 요청된 데이터를 찾지 못하면, 그 데이터 요청은 데이터 서버로 전송되어진다. 평균 지연을 계산하기 위하여, 다음 3가지 경우가 고려되어야 한다.

1. 요청된 데이터가 로컬 캐시에 있다.

2. N_i가 요청된 데이터를 캐시하고 있다는 것을 가리키는 어떤 경로가 로컬 캐시에서 발견된다.
   a. 유효한 데이터 아이템이 N_i에서 발견된다.
   b. 파손된 경로 또는 TTL 만기 때문에 N_i의 데이터 아이템은 사용 불가능하다.
3. 로컬 캐시에서 데이터나 경로가 발견되지 않았다.

경우 1, 2(a), 2(b), 3의 확률은 각각 P_d^{hc}, $(1 - P_d^{hc}) P_p^{hc} (1 - P_p^x)$, $(1 - P_d^{hc}) P_p^{hc} P_p^x$ 그리고 $(1 - P_d^{hc})(1 - P_p^{hc})$이다. 그러므로 평균 응답 지연은 식 (4.7)과 같다.

$$\begin{aligned} D_{HybridCache} &= L_p^{hc}(1 - P_d^{hc})P_p^{hc}[(1 - P_p^x) + L_s P_p^x] + L_s(1 - P_d^{hc})(1 - P_p^{hc}) \\ &= (1 - P_d^{hc})[L_p^{hc} P_p^{hc}[(1 - P_d^x) + L_s P_d^x] + L_s(1 - P_p^{hc}) \end{aligned} \tag{4.7}$$

평균 에너지 비용은 식 (4.8)과 같다.

$$E_{HybridCache} = (1 - P_d^{hc})[L_p^{hc} P_p^{hc}[(1 - P_p^x) + L_s P_d^x] + L_s(1 - P_p^{hc})] \tag{4.8}$$

(5) COOP 방법

COOP의 캐시 해결은 협력 지역에서 데이터 요청 해결을 위한 첫 번째 시도로 hop-by-hop 해결과 협력 지역 해결을 조합한 칵테일 방법을 사용한다. 만일 그것이 실패하면, 그 요청은 중간 노드들이 그 요청을 포착하고 해결하는 것을 허용하는 서버로 전송된다. 여기서 r 을 협력 지역의 반경, 그리고 P_d^{cz}를 협력 지역 내의 어떤 노드가 d를 캐시할 확률이라 한다. 그 협력 지역으로부터 요청된 데이터 d를 얻을 확률은 다음과 같다.

$$P(r) = 1 - (1 - P_d^{cz})^{\rho\pi r^2 - 1}$$

전달하는 노드가 데이터 요청을 해결할 확률을 P_f^{co}라 한다. COOP에서, 가능한 상황의 확률들은 다음과 같다. 그 요청이 로컬 캐시에서 해결될 확률은 P_d^{co}, 그 요청이 협력 지역에서 해결될 확률은 $P(r)$, 그 요청이 협력 지역을 넘어서 i (i=1, 2, ..., $L_s - r - 1$) 홉에서 해결될 확률은 $(1-P(r))(1-P_f^{co})^{i-1}P_f^{co}$, 그리고 그 요청이 데이터 서버에서 해결될 확률은 $(1 - P(r))(1 - P_f^{co})^{L_s - r - 1}$이다. 따라서 평균 응답 지연은 식 (4.9)와 같다.

$$\begin{aligned} D_{COOP} &= (1 - P_d^{co})[rP(r) + (1 - P(r))[r + P_f^{co}(1 + 2(1 - P_f^{co}) + \cdots \\ &\quad + (L_s - r - 1)(1 - P_f^{co})^{L_s - r - 2}) + (L_s - r)(1 - P_f^{co})^{L_s - r - 1}]] \\ &= (1 - P_d^{co})[rP(r) + (1 - P(r))[r + P_f^{co} \sum_{i=1}^{L_s - r - 1} i(1 - P_f^{co})^{i-1} \\ &\quad + (L_s - r)(1 - P_f^{co})^{L_s - r - 1}]] \end{aligned} \tag{4.9}$$

로컬 실패이고 만일 그 데이터가 협력 지역에서 발견된다면, $\rho\pi r^2$의 에너지 비용이 든다. 따라서 평균 에너지 비용은 식 (4.10)과 같다.

$$\begin{aligned} E_{COOP} &= (1 - P_d^{co})[(\rho\pi r^2)P(r) + (1 - P(r)) \\ &\quad [r + P_f^{co} \sum_{i=1}^{L_s - r - 1} i(1 - P_f^{co})^{i-1} + (L_s - r)(1 - P_f^{co})^{L_s - r - 1}]] \end{aligned} \tag{4.10}$$

(6) CFCC 방법

CFCC 방법의 평균 응답 지연은 5가지의 가능한 경우에 기초하여 계산되어진다.

1. 요청된 데이터가 로컬 캐시에서 해결된다.
2. 요청된 데이터가 클러스터내의 노드에서 해결된다.
   a. 요청된 데이터가 로컬 클러스터내의 노드에 있다.
   b. 요청된 데이터가 이웃 클러스터내의 노드에 있다.
3. 요청된 데이터가 캐시 된 경로가 가리키는 노드에서 해결된다.
4. 요청된 데이터가 데이터 서버로의 경로 상의 전송 노드에서 해결된다.
5. 요청된 데이터가 데이터 서버에서 해결되어 진다.

상기 데이터 요청에 대한 캐시 해결 과정은 그림 4.1과 같이 나타낼 수 있다.

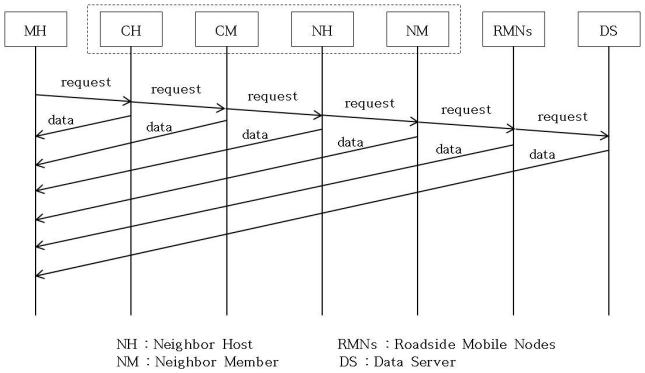

그림 4.1 데이터 요청에 대한 캐시 해결 과정

CFCC의 분석적 평가를 위해 r_1을 로컬 클러스터의 반경, r_2를 로컬 클러스터 헤드에서 이웃 클러스터까지의 반경이라 하고, 그리고 P_d^{cf}를 요청 데이터가 로컬 캐시에 있을 확률, P_d^{lc}를 요청 데이터가 로컬 클러스터내의 어떤 노드에 있을 확률, P_d^{nc}를 요청 데이터가 이웃 클러스터내의 어떤 노드에 있을 확률, P_p^{cf}를 요청한 데이터가 캐시된 경로가 가리키는 노드에 있을 확률이라 한다. 따라서 CFCC 방법의 평균 응답 지연은 식 (4.11)과 같다.

를 요청한 데이터가 캐시된 경로가 가리키는 노드에 있을 확률이라 한다. 따라서 CFCC 방법의 평균 응답 지연은 식 (4.11)과 같다.

$$\begin{aligned}
D_{CFCC} &= (1-P_d^{cf})(r_1+1)P_d^{lc} + (1-P_d^{cf})(1-P_d^{lc})(r_2+1)P_d^{nc} \\
&\quad + (1-P_d^{cf})(1-P_d^{lc})(1-P_d^{nc})P_p^{cf}(1-P_p^{x})L_p^{cf} + (1-P_d^{cf})(1-P_d^{lc}) \\
&\quad (1-P_d^{nc})[1-P_p^{cf}(1-P_p^{x})][r_2 + P_f^{cf}(1+2(1-P_f^{cf}) + \cdots \\
&\quad + (L_s - r_2 - 1)(1-P_f^{cf})^{L_s-r_2-2}) + (L_s - r_2)(1-P_f^{cf})^{L_s-r_2-1}] \\
&= (1-P_d^{cf})[(r_1+1)P_d^{lc} + (1-P_d^{lc})[(r_2+1)P_d^{nc} + (1-P_d^{nc}) \\
&\quad [P_p^{cf}(1-P_p^{x})L_p^{cf} + [1-P_p^{cf}(1-P_p^{x})][r_2 + P_f^{cf}\sum_{i=1}^{L_s-r_2-1} i(1-P_f^{cf})^{i-1} \\
&\quad + (L_s - r_2)(1-P_f^{cf})^{L_s-r_2-1}]]]]
\end{aligned} \tag{4.11}$$

CFCC 방법의 평균 에너지 비용에서는 모바일 노드가 어떤 데이터 아이템을 캐시 하였을 때, 그 모바일 노드는 로컬 클러스터와 그 로컬 클러스터의 이웃 클러스터에게 캐시 상태 정보를 전송한다.

CFCC의 평균 에너지 비용을 계산하기 위하여 액세스-캐시 비율 (access-cache ratio)을 정의한다. 액세스-캐시 비율, R_a^c은 어떤 모바일 노드가 데이터 액세스 요청 당 그 요청된 데이터를 캐시 하는 비율을 나타낸다. 따라서 R_a^c을 고려한 CFCC의 평균 에너지 비용은 식 (4.12)와 같다.

$$\begin{aligned} E_{CFCC} = (1-P_d^{cf})[(r_1+1)P_d^{lc} + (1-P_d^{lc})[(r_2+1)P_d^{nc} + (1-P_d^{nc}) \\ [P_p^{cf}(1-P_p^x)L_p^{cf} + [1-P_p^{cf}(1-P_p^x)][r_2 + P_f^{cf}\sum_{i=1}^{L_s-r_2-1} i(1-P_f^{cf})^{i-1} \\ + (L_s-r_2)(1-P_f^{cf})^{L_s-r_2-1}]]] + R_a^c(4r_2-3)] \end{aligned} \tag{4.12}$$

CFCC 방법에서, 요청된 데이터가 로컬 캐시에 있을 확률은 다음과 같이 계산되어진다.

$$P_d^{cf} = P_{cf}^{cd} + P_{pd}^h$$

여기서 P_{pd}^h는 식 (4.13)에 의해 계산되어진다.

$$P_{pd}^h = \frac{\lambda \times P_{xpd}^r \times P_d^p}{TTL} \tag{4.13}$$

우리는 데이터 액세스 율은 Zipf-like 분포 (Breslau 등, 1999)를 따른다고 가정한다. Zipf-like 분포에서 i-번째 인기 있는 데이터 아이템의 액세스 확률을 계산하는 수식은 다음과 같다.

$$P(i) = \frac{1}{i^\alpha \sum_{k=1}^{n} \frac{1}{k^\alpha}}$$

여기서 매개변수 n은 데이터 아이템의 전체 개수이고, α는 분포의 기울어진 정도를 나타낸다. α가 클수록 더 기울어진 액세스 분포가 발생한다.

그림 4.2는 n=100이고 $\alpha = 0.8$인 경우의 Zipf-like 분포를 보여준다. 여기서 인기 있는 아이템의 인기도 지수의 임계값을 0.015로 하면, 전체 100개의 데이터 아이템 중에서 13개의 인기 있는 데이터 아이템만이 있고, 인기 있는 데이터 아이템을 액세스할 확률의 합은 0.49가 된다.

선 인출되는 데이터들은 인기 있는 데이터이기 때문에 P_{xpd}^r을 0.9라 가정하고, 그리고 데이터 액세스 율 λ를 10, TTL을 200이라 가정하면, P_{pd}^h는 식 (4.13)에 의해 다음과 같이 구해진다.

$$P_{pd}^h = \frac{10 \times 0.9 \times 0.49}{200} \approx 0.022$$

CFCC 방법의 성능 평가를 위하여 표 4.2와 같은 매개변수와 값을 사용하여 데이터 요청의 평균 응답 지연 시간과 평균 에너지 비용을 평가한다. 여기서 평균 응답 지연은 데이터 요청 메시지가 운행한 홉 개수에 비례하기 때문에 데이터 요청 메시지가 운행한 홉의 개수로 평가하였고, 평균 에너지 비용은 캐시 해결을 위하여 전송된 데이터 요청 메시지 개수에 비례하기 때문에 전송된 요청 메시지 개수로 평가하였다.

그림 4.3은 $r = 3$일 때, 데이터 서버 거리에 따른 요청 메시지의 평균 지연을 보여준다. 여기서 TTL이 만기된 인기 있는 데이터 아이템을 선인출하여 로컬 캐시 적중률을 향상시켰고, 아울러 이웃 클러스터 헤드들 간의 캐시 정보 교환으로 이웃 클러스터로부터 캐시 해결이 되기 때문에 데이터 서버의 거리에 관계없이 CFCC의 성능이 다른 모바일 캐싱 방법들에 비해 성능이 우수함을 확인하였다.

그림 4.4는 $L_s = 20$ 이고 $R_a^c = 0.2$일 때, 협력지역 반경에 따른 예상 에너지 비용을 보여준다. 여기서 지역 기반 캐시 방법과 COOP 방법은 협력지역의 반경이 커질수록 예상 에너지 비용도 선형적으로

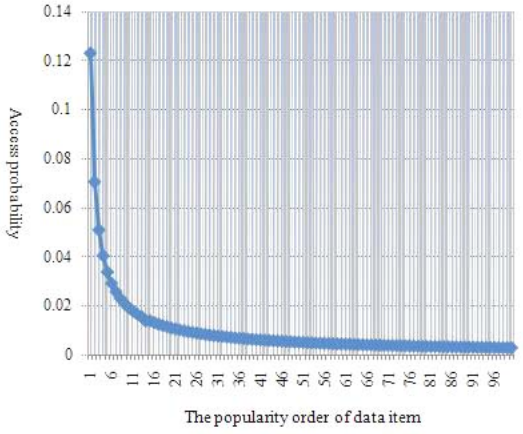

그림 **4.2** Zipf-like 분포 (n=100, α =0.8)

표 **4.2** 성능 평가를 위한 매개변수와 값

매개변수	값
P_d^s, P_d^h, P_d^l, P_d^{hc}, P_d^{co}	0.5
P_d^{cf}	0.522
P_f^h, P_f^{cf}, P_d^z, P_d^{cz}	0.03
P_p^{hc}, P_p^{cf}	0.2
P_d^{lc}	0.15
P_d^{nc}	0.1
P_p^x	0.01
r (hops)	3~6
L_p^{hc}, L_p^{cf} (hops)	4
ρ	0.5
R_a^c	0.15~0.45
L_s (hops)	10~25

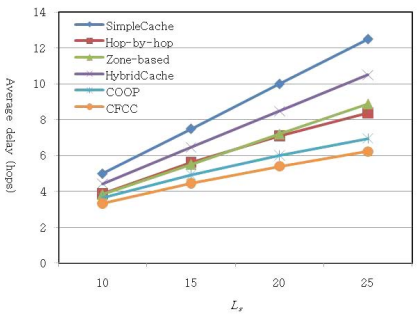

그림 **4.3** 데이터 서버 거리에 따른 평균 지연

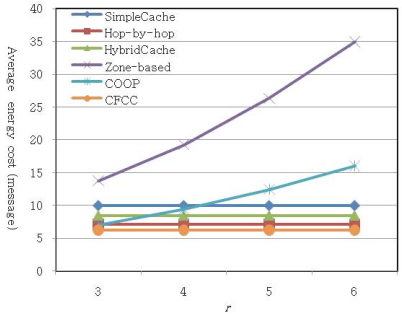

그림 **4.4** 협력지역 반경에 따른 평균 에너지 비용

증가함을 알 수 있다. 그러나 CFCC에서는 클러스터 크기가 고정되어 있고, 로컬 캐시와 로컬 클러스터에서 캐시 해결이 실패한 경우에 협력지역에 요청 메시지를 플로딩하는 것이 아니라 이웃 클러스터 헤드들에게만 요청 메시지를 전송하기 때문에 전송되는 메시지 개수가 줄어들어 든다. 따라서 CFCC 방법이 협력지역의 반경에 관계없이 가장 낮은 평균 에너지 비용을 보인다.

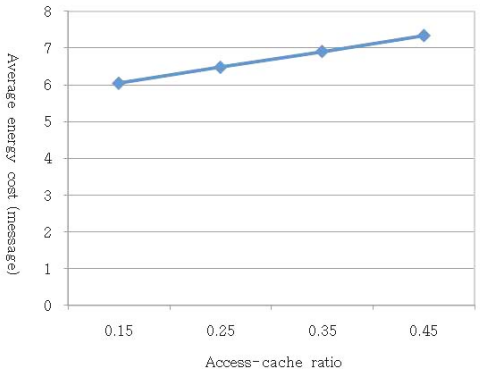

그림 4.5 액세스-캐시 율에 따른 CFCC 방법의 평균 에너지 비용

그림 4.5는 R_a^c에 따른 CFCC 방법의 평균 에너지 비용을 보여준다. 액세스-캐시 비율이 증가할수록 모바일 노드와 로컬 CH, 로컬 CH와 이웃 CH간의 캐시 상태 정보를 포함하는 메시지 전송에 기인하여 증가하는 에너지 비용은 매우 작음을 알 수 있었다.

5. 향후연구 및 결론

애드-혹 망에서, 빈번한 망 분할에 기인하여, 데이터 가용성은 일반적인 유선망의 데이터 가용성 보다 낮다. 이 문제는 모바일 호스트 상에 데이터 아이템들을 캐싱하여 해결할 수 있다. 그러나 노드들의 이동성, 제한된 저장 공간 그리고 빈번한 단절은 가용성을 제한한다. 애드-혹 망에서 자주 액세스 되는 데이터의 캐싱으로 데이터 액세스 성능 그리고 가용성을 향상시킬 수 있다.

본 논문에서는 모바일 애드-혹 망을 위한 데이터 캐싱을 위한 클러스터 기반 퍼지 협력 캐싱, CFCC 방법을 제안한다. CFCC에서, 망 위상은 물리적 망 근접성에 기초하여 겹치지 않는 클러스터로 분할하고, 각 노드는 데이터 캐싱을 위해 노드 활동 상태와 데이터 유용성에 따른 퍼지 논리에 기초하여 CacheData 방법 또는 CachePath 방법을 적응적으로 사용한다. 어떤 MH의 정보 검색 연산은 로컬 캐시, CHs, 경로 기반 노드, Roadside 노드들, 그리고 마지막으로 데이터 서버 순으로 데이터 아이템을 요청한다. 모바일 애드-혹 망의 데이터 캐싱을 위한 클러스터 기반 퍼지 협력 캐싱, CFCC 방법의 성능을 분석적 모델로 평가한 결과 CFCC 방법은 협력지역의 반경에 관계없이 가장 낮은 평균 에너지 비용과 데이터 서버 거리에 따른 요청 메시지의 평균 지연에서 가장 낮은 값을 나타냈다. 따라서 제안한 CFCC 방법의 성능이 기존의 다른 모바일 캐싱 방법들에 비해 데이터 요청 메시지의 평균 지연과 에너지 비용이 모두 우수함을 확인하였다.

향후 연구과제로는 모바일 애드-혹 망을 위한 데이터 캐싱을 위한 클러스터 기반 퍼지 협력 캐싱, CFCC 방법의 성능을 다양한 매개변수의 변화에 따른 모의실험을 통하여 분석 평가하는 것이다.

참고문헌

Bae, I. H. (2009). Design and analytical evaluation of a fuzzy proxy caching wireless internet. *Journal of the Korean Data & Information Science Society*, **2**, 1177-1190.

Bae, I. H. (2010). Design and evaluation of a fuzzy cooperative caching scheme for MANETs. *Journal of the Korean Data & Information Science Society*, **21**, 605-619.

Breslau, L., Cue, P., Cao, P., Fan, L., Phillips, G. and Shenker, S. (1999). Web caching and zipf-like distributions: Evidence and implications. *IEEE International Conference on Computer Communications*, 126-134.

Cao, G. and Yin, L. (2004). Cooperative cache based data access in ad hoc networks. *IEEE Computer*, **37**, 32-39.

Chand, N., Joshi, R. C. and Misra, M. (2007). Cooperative caching strategy in mobile ad hoc networks based clusters. *Wireless Personal Communication*, **43**, 41-63.

Chow, C-Y., Leong, H. V. and Chan A. (2004). Peer-to-peer cooperative caching in mobile environments. *International Conference on Distributed Computing Systems Workshops*, 528-530.

Denko, M. K., Jun, T., Nkwe, T. and Obaidat, M. S. (2009). Cluster-based cross-layer design for cooperative caching in mobile ad hoc networks. *IEEE System Journal*, **3**, 499-508.

Du, Y., Gupta, S. K. S. and Varsamopoulos, G. (2009). Improving on-demand data access efficiency in MANETs with cooperative caching. *Ad Hoc Networks*, **7**, 579-598.

Du, Y. and Gupta, S, K. S. (2005). COOP - A cooperative caching service in MANETs. *Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services*, 58-63.

Kumar, P. and Chauhan, N. (2010). A review of cooperative cache management techniques in MANETs. *International Conference on Recent Trends in Soft Computing and Information Technology*, 1-7.

Li, J., Blake, C., De Couto, D. S. J., Lee, H. I. and Morris, R. (2001). Capacity of ad hoc wireless networks. *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, 61-69.

Yin, L. and Cao, G. (2006). Supporting cooperative caching in ad hoc networks. *IEEE Transactions on Mobile Computing*, **5**, 77-89.

Journal of the Korean Data & Information Science Society
2011, **22**(2), 269–285

한국데이터정보과학회지

Design and evaluation of a cluster-based fuzzy cooperative caching method for MANETs

Eun-Ju Lee[1] · Ihn-Han Bae[2]

[12]School of Computer and Information Communications Engineering,
Catholic University of Daegu

Received 5 February 2011, revised 17 March 2011, accepted 22 March 2011

Abstract

Caching of frequently accessed data in mobile ad-hoc networks is a technique that can improve data access performance and availability. Cooperative caching, which allows sharing and coordination of cached data among several clients, can further enhance the potential of caching techniques. In this paper, we propose a cluster-based fuzzy cooperative caching method for mobile ad-hoc networks. The performance of the proposed caching method is evaluated through an analytical model and is compared to that of other cooperative caching methods.

Keywords: Clustering, cooperative caching, fuzzy logic, mobile ad-hoc networks.

[1] Graduate student, Department of Computer and Information Communication Engineering, Catholic University of Daegu, Gyeongbuk 712-702, Korea.
[2] Corresponding author: Professor, School of Computer and Information Communication Eng., Catholic University of Daegu, Gyeongbuk 712-702, Korea. E-mail: ihbae@cu.ac.kr