MANET에서 데이터 가용성 향상을 위한 캐시 기법

양환석*

요약

MANET(Mobile Ad-hoc Network)은 이동 노드로만 구성된 형태로서 다중 홉 통신을 이용하여 서로 통신하기 때문에 모든 노드가 데이터 전달 기능을 수행하는 라우터 역할을 해야 한다. MANET에서 노드들의 이동성과 다중 홉 등 제한된 시스템 자원으로 인해 데이터 접근 성능이 낮아지는 단점을 가지고 있다. 이러한 데이터 접근 성능과 가용성을 향상시킬 수 있는 방법이 캐싱 기법이다. 본 논문에서는 데이터 검색을 위한 쿼리 지연시간을 줄이고 노드들이 저장하고 있는 중복된 데이터들의 일관성을 높임으로써 데이터 가용성을 향상시키기 위한 캐시 기법을 제안하였다. 제안한 기법의 성능 평가를 위하여 GC, COOP기법과 비교실험 하였으며, 실험을 통해 제안한 기법의 효율성을 확인할 수 있었다.

Cache Technique for Improvement Data Availability in Mobile Ad-hoc Networks

Hwan-Seok Yang*

ABSTRACT

MANET(Mobile Ad-hoc Network) is consisted of only mobile nodes. Every node plays a role as router which performs data transfer function because it communicates each other using multi-hop communication, It has disadvantage that data access performance gets lower by moving of nodes, multi-hop, and limited system resource. Caching scheme improves this data access performance and availability. In this study, we proposed cache scheme to improve data availability by enhancing consistency of overlapping data which nodes are saving and decreasing query delay time to search data. We compared with GC, COOP method and experienced to evaluate performance of proposed method and identified efficiency of this method.

Key Words : Cache Technique, Clustering, Data Availability, Cache hit, Mobile Ad-hoc Network

* 중부대학교 정보보호학과(✉yanghs@joongbu.ac.kr)
· 제1저자(First Author) : 양환석 · 교신저자(Correspondent Author) : 양환석
· 접수일(2012년 8월 23일), 수정일(1차 : 2012년 10월 16일), 게재확정일(2012년 10월 19일)

Ⅰ. 서 론

MANET(Mobile Ad-hoc Network)은 기지국이나 기존의 다른 네트워크 인프라스트럭처 도움 없이 무선 이동 노드로만 구성된 네트워크이다[1]. MANET을 구성하는 이동 노드들은 무선 다중 홉 통신을 이용하여 서로가 통신 할 수 있기 때문에 모든 노드들은 라우터와 같은 데이터 전달 기능을 수행해야만 한다. 하지만 노드들의 이동으로 인해 네트워크 토폴로지가 수시로 변화하기 때문에 유효한 경로 유지가 쉽지 않고 데이터 전송 지연시간이 높다. 그리고 무선 통신의 낮은 대역폭과 이동 노드들이 제한된 자원 및 연산 능력 때문에 효율적인 자원의 이용이 반드시 필요하다[2]. MANET에서는 노드들 간의 연결을 위한 라우팅 기술도 중요하지만 그 보다 이동 노드에서 제공되는 데이터에 접근이 무엇보다 중요하다고 할 수 있다. 따라서 이동 노드가 원하는 데이터에 빠르게 접근할 수 있도록 성능을 향상시키기 위해서는 캐싱 기법이 매우 중요하다[3-6].

본 논문에서는 이동 노드들의 제한된 캐시 크기와 데이터 검색을 위한 쿼리 지연 시간을 줄여서 데이터 가용성을 향상시키기 위한 캐시 기법을 제안하였다. 제안한 기법은 MANET을 구성하는 노드들을 클러스터 형태로 구성하였고, 이동 노드들은 로컬 캐시 테이블을 갖고 있으며 클러스터 헤드는 멤버 캐시 테이블과 클러스터 캐시 테이블을 관리하도록 하였다. 그리고 데이터의 유효성 검사를 위하여 TTL을 사용하였다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 캐싱 기법에 대하여 살펴보고 3장에서는 본 논문에서 제안한 캐싱 기법에 대하여 기술하였다. 4장에서는 제안한 기법의 성능을 GC, COOP기법과 비교실험을 통해 평가하였으며 마지막으로 5장에서는 결론을 맺는다.

Ⅱ. 관련연구

Group caching 기법은 각 노드들이 주기적으로 hello 메시지를 송신하여 자신과 1-홉 거리에 있는 이웃 노드들과 그룹을 형성한다[7]. 이렇게 형성된 그룹에는 마스터 노드가 있으며, 멤버 노드들과 직접 통신을 한다. 그룹 내의 각 노드의 캐시 공간을 활용하기 위하여 마스터 노드는 캐시 제어 메시지를 사용하여 그룹 멤버 노드들의 캐싱 상태를 점검한다. 이러한 방법 때문에 캐시 공간에 저장되는 데이터의 중복성이 낮아지고 데이터 접근성을 향상시킬 수 있게 되었다. 멤버 노드들은 캐싱 상태를 저장하기 위하여 self table과 group table을 사용한다. 이동 노드가 데이터 요청을 수신하면 가장 먼저 자신의 self table을 검색한다. 만약 여기에서 원하는 정보를 검색하지 못한다면 group table에 있는 그룹 멤버들에게 요청을 한다. 이와 같은 방법으로 원하는 데이터의 정보를 찾기 위해 인접한 그룹으로 요청 메시지를 전달하게 된다. <그림 1>은 group caching 기법의 캐시 검색 과정을 보여주고 있다.

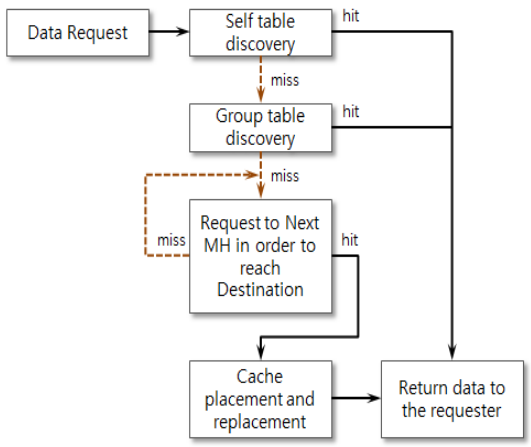

그림 1. GC 기법의 요청 처리 과정
Fig. 1. Process of a request by GC

이 기법에서 캐시에 데이터를 저장하는 방법은 크게 두 가지로 이루어진다. 첫 번째로 이동 노드에 충분한 캐시 공간이 있다면 수신한 데이터를 저장한다. 하지만 그렇지 않다면 그룹 멤버들의 캐시 상태를 검사하여 캐시 공간이 여유가 있는 멤버 노드에 데이터를 저장하게 된다. 두 번째로 만약 모든 그룹 멤버에 수신한 데이터를 저장하기에 캐시 공간이 충분하지 못하다면 그룹 테이블을 검사하여 수신한 데이터의 저장 여부를 확인하게 된다. 만약 데이터가 저장되어 있지 않다면 group table에 저장되어 있는 데이터 중에서 가장 오래된 TTL을 갖는 데이터와 교체하게 된다. 이 기법에서는 이웃 노드의 캐시 상태 정보와 그룹을 떠나거나 들어오는 노드들에 대한 정보를 업데이트하기 위해 주기적으로 hello 메시지를 보내야하기 때문에 노드들의 에너지 소모가 증가하게 된다. 그리고 제어 메시지의 양이 증가하여 네트워크의 성능이 떨어지는 단점이 있다.

Yu Du가 제시한 COOP 기법은 데이터 접근에 대한 협동 캐시 구조이다[8]. 이 기법은 이동 노드의 로컬 자원들을 협력하도록 하여 접근 효율성과 데이터 가용성을 향상시키는 것이다. 캐시 노드들의 협력은 두 단계로 이루어진다. 첫 번째로 캐싱 노드는 다른 노드들로부터 데이터 요청에 응답할 수 있다. 두 번째로 캐싱 노드는 자신이 필요할 때나 다른 노드들의 필요에 의해서나 언제든 데이터를 저장할 수 있다. 이 기법에서의 목표는 기존의 캐시 기법에서 시간, 에너지 그리고 대역폭에 대한 최소한의 비용으로 데이터 요청을 처리할 수 있는 방법과 로컬 캐시에 데이터 저장 여부를 결정하는 캐시 관리 방법이다. 먼저 통신비용을 줄이기 위하여 프로파일과 포워딩 노드들을 활용하여 데이터 소스 검색 방법을 이용하였다. 그리고 이웃 노드들 간의 캐시 복사를 최소화시키고 전체적인 성능이 향상될 수 있도록 보다 많은 데이터를 저장하는 협동 캐시를 사용하였다. 이 기법의 단점은 캐시 교체시 크기나 일관성과 같은 요소를 고려하지 않았고 데이터를 검색할 때 플러딩에 대한 높은 오버헤드가 발생하게 된다. <그림 2>는 COOP 구조를 보여주고 있다.

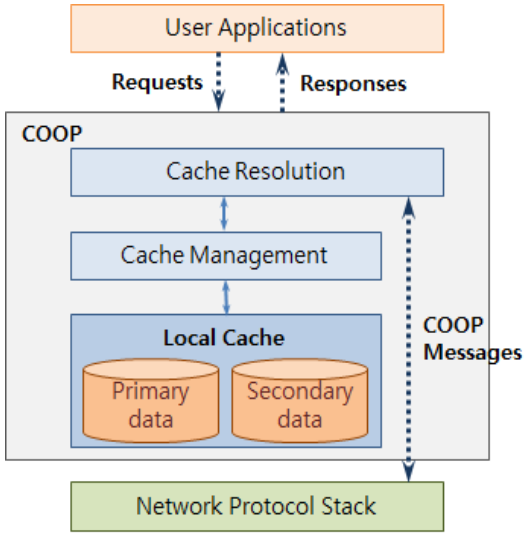

그림 2. COOP 구조
Fig. 2. COOP Architecture

III. 제안한 기법

본 장에서는 정보 접근의 효율성을 향상시키고 접근 지연과 대역폭 사용을 줄일 수 있는 캐싱 기법에 대하여 기술한다.

3.1 클러스팅과 캐시 테이블

MANET을 구성하는 노드들의 이동으로 인한 캐시 업데이트 횟수 증가를 막고 캐시 테이블의 관리를 효율적으로 수행하기 위하여 본 논문에서는 클러스터 구조를 이용하였다. 먼저 전체 네트워크를 구성하는 노드들을 1-홉 거리에 있는 이웃 노드들을 기초로 하여 클러스터를 구성하였다. 노드들의 배터리 양과 연결 수를 근거로 클러스터 헤드를 선출하게 된다. 클러스터 헤드가 많은 노드와 연결되어 있기 때문에 노드

들의 이동으로 인해 발생하는 데이터 요청에 대한 오버헤드를 줄일 수 있다. 그리고 클러스터 기법을 사용함으로서 캐시 테이블 관리가 용이하게 된다. 네트워크를 구성하는 모든 멤버 노드들은 로컬 캐시 테이블을 가지고 있다. 그리고 선출된 클러스터 헤드는 자신의 클러스터내의 멤버 노드들에게 정보를 효율적으로 제공해주기 위하여 멤버 캐시 테이블과 클러스터 캐시 테이블을 갖는다. 먼저 멤버 노드들은 여러 노드들이 같은 데이터를 저장하고 있을 때 이 데이터들에 대한 일관성이 문제가 된다. 따라서 로컬 캐시 테이블은 데이터 항목과 데이터 항목이 성공적으로 저장되었을 때의 시간을 저장하고 있다. 그리고 클러스터 헤드가 갖고 있는 멤버 캐시 테이블은 멤버 노드들에 의해 캐시된 데이터 항목에 대한 자세한 정보를 담고 있으며 클러스터 캐시 테이블은 이웃 클러스터들의 캐시 데이터 정보를 저장하고 있다. <그림 3>은 각 캐시 테이블의 구조를 보여주고 있다.

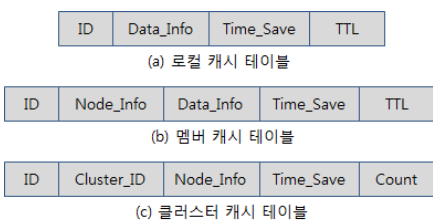

그림 3. 캐시 테이블 구조
Fig. 3. Cache Table Structure

<그림 3>과 같은 캐시 테이블을 활용함으로써 GC 기법과 COOP 기법에 비해 캐시 테이블의 관리가 잘 이루어지기 때문에 캐시 일관성 유지에 큰 장점이 있다.

3.2 쿼리 처리

멤버 노드들이 수행하는 쿼리 처리는 크게 다른 노드의 데이터 요청 쿼리를 수신하였을 때와 노드 자신이 데이터를 요청하였을 때로 나누어 볼 수 있다. 먼저 다른 노드로 부터 데이터 D_i에 대한 요청을 수신하였을 때는 자신의 로컬 캐시 테이블을 확인한다. 로컬 캐시 테이블에서 요청한 데이터 D_i를 찾는다면 해당 데이터 D_i의 저장된 시간(Time_Save)을 데이터 요청을 수신한 시간에서 뺀 값이 TTL보다 작으면 해당 데이터가 유효한 것으로 간주하여 해당 데이터에 대한 정보를 전송해준다. 하지만 TTL 보다 값이 크다면 해당 데이터 D_i의 모든 정보를 로컬 캐시 테이블로부터 삭제하게 된다. 그리고 해당 데이터 D_i에 대한 요청을 자신이 속한 클러스터 헤드에게 전달하게 된다. 만약 해당 클러스터 헤드로부터 요청한 데이터 D_i 정보를 수신하게 된다면 자신의 로컬 캐시 테이블을 갱신하게 된다. 이와 같은 방법으로 노드들은 자신이 갖고 있는 로컬 캐시 테이블을 갱신하게 된다. <그림 4>는 앞에서 설명한 데이터 요청 쿼리를 수신하였을 때 처리 과정을 보여주고 있다.

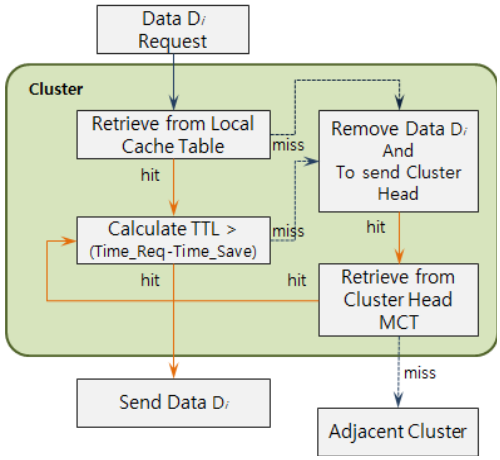

그림 4. 요청 쿼리 처리과정
Fig. 4. Request query processing

이동 노드 자신이 데이터 D_i가 필요할 때는 가장 먼저 자신의 로컬 캐시 테이블을 검사한다. 여기서 원하는 데이터 D_i를 찾지 못한다면 자신의 클러스터 헤드에게 요청 쿼리를 전송한다. 이를 수신한 클러스터 헤

드는 클러스터내의 이동 노드들에 대한 정보가 담겨 있는 멤버 캐시 테이블을 검사하게 된다. 요청한 데이터 D_i를 찾는다면 데이터 요청 시간에서 데이터의 저장된 시간(Time_Save)를 뺀 후 TTL값보다 작으면 해당 데이터 D_i의 정보를 이동 노드에게 전달한다. 하지만 TTL값보다 크다면 해당 멤버 노드에게 정보를 다시 요청한 후 멤버 캐시 테이블의 내용을 업데이트하고 요청 노드에게 해당 데이터 D_i의 정보를 전송한다. 만약 멤버 캐시 테이블에서 요청한 데이터 D_i에 대한 정보를 찾지 못하게 된다면 클러스터 캐시 테이블을 이용하여 인접한 클러스터 헤드에게 요청 쿼리를 송신한다. 인접한 클러스터로부터 해당 데이터 D_i의 정보를 수신하면 이를 요청한 노드에게 전달해 주고 자신의 멤버 캐시 테이블을 업데이트하여 테이블을 관리하게 된다. <그림 5>는 캐시 검색 과정의 의사 코드를 보여주고 있다.

```
IF MHi request data Di
Then
 Search data Di from LCT
 IF LCT not contains requested data Di
 Then
  SendRequestClusterHead(Di)
 End IF
End IF
While Di > 0 Do
 IF Data Di found from MCT
 Then
  Valid = Time_Req - Time_Save
  IF TTL > Valid
  Then
   SendtoMHi(Di)
  Else
   SendRequestMobileHost(Di)
   Update MCT
  End IF
 End IF
End While
```

그림 5. 캐시 검색 의사 코드
Fig. 5. Pseudo-code of the Algorithm

GC 기법은 그룹 내에서 캐시 검색을 실패한 경우와 업데이트시 많은 제어 메시지가 발생하고 지연시간이 길어지는 단점이 있으며, COOP 기법도 이웃 노드들과 협력을 이용한 데이터 가용성은 높였지만 쿼리 요청시 많은 오버헤드가 발생한다. 그러나 제안한 기법은 클러스터 헤드가 관리하는 두 개의 테이블에 의해 쿼리 처리가 이루어지기 때문에 제어 메시지를 줄여 네트워크 성능을 향상시키고 지연 시간을 줄이는 장점이 있다.

IV. 모의실험 및 결과

4.1 모의실험 환경

본 논문에서 제안한 기법의 효율성을 평가하기 위하여 GC 기법, COOP 기법과 비교 실험하였다. 모의실험을 위해 ns-2 시뮬레이터를 이용하였고 <표 1>에서 제시한 환경 변수 값을 사용하였다.

표 1. 실험에 사용한 환경변수
Table 1. Simulation parameters

Parameter	Value
Network size	1500m × 1500m
Mobility model	Random waypoint
Number of nodes	50 to 150
Transmission range	200m
Bandwidth(MB)	2
Cache size(KB)	200 to 1000
Average Query Rate(Sec)	0.5
TTL(Sec)	200 to 400
Pause time(Sec)	30

4.2 결과

본 논문에서 제안한 기법의 성능을 평가하기 위하여 캐시 크기와 노드 수에 따른 캐시 적중률, 평균 쿼리 지연시간을 평가 기준으로 하였다. 캐시 적중률은 데이터를 요청한 노드의 로컬 캐시 테이블과 클러스

터 헤드의 멤버 캐시 테이블과 클러스터 캐시 테이블의 적중률을 모두 합한 값이다. 그리고 평균 쿼리 지연시간은 노드가 데이터 요청 쿼리를 발생한 후 그에 대한 응답을 수신한 시간을 계산한 것이다.

<그림 6>과 <그림 7>은 캐시 크기에 따른 캐시 적중률과 평균 쿼리 지연시간을 보여주고 있다. <그림 6>에서도 보듯이 세 가지 기법 모두 캐시 크기가 증가할수록 적중률도 증가하였다. COOP 기법은 캐시 교체시 크기나 일관성을 고려하지 않기 때문에 성능이 가장 좋지 않았고, GC 기법은 그룹 멤버들의 캐시 테이블을 사용하기 때문에 다소 좋은 결과를 보였다. 제안한 기법에서는 정보 접근과 일관성을 향상시키기 위하여 TTL 계산을 이용한 캐시 테이블의 내용을 관리하고 클러스터 헤드에 의한 캐시 테이블 관리가 이루어져 가장 좋은 성능을 보였다.

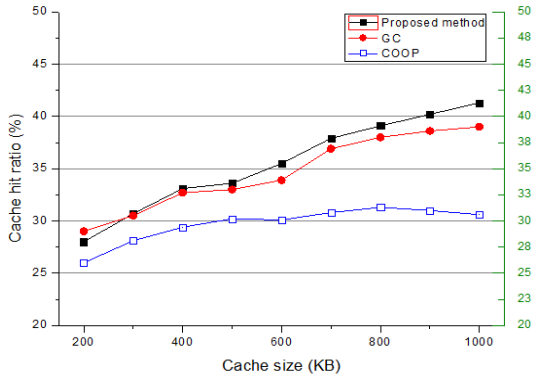

그림 6. 캐시 크기에 따른 캐시 적중률
Fig. 6. Cache hit ratio under different cache size

<그림 7>에서 평균 쿼리 지연시간은 제안한 기법의 성능이 가장 좋게 나타나는 것을 확인할 수 있다. 캐시 크기가 작을수록 로컬 캐시 테이블과 클러스터 헤드가 갖고 있는 캐시 테이블의 검색 시간이 길어져 지연시간이 높게 나타나는 것을 볼 수 있다. 그리고 COOP 기법은 데이터 검색 시 플러딩으로 인한 높은 오버헤드가 발생하여 가장 나쁜 결과를 보였다.

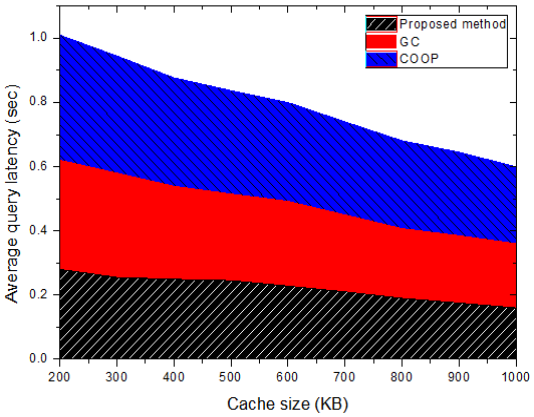

그림 7. 캐시 크기에 따른 평균 쿼리 지연
Fig. 7. Average query latency under different cache size

<그림 8>과 <그림 9>는 캐시 크기를 600KB로 하였을 때, 노드 수에 따른 캐시 적중률과 평균 쿼리 지연시간을 보여주고 있다. <그림 8>에서 캐시 적중률은 노드의 수에 비례하였는데 이는 노드의 수가 증가할수록 클러스터내의 노드 밀집도가 증가하여 클러스터 헤드의 캐시 테이블에 보다 많은 데이터 정보를 저장할 수 있었기 때문이다.

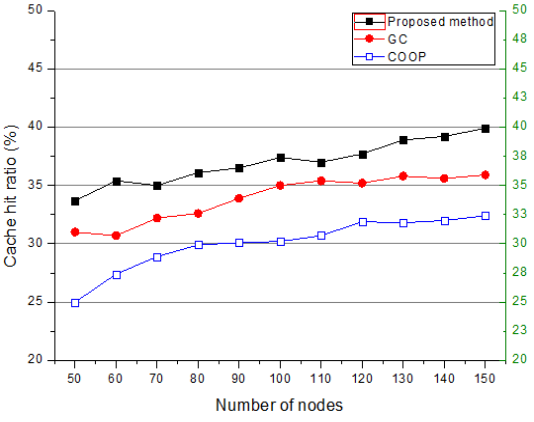

그림 8. 노드 수에 따른 캐시 적중률
Fig. 8. Cache hit ration under different number of nodes

<그림 9>의 평균 쿼리 지연시간은 노드의 수가 증가할수록 제한된 대역폭으로 인해 지연시간이 증가하는 것을 보였다. 제안한 기법이 다른 두 기법에 비해 지연시간이 낮은 이유는 전체 노드들의 수가 많아질수록 클러스터내의 노드들 수가 많아져 빠른 데이터 검색이 이루어졌기 때문이다.

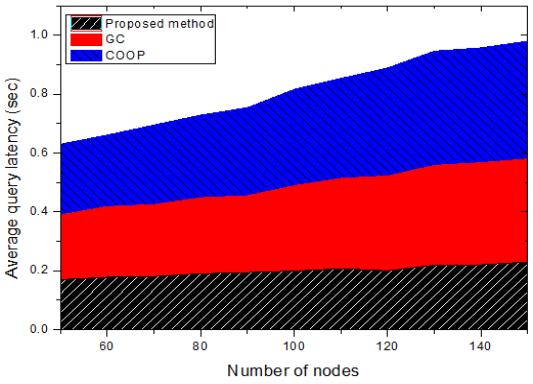

그림 9. 노드 수에 따른 평균 쿼리 지연
Fig. 9. Average query latency under different number of nodes

V. 결 론

본 논문에서는 제한된 캐시 크기를 갖는 이동 노드들의 데이터 검색 시간을 줄이고 데이터 가용성을 향상시키기 위한 기법을 제안하였다. 모든 이동 노드들은 로컬 캐시 테이블을 갖고 있으며 클러스터 헤드는 멤버 캐시 테이블과 클러스터 캐시 테이블을 갖고 있다. 네트워크내의 많은 노드들이 같은 데이터를 저장하고 있을 때 이들에 대한 일관성을 증가시키기 위하여 각 데이터에 대한 TTL을 계산하였다.

각 노드들은 데이터 요청을 수신하였을 때 데이터를 저장한 시간과 요청 시간을 뺀 후 TTL과 비교하여 데이터 정보를 유지할 것인지 새롭게 갱신할 것인지를 결정한다. 캐시 업데이트 결과를 클러스터 헤드의 멤버 캐시 테이블에서도 저장, 관리함으로써 데이터의 가용성 및 캐시 적중률이 향상되는 것으로 나타났다.

향후 연구로는 캐시 공격에 안전성을 보장하기 위한 보안 기법들에 대해 연구되어져야 할 것이다.

참고문헌

[1] M. Frodigh, P. Johansson, and L. Larsson, "Wireless Ad Hoc Networking" *Erricsson Review*, No. 4, 2000.

[2] J. P. Walters, Z. Liang, W. Shi, and V. Chaudhary,"Wireless Sensor Network Security: A Survey," *Security in Distributed Grid and Pervasive Computing*, pp.1-50, 2006.

[3] L. Yin, and G Cao, "Supporting Cooperative Caching in Ad Hoc Networks," *IEEE INFOCOM*, pp.2537-2547, 2004.

[4] Takahiro Hara, "Replica Allocation Methods in Ad Hoc Networks with Data Update," *Kluwer Journal of Mobile Networks and Applications*, Vol. 8, No. 4, pp.343-354, 2003.

[5] G. Cao, "A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 15, No. 5, pp.175-186, 2004.

[6] S. Lim, W. C. Lee, G. Cao, and C. R. Das, "A Novel Caching Scheme for Improving Internet Based Mobile Ad Hoc Networks Performance," *Elsevier Journal of Ad Hoc Networks*, Vol. 4, No. 2, pp.225-239, 2006.

[7] N. Chand, R. C. Joshi, and M. Misra, "Efficient Cooperative Caching in Ad Hoc Networks," *IEEE Transactions*, pp.1-8, 2006.

[8] Yu Du, and S. Gupta, "Improving On-demand Data Access Efficiency in MANETs with Cooperative Caching," *Ad hoc Networks, Elsevier*, pp.579-598, 2009.

저자소개

양환석(Hwan-Seok Yang)

1998년 조선대학교 대학원 전산통계학과 (이학석사)
2005년 조선대학교 대학원 전산통계학과 (이학박사)

2011년~현재 중부대학교 정보보호학과 조교수
※ 관심분야 : 정보보호, 모바일 보안, 시스템 보안