情報保護學會論文誌
第15券 第2號, 2005. 4

무선랜에서 낮은 지연 특성을 가지는 인증유지 핸드오프 기법과 트래픽 관리 기법

최 재 우,† 강 전 일,‡ 양 대 헌
인하대학교

Authenticated Handoff with Low Latency and Traffic Management in WLAN

Jae-woo Choi,† Jeon-il Kang,‡ Dae-hun Nyang
Inha University

요 약

최근 무선랜 환경은 우리 근처에 널리 퍼져있고 많은 사람들이 PDA를 비롯한 multimedia application들과 같은 휴대장치를 많이 사용함으로써 단말 장비들의 이동이 계속 늘어나고 있다. 그러나 이러한 단말 장비들이 이동하여 현재 접속되어 있는 AP(Access Point)를 벗어나 다른 AP와 접속을 시도할 때 두 AP들 사이에는 핸드오프 지연이 발생한다. 이러한 핸드오프 지연을 줄이기 위해 이 논문에서는 효과적인 데이터 구조를 사용하는 WFH(Weighted Frequent Handoff)를 제안한다. WFH는 FHR(Frequent Handoff Region)에서 클라이언트의 이동확률 개념을 도입한 새로운 cache replacement algorithm을 사용하여 cache hit ratio를 높이고 토폴로지에서 불필요하게 중복되는 트래픽도 줄여준다. 이 알고리즘은 QoS를 기반으로 하는 클라이언트의 레벨과 이동 패턴에 따라서 인증되는 범위가 달라지는 FHR과 네트워크 이동 토폴로지를 동적으로 캡쳐하는 neighbor graph을 이용하고 있다.

ABSTRACT

Recently, wireless LAN circumstance is being widely deployed in public spots. Many people use portable equipments such as PDA and laptop computer for multimedia applications, and also demand of mobility support is increasing. However, handoff latency is inevitably occurred between both APs when clients move from one AP to another. To reduce handoff latency, in this paper, we suggest WFH(Weighted Frequent Handoff) using effective data structure. WFH improves cache hit ratio using a new cache replacement algorithm considering the movement pattern of users. It also reduces unessential duplicate traffics. Our algorithm uses FHR(Frequent Handoff Region) that can change pre-authentication region according to QoS based user level, movement pattern and Neighbor Graph that dynamically captures network movement topology.

Keywords : *handoff, proactive caching, traffic, cache-hit, WLRU(Weighted Least Recently Used)*

Ⅰ. 서 론

IEEE 802.11 standard (Wi-Fi)를 기본으로 하는 무선 네트워크는 낮은 비용과 규제되지 않은 대역폭 때문에 빠른 성장을 하고 있고 무선장비를 사용하는 사용자들이 늘어나고 있다. 사용자들이 무선장비를 이용하는 가장 큰 이유는 무선 서비스가 제공되는 범위에서 장소에 구애받지 않고 이동하면서 무선 서비스를 이용할 수 있다는 것이다. 많은

접수일 : 2005년 2월 15일 ; 채택일 : 2005년 3월 29일
† 주저자 : elvin0@seclab.inha.ac.kr
‡ 교신저자 : dreamx@seclab.inha.ac.kr

데이터량을 요구하지 않는 서비스들은 핸드오프가 발생했을 때 사용자들이 불편함을 느끼지 못하지만 voice와 기타 multimedia 서비스는 실시간으로 많은 데이터 량을 요구하므로 핸드오프 지연으로 인한 끊김 현상에 대해 불편함을 느낄 수 있다. 그러므로 AP들 사이에서의 핸드오프 처리는 빨리 수행되어져야 하며 IAPP(Inter-Access Point Protocol)를 통해서 핸드오프가 일어난 뒤에도 클라이언트와 관련된 여러 상태 정보들이 핸드오프가 일어나기 이전의 상태 정보와 같아야 한다. 즉, 클라이언트들의 세션과 QoS 그리고 security context information이 핸드오프가 일어나기 전과 동일한 상태로 유지되어야 하는 것이다.

핸드오프의 전체 지연을 줄이기 위한 한 가지 방법은 proactive 기법을 이용하여 미리 mobile 클라이언트의 security context information을 이동할 이웃 AP들에게 보내는 것이다. 그러나 proactive방식에는 한 가지 문제점이 있다. 그것은 "클라이언트가 어디로 이동할지 모르는 상황에서 이웃 AP들을 어떻게 미리 적절하게 선택하느냐?" 이다. 이 문제를 neighbor graph를 이용한 context caching에서는 이웃 AP들에게 클라이언트의 정보를 전송하고 있다. 하지만 무조건 이웃 AP들에게 클라이언트의 정보를 전송하는 것은 트래픽 낭비이다.

이 논문에서는 이러한 문제점을 해결하기 위해 새로운 트래픽 관리 알고리즘을 제안하고, 또한 이동성이 적은 사용자들의 cache hit ratio를 희생하지 않고 이동성이 많은 사용자들의 hit ratio을 높여서 핸드오프 지연을 최소화 하기위해 WLRU(Weighted Least Recently Used) 알고리즘을 제안한다. 2장과 3장에서는 기존의 FHR과 neighbor graph에 관하여 설명하고 4장에서는 WFH 알고리즘에 대해 설명하며, 5장에서는 시뮬레이션을 통한 WFH의 성능을 보여준다.

II. Frequent Handoff Region (FHR) 선택

FHR은 클라이언트에 대한 인증정보를 인증 서버가 이웃 AP들 에게 전송하는 방식이기 때문에 AP의 위치와 클라이언트의 이동 패턴에 대한 정보를 인증 서버가 관리하게 된다. 즉, FHR은 클라이언트가 통신하기 쉬운 AP들로 구성되며, AP 사이의 이동횟수와 이동하기 전의 AP에서 머무른 시간에 따라 이동 비율이 계산되며 이동비율은 일반적으로 AP의 위치와 클라이언트 이동성에 의해서 정해진다[1]. 클라이언트 레벨은 클라이언트의 서비스 요구 정도를 나타내며 이 값이 높으면 데이터 통신의 두절이 없는 실시간 서비스를 요구하는 것을 말한다. AP사이의 이동비율을 측정하기 위해, 표 1의 event log database 시스템을 사용한다. 이 테이블은 사용자의 로그인과 핸드오프 이벤트들을 기록하는 데이터베이스의 예제를 보여준다[2].

표 1. Event Log Database 예제

Number	User ID	Login Time	Handoff Time	Handoff Time
1	2314	07:54:57/3		
2	3452	08:00:55/2	08:05:18/5	
3	1093	08:04:23/3	08:14:03/6	08:15:17/4

다음에 보여주는 weight 값은 핸드오프 비율에 의해 결정된다. Weight 값이 0이면 AP 자기 자신을 나타내며 ∞는 이웃 해 있지 않은 AP를 나타낸다. FHR을 선택할 때는 클라이언트의 레벨을 고려

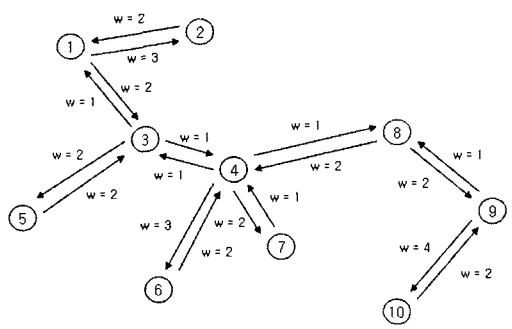


그림 1. 토폴로지

$$W=\begin{pmatrix} 0 & 3 & 2 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ 2 & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & 0 & 1 & 2 & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 1 & 0 & \infty & 3 & 2 & 1 & \infty & \infty \\ \infty & \infty & 2 & \infty & 0 & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 2 & \infty & 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 1 & \infty & \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & 2 & \infty & \infty & \infty & 0 & 2 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 2 & 0 & 4 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 2 & 0 \end{pmatrix}$$

그림 2. Weighted Graph

하여 인증 범위가 정해지고 그림 2에서 보여주는 weighted graph를 보고 각 AP에서 클라이언트 레벨에 따라 FHR 범위를 정할 수 있다. 예를 들어, AP_4에서 레벨이 3인 클라이언트는 AP_3, AP_6, AP_7, AP_8을 FHR 범위로 설정하여 미리 인증하게 된다.

Ⅲ. Neighbor Graphs

Neighbor graph(NG)는 FHR과 동일한 proactive 기법을 사용하지만 클라이언트가 이동할 확률은 고려하지 않았다. NG는 서버 도움 없이 IAPP 메시지를 이용하여 한 홉 거리에 있는 이웃 AP들을 미리 인증하는 방식이며 방향성 없는 edge를 가정한다.

3.1 Neighbor Graph 생성

NG의 초기화 상태에서는 처음부터 이웃 AP들을 알 수는 없다. 그래서 초기에는 클라이언트가 이동하면서 이웃 AP들을 방문하게 되어 각각의 이웃 AP들을 알 수 있게 된다. NG는 각 AP에 의해 자동적으로 생성 되며 다음 두 상태로부터 AP가 NG에서의 edge를 찾아낼 수 있다[3].

1) AP가 클라이언트로부터 Reassociation Request 메시지를 받았을 때: 하나의 클라이언트가 old AP에서 new AP로 재협상 할 때 new AP의 이웃 AP로서 old AP를 추가한다.
2) IAPP를 통해 다른 AP로부터 Move-Notify 메시지를 받았을 때: old AP가 new AP로부터 Move-Notify 메시지를 받았을 때 old AP의 이웃 AP로서 new AP를 추가한다.

각각의 AP들이 위의 과정을 통해 이웃 AP들을 알게 되면, 이것을 이용한 context caching 알고리즘은 다음과 같다. 한 개의 클라이언트가 어느 한 AP에 접속했을 때 이 AP는 클라이언트의 context 정보를 자신의 이웃 AP들에게 전달해주기 때문에 자신과 이웃 AP들을 미리 인증하게 되는 것이다. 하지만 NG를 이용한 context caching에서도 핸드오프 지연을 증가시키는 두 가지 형태의 cache miss 상황들이 있다.

3.2 Cache Miss 발생 상황

1) AP가 자신의 이웃 AP들를 모르는 상태:이웃 AP들이 없는 두 AP사이에서 재협상이 일어났을 때 context 정보를 전달할 수가 없게 되어 cache miss를 발생하고 혹은 wireless network이 멈추거나 재부팅 되어 초기화 되었을 때 cache miss가 발생한다.
2) 다른 클라이언트가 이웃 AP에 접속했을 때: 이것은 이웃 AP에 협상하려는 다른 클라이언트들 때문에 발생하게 된다. 즉 한 클라이언트가 old AP에서 new AP로 이동했을 동안에 old AP의 캐쉬는 자신의 이웃 AP들에 접속한 클라이언트들의 context 정보로 인해 채워지게 되고 이때 캐쉬가 Full인 상태가 되면 LRU 알고리즘에 의해 최근에 사용하지 않은 context 정보를 지우게 된다. 그리고 다시 new AP로 이동했던 클라이언트가 다시 old AP로 이동할 때 old AP에서는 cache miss가 발생하게 된다. NG를 이용한 context caching에서는 두 번째 cache miss 상황을 줄이기 위해서 Cache Invalidation 개념을 도입하였다.

Cache-Invalidation 메시지는 new AP에 접속했던 클라이언트가 자신의 이웃 AP로 이동했을 때 old AP가 자신의 이웃 AP들에게 Cache-Invalidation 메시지를 전송하고 이 메시지를 받은 AP는 cache에서 해당 클라이언트의 정보를 제거하여 cache hit ratio를 높여준다. 하지만, 이 논문에서는 IEEE 802.11f에서 사용하는 Move-Response 메시지를 이용하여 표준을 따르면서, Cache-Invalidation을 효율적으로 구현하는 방법을 제안한다. [3]에서는 Cache-Invalidation을 무선랜에 어떻게 적용할지 언급하고 있지 않았다.

Ⅳ. Weighted Frequent Handoff(WFH)

기존의 FHR은 서버가 클라이언트의 이동 패턴과 AP의 위치 그리고 클라이언트의 레벨에 따라서 이웃 AP들을 선택하여 자주 가는 AP들을 미리 인증하는 방식이며 NG를 이용한 context caching

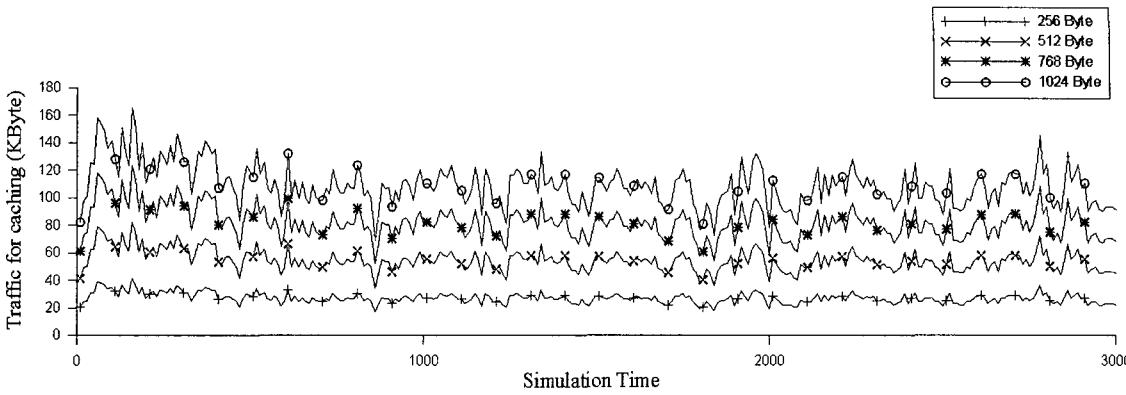


그림 3. 임의의 한 AP에서 발생하는 Cache-Notify 트래픽량

은 서버 도움 없이 AP들 간에 시간이 지남에 따라 각각의 AP들이 자신의 이웃 AP들을 알 수 있기 때문에 미리 한 홉 거리에 있는 이웃 AP들을 인증하는 방식이다.

FHR과 NG는 proactive 방식을 사용하여 빠른 인증을 제공하고 있다. 하지만, 이 두 알고리즘은 빠른 인증에 사용된 중복 트래픽으로 인해 발생되는 네트워크 부하를 고려하지 않고 있다. 중복 트래픽은 클라이언트가 핸드오프 이후 이미 인증된 AP들에게 동일한 메시지를 전송시켜서 불필요한 메시지를 발생시키는 것이다. 이것은 클라이언트 수가 많아지면 많아질수록 불필요한 트래픽을 증가시키게 된다.

FHR 내에서 핸드오프가 일어나면 재인증 과정이 일어나지 않게 되어 클라이언트의 정보가 담긴 메시지를 이웃 AP들에게 전송하지 않지만, FHR 범위를 벗어나게 되면 중복 메시지를 발생한다. 기존 FHR 범위를 벗어나면, 새로운 FHR 범위가 만들어 지며 이때 기존의 FHR 범위와 새로 생성된 FHR 범위가 겹쳐지게 되어 겹쳐진 부분에 동일한 중복 메시지 전송하게 된다. NG를 이용한 context caching에서는 한 홉으로 이동할 때마다 무조건 이웃 AP들에게 메시지를 전송시키기 때문에 old AP와 new AP의 이웃 AP가 동일한 AP이면 이 AP는 같은 클라이언트의 정보를 담은 중복된 메시지를 전송받게 된다.

그림 3은 NG환경에서 1부터 10까지의 시뮬레이션 기간 동안 클라이언트들의 이동이 많을 것 같은 한 AP에서 빠른 인증을 위해 사용하는 Cache-Notify 평균 트래픽량을 보여주고 있다. 이 메시지 크기는 256, 512, 768, 1024 Byte로 설정하였다. WFH와 NG에서 발생하는 평균 트래픽량의 비교는 5장 시뮬레이션에서 보여준다.

WFH에서는 이러한 중복 메시지를 줄일 뿐만 아니라 새로운 cache replacement algorithm인 WLRU를 사용하여 많은 이동성(mobility가 높은)을 가진 클라이언트에 대해서 높은 cache hit ratio도 제공하고 있다. NG에서는 기본적으로 LRU 캐쉬 알고리즘을 사용했지만 WFH에서는 LRU 캐쉬 알고리즘에 이동해올 확률을 고려하였기 때문에 이동성이 적은 클라이언트의 cache hit ratio를 희생하지 않으면서, 이동성이 많은 클라이언트에게 높은 hit ratio를 제공해 준다. 클라이언트의 hit ratio가 높다는 것은 핸드오프 시 발생하는 인증지연 시간횟수를 줄임으로써 빠른 인증을 받는 횟수가 많다는 것이다.

이 논문에서 제안하는 알고리즘은 핸드오프가 일어날 때 전체 토폴로지에서 불필요하게 중복되는 데이터의 트래픽량을 최소한으로 줄이는 것이며, 또한 NG를 이용한 context caching에서 언급한 두 번째 경우의 cache miss가 일어날 확률을 줄여서 핸드오프의 지연을 최소화 시키는 것에 목표를 두고 있다. WFH는 NG의 메시지 전송방식에 FHR의 클라이언트 이동성을 고려하여 설계하였기 때문에 클라이언트가 핸드오프 할 때 FHR처럼 따로 인증서버를 요구하지 않는다.

4.1 트래픽 관리 알고리즘을 이용한 AP Pre-Authentication

NG 환경에서 핸드오프가 발생했을 때, 인증지연

을 최소화하기 위해서 proactive 기법을 이용하여 클라이언트가 접속한 AP의 이웃 AP들에게 클라이언트의 인증정보가 포함된 메시지(Cache-Notify)를 전송한다. 하지만, 핸드오프가 발생하기 전 old AP의 이웃 AP와 핸드오프 발생 후 new AP의 이웃 AP가 동일한 AP인 경우 불필요하게 메시지를 두 번 전송하게 되어 메시지 중복을 야기한다. 이 논문에서는 이러한 문제점을 해결하기 위해서 다음을 정의하였다.

그림 4는 WFH 알고리즘을 보여주며, 알고리즘에 사용되는 데이터구조 및 함수를 정의한다.

1) c.context 클라이언트 c에 대한 security context 정보이다.
2) AP_i.cache AP_i가 유지하고 있는 클라이언트에 대한 캐쉬 데이터 구조이다.
3) AP_i.AAT(Authenticated_AP_Table) AP마다 인증 테이블을 유지하고 있으며 표 2는 AAT의 필드를 설명한다. 자신의 AAT에서 '각 클라이언트에 대한 이웃 AP(ID)의 인증 상태' 필드를 확인하면 한 홉 떨어져 있는 AP에서 이동해올 수 있는 클라이언트에 대한 인증된 AP들의 ID를 알 수 있다. 핸드오프 이후, AAT를 이용하여 중복된 데이터 트래픽 전송을 막을 수 있다.
4) c.AAL(Authenticated_AP_List) 이것은 Cache-Notify 메시지에 포함되며 클라이언트가 접속한 AP의 이웃 AP들의 ID들이다.
5) L2_Update(AP_{new}, c, p, AP_{new-n}) 이 논문에서는 불필요한 Cache-Notify 메시지를 전송하지 않으면서 마치 메시지를 전송한 것과 같은 효과를 보기위해 Layer 2 Update 프레임을 이용한다. L2_Update 메시지는 AP_{new}에서 AP_{new-n}로 전송되며, AP_{new-n}는 캐쉬

```
APold = old AP, APnew = new AP, APc = current AP,
APold-n = old AP의 이웃 AP, APnew-n = new AP의 이웃 AP, c = 클라이언트

1: if client c associates to APnew then
2: for all APnew-n ∈ neighbor(APnew) do
3: L2_Update(APnew, c, p, APnew-n)
4: Cache_AAT_Update(APnew-n, c, p)
5: Context_Table_Notification(APnew, c.context, p, c.AAL, APnew-n)
6: end for
7: end if
8: if client c re-associates from APold to APnew then
9: for all APnew-n ∈ neighbor(APnew) do
10: L2_Update(APnew, c, p, APnew-n)
11: Cache_AAT_Update(APnew-n, c, p)
12: if APnew-n ∉ APnew.AAT then
13: Context_Table_Notification(APnew, c.context, p, c.AAL, APnew-n)
14: end if
15: end for
16: end if
17: if client c re-associates from APold to APnew then
18: for all APold-n ∈ neighbor(APold) do
19: if APold-n ∉ neighbor(APnew) then
20: Delete_Context_Table(APold, c, APold-n)
21: Delete_AP_ID(APold, APold-n)
22: end if
23: end for
24: end if
25: if APnew-n received c.context, p and c.AAL from APnew then
26: Insert_AP_Cache(APnew-n, c.context, p, c.AAL)
27: end if
```

그림 4. Proactive Caching을 위한 효율적인 트래픽 관리 알고리즘

표 2. AAT 필드정의

필 드		내 용
AP_i		각 클라이언트들의 인증정보를 가지고 있는 AP_i
APi에서 각 클라이언트에 대한 이웃 AP(ID)의 인증상태		APi의 이웃 AP들의 ID, 각각의 ID들은 해당 클라이언트에 대해서 인증이 되었는지를 표시
클라이언트 ID		AP_i가 유지하고 있는 인증된 클라이언트들의 ID
인증된 이웃 AP	"O"	클라이언트의 인증정보를 O/X로 표시
	"(IDs)"	클라이언트의 인증정보를 가지고 있는 해당 AP의 이웃 AP들의 ID (AP_i는 이 필드를 확인하여 Cache-Notify 메시지를 줄일 수 있다.)
Old AP		L2-Update 혹은 Cache-Notify 메시지를 전송한 AP의 ID(메시지를 받으면 갱신된다.)

와 테이블에서 클라이언트 c에 대한 정보를 갱신한다. 이 함수는 같은 DSM(Distribution System Medium)내의 모든 AP_{new-n}들에게 broadcast 한다. 우선순위(priority)는 IAPP 표준 메시지의 예약필드(1byte)에서 2bit를 이용한다.

6) Cache_AAT_Update(AP_{new-n}, c, p) 이 함수는 AP_{new}로부터 L2_Update 메시지를 전송받으면 AP_{new-n}에서 다음과 같은 수행을 한다.

① 캐쉬 갱신

캐쉬에 클라이언트의 정보가 없으면 무시한다. 클라이언트의 정보가 캐쉬에 기록되어 있다면, 그 위치를 최상위가 아닌 p순서에 따라 위치시킨다. 클라이언트에 대한 'Old AP' 필드는 Cache-Notify 메시지를 전송한 AP_{new}의 ID로 갱신한다.

② AAT 갱신

AAT에서 클라이언트의 정보가 없으면 무시한다. 클라이언트의 정보가 AAT의 '각 클라이언트에 대한 이웃 AP(ID)의 인증상태' 필드에 기록되어 있다면, 해당 클라이언트에 대한 'Old AP' 필드를 AP_{new}의 ID로 갱신한다.

'Old AP' 필드는 해당 클라이언트와 관련해서 가장 최근에 메시지를 보낸 AP를 나타낸다.

7) Context_Table_Notification(AP_{new}, c.context, p, c.AAL, AP_{new-n}) AP_{new}에서 AP_{new-n}으로 Cache-Notify 메시지를 이용하여 전송하며, 클라이언트 c에 대한 context 정보와 캐쉬에 저장될 위치를 알려주는 p 그리고 c.AAL을 포함한다. 그림 5는 클라이언트 #2가 AP_C로 핸드오프 하기 전 AP_A에서 AP_B, AP_C, AP_D에게 Cache_Notify 메시지 전송을 보여주며 그림 6은 핸드오프 이후 AP_C에서 AP_E와 AP_F에게 Cache-Notify 메시지를 전송하는 것을 보여준다. AP_C.AAT에서 '각 클라이언트에 대한 이웃 AP(ID)의 인증상태' 필드를 확인하여 AP_A, AP_B, AP_D들의 ID가 클라이언트 #2에 대해서 이미 인증되었다는 것을 인식하여 AP_E와 AP_F에게만 Cache-Notify 메시지를 전송하고 있음을 보여주고 있다.

8) Insert_AP_Cache(AP_{new-n}, c.context, p, c.AAL) 이 함수는 AP_{new-n}가 AP_{new}로부터 Cache-Notify 메시지를 받으면 다음과 같은 수행을 한다.

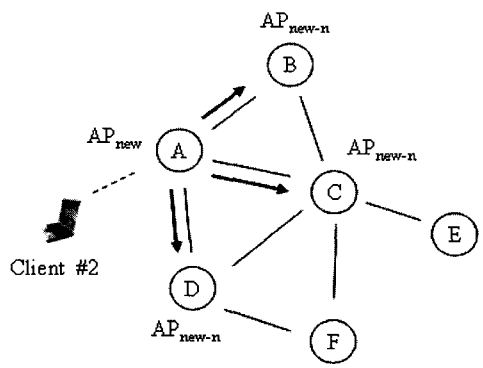


그림 5. 핸드오프 전 Cache_Notify 메시지 전송

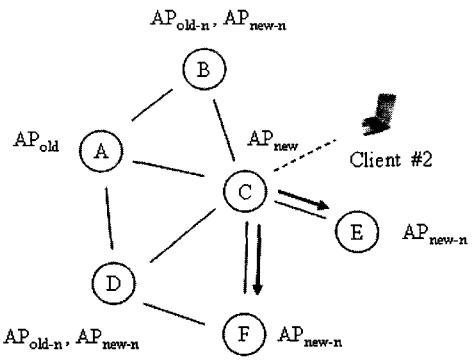


그림 6. 핸드오프 후 Cache_Notify 메시지 전송

표 3. 핸드오프 전 AP_C.AAT

AP_C		AP_C에서 각 클라이언트에 대한 이웃 AP(ID)의 인증상태				
클라이언트 ID	클라이언트의 인증정보를 가진 이웃 AP/Old AP	AP_A	AP_B	AP_D	AP_E	AP_F
#1	인증된 이웃 AP	O(AP_B, AP_D)	O(AP_A)	O(AP_A, AP_F)	O	O(AP_D)
	Old AP	AP_C	AP_C	AP_C	AP_C	AP_C
#2	인증된 이웃 AP	O(AP_B, AP_D)	O(AP_A)	O(AP_A)	X	X
	Old AP	AP_A	AP_A	AP_A	X	X

① 캐쉬 갱신

AP_{new-n}는 캐쉬에 c.context를 기록하며, 기록하는 위치는 최상위가 아닌 p순서를 따른다. 클라이언트에 대한 'Old AP' 필드는 Cache-Notify 메시지를 전송한 AP_{new}의 ID로 갱신한다. p는 4.2.2 우선순위에서 설명한다.

② AAT 갱신

AP_{new-n}.AAT의 인증상태 필드에 클라이언트 c에 대한 c.AAL을 저장하며 'Old AP' 필드에는 AP_{new}의 ID로 기록한다. 표 3은 클라이언트 #2가 AP_A에 접속하여 Cache-Notify 메시지 AP_B, AP_C, AP_D로 전송한 이후 AP_C.AAT 상태를 나타내며, 클라이언트 #1은 AP_C에 먼저 접속되어 있는 사용자이다. 표 4는 클라이언트 #2가 AP_C로 이동한 다음 AP_E와 AP_F로 메시지를 전송한 이후의 AP_C.AAT 상태를 나타낸다.

9) Delete_Context_Table(AP_{old}, c, AP_{old-n})

AP_{old}는 $AP_{old\ n}$의 캐쉬와 테이블에서 클라이언트에 대한 정보를 제거하기위해 AP_{old-n}으로 메시지를 전송한다. NG에서는 Cache-Invalidation 메시지를 따로 정의하고 있지만 이 논문에서는 IAPP 표준의 Move-Response 메시지에 포함되는 클라이언트의 context 정보를 제외하고 이용하였다.NG는 cache hit ratio를 높이기 위해 Cache-Invalidation 메시지 를 AP_{old}에서 AP_{old-n}들로 전송하여 AP_{old-n}.cache에서 클라이언트에 대한 정보를 간단하게 삭제한다. 이 논문에서 제안하는 트래픽 관리 알고리즘도 NG에서와 같은 효과를 보기위해 이 알고리즘을 이용하였지만 문제점이 발생하였다. 그것은, 트래픽 관리 알고리즘이 트래픽을 줄이기 위해 AAT를 사용하기 때문이다. NG에서처럼 Cache-Invalidation 메시지를 받은 AP들은 메시지를 보낸 AP_{new}의 ID와 'Old AP' 필드의 ID를 확인하여 일치하는 AP들의 캐쉬와 AAT에서 클라이언트에 대한 정보를 삭제하면 되지만, 이 메시지의 목적은 클라이언트에 대한 정보를 AP_{old-n}들에게 전부 삭제하라고 요청하는 메시지이기 때문에, Cache-Invalidation 메시지를 보내는 AP_{old}.AAT의 인증상태 필드에서 메시지를 수신할 AP_{old-n}의 ID를 삭제해야 한다. 메시지가 AP_{old}에서 AP_{old-n}들로

표 4. 핸드오프 후 AP_C.AAT

AP_C		AP_C에서 각 클라이언트에 대한 이웃 AP(ID)의 인증상태				
클라이언트 ID	클라이언트의 인증정보를 가진 이웃 AP/Old AP	AP_A	AP_B	AP_D	AP_E	AP_F
#1	인증된 이웃 AP	O(AP_B, AP_D)	O(AP_A)	O(AP_A, AP_F)	O	O(AP_D)
	Old AP	AP_C	AP_C	AP_C	AP_C	AP_C
#2	인증된 이웃 AP	O(AP_B, AP_D)	O(AP_A)	O(AP_A, AP_F)	O	O(AP_D)
	Old AP	AP_C	AP_C	AP_C	AP_C	AP_C

전송되어 AP_{old}.AAT의 인증상태 필드로부터 클라이언트에 대해 인증된 AP_{old-n}의 ID가 전부 삭제되고, AP_{new}로 갔던 클라이언트가 다시 AP_{old}로 이동해 온다면 NG와 같이 중복된 메시지를 전송하게 되어 트래픽 이득이 없게 된다.

이러한 문제점은 이 논문에서 제안한 AAT를 이용하면 해결할 수 있다. 클라이언트가 AP_{new}로 이동한 후 AP_{old}는 AP_{new}로부터 Move-Notify 메시지를 받는다. 이 메시지로부터 AP_{old}는 클라이언트가 어디로 이동했는지 알 수 있다. 그리고 AP_{old}는 표 2와같이 AAT의 '인증된 이웃 AP' 필드를 확인하여 이동한 AP_{new}의 ID가 포함되지 않은 이웃 AP들에게만 Cache-Invalidation 메시지를 AP_{old-n}로 전송한다. 즉, AP_{old}는 Cache-Invalidation 메시지를 AP_{new}와 두 홉 떨어진 AP_{old-n}들에게만 전송하는 것이다. 이 논문에서는 Cache-Invalidation 메시지 역할을 c.context가 포함되지 않아 Move-Notify 메시지가 대신한다.

AAT를 이용하게 되면, 중복되는 Cache-Notify 메시지뿐만 아니라 NG에서 사용하는 중복된 Cache-Invalidation 메시지도 줄이는 효과도 얻을 수 있다. 이 논문에서는 트래픽 관리 알고리즘을 이용하여 Cache-Notify 메시지에 대한 트래픽 이득만을 보여준다.

10) Delete_AP_ID(AP_{old}, AP_{old-n})

이 함수는 AP_{old}가 AP_{old-n}으로 Move-Response 메시지를 전송할 때, AP_{old}에서 다음과 같은 수행을 한다.

① AAT 갱신

AP_{old}.AAT에서 클라이언트가 인증했던 AP_{old-n}의 ID를 제거한다. 다시 말해, AP_{old}.AAT 에서 Delete_Context_Table(AP_{old}, c, AP_{old-n})을 수행한 AP_{old-n}의 ID만 '각 클라이언트에 대한 이웃 AP(ID)의 인증상태' 필드에서 제거한다.

그림 7은 핸드오프가 발생할 때 전체 메시지 전송과정을 보여준다.

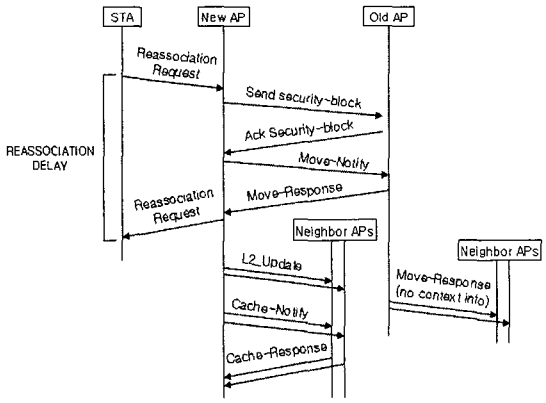
4.2 WLRU(Weighted Least Recently Used)

이 절에서는 cache hit ratio를 높이기 위해 기존의 LRU가 아닌 클라이언트들이 이동해 올 확률을 고려한 새로운 cache replacement algorithm을 제안한다. WFH 알고리즘은 NG가 생성된 후, FHR에서 언급한 AP의 위치와 클라이언트의 이동 패턴에 따라서 AP들 사이에 서로 다른 방향성 있는 edge를 가지며 각 edge는 w(weight)를 가지게 된다. 이렇게 이웃 AP마다 다른 w값은 최근에 얼마나 자주 이동했는가를 나타내며 이 값을 cache replacement에 이용한다.

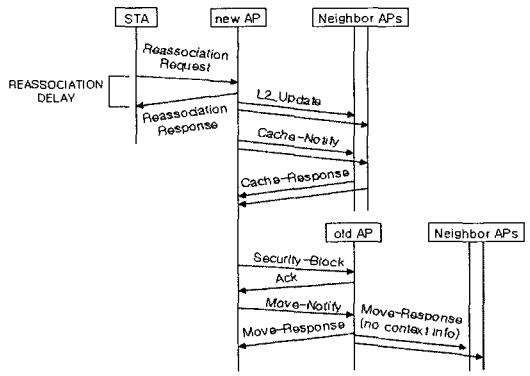
LRU 캐쉬 알고리즘은 최근의 정보를 캐쉬 맨 상위에 저장하는 반면 이 논문에서의 제안하는 WLRU는 w를 이용하여 이동해올 확률이 적은 것(w가 낮은)에 우선순위를 높게 하여 저장한다.

4.2.1 Weight 계산

그림 8과 표 5는 그림 5에 있는 AP_A에서의 event



(a) IAPP 재협상과 cache miss



(b) IAPP 재협상과 cache hit

그림 7. 핸드오프 동안 메시지 전송순서

Number	Login Time	Handoff Time	Neighbor APs
1	07:54	08:05	B
2	08:24	10:00	C
3	09:11	09:40	B
4	10:15	11:20	D
.	.	.	.
.	.	.	.

그림 8. AP_A의 Event Log Database

표 5. AP_A의 Weight Table

이웃 AP들	Weight
B	10
C	2
D	5

log database와 이것을 이용한 w의 예를 들었다.

Event Log Database : 각각의 AP들은 eventlog database를 가지고 있으며 클라이언트가 현재 접속한 시간과 이웃 AP로 핸드오프 한 시간을 기록한다.

Weight Table : AP는 event log database를 이용하여 한 홉 떨어진 이웃 AP들과의 이동률을 계산하여 weight table에 기록하게 된다.

Weight 계산 : H(i, j)는 핸드오프 비율이고, N(i, j)은 AP(i)에서 AP(j)로부터 핸드오프 이벤트 발생 횟수이다. R(i, j)은 AP(i)에서 AP(j)로 핸드오프 이벤트가 발생할 때 AP(i)에 머무른 시간을 나타낸다[11]. AP들 사이의 w는 핸드오프 비율에 의해 결정된다.

$$H(i,j) = \frac{N(i,j)}{R(i,j)} \quad (1)$$

$$w(i,j) = \frac{1}{H(i,j)} \quad (2)$$

```
1: if 1 ≤ w(api, apj) ≤ 3
2: return 0;
3: else if 4 ≤ w(api, apj) ≤ 6
4: return 1;
4: else if 7 ≤ w(api, apj) ≤ 9
5: return 2;
6: else if 10 ≤ w(api, apj) ≤ 12
7: return 3;
```

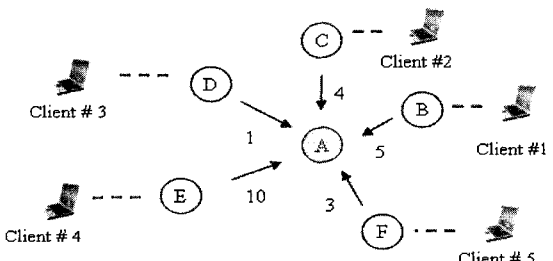
그림 9. 우선순위 알고리즘

Weight 유지 : AP들 사이의 w는 최근 접속기록을 바탕으로 하는 event log database의 시간과 w계산에 의해 실시간으로 바뀌며 유지된다.

4.2.2 우선순위(priority)

Cache-Notify 혹은 L2-Update 메시지에 포함되는 우선순위는 w값에 의해 결정되며 메시지를 보내는 AP가 다음과 같은 함수를 사용하여 우선순위를 결정한다. $p(AP_i, AP_j)$: AP_i에서 AP_j의 방향으로, AP_j.cache에 저장될 우선순위이며 그림 9와 같다.

그림 10은 WLRU 캐쉬 알고리즘이며 AP_A가 Cache-Notify 메시지를 클라이언트 #1부터 #5까지 순서대로 받아서 p 의 값에 따라 캐쉬에 저장하



Client #	Direction	Weight	Priority
Client #1	AP_B -> AP_A	5	1
Client #2	AP_C -> AP_A	4	1
Client #3	AP_D -> AP_A	1	0
Client #4	AP_E -> AP_A	10	3
Client #5	AP_F -> AP_A	3	0

AP_A의 우선순위

	1	2	3	4	5
0	Client #1	Client #1	Client #3	Client #3	Client #5
1		Client #2	Client #1	Client #1	Client #3
2			Client #2	Client #2	Client #1
3				Client #4	Client #2

AP_A에서 우선순위에 따른 캐쉬상태의 변화

그림 10. WLRU 캐쉬 알고리즘

는 것을 보여준다. 각 화살표는 AP사이의 w를 나타낸다.

V. 시뮬레이션

C++로 구현된 시뮬레이션에서, AP는 제한된 캐쉬 사이즈를 가지며 클라이언트들은 시뮬레이션 동안 자신의 mobility에 따라서 핸드오프를 한다.

• 시뮬레이션 목적 :

클라이언트수와 네트워크 토폴로지에 따라서 기존의 NG보다 얼마만큼의 중복 메시지를 줄일 수 있는지 관찰하며, 제한된 캐쉬에서 클라이언트가 늘어날 때 NG에서 사용하는 LRU와 이 논문에서 제안하는 WLRU의 cache hit ratio를 비교하여 그 효율성을 알아본다.

• 시뮬레이션 모델과 가정 :

1) 네트워크 토폴로지 : 초기 AP의 위치는 랜덤하게 설정되며 이 토폴로지는 시뮬레이션이 끝날 때 까지 변하지 않는다.
2) 초기 클라이언트의 위치 : 클라이언트들은 전체 토폴로지에 고루 분산되어 있다.
3) 클라이언트 mobility : 시뮬레이션 동안 주어진 시간에 클라이언트가 이동하는 확률로서 한 개의 클라이언트 이동성을 정의한다[3].

• 시뮬레이션 환경 :

1) 시뮬레이션은 초기 AP를 랜덤하게 위치하였다.
2) weight : w의 범위가 너무 크면 한곳으로 클라이언트가 집중되는 현상이 일어날 수 있으므로 그 범위를 1~12까지 제한하며 클라이언트의 랜덤한 움직임에 따라 실시간으로 고루 분포된다.
3) 시뮬레이션 기간 : 클라이언트의 mobility에 따라서 균일하게 분산된 3만번의 재협상 과정을 다섯 번 이행하여 평균을 내었고 이것은 결과 값이 만족스러운 신뢰성을 줄 만큼의 충분한 기간이다.

• 시뮬레이션 결과:

1) 클라이언트 증가에 따른 cache hit ratio: 그림 11은 AP(50)인 토폴로지에서 클라이언트수를 증가시켜 가면서 LRU와 WLRU의 cache hit ratio를 비교하였다. 캐쉬 크기가 충분할 때는 두 알고리즘이 비슷한 cache hit ratio를 보여주지만 캐쉬 크기가 충분하지 못할 때는 WLRU가 LRU보다 높게 나타났다. 클라이언트가 500일 때와 400일 때 cache hit ratio의 이득이 비슷한데 이것은 캐쉬 크기가 일정한 크기 이상으로 작아져도 그 이득은 비슷하다는 것을 보여주며, 제안한 알고리즘이 mobility가 낮은 사용자의 cache hit ratio를 희생하지 않으면서 mobility가 높은 사용자에게는 더 높은 hit ratio를 제공하는 것을 알 수 있다.
2) 1)과 동일한 환경에서 NG와 WFH의 총 트래픽량: 그림 12는 1)번 시뮬레이션 환경에서 기존 proactive caching을 사용하는 NG와 제안한 WFH에서의 트래픽량을 비교하여 보여주고 있다. 클라이언트수가 500이였을 때 이웃 AP들에게 무조건 Cache-Notify 메시지를 전송하는 NG보다 자신의 AP에서 AAT을 확인하여 중복 메시지를 줄이는 WFH가 시뮬레이션 기간 동안 이백만개 이상의 트래픽 이득을 보이고 있다.

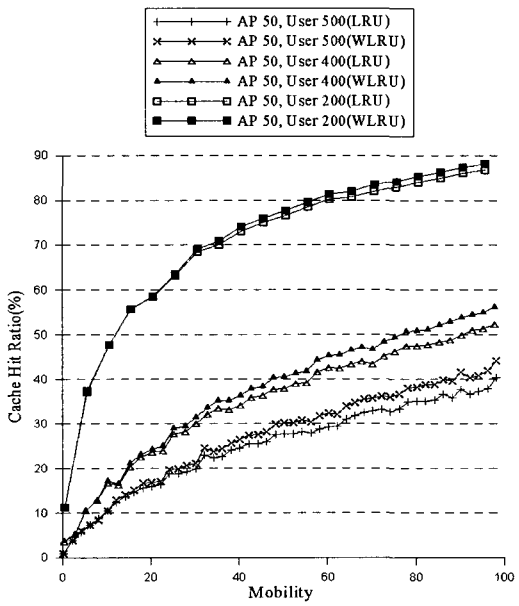


그림 11. 클라이언트 증가에 따른 cache hit ratio

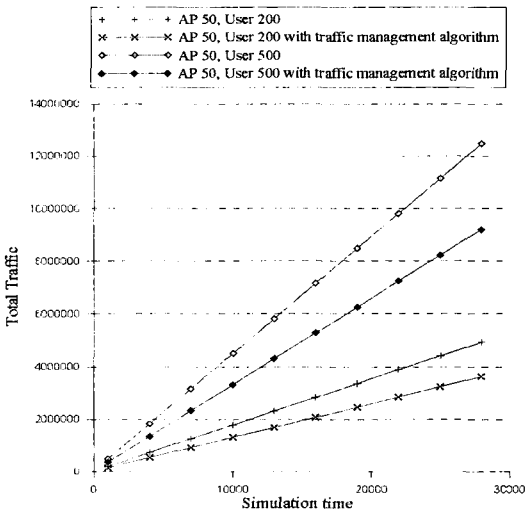


그림 12. 클라이언트 증가에 따른 전체 트래픽량

이 개수는 Cache-Notify 메시지를 전송한 횟수를 나타냈으며 실제 데이터 량의 이득은 클라이언트 context 정보의 크기에 영향을 받으며 이 정보가 크면 클수록 그 이득은 커진다.

3) 클라이언트수와 캐쉬 크기에 따른 cache hit ratio: 그림 13은 클라이언트를 증가시켜가면서 각 캐쉬 크기에서 LRU와 WLRU의 평균 cache hit ratio를 비교하였다. 클라이언트수와 비례하여 캐쉬 크기가 충분하지 못한 상태에서 WLRU가 LRU보다 hit ratio가 높게 나타나는 것을 알 수 있다.

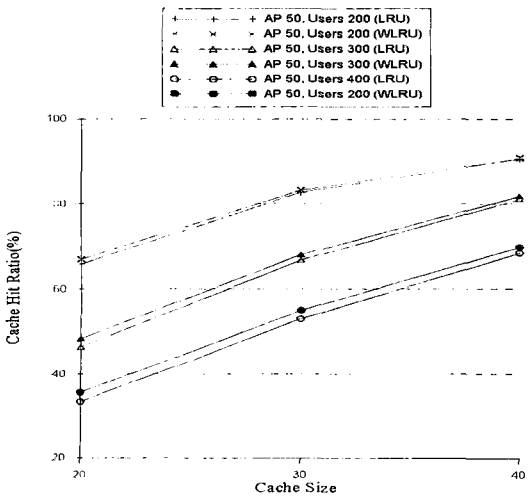


그림 13. 클라이언트수와 캐쉬 크기에 따른 cache hit ratio

4) 겹쳐진 서비스 범위에 따른 트래픽 이득차이: 그림 14는 AP(20), 클라이언트(100)으로 일정하고 AP의 위치를 변경하면서 기존 proactive caching을 사용하는 NG와 제안한 WFH에서의 트래픽 이득차이를 겹쳐진 서비스 범위의 개수를 늘려가며 비교한다. 토폴로지는 랜덤하게 위치하였고 AP의 서비스 범위가 서로 겹쳐진 환경을 증가시켰다. 가로축 The number of overlaped ranges in topology는 표 6과 같다.

5) AP수와 클라이언트 증가에 따른 트래픽량: 그림 15는 AP수와 클라이언트수를 일정비율로 증가시키면서 발생하는 트래픽량을 기존 proactive caching기법 NG와 트래픽 관리 알고리즘을 사용한 WFH를 비교하였다. AP수가 증가할수록 서로 겹쳐지는 서비스 지역이 많아지고 클라이언트수가 증가할수록 트

표 6. AP의 위치에 따른 겹쳐진 서비스범위 수

The number of overlaped ranges	총 겹쳐진 서비스범위 수	AP 3개가 서로 겹쳐진 서비스 범위 수
1	19	0
2	20	1
3	24	3
4	26	7

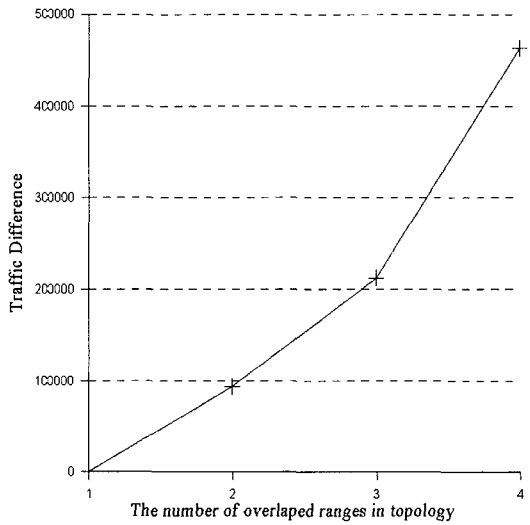


그림 14. AP 위치에 따른 트래픽 이득

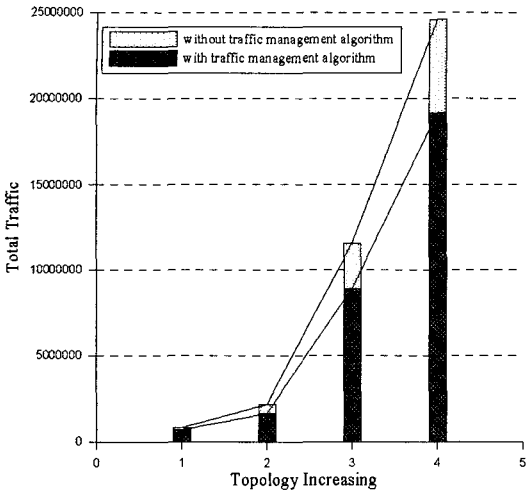


그림 15. AP수와 클라이언트 증가에 따른 트래픽량

래픽량도 증가하기 때문에 점점 트래픽 이득의 차이가 커지게 된다. 가로축의 Topology Increasing은 다음 표 7과 같다.

6) 시뮬레이션동안 한 AP에서 발생하는 트래픽량 : 그림 16의 시뮬레이션 환경은 AP(50), 클라이 언트(1000), 시뮬레이션 시간(3000)이며, 빠른 인증을 위해 사용하는 Cache-Notify 메시지크 기는 256과 1024 Byte로 설정하였다. 1000개 의 클라이언트들이 고루 분포된 AP들 중에서 비 교적 클라이언트들의 이동이 잦은 임의의 한 AP 를 선택하여 1부터 10까지의 시뮬레이션 기간에 서 발생하는 평균 Cache-Notify 메시지량을 NG 와 WFH를 비교하여 보여주고 있다. 그림 17은 임의의 한

표 7. Topology Increasing

Topology Increasing	AP 수	클라이언트 수
1	10	50
2	20	100
3	100	500
4	200	1000

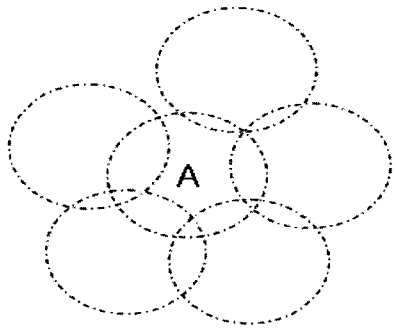


그림 17. 5개의 이웃AP를 가진 AP_A

AP 구조이다. WFH가 NG보다 트래 픽량을 약 40~45% 줄이는 것을 알 수 있다.

Ⅵ. 결 론

이 논문에서는 서버가 직접 인증 범위를 지정해주는 FHR과 핸드오프 발생 시 AP들이 서로 자동적으로 이웃 AP를 생성하는 Neighbor Graph를 살펴보았고, 이를 개선할 수 있는 새로운 트래픽 관리 기법과 cache replace algorithm을 제안하였다.

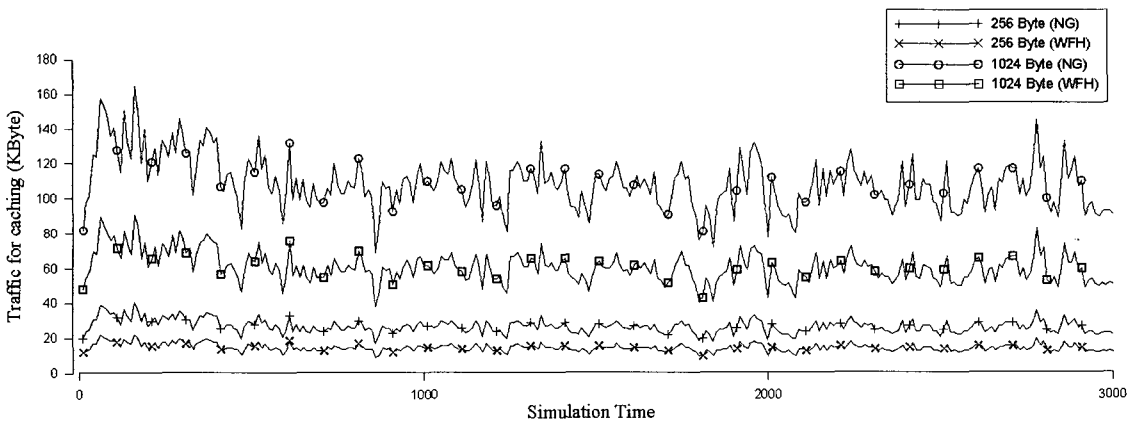


그림 16. 임의의 한 AP에서 발생하는 Cache-Notify 트래픽량

FHR은 IAPP 메시지를 이용하지 않고 서버를 이용한 중앙 집중방식(centralized fashion)의 알고리즘이며 한 홉 이상의 이접한 AP들을 인증하고 있다. 하지만 클라이언트가 인증 범위를 벗어날 때 마다 새로운 인증 범위를 설정하는데 이것은 이전의 인증 범위와 새로운 인증 범위가 겹쳐지게 될 가능성을 말해주며 이렇게 겹쳐지는 인증 범위가 클수록 더 많은 데이터 트래픽의 낭비를 가져오게 된다.

FHR은 클라이언트의 정보가 포함된 메시지를 전송시키기(인증 범위를 설정하기) 위해 그림 2에서 서버가 n×n matrix에 대해 $O(n^2)$의 계산을 하고 있다. n은 네트워크에서 AP의 수를 의미한다. NG에서는 한 AP가 자신의 이웃 AP들만 인증하기 때문에 각 AP마다 O(n)의 계산을 수행한다. NG는 FHR보다 인증범위를 설정하는 계산을 적게 요구하지만 FHR과 마찬가지로 핸드오프가 발생할 때마다 모든 이웃 AP들에게 context 정보를 전송하기 때문에 old AP와 new AP의 인증범위가 겹치게 되어 데이터 트래픽 낭비를 가져온다.

이 논문에서 제안한 WFH 알고리즘은 각 AP마다 AAT를 가지고 있기 때문에 인증범위를 설정하기위해 O(n)의 계산을 하며, old AP와 new AP의 인증범위가 겹쳐지게 되어도 불필요한 중복 메시지를 전송하지 않으므로 전체 네트워크 환경에서 트래픽 부하를 줄일 수 있다. 또한 WFH는 NG와 같이 분산된 방식(distributed fashion)이기 때문에 토폴로지 변화에 빨리 적응할 수 있다.

트래픽량을 줄이는 방법 이외에도 cache hit ratio를 높이기 위해 클라이언트가 이동해올 확률을 고려하여 캐쉬에 저장하는 방식인 WLRU 알고리즘을 사용하고 있다. WLRU를 이용하여 hit ratio를 높임으로써 그만큼 핸드오프 지연을 줄일 수 있게 된다.

이렇게 트래픽량을 줄이는 기술과 cache hit ratio를 높이는 방법은 현재 AP와 단말기사이의 무선 환경보다도 기지국과 단말기 사이처럼 속도가 빠른 휴대 인터넷환경에 더 많이 이용될 것으로 기대하며, 향후 휴대인터넷 환경에 적합한 pre-authentication 방법 등에 적용할 수 있을 것이다.

참 고 문 헌

[1] Sangheon Pack and Yanghee Choi, "Fast Inter-AP Handoff using Predictive-Authentication Scheme in a Public Wireless LAN," *Networks 2002 (Joint ICN 2002 and ICWLHN 2002)*, August 2002.

[2] Sangheon Pack and Yanghee Choi, "Pre-Authenticated Fast Handoff in a Public Wireless LAN based on IEEE 802.1x Model," *IFIP TC6 Personal Wireless Communications 2002 (To Appear)*, October 2002.

[3] A. Mishra, M. Shin, and W. A. Arbaugh, "Context Caching using Neighbor Graphs for Fast Handoffs in a Wireless Network," *IEEE Infocom 2004*, Mar. 2004.

[4] IEEE, "Recommended Practice for Multi-Vendor Access Point Interoperability via an Inter-Access Point Protocol Across Distribution Systems Supporting IEEE 802.11 Operation," *IEEE standard 802.11f*, July 2003.

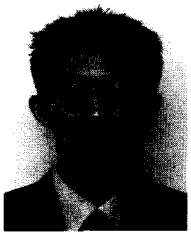
[5] Arunesh Mishra, Minho Shin, and William Arbaugh, "An Empirical Analysis of the IEEE 802.11 MAC Layer Handoff Process," *ACM Computer Communications Review*, Apr. 2003.

[6] IEEE, "IEEE Standard for Local and metropolitan area networks-Port-Based Network Access Control", *IEEE Standard 802.1x-2001*, June 2001

[7] IEEE, "Draft 5 Amendment for Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands", *IEEE P8-02.16e/D5*, September 2004

[8] Hyung-Seop Hong, Ae-Soon Park, Da-Hye Choi, Sang-Ha Kim, Department of Computer Science, Chungnam National University, Electronics and Telecommunications Research Institute, "A Security Mechanism to Guarantee Mobility of Mobile Stations in Wireless LAN based on 802.11b", *JCCI*, Apr. 2003

〈著者紹介〉



최 재 우 (Jae-woo Choi) 정회원
2003년 2월: 안동대학교 정보통신공학과 졸업
2003년 9월~현재: 인하대학교 정보통신대학원 석사과정
〈관심분야〉 WLAN 보안, 휴대인터넷 보안

강 전 일 (Jeon-il Kang) 정회원
2003년 2월: 인하대학교 컴퓨터공학과 졸업
2004년 3월~현재: 인하대학교 정보통신대학원 석사과정
〈관심분야〉 정보보호, RFID 보안

양 대 헌 (Dae-hun Nyang) 정회원
1994년 2월: 한국과학기술원 과학기술 대학 전기 및 전자 공학과 졸업
1996년 2월: 연세대학교 컴퓨터과학과 석사졸업
2000년 8월: 연세대학교 컴퓨터과학과 박사졸업
2000년 9월~2003년 2월: 한국전자통신연구원 정보보호연구본부 선임연구원
2003년 9월~현재: 인하대학교 정보통신대학원 교수
〈관심분야〉 암호이론, 암호프로토콜, 인증프로토콜, 무선 인터넷 보안, RFID 보안, 휴대인터넷 보안