PCN 환경을 위한 Working Set 기반의 위치 관리 기법

임 성 빈† · 엄 영 익††

요 약

최근 휴대용 컴퓨터 기술과 무선 통신 기술의 결합으로 이루어진 이동 컴퓨팅 환경에 대한 연구가 이루어지고 있다. 이동 컴퓨팅 환경은 기존의 네트워크 환경에서 이동 호스트(mobile host)를 지원할 수 있도록 구성되어 있다. 이동 컴퓨팅 환경 중 PCN 환경의 구축을 위해서는 여러 가지 기법들의 지원이 필요하지만, 특히 이동 호스트가 가진 특성인 이동성(mobility)을 지원하기 위하여 위치 관리 기법을 필요로 한다. PCN 환경에서의 위치 관리 기법은 현재 IS-41 기법을 기반으로 하고 있지만, 이 기법은 이동 호스트가 위치 서버(location server)를 변경할 때마다 이동 호스트의 홈 위치 서버에 갱신 메시지가 전달되어 네트워크 부하를 가중시킨다는 문제점을 안고 있다. 본 논문에서는 CDMA의 HLR/VLR 기법에서 기존 IS-41 기법이 가진 문제점을 개선하기 위하여 이동 호스트가 최근 방문한 위치 서버들을 동적 working set으로 설정하여 활용하는 갱신 기법 및 탐색 기법을 제안한다 또한, 제안 기법과 IS-41 기법을 사용하여 위치 관리를 실시할 때 각각에 대한 메시지 수, 데이터베이스 접근 수, 그리고 홉 수를 비교하는 성능 평가 및 분석 결과를 보인다.

Location Management Strategy Based on Working Set for Personal Communications Network

Seong-Bin Im† · Young-Ik Eom††

ABSTRACT

Recently, research activities for combining portable computer technology and wireless communication technology are in progress to realize mobile computing environment that designed to support mobile hosts in the existing network environments. For PCN environments, the location management strategies are particularly emphasized among various strategies necessary for the mobile computing environment. Location management strategy for PCN environments, which is mostly based on the IS-41 standards brings an issue that additional network traffics are generated to deliver update messages to home location server whenever a mobile host changes its location server. In this paper, we propose an innovative update strategy and a search strategy that resolves the problem shown in the IS-41 scheme for HLR/VLR structure of CDMA environments. Also, this paper presents the performance comparison of proposed scheme and scheme used in IS-41 obtained from a simulation. The simulation was conducted based on the number of messages, the rate of database accesses, and the number of hops visited when the location management scheme is executed.

† 정 회 원 ㈜데이터로직 연구원
†† 종신회원 · 성균관대학교 전기전자및컴퓨터공학부 교수
논문접수 1999년 9월 9일, 심사완료 2000년 3월 6일

1. 서 론

최근 무선 통신 및 이동 통신 기술과 휴대용 컴퓨터 기술의 발전으로 인하여 이동 컴퓨팅 환경(mobile computing environment)이 탄생하게 되었다. 이동 컴퓨팅 환경이란 기존의 고정 네트워크 환경에서 이동 호스트(mobile host)를 지원할 수 있도록 확장된 컴퓨팅 패러다임이다. 이동 컴퓨팅 환경의 기본 속성들 중의 하나는 호스트의 이동성(mobility)이다. 이와 같이 호스트의 이동성을 지원하기 위해서는 이동 호스트들을 위한 위치 관리(location management) 정책 등이 필요하게 된다.

이동 컴퓨팅 환경에서의 위치 관리 기법은 이동 호스트의 위치 정보에 대한 갱신(update), 탐색(search), 그리고 탐색-갱신(search-update) 모듈들로 구성된다. 갱신은 이동 호스트가 위치를 변경할 때 발생하고, 탐색은 한 호스트가 알려지지 않은 위치에 있는 이동 호스트와 통신을 원할 때 발생하며, 탐색-갱신은 성공적으로 탐색이 일어난 후 탐색 대상 호스트의 위치 정보를 갱신할 때 발생한다[2, 6].

본 논문에서는 PCN 환경을 대상으로 이와 같은 이동 호스트의 위치 관리를 효율적으로 수행하기 위해 동적인 working set 개념을 이용한 기법을 제시한다. 본 논문의 2장에서는 이동 컴퓨팅 환경에서의 위치 관리 정책과 관련한 기존의 연구 결과들을 소개하고, 3장에서는 PCN 환경에 대한 시스템 모델을 제시한다. 4장에서는 PCN 환경에서 동적인 working set 개념을 이용한 위치 관리 기법을 제시하고, 5장에서는 제시한 기법에 대한 성능 평가 및 분석 결과를 보이며, 마지막으로 6장에서 결론을 맺는다.

2. 관련 연구

위치 관리 기법은 이동 컴퓨팅 환경을 구축하는데 중요한 기법이다. 위치 관리 기법에 의해 이동 호스트들의 위치 정보를 유지, 관리함으로써 차후 탐색 시에 불필요한 메시지의 전송을 막아 네트워크의 부하를 줄일 수 있고, 이동 호스트에 대한 빠른 탐색을 통하여 신속한 서비스를 제공할 수 있게 된다. 지금까지 이동 컴퓨팅 환경을 위한 위치 관리 기법들이 다수 제시되어 있으며 이들은 다음과 같다.

Krishna 등이 제안한 기술 보고서에서는 북아메리카의 EIA/TIA에서 표준안으로 제시한 IS-41을 기본 바탕으로 이를 확장하는 방안을 모색하였다[3, 6]. 이 보고서에서는 항상 홈 위치 서버(HLS : Home Location Server)에 보내지는 갱신 메시지의 부하를 줄일 수 있도록 이전의 위치 서버에 다음 위치 서버로의 포인터를 유지하도록 하였고, 차후 탐색 시에는 위치 서버가 다음 위치 서버로의 포인터를 추적하여 이동 호스트를 찾을 수 있도록 하는 기법을 제시하였다.

Badrinath 등의 논문에서는 PCN에서의 탐색 전략을 주로 소개하고 있는데, 탐색 전략들은 가장 상위 위치 서버에서부터 질의를 병렬적으로 하위 위치 서버에 보내어 마지막 종단까지 이동 호스트를 찾는 flat 전략과 먼저 홈 위치 서버에 질의를 보내어 하위에 이동 호스트가 없을 시에 다시 상위의 부모 위치 서버에 질의를 보내되 이를 순차적으로 상위로 전개하는 expanding 전략, 그리고 flat 전략과 expanding 전략을 혼용한 hybrid 전략 등을 제시하였다[1].

Shivakumar 등은 사용자의 profile을 중복 배치함으로써 탐색 시간을 줄이고자 하였는데, 이에 그래프 이론을 적용하여 min-cost max-flow를 찾아 정보의 중복을 통한 사용자의 탐색 시간을 줄이는 방안을 제시하였다[5].

Tabbane은 사용자의 위치 정보를 profile의 존재 확률로 정리한 후 검색시에 존재 확률에 따른 검색을 수행하였는데[12], 이때 사용자의 위치가 사용자의 위치 정보 profile에 존재하는 경우만을 고려하였다.

Pollini 등은 calling history 정보를 사용하여 위치 예측을 하는 방법을 제안하였는데[13], 이는 호출 빈도가 높고 이동 빈도가 낮은 상황을 고려하여 사용자의 calling history로 갱신 작업을 대신하도록 하였다.

정동근 등은 사용자의 이동 빈도가 많고 적음에 따라 분류하여 클래스를 만들고, 이에 따라 사용자들에 대한 탐색 시에 클래스별로 다른 알고리즘을 적용하여 사용자를 찾을 수 있는 방법을 제안하였다[14].

위치 관리 기법에서 탐색 시간을 줄이기 위해서는 위치 정보를 많은 위치 서버에 중복해야 하는 부하가 생기고 반대로 위치 정보를 적게 둔다면 탐색 시간이 많이 걸리는 trade-off 문제가 발생한다[7, 8]. 따라서 위치 정보를 적절히 두고 탐색 시간을 일정 수준으로 유지하도록 하는 것은 위치 관리 기법에서의 중요한 문제중의 하나이다[6].

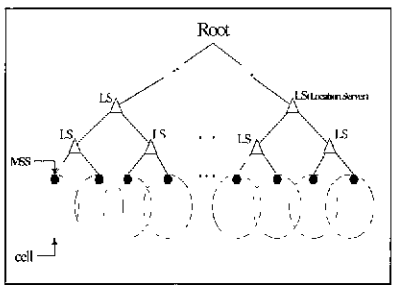
본 논문에서 제안하는 기법은 다음과 같은 점에서 위에서 제시한 기법들[12, 13, 14]과 차이점을 보인다.

본 논문에서 제안한 기법은 기존 시스템의 구조적 변화 없이 세부 알고리즘의 변형만을 요구하며, 사용자의 위치가 과거 기록으로부터 얻어진 profile에 존재하지 않는 경우까지 고려함으로써 어느 정도의 부하는 있지만 예외적인 상황에 대처할 수 있도록 제안되었다. 이에 반해 Tabbine의 기법은 사용자 profile 이외의 지역에 존재하는 사용자에 대한 검색이 이루어지지 않는 단점이 있으며[12], Pollini의 기법은 이동 빈도가 높은 사용자에 대한 취약성이 너무 크다는 단점을 갖는다[13]. 또한, 정동근 등이 제시한 기법은 검색 알고리즘을 클래스별 적용함으로써 클래스의 분류와 각 클래스에 대한 별도의 처리를 해야 하는 오버헤드 문제가 발생하게 된다[14].

3. 시스템 모델

본 논문에서 제시하는 시스템 모델은 (그림 1)에서와 같다. 위치 서버들은 계층적 구조로 구성되며, 이의 가장 하위에 이동 지원국(MSS : Mobile Support Station)들이 존재한다. 이 종단 노드를 레벨-0으로 하였을 경우 레벨 1부터 레벨 n까지는 위치 서버들의 집합이다. 계층적인 위치 서버들 중 레벨 1의 위치 서버들은 working set의 집합이 될 수 있으며, 실질적으로 지역적인 이동 호스트들의 정보를 관리하는 주체가 된다. 레벨 2의 위치 서버들은 홈 위치 서버들의 계층이며, 이들은 하위 위치 서버들의 정보를 보조하고, 각 이동 호스트의 홈 위치 서버로서 이동 호스트들에 대한 working set 정보를 저장한다. 홈 위치 서버 상위의 위치 서버들은 전역 위치 서버로서 지역 위치 서버들을 관리하고, 하위에 존재하는 위치 서버들의 관리 및 지역 그룹간의 통신 요구에 대한 중간 매체가 될 수 있다

가장 상위의 위치 서버는 네트워크간의 접점이 되며



(그림 1) 시스템 모델

하위 위치 서버들의 관리를 목적으로 한다. 본 논문에서는 root 위치 서버에 대한 기능은 언급하지 않는다. 레벨-0의 이동 지원국은 자신이 관리하는 셀(cell)에 위치한 이동 호스트와 직접 무선 통신을 하는 주체가 된다.

4. 제안 기법

본 논문에서 제안하는 기법은 동적 working set의 운영으로 인하여 데이터베이스 부하를 증가시키는 오버헤드를 갖지만, 궁극적으로 이동 호스트의 위치 정보 저장 시에 갱신 메시지의 수를 줄여 네트워크 트래픽을 줄이는 것을 목적으로 한다. 4.1 절에서는 working set을 설정하기 위한 기본 방침과 이를 위한 자료구조를 보이고, 4.2 절에서는 동적인 working set을 설정하기 위하여 각 이동 호스트의 버퍼 큐의 운영과 큐의 내용을 바탕으로 working set 설정하는 과정에 대하여 설명하겠고, 4.3 절에서 이를 기반으로 한 갱신 기법과 탐색 기법을 제시한다.

4.1 자료구조

각 구성요소들, 즉, 홈 위치 서버, 위치 서버, 그리고 이동 지원국들이 갖는 자료 구조는 지속적인 위치 갱신 메시지가 홈 위치 서버로 집중되는 것을 줄일 수 있도록 설계하였다. 그리고 각 구성 요소들은 다음과 같은 자료 구조를 이용하여 위치 갱신 정보를 저장하게 된다

홈 위치 서버는 각 이동 호스트들에 대한 식별자를 저장하고, 또한 각 이동 호스트마다의 working set 정보와 시간대별로 이동 호스트가 각 working set에 존재할 확률을 유지함으로써 차후 호출자(caller)가 피호출자(callee)를 탐색할 때 이 정보들을 유용하게 사용하도록 한다. 홈 위치 서버가 유지해야 할 자료구조는 다음과 같다.

- 홈 위치 서버에 등록된 이동 호스트들의 목록
  - HLS(ls) = {mh_1, mh_2, ⋯, mh_k}
- 자신에게 등록된 이동 호스트들에 대한 working set 정보
  - WS(mh) = {ls_1, ls_2, ⋯, ls_l}
- 이동 호스트가 working set 내의 각 지역에 존재할 확률

- Prob(mh) = {{P_{11}, P_{12}, ···, P_{1l}}, {P_{21}, P_{22}, ···, P_{2l}}, ···, {P_{m1}, P_{m2}, ···, P_{ml}}}
- 시간대를 m개의 영역으로 분할.
- P_{ni}는 이동 호스트 mh가 i 시간대에 ls_j 지역에 존재할 확률
- 단, $\sum_{i=0} P_{ni} \leq 1,\ 1 \leq n \leq m$

위치 서버들은 현재 어떤 이동 호스트들이 하위 이동 지원국에 진입하여 머무는 지에 대한 정보와, 이동 호스트가 working set이 아닌 다른 위치 서버로 이동했을 경우 이에 대한 정보 등을 갖는다. 현재 위치 서버 하위의 이동 지원국에 이동 호스트가 존재한다면 위치 서버는 MH_CUR(ls) 테이블에 이동 호스트의 식별자와 이동 호스트가 존재하는 이동 지원국의 식별자를 기록, 유지하고, 이동 호스트가 working set이 아닌 다른 위치 서버로 이동하였을 경우는 MH_FPTR(ls) 테이블에 이동 호스트의 식별자와 이동한 위치 서버로의 전방 포인터(forwarding pointer)의 정보를 기록, 유지한다. 따라서 위치 서버들이 갖는 정보들은 다음과 같다.

- 위치 서버 하위의 이동 지원국에 존재하는 이동 호스트들의 정보
  - MH_CUR(ls_p) = {mh_{p1}, mh_{p2}, ···, mh_{pm}}
  - mh_{pi} : (mh_id, fp_mss)
  - fp_mss는 위치 서버 ls_p 하위의 이동 지원국에 대한 식별자
- 현재 ls_p가 전방 포인터를 유지하고 있는 이동 호스트들에 대한 정보
  - MH_FPTR(ls_p) = {mh_{l1}, mh_{l2}, ···, mh_{lo}}
  - mh_{li} : (mh_id, fp_ls)
  - fp_ls는 해당 이동 호스트가 존재하는 위치 서버로의 포인터

이동 지원국은 현재 자신의 영역에 존재하는 이동 호스트에 대한 정보만을 저장하는데 이는 다음과 같다.

- 현재 해당 이동 지원국에 존재하는 이동 호스트들에 대한 정보
  - MH_CUR(mss_s)={mh_{s1}, mh_{s2},···, mh_{st}}
  - t는 현재 이동 지원국에 존재하는 이동 호스트의 개수

이동 호스트들은 자신이 이동한 지역의 위치 서버에 대한 식별자 정보를 얻게 되며, 위치 서버를 변경할 때마다 자신의 큐에 위치 서버의 식별자와 그 위치 서버에 머무른 시간을 저장한다. 또한, 차후 working set을 설정할 때 현재의 상태가 working set 설정에 커다란 영향을 미치는 것을 방지하기 위하여 working set 설정 시에 현재 메인 큐의 내용을 보조 큐에 저장하도록 한다.

Working set의 설정은 Δt 시간마다 이루어지며, Δt시간이 지나면 메인 큐의 내용은 보조 큐로 이동하게 되고, 메인 큐는 초기화 후 다시 Δt 시간동안 위치 서버 및 위치 서버에 존재한 시간을 지속적으로 기록하게 된다. 또한 이동 호스트는 자신의 working set이 되는 위치 서버들의 정보를 가지고 있으므로 위치 서버 변경 시 홈 위치 서버와의 메시지 교환 없이 새로운 위치 서버에 자신의 위치를 등록하게 된다.

- 메인 큐 정보
  - Track_MH(mh_m) = {{ls_{a1},t_{a1}}, { ls_{a2}, t_{a2}}, ···, {ls_{am},t_{am}}}
  - 위 기록은 일정시간 Δt 동안 기록됨
  - $\sum_{i=0} t_{ai} = \Delta t$
  - t_{ai}는 위치 서버 ls_{ai}에 존재한 시간
  - 이동 지원국을 변경하여도 위치 서버를 변경하지 않은 경우는 한 위치 서버에 기록
- 보조 큐 정보
  - Past_Track_MH(mh_{pa}) = {{ls_{pa1},t_{pa1}}, { ls_{pa2}, t_{pa2}}, ···, {ls_{pan},t_{pan}}}
  - working set을 설정할 시기까지 Track_MH(mh_{pa})에 저장되어 있던 내용
- 이동 호스트 자신의 working set 정보
  - WS(mh) = {ls_1, ls_2, ···, ls_l}
  - 메인 큐와 보조 큐의 정보를 바탕으로 설정됨

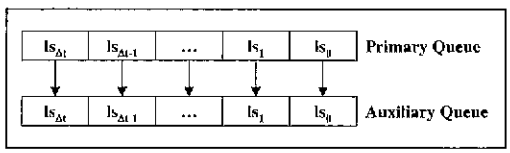
이동 호스트가 위치 서버를 변경할 때에는 새로운 위치 서버와 working set의 내용을 비교하여 이전 위치 서버에서 전방 포인터를 유지할 것인지 아니면 이전 위치 서버의 기록을 삭제할 것인지를 결정하게 된다.

4.2 큐의 운영과 working set의 설정

4.1절에서는 각 구성 요소들이 유지하는 자료구조를 살펴보았다. 이러한 자료구조를 기반으로 이동 호스트

의 working set을 설정하기 위한 큐의 운영과 working set의 설정, 그리고 working set의 운영 방법은 다음과 같다.

동적인 working set을 얻기 위해서는 이동 호스트가 이동한 지역에 대한 정보가 필요하게 된다. 이때 각 이동 호스트는 자신들의 이동 기록을 큐에 유지하게 된다. 큐에 기록되는 정보는 이동 호스트가 최근 Δt 시간동안 이동한 위치 서버들에 대한 기록이다. 이동 호스트의 위치 서버 변경에 대한 정보는 지속적으로 이동 호스트의 큐에 삽입되고, Δt 시간 후에 working set이 설정되면 메인 큐의 내용은 보조 큐로 복사되어 차후 working set 설정 시에 working set 설정에 반영되도록 한다. 이는 이동 호스트가 이동 호스트의 최근 위치 서버 변경 기록으로만 이루어진 큐의 내용으로 working set을 설정하면 이동 호스트의 예기치 않은 이동이 working set 설정에 큰 영향을 주게 되기 때문이다. 본 제안 기법의 평가를 위한 시뮬레이션 과정에서는 Δt 값을 1일로 설정하였다. 이러한 큐의 운영은 다음 (그림 2)에서와 같이 표현 할 수 있다



(그림 3) Working set 설정을 위한 큐의 운영

이때 working set의 설정은 이동 호스트의 현재 큐의 내용과 보조 큐의 내용을 반영하여 이루어지게 되는데 그 과정은 알고리즘 1에서와 같다 홈 위치 서버는 해당 이동 호스트에 대한 working set을 설정하기 위해서 메인 큐와 보조 큐의 정보를 기반으로 Δt 시간에 대하여 이동 호스트가 각 위치 서버들에 머무른 비율을 구한다. 이후 이동 호스트가 메인 큐의 각 위치 서버에 존재할 비율에 주어진 가중치 W_m을 곱하고, 보조 큐의 각 위치 서버에 존재할 비율에 가중치 W_a를 곱하여 더한 후 존재 확률이 주어진 시스템 파라메터 P보다 작은 위치 서버는 working set에서 제외하고 나머지 위치 서버들을 working set으로 설정한다. 이때 가중치 W_m은 W_a보다 큰 임의의 가중치로써 $W_m + W_a = 1$이다. 차후 시뮬레이션에서 W_m는 0.75, W_a는 0.25, 그리고, P는 0.05로 설정하여 결과를 얻을 수 있도록 하였다. Working set 그룹이 설정되면 이동 호스트가 시간대별 working set 위치 서버에 존재할 비율을 구하여 홈 위치 서버에 저장한다.

<알고리즘 1> Working set 설정 알고리즘

```
① from primary queue
   - extract location server's identifier
     from primary queue
   - compute mobile host's existence
     percentage for each location server
② from secondary queue
   - extract location server's identifier
     from secondary queue
   - compute mobile host's existence
     percentage for each location server
③ working set establishment
   Wm ← weight of primary queue;
   Wa ← weight of auxiliary queue,
   P ← given system parameter,
   for (i=0; i < queue_size; i++) {
       P(lsi) = Pp(lsi)*Wm + Ps(lsi)*Wa;
       if (P(lsi) > P)
          include lsi to the current working set;
   }
④ compute existence percentage for each time zone;
```

4.3 갱신 기법 및 탐색 기법

본 절에서는 동적인 working set 개념을 활용한 갱신 기법과 탐색 기법을 제시한다. 4.3.1절에서는 홈 위치 서버에 집중되는 위치 등록 메시지를 줄이기 위한 갱신 기법을 제시하고, 4.3.2절에서는 제안 갱신 기법을 기반으로 하는 탐색 기법을 제시한다.

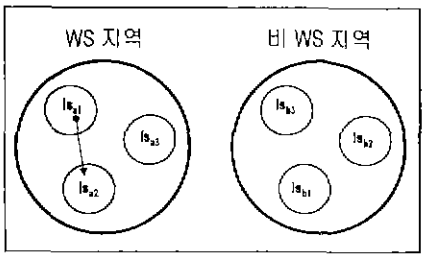
4.3.1 갱신 기법

갱신 기법에서는 4.1절의 자료구조와 4.2절의 working set 설정 기법을 사용한다. 이동 호스트가 이동 지원국 mssa에서 mssb로 이동하였을 경우 mssa를 old_mss로, mssb를 new_mss로 표기하고, old_mss가 속하는 위치 서버를 old_ls, new_mss가 속하는 위치 서버를 new_ls라 표기한다. 또한 이동 호스트는 mh로 표기한다

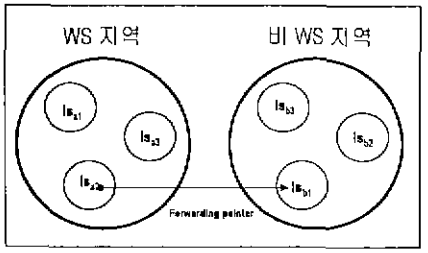
위치 갱신이 이루어지는 경우는 이동 호스트가 이동한 지역의 위치 서버와 현재 working set 정보로 비교할 때 다음 <표 1>에서와 같이 분류해 볼 수 있으며,

<표 1> 이동 호스트의 이동 형태 분류

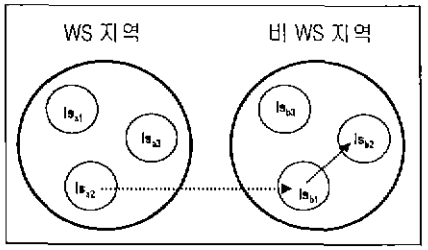
이 동	경우	표 현
WS→WS	1	new_ls∈WS(mh) and old_ls∈WS(mh)
WS→비WS	2	new_ls∉WS(mh) and old_ls∈WS(mh)
비WS→비WS	3	new_ls∉WS(mh) and old_ls∉WS(mh)
비WS→비WS	4	new_ls∈WS(mh) and old_ls∉WS(mh)



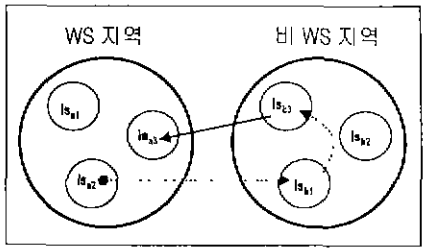
(그림 3) 경우-1의 이동 모델



(그림 4) 경우-2의 이동 모델



(그림 5) 경우-3의 이동 모델



(그림 6) 경우-4의 이동 모델

각 경우를 (그림 3), (그림 4), (그림 5), 그리고 (그림 6)에서 구체적으로 보인다.

위에서 제시한 4가지 경우에 대한 위치 갱신 알고리즘은 다음과 같다. 우선 경우-1에서와 같이 WS 지역에서 WS 지역으로 이동하는 경우에는 해당 이동 호스트가 working set 내의 위치 서버들을 순방하는 경우로써, 갱신 정보는 new_mss가 유지하는 MH_CUR(new_mss) 데이블에 이동 호스트의 식별자를 추가하고, new_ls가 유지하는 MH_CUR(new_ls) 테이블에 이동 호스트가 위치한 이동 지원국의 식별자를 추가한다. 그리고, old_ls와 new_ls가 다른 위치 서버라면 이동 호스트의 Track_MH(mh) 테이블에 old_ls에 머문 시간을 먼저 첨가하고 이후에 new_ls의 식별자를 메인 큐에 추가한다. 이와 같은 과정을 알고리즘으로 표현하면 알고리즘 2에서와 같다.

<알고리즘 2> 경우-1에 대한 알고리즘

```
if (new_ls ∈ WS(mh) and old_ls ∈ WS(mh)) {
  if (new_ls == old_ls) {
    insert mh into MH_CUR(new_mss) table of new_mss,
    update mh information of MH_CUR(new_ls)
      to (mh_id, new_mss);
    delete from MH_CUR(old_mss) table of old_mss;
  }
  else {
    insert mh into MH_CUR(new_ls) table of new_ls,
    insert mh into MH_CUR(new_mss) table of new_mss,
    delete mh from MH_CUR(old_ls) table of old_ls,
    delete mh from MH_CUR(old_mss) table of old_mss,
    insert new_ls into Track_MH(mh) table of mh;
  }
}
```

경우-2에서와 같이 mh가 WS 지역에서 비 WS 지역으로 이동하는 경우에는 working set 지역에서 mh의 이동에 대한 정보를 유지하도록 한다. 이를 위해 MH_CUR(new_mss) 테이블에는 이동 호스트 mh의 식별자가 추가되고, working set 지역의 위치 서버, 즉, old_ls는 MH_FPTR(old_ls) 테이블에서 new_ls로의 전방 포인터를 유지하도록 한다. 이와 같은 경우의 과정을 알고리즘 3에서 보인다.

<알고리즘 3> 경우-2에 대한 알고리즘

```
if (new_ls ∈ WS(mh) and old_ls ∈ WS(mh)) {
  insert mh into MH_CUR(new_ls) table of new_ls,
  insert mh into MH_CUR(new_mss) table of new_mss,
  insert pointer to new_ls into MH_FPTR(old_ls) of
    old_ls;
  delete mh from MH_CUR(old_ls) table of old_ls,
  delete mh from MH_CUR(old_mss) table of old_mss;
  insert new_ls into Track_MH(mh) table of mh;
}
```

경우-3에서와 같이 mh가 비 WS 지역에서 비 WS 지역으로 이동하는 경우에는 이동 호스트의 위치 정보를 new_ls에 저장하고, old_ls는 MH_FPTR(old_ls) 테

이블에서 new_ls로의 전방 포인터를 유지하도록 한다. 이러한 과정을 알고리즘 4에서 보인다

<알고리즘 4> 경우-3에 대한 알고리즘

```
if (new_ls ∈ WS(mh) and old_ls ∉ WS(mh)) {
  if (old_ls == new_ls) {
    insert mh into MH_CUR(new_mss) table of
      new_mss;
    update mh information of MH_CUR(new_ls) to
      (mh_id, new_mss),
    delete mh from MH_CUR(old_mss) table of
      old_mss;
  }
  else {
    insert mh into MH_CUR(new_ls) table of new_ls;
    insert mh into MH_CUR(new_mss) table of
      new_mss,
    insert pointer to new_ls into MH_FPTR(old_ls)
      of old_ls;
    delete mh from MH_CUR(old_ls) table of old_ls;
    delete mh from MH_CUR(old_mss) table of
      old_mss,
    insert new_ls into Track_MH(mh) table of mh,
  }
}
```

경우-4에서와 같이 mh가 비 WS 지역에서 WS 지역으로 이동하는 경우에는 이동 호스트의 위치 갱신 정보를 new_ls 및 new_mss에 삽입하고, old_ls 및 old_mss에 존재하는 정보는 삭제된다 그리고 위치 정보가 갱신되는 동안 new_ls는 이동 호스트의 홈 위치 서버에게 그 동안 유지되던 전방 포인터의 삭제를 요구한다. 홈 위치 서버는 전방 포인터 삭제 시 이동 호스트의 working set 위치 서버로부터 유지되고 있는 전방 포인터를 순차적으로 제거한다 이 경우의 동작 과정을 알고리즘 5에서 보인다.

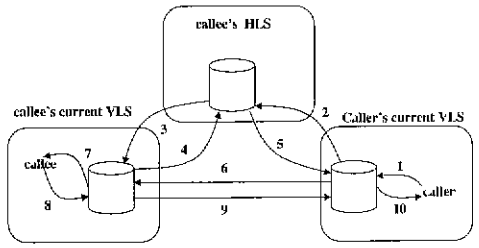
<알고리즘 5> 경우-4에 대한 알고리즘

```
if (new_ls ∈ WS(mh) and old_ls ∉ WS(mh)) {
  insert mh into MH_CUR(new_ls) table of new_ls,
  insert mh into MH_CUR(new_mss) table of new_mss,
  insert new_ls into Track_MH(mh) table of mh;
  purge forwarding pointers maintained for the mh,
}
```

4.3.2 탐색 기법

탐색은 임의의 이동 호스트 또는 고정 호스트가 목적 이동 호스트와 통신을 하고저 할 때 필요한 기법이다. 기존 IS-41의 탐색 기법은 다음에 보여지는 (그림 7)에서와 같다.

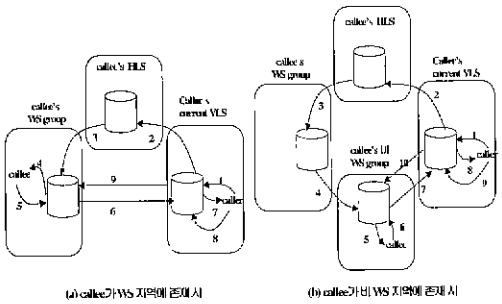
IS-41의 고전적인 탐색은 caller가 callee를 호출하였을 경우에 caller의 현재 지역 위치 서버(VLS)의 데이터베이스에서 callee가 존재하는지에 대한 검색을 수행



(그림 7) IS-41의 탐색 기법

하고, 이에 callee의 식별자가 존재하지 않을 경우에는 caller의 지역 위치 서버가 callee의 홈 위치 서버에게 callee의 위치를 질의하게 된다. callee의 홈 위치 서버는 callee의 위치 정보를 검색하여 callee의 지역 위치 서버에 callee의 존재 여부를 질의한다. callee의 지역 위치 서버는 callee의 상태(status)를 확인하고, 이를 포함한 정보를 callee의 홈 위치 서버에 응답한다. callee의 홈 위치 서버는 다시 이 응답을 caller의 지역 위치 서버에게 전달한다. 이후 caller의 지역 위치 서버는 callee의 지역 위치 서버에게 연결을 요구하고, 이 요구가 callee에게 전달되어 연결이 이루어지게 된다.

본 논문에서 제안하는 탐색 기법은 4.3.1절에서 제안한 갱신 기법을 기반으로 위치 파악 지연을 줄이기 위하여 (그림 8)에서와 같은 순서로 이루어진다.



(그림 8) Working set을 이용한 탐색 기법

(그림 8(a))는 callee가 working set 지역에 존재하는 경우의 탐색 과정이다. 이 과정에서는 caller가 callee를 호출할 때 callee의 식별자를 caller의 지역 위치 서버에 전달한다. caller의 지역 위치 서버는 callee가 자신의 지역 위치 서버에 존재하는지를 검사하고, 존재하지 않을 경우에는 caller의 지역 위치 서버가 callee의

홈 위치 서버로 callee의 위치를 질의하게 된다. callee의 홈 위치 서버는 callee에 대한 working set 정보를 기반으로 시간대별 확률에 따라 working set을 정렬한 다음 현재 callee가 존재할 확률이 가장 큰 위치 서버로 callee에 대한 위치 질의를 하게 된다. 이 질의는 callee가 존재하는 위치 서버까지 순차적으로 전달된다. 즉, 질의된 위치 서버에 callee가 존재하지 않는다면, 다음 위치 서버로 질의가 전달되는 과정을 반복하게 된다. 질의가 전달되는 과정 도중 callee가 working set 위치 서버 중에 존재하면, callee의 지역 위치 서버는 callee에게 상태를 질의하고, callee가 수신 가능한 상태라면 caller의 지역 위치 서버에게 연결을 요구한다

(그림 8(b))는 callee가 비 working set 지역에 존재하는 경우의 탐색 과정이다. 이 경우에는 (그림 8(a))에서와 비슷한 과정을 거치다가 callee의 홈 위치 서버로부터 working set 위치 서버들에게 피 caller의 위치를 질의하는 과정에서 병렬적으로 callee의 전방 포인터를 검색하게 된다. 이때 callee에 대한 전방 포인터가 발견되면, callee의 전방 포인터를 추적하게 되며, 이 과정에서 callee의 위치가 파악되면, callee의 방문 위치 서버는 callee의 상태를 질의하고, 통신이 가능한 상태라면, caller의 지역 위치 서버에게 직접 연결을 요구하도록 한다. 이의 같은 과정을 알고리즘 6에서 보인다.

<알고리즘 6> 탐색 과정

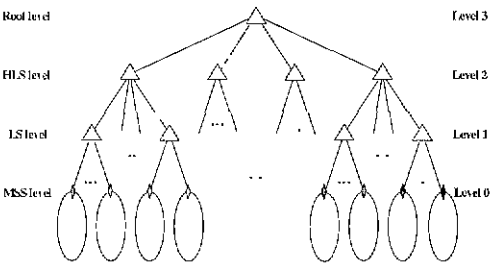
```
// caller
caller sends callee's ID to its VLS;

//caller's VLS
if (callee ∈ MH_CUR(caller's VLS))
  if (callee_status == Ready)
    caller's VLS requests connection to callee.
  else
    caller's VLS replies to caller that the callee
      is busy or off;
caller's VLS queries callee's HLS;

//callee's HLS and provable VLS's
callee's HLS sorts the callee's working set;
for (i = 1; i < workingset_size; i++) {
    ls_target = workingset[i].
    if (callee ∈ MH_CUR(ls_target))
      goto search.
    else if (callee ∈ MH_FPTR(ls_target)) {
      ls_target = MH_FPTR(ls_target) fp_ls;
      break;
    }
}
while(1) {
   if (callee ∈ MH_CUR(ls_target))
     goto search;
   else if (callee ∈ MH_FPTR(ls_target))
     ls_target = MH_FPTR(ls_target) fp_ls.
   else
     notify that connection is impossible.
}
search:
if (callee_status == Ready)
  callee's VLS requests connection to caller's VLS.
else
  callee's VLS replies to caller's VLS that callee
    is busy or off;
```

5. 성능 평가

위치 관리 정책에서는 갱신 비용과 탐색 비용 사이에 trade-off가 존재하게 된다. 또한 이 trade-off는 이동 호스트의 위치 정보를 저장하는 저장 방법과 이 저장 정보를 사용하여 탐색하는 탐색 방법에 영향을 받게 되고, 모든 기법들은 이동 빈도와 호출 빈도의 증감에 따라 그 성능의 차이를 보이게 된다 따라서 본 절에서는 이동 빈도와 호출 빈도의 변화가 갱신 비용과 탐색 비용에 미치는 영향을 평가한다. 특히 위치 관리 기법은 사용자의 CMR(Call-to-Mobility Ratio)의 변화에 따라 그 성능의 변화가 생기는데 본 논문은 이를 중심으로 시뮬레이션 결과를 얻을 수 있도록 한다. 여기에서 CMR은 이동 빈도에 대한 호출 빈도로써 이동 빈도는 단위 시간당 cell을 통과하는 비율을 말한다.

5.1 시스템 환경 모델

성능 평가를 위해 설정된 시스템의 논리적인 구조는 하위에 4개의 노드를 갖는 계층적인 트리 구조로서 깊이가 4인 구조를 가정한다. 따라서 모든 위치 서버의 수는 $4^{(4-1)}-(4-1)$, 즉, 21개가 되고 이동 지원국은 $4^{(4-1)}$, 즉, 총 64개를 갖도록 구성된다. 위치 서버들 중에는 root 위치 서버 한 개와 네 개의 홈 위치 서버가 포함된다. 이와 같은 구조를 (그림 9)에서 보인다.

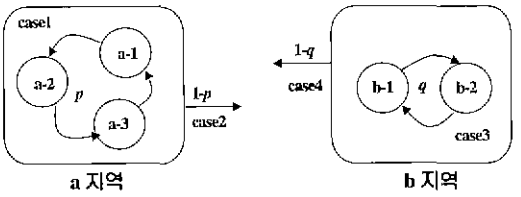


(그림 9) 성능 평가 모델의 구조

각 노드의 식별자는 상위 계층으로부터 부여하여 하

위에 존재하는 위치 서버나 이동 지원국 등은 상위 위치 서버의 식별자를 인코드(encode) 하도록 하였고, 이동 호스트도 마찬가지로 홈 위치 서버의 식별자를 포함하도록 하여 임의의 위치 서버가 이동 호스트의 홈 위치 서버를 찾을 수 있도록 설정하였다.

시뮬레이션을 위하여 몇 가지 고려해야 할 사항들로는 이동 호스트의 이동 상황과 각 이동 호스트에 대한 호출 형태, 그리고 이동 빈도와 호출 빈도 사이의 관계 등이 있다. 먼저 이동 호스트는 제안 기법에서 제시한 것처럼 working set 지역과 비 working set 지역을 이동하게 된다. 이러한 상황을 이동 확률과 함께 (그림 10)에서 보인다.

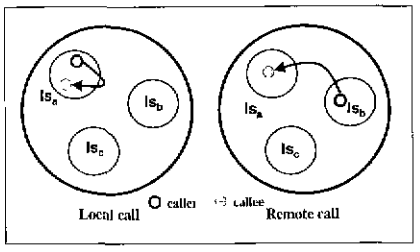


(그림 10) 사용자들의 일반적 이동 모델

(그림 10)에서 a 지역을 working set 지역이라 하고 b 지역을 비 working set 지역이라 가정하였으며, 시뮬레이션 과정에서 두 지역간의 이동 확률을 다음과 같이 설정하였다.

- Prob(a 지역 → a 지역) = p = {0.7, 0.75, 0.8, 0.85, 0.9}
- Prob(a 지역 → b 지역) = 1-p
- Prob(b 지역 → b 지역) = q = 0.25
- Prob(b 지역 → a 지역) = 1-q = 0.75

호출 모델에서는 caller의 callee에 대한 근거리(local) 호출과 원거리(remote)호출을 고려하여 근거리 호출에 대한 비율을 p_c 정도로, 원거리 호출에 대한 비율을 $(1-p_c)$ 정도로 유지하도록 하였다. 시뮬레이션 과정에서는 p_c를 0.25로 설정하였다. 근거리 호출과 원거리 호출에 대한 정의는 (그림 11)에서와 같다.



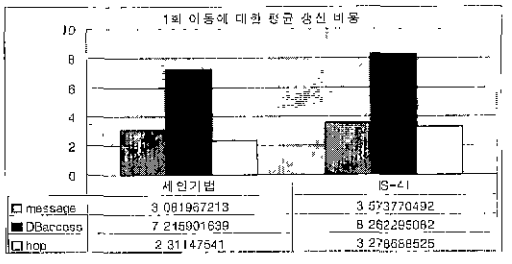
(그림 11) 근거리 호출과 원거리 호출

5.2 성능 분석 및 평가

성능 평가는 이동 호스트가 자신의 위치를 변경하는 상황에서 working set 지역으로 이동하는 비율과 비 working set 지역으로 이동하는 비율을 시스템 환경 모델에서 설정한 값들을 고려하여 수행하였다. 우선 IS-41 기법에서의 갱신 비용과 제안 기법의 갱신 비용을 비교하였고, 다음으로 각 기법의 탐색 비용을 비교하였으며, 마지막으로 이동 호스트의 CMR 변화에 따른 위치 관리비용을 비교, 분석하였다.

5 2.1 갱신 비용 분석

(그림 12)에서는 임의의 이동 호스트가 여러 셀을 이동하는 경우의 갱신 비용을 보이는데 이때 사용된 working set 지역으로의 이동 비율은 0.9로 하였을 경우이다. 이 그림은 IS-41의 갱신 기법과 제안 기법의 갱신 기법을 사용할 경우 각각에 대해 1회 이동시 발생하는 메시지의 수, 데이터베이스 접근 수, 그리고 이동 지역간의 홉 수의 평균을 보인다.

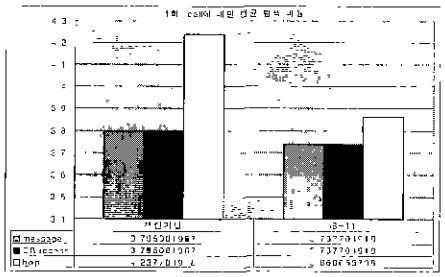


(그림 12) 1회 갱신 시의 운영 비용 비교

두 기법의 평가 결과를 비교해 보면, 이동 호스트의 1회 이동시 갱신과 관련하여 전송되는 메시지의 수로 볼 때 제안 기법이 IS-41 기법에 비해 약 14% 정도의 비용 감소를 보였고, 데이터베이스 접근과 관련해서는 약 3% 정도의 비용 감소를 보였다. 마지막으로 갱신 메시지가 전송되는 과정에서의 홉 수와 관련해서는 약 30%정도의 비용이 감소되었다.

5.3.2 탐색 비용 분석

(그림 13)에서는 임의의 한 이동 호스트에 대한 1회 호출 시 발생하는 평균 비용의 결과를 보이는데 이때 사용된 working set 지역으로의 이동 비율은 0.9로 하였을 경우이다. 이 결과에 따르면 탐색 시에 전송되는 메시지의 수와 데이터베이스 접근에서는 약 2%정도 비용이 증가되었고, 탐색 메시지의 전송에 필요한 홉 수에서는 약 10%정도 비용이 증가하였다.

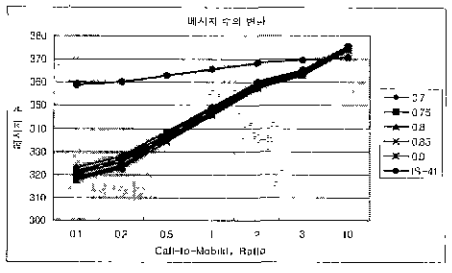


(그림 13) 1회 탐색 시의 운영 비용 비교

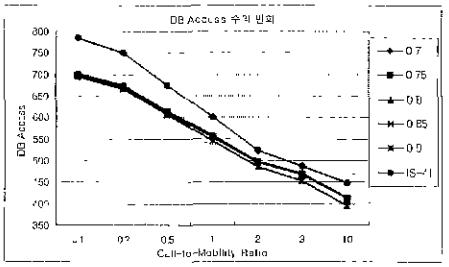
5.2.3 CMR의 변화에 따른 분석

위의 성능 평가 결과로 볼 때 IS-41 기법과 제안 기법이 탐색 비용과 갱신 비용 면에서 각각 효율성을 보이고 있음을 알 수 있다. 본 절에서는 호출 빈도에 대한 이동 빈도의 비율, 즉, CMR에 따른 각 기법의 성능을 분석해 본다.

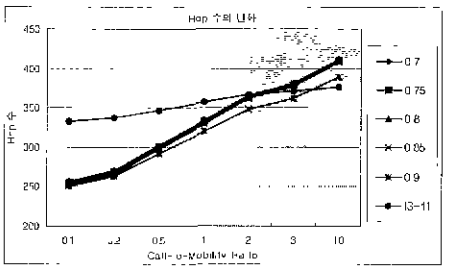
(그림 14)와 (그림 15) 그리고 (그림 16)에서는 CMR 변화와 이동 비율에 따라 제안 기법을 적용하였을 경우와 IS-41 기법을 적용하였을 경우의 메시지 수, 데이터베이스 접근 수, 그리고 홉의 수에 대한 결과를 각각 보인다. 위 그림에서 보이는 바와 같이 모든 성능 평가 지표에 대해 대체로 CMR이 낮은 경우에는 제안 기법이 효율적이지만, 반대로 CMR이 6이상 크게 높아지는 경우에는 성능이 역전되는 현상을 보이



(그림 14) CMR 변화에 따른 메시지 수



(그림 15) CMR 변화에 따른 DB 접근 수



(그림 16) CMR 변화에 따른 홉 수

고 있다 결과적으로 본 제안 기법은 CMR이 극히 크지 않은 경우에 효율적인 기법으로 사용될 수 있음을 보이는 것이다

6. 결 론

본 논문에서는 동적인 working set 개념을 이용하는 위치 관리 기법을 제시하였다. 제안 기법은 홈 위치 서버로 지속적으로 송신되는 갱신 메시지의 수를 줄이기 위하여 이동 호스트가 이동한 지역의 위치 서버와 이동 전의 위치 서버에 위치 정보를 갱신하도록 하며, 이에 따라 홈 위치 서버의 통신 부하를 줄일 수 있게 하였다. 제안한 동적 working set 기법에 따라 이동 호스트의 위치 갱신에 필요한 비용을 줄일 수 있었으나, 갱신 비용을 줄이기 위해 발생하는 부하들로 인하여 탐색 비용이 약간 증가함을 알 수 있었다

성능 평가 및 분석 결과에서 이동 호스트의 이동 상황이 working set 지역에서 working set 지역으로 이루어지는 비율이 0.9인 경우를 보면 갱신 비용의 경우 제안 기법의 갱신 비용이 IS-41 기법의 갱신 비용보다 작게는 3%에서 크게는 30% 정도까지 효율이 좋게 나타난 반면, 탐색 비용에서는 약 2% 내지 10% 정도의 효율이 저하되는 결과를 보였다 비록 이동 호스트가

working set 지역에서 working set 지역으로 이동하는 비율이 0.7인 경우라 할 지라도 IS-41 기법에서 보이는 성능보다는 대체로 좋은 결과를 얻을 수 있었다. 제안 기법의 성능은 이동 호스트가 working set 지역에서 working set 지역으로 이동하는 비율이 0.7, 0.75, 0.8, 0.85, 그리고 0.9인 상황에서 대체로 좋은 결과를 보였다. 간혹 CMR이 높은 경우에는 메시지 수와 홉 수에 있어서 제안 기법이 IS-41 기법에서 보이는 성능보다 떨어지는 것을 볼 수 있었다. 각 기법들의 성능을 CMR의 변화에 따라 비교한 결과 CMR이 크게 높지 않은 경우 제안 기법이 IS-41 기법보다 효율적임을 알 수 있었다

참 고 문 헌

[1] B. R. Badrinath, T Imielinski and A. Virmani, "Location Strategy for Personal Communication Networks," Proc. of the IEEE GLOBECOM Workshop on Networking of Personal Communication, Dec. 1992.

[2] P. Krishna, N. H. Vaidya and D. K. Pradhan, "Static and Dynamic Location Management in Distributed Mobile Environments," Technical Report #94-030, Dept. of Computer Science, Texas A&M University. Jun. 1994

[3] S. Mohan and R. Jain, "Two User Location Strategies for Personal Communication Services," IEEE Personal Communications, Vol.12, No.8, Oct. 1994.

[4] S. Rajagopalan and B R. Badrinath, "An Adaptive Location Management Strategy for Mobile IP," In Proc of First ACM MOBICOM 95, Nov. 1995.

[5] N Shivakumar and J. Widom, "User Profile Replication for Faster Location Lookup in Mobile Environments," Proc. of ACM Int'l Conference on Mobile Computing and Networking, Nov. 1995.

[6] P Krishna, N. H. Vaidya and D. K. Pradhan." Efficient Location Management In Mobile Wireless Networks," Technical Report #96-030, Dept. of Computer Science, Texas A&M University, Jul 1996

[7] D. Lam, J. Jannink, D. C. Cox, and J. Widom, "Modeling Location Management in Personal Communication Services." Proc. of 1996 IEEE Int'l Conference on Universal Personal Communications (ICUPC'96), Sep. 1996

[8] B. R. Badrinath and T Imielinski, "Replication and Mobility," To appear in the Proc. of the Second Workshop on the Management of Replicated Data, Nov 1992.

[9] P Bhagwat, S. Tripath and C. Perkins, "Network Layer Mobility: an Architecture and Survey," Sep. 1995

[10] G. Y. Lui and G. Q. Maguire," A Predictive Mobility Management Scheme for Supporting Wireless Mobile Computing," Feb. 1995.

[11] P. Krishna, N H Vaidya and D. K. Pradhan,"Efficient Location Management In Mobile Wireless Networks." Technical Report #95-011, Dept. of Computer Science, Texas A&M University, Mar. 1995

[12] S. Tabbine, "An alternative strategy for location tracking," IEEE J. Select. Areas Communication, Vol.13, No.5, pp 880-892. Jun 1995.

[13] G. P. Pollini and C. L. I, "A profile-based location strategy and its performance," IEEE J. Select. Areas Communication, Vol.15, No.8, pp.1415-1424, Oct. 1997.

[14] D. G. Jeong and W. S Jeon, "Effective location management strategy based on user mobility calsses." In Proc of IEEE GLOBCOM '98, Sydney, Australia, Nov. 1998.

임 성 빈

e-mail : aprince@kebi.com

1997년 대전대학교 컴퓨터공학과 졸업

1999년 성균관대학교 전기전자및 컴퓨터공학부 대학원 석사

1999년~현재 ㈜데이터로직 연구원

관심분야 : 이동 컴퓨팅, 분산 객체 컴퓨팅

엄 영 익

e-mail yieom@ece.skku.ac.kr

1983년 서울대학교 계산통계학과 졸업(학사)

1985년 서울대학교 대학원 전산과학전공(석사)

1991년 서울대학교 대학원 전산과학전공(박사)

1983년~1986년 서울대학교 도서관 조교

1986년~1993년 단국대학교 전자계산학과 부교수

1993년~현재 성균관대학교 전기전자및컴퓨터공학부 교수

관심분야 : 분산 시스템, 분산 객체 컴퓨팅, 이동 컴퓨팅, 운영체제