트랜잭션의 중요도와 데이터의 유효성을 고려한 실시간 이동 트랜잭션 관리자의 스케줄링 기법

조 숙 경[†]·김 경 배[††]·이 순 조[†††]·배 해 영[††††]

요 약

본 논문에서는 이동 컴퓨팅 환경에서 발생하는 실시간 이동 트랜잭션을 처리하기 위한 트랜잭션 관리자의 스케줄링 기법을 제안한다. 제안된 스케줄링 기법은 기존의 종료시한만 고려하던 방법과는 달리 이동 호스트 때문에 발생하는 실시간 데이터의 유효성도 함께 고려하여 트랜잭션의 실행여부를 판단한다. 그 후, 트랜잭션의 중요도와 종료시한을 고려한 우선순위 큐에 최적의 실행 윈도우를 생성하여 스케줄링하고, 실행될 트랜잭션을 선택할 때는 이동 호스트와의 접속 단절을 고려한다. 따라서, 제안된 기법은 실시간 데이터의 유효성 제약조건 위반에 따른 트랜잭션의 철회 후 재시작을 감소시키며, 종료시한을 만족시키는 실시간 이동 트랜잭션의 중요도의 합을 최대로 하는 장점을 가진다. 또한 제안된 기법은 성능 평가를 통해서 기존의 기법에 비하여 실시간 이동 트랜잭션의 종료시한 만족 비율이 20% 정도 향상됨을 보였다. 이 기법은 이동 통신, 응급 재난 구조 시스템 등에서 발생하는 실시간 이동 트랜잭션의 관리자의 스케줄링 기법으로 적합하다.

Scheduling Method of Real-Time Mobile Transaction Manager considering Value of Transactions and Validity of Real-Time Data

Sook-Kyoung Cho[†] · Gyoung-Bae Kim[†] · Soon-Jo Lee[††] · Hae-Young Bae[†††]

ABSTRACT

In this paper, we present a scheduling method for real-time mobile transaction manager in mobile computing environment. The proposed method checks whether a transaction is executable or not. It is able to by considering not only the deadline of real-time transaction but also the validity of real-time data in mobile hosts. And then, it schedules the real-time mobile transactions by making optimal execution window based on the priority queue, while considering the transaction value and deadline. Disconnection with mobile hosts is monitored in selecting the transaction for execution. Using the proposed method reduces the number of restarting times after transaction aborts which is caused by the violation of the validity constraints of real-time data. And it has merits of maximizing the sum of values of real-time mobile transactions which meet the deadline. The performance evaluation demonstrates that the number of committed real-time transactions within the deadline is improved by 20%. This method can be used in real-time mobile transaction manager in such environments as cellular communications, emergency medicine information systems and so on.

키워드 : 실시간 이동 트랜잭션(real-time mobile transaction), 스케줄링(scheduling), 우선순위 할당(priority assignment), 실시간 데이터베이스 시스템(real-time database systems)

1. 서 론

최근 들어 휴대용 컴퓨터의 대중화와 무선 매체들의 급속한 발전으로 인해 사용자들 기존의 유선 네트워크의 장소 제한성을 탈피하여 사용자가 네트워크의 접속을 유지하면서 원하는 장소로 자유로운 이동을 할 수 있는 이동 컴퓨팅(mobile computing)[1-3]이 가능하게 되었다. 이러한 장소의 제약성이 없는 이동 컴퓨팅 환경의 등장으로 인해 시스템 사용자들이 고정망에서 뿐만 아니라 이동 컴퓨팅 환경에서도 고정망 수준의 실시간 처리 능력을 요구하게 됨으로써 이동 컴퓨팅 환경에서의 실시간 데이터베이스 시스템에 대한 요구가 크게 증대되었다[4, 5]. 지금까지 이동 컴퓨팅 환경에 대한 연구는 위치 관리(location management), 핸드오버(handover) 처리, 데이터 관리 등만 활발히 진행되고 있고 이동 컴퓨팅 환경에서 발생하는 실시간 데이터 및 실시간 이동 트랜잭션을 관리하는 실시간 데이터

※ 본 논문은 정보통신부의 대학소프트웨어 연구센터 지원사업의 연구결과임.
† 준 회 원 : 인하대학교 대학원 전자계산공학과
†† 준 회 원 : 한국전자통신연구원 선임연구원
††† 종신회원 : 서원대학교 컴퓨터교육학과 교수
†††† 종신회원 : 인하대학교 전자계산공학과 교수
논문접수 : 2001년 6월 1일, 심사완료 : 2001년 8월 17일

베이스 시스템에 관한 연구는 미비한 실정이다. 따라서, 본 논문에서는 이동 컴퓨팅 환경을 위한 실시간 데이터베이스 시스템의 트랜잭션 관리 및 스케줄링에 대해 연구해 본다.

이동 컴퓨팅 환경을 위한 실시간 데이터베이스 시스템에서 발생하는 실시간 이동 트랜잭션(real-time mobile transaction)은 이동 트랜잭션(mobile transaction)[6]과 실시간 트랜잭션(real-time transaction)[7, 8]이 결합된 형태이다.

이동 트랜잭션은 트랜잭션 결과의 정확성이 위치에 영향을 받는 위치 제약 조건을 가지며, 이것은 이동 트랜잭션의 위치 데이터가 실시간 데이터임을 의미한다. 실시간 트랜잭션은 종료시한(deadline)이라는 시간 제약 조건을 가지게 된다. 시간 제약 조건은 트랜잭션 결과의 가치가 정확성뿐만 아니라 시간에도 영향을 받는 조건이다. 또한 실시간 트랜잭션은 실시간 데이터를 요구하기도 하는데, 이러한 실시간 데이터는 동적으로 끊임없이 변화하므로 트랜잭션의 종료 시점까지 유효해야 한다. 이것을 실시간 데이터의 유효성(validity) 제약 조건이라 한다.

이동 컴퓨팅 환경을 위한 실시간 데이터베이스 시스템에서 발생하는 실시간 이동 트랜잭션(real-time mobile transaction)은 시간 제약 조건과 더불어 데이터의 유효성 제약 조건을 만족시켜야만 정확한 결과를 도출할 수 있다. 그러나, 이동 컴퓨팅 시스템 환경의 실시간 데이터베이스 시스템에서는 이동 호스트의 이동 특성에 의해 시간은 더욱더 귀중한 자원이 되어 실시간 이동 트랜잭션의 지원을 어렵게 한다. 이동 호스트의 이동성과 낮은 신뢰성 등으로 인해 트랜잭션 처리 시간이 지연되고, 좁은 대역폭의 무선통신 채널 사용으로 인해 이동 호스트와 고정망과의 통신 시간 지연으로 인해 전체적인 트랜잭션의 처리 시간이 길어지게 되고, 이는 트랜잭션의 시간적인 제약조건을 만족시키지 못한다는 문제점이 발생한다[9]. 그러므로, 본 논문은 이러한 문제점을 해결하고 실시간 이동 트랜잭션의 종료시한과 실시간 데이터의 유효성을 보장하기 위한 실시간 이동 트랜잭션 관리자의 스케줄링 기법을 제안한다.

제안된 스케줄링 기법을 위한 트랜잭션 관리자는 실행여부 검사기, 우선순위 할당기, 스케줄러, 디스패처(dispatcher)로 구성되어 있다. 실행여부 검사기는 트랜잭션이 수행되기 전에 종료시한 내에 수행 가능하고, 사용되는 실시간 데이터가 트랜잭션 완료(commit) 시점까지 유효성을 상실하지 않는지를 검사하여 실행 여부를 판단한다. 실행이 결정된 트랜잭션은 우선순위 할당기로 보내지며, 우선순위 할당기에서는 실시간 트랜잭션일 경우에는 EDF(Earliest Deadline First) 기법[10]과 트랜잭션의 중요도(value)를 기반으로 한 우선순위 큐에 트랜잭션을 삽입하고 비실시간 트랜잭션일 FCFS(First Come First Served)[11] 기법을 기반으로 한 큐에 삽입한다. 스케줄러에서는 우선순위 큐에 현재 시간, 트랜잭션의 수행 정보, 중요도, 그리고 종료시한을 고려한 종료시한 내에 완료가 가능한 트랜잭션들의 집합인 실행 윈도우를 생성한다. 스케줄러는 실행 윈도우 내에 있는 트랜잭션들을 스케줄링한다. 디스패처는 트랜잭션의 종류에 따라 해당 큐에 트랜잭션들을 할당하고, 수행중인 트랜잭션이 완료되면 스케줄링된 트랜잭션 중에서 가장 우선순위가 높은 트랜잭션을 선택하는 역할을 한다. 이때 디스패처는 트랜잭션의 접속이 단절되었는가를 검사하여 연결시에만 수행을 시키며, 단절시에는 다음 우선순위의 트랜잭션을 선택한다.

제안된 스케줄링 기법은 실행여부검사기를 통해 실시간 이동 트랜잭션을 위한 예측성을 향상시켜 종료시한을 만족하는 실시간 이동 트랜잭션의 수를 최대화할 수 있다. 또한 실시간 데이터의 유효성에 따른 트랜잭션의 지연으로 트랜잭션의 철회를 감소시키며, 이동 호스트와 고정망간의 접속 단절로 인한 전체 트랜잭션의 지연이나 철회를 감소시킨다.

본 논문의 구성은 다음과 같다. 2장에서 이동 컴퓨팅 시스템의 특성과 기존의 스케줄링 기법에 대하여 살펴보고, 3장에서는 이동 컴퓨팅 환경에서 실시간 트랜잭션과 실시간 데이터의 제약조건을 준수하면서 사용자가 요구한 트랜잭션의 중요도를 우선으로 하여 실행 윈도우를 이용한 트랜잭션 관리자를 설계하며, 4장에서는 제안된 스케줄링 기법과 기존의 기법을 비교하여 성능 평가를 하며, 마지막으로 5장에서 본 논문에 대한 결론과 향후 연구 방향에 대하여 논한다.

2. 관련 연구

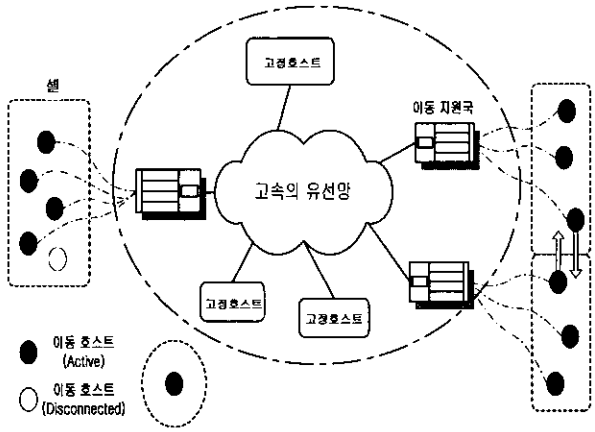
이 장에서는 관련 연구로써 이동 컴퓨팅 시스템의 구조와 기존의 실시간 데이터베이스 시스템을 위한 스케줄링 기법에 대해 조사한다.

2.1 이동 컴퓨팅 시스템의 특성

이동 컴퓨팅 환경이란 기존의 분산 컴퓨팅 환경이 이동 호스트들을 지원할 수 있도록 구성된 환경을 의미한다[2]. 현재 널리 이용되고 있는 이동 컴퓨팅 환경의 시스템에 대한 구조는 (그림 1)과 같이 이동 호스트(Mobile Host), 이동 지원국(Mobile Support Station), 그리고 고정 호스트(Fixed Host)들로 구성되어 있다[3].

이동 호스트는 노트북 컴퓨터 혹은 이동 컴퓨팅 환경에 적합하게 개발된 휴대용 컴퓨터를 사용하며 이동지원국을 통하여 고정 컴퓨터 통신망에 접속할 수 있다. 하나의 이동 지원국이 포괄하는 통신 지역을 무선 통신 셀(cell)이라 하며, 해당 이동지원국은 자신 셀 안에 있는 모든 이동 호스트와의 통신을 관장한다. 이동 지원국은 관장하고 있는 셀 내에 위치한 이동 호스트의 소속 정보, 보안 정보, 위치 정보 등을 자체 관리하고 있으며, 관련 정보를 이용하여 이동 호스트 사용자의 정보 접근을 조정한다. 고정 호스트는 고

정 사용자 및 이동 사용자에게 제공할 컴퓨팅 능력 및 정보를 관리한다. 고정 호스트에는 메인 프레임, 워크스테이션 및 개인용 컴퓨터 등을 용도 및 처리 능력에 따라 설치하여 사용한다.



(그림 1) 이동 컴퓨팅 시스템의 구조

이동 호스트와 고정 호스트의 연결성은 고정망에 안정적으로 연결되어 이동 호스트와 고정망이 원활한 데이터의 전송이 이루어지는 완전 연결 상태(fully connected), 이동 호스트가 이동 지원국의 셀 영역을 벗어나 고정망과 연결이 단절된 상태인 접속 단절 상태(disconnected), 이동 호스트가 셀 영역의 경계지역이나 전파차단 지역 등에 위치하여 고정망과의 이동 호스트간의 접속이 불안전한 상태에 있는 약한 연결 상태(weakly connected), 그리고 사용자가 이동 호스트의 전원의 절약 등을 목적으로 의도적으로 망과의 접속을 분리하는 의도적 접속 단절 상태(spurious disconnected)가 있다[12, 13].

2.2 실시간 트랜잭션 스케줄링 기법

기존의 스케줄링 알고리즘은 시스템 사용과 작업처리량을 최대화하기 위해 중앙 처리 장치 한계와 입출력 한계의 균형을 목적으로 대부분의 기존 운영체제에서 사용되었으며, 프로세스들을 공평하게 사용하도록 설계되었다. 즉, 각각의 프로세스들은 시스템 자원의 공정한 사용을 위해 수행 작업 시간, 대기시간, 그리고 응답시간 등을 고려한 복잡한 알고리즘들을 사용한다. 그러나 이러한 기법은 종료시한의 만족율을 중요시하는 실시간 트랜잭션의 스케줄링에 적합하지 않다.

데이터와 시스템의 자원에 대한 충돌을 해결하고, 트랜잭션이 지닌 시간적인 제약조건을 만족시키기 위해서 실시간 데이터베이스 시스템에서는 트랜잭션에 우선순위를 부과하는 기법을 사용한다[7, 10, 11, 14]. 이러한 우선순위 기법으로는 FCFS 기법[11], EDF 기법[10], LSF 기법[14]이 있으며 이 절에서 각각의 특징을 고찰해 본다.

가장 먼저 준비된 트랜잭션에 높은 우선순위를 할당하는 기법인 FCFS 기법은 스케줄링 간단하다는 장점을 가진다 그러나, 트랜잭션의 종료시한에 관한 정보를 사용하지 않으므로 긴급한 트랜잭션이 도착한 경우에도 먼저 대기하고 있는 트랜잭션이 앞서 처리되므로 긴급한 트랜잭션에 대해서는 대응하기가 어렵다. 이 기법은 모든 트랜잭션이 도착시간부터 항상 일정한 양의 시간을 종료시한으로 가지는 환경에 유리하다.

EDF 기법은 실시간 데이터베이스 시스템의 스케줄링에서 가장 기본적으로 이용되고 있는 기법으로 종료시한에 가장 근접한 트랜잭션에 높은 우선순위를 할당한다.

EDF 기법은 종료시한 정보를 사용하므로 긴급한 트랜잭션의 처리에 용이하고, 트랜잭션들이 적당히 적재되는 경우에는 좋은 성능을 유지하지만, 트랜잭션들이 과도하게 적재되는 경우에는 종료시한에 근접하거나 종료시한을 초과하는 트랜잭션에 높은 우선순위가 할당되는 단점을 가지고 있다.

LSF 기법은 트랜잭션이 자신의 종료시한을 만족시키면서 수행을 얼마나 늦출 수 있는 가를 나타내는 슬랙을 이용하는 기법이다. 어떤 트랜잭션의 슬랙시간은 다음과 같이 계산한다.

$$\text{슬랙시간} = \text{종료시한} - (\text{현재시간} + \text{잔여 실행시간})$$

계산된 슬랙시간이 0보다 작으면 트랜잭션은 종료시한 내에 끝나는 것이 불가능하다. 그러므로 이 기법은 트랜잭션간에 충돌이 발생할 경우, 높은 우선순위를 가진 트랜잭션이 슬랙시간 내에 처리 가능하다면 낮은 우선순위를 가진 트랜잭션을 철회하지 않고 수행함으로써 자원의 이용을 극대화할 수 있으나, 트랜잭션의 수행시간과 이를 고려한 슬랙시간을 계산하기가 어려울 뿐만 아니라 EDF기법에 비해 부하가 크다.

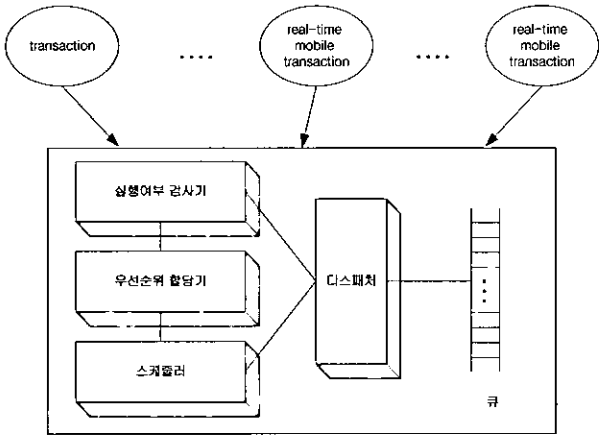
3. 실시간 이동 트랜잭션 관리자를 위한 스케줄링 방법

이 장에서는 이동 컴퓨팅 환경에서 발생되는 실시간 이동 트랜잭션을 처리하기 위한 트랜잭션 관리자의 스케줄링 방법을 제안한다. 제안된 트랜잭션 관리자는 실행여부 검사기, 우선순위 할당기, 스케줄러, 디스패처로 구성되어 있다. 다음의 절들에서 각각에 대한 세부 사항을 논한다.

3.1 실시간 이동 트랜잭션 관리자

이동 컴퓨팅 환경을 위한 실시간 데이터베이스 시스템에서는 실제 실시간 트랜잭션과 비 실시간 트랜잭션이 동시에 발생하게 된다. 즉, 시간적인 제약조건을 갖는 실시간 트랜잭션과 평균적인 응답속도를 최소화해야 하는 비 실시

간 트랜잭션이 동시에 발생하게 된다. 따라서 실시간 트랜잭션뿐만 아니라 비 실시간 트랜잭션까지 효과적으로 처리할 수 있어야 한다. 제안된 실시간 이동 트랜잭션 관리자는 (그림 2)와 같다.



(그림 2) 실시간 이동 트랜잭션 관리자의 구조

대기 큐에서 대기하던 트랜잭션이 트랜잭션 관리자로 삽입되면, 그 트랜잭션의 유형에 따라 실시간 이동 트랜잭션일 경우에는 실행여부 검사기로 보내고, 비 실시간 트랜잭션일 경우에는 큐에 삽입한다. 실행여부 검사기로 보내진 1장 높은 우선순위를 가졌다면 디스패처가 다음 실행될 트랜잭션으로 선택한다. 다음 알고리즘은 트랜잭션 관리자 알고리즘이다.

```
입력 : 트랜잭션 Ti,
       참조될 실시간 데이터 집합 RTDS(Ti),
       우선순위 큐 Q,
       큐 q
출력 : 실행이 결정된 트랜잭션 Ti
변수 : 실행 여부를 가지는 result,
       실행윈도우 EW
01 :   while (1) {
02 :     대기 큐에서 Ti 삽입 ;
03 :     if (Ti의 유형이 real-time이면) {
04 :       result = ExecutionTester (Ti , RTDS(Ti) ) ;
                 //실행여부 검사
05 :       if (result가 true이면) {
06 :         PriorityAssigrment (Ti, Q); //우선순위 할당
07 :         SchedulingInExecutionWindow(Q) ;
                   //트랜잭션 스케줄링
08 :         Dispatcher (EW); //트랜잭션 실행
09 :       }
10 :       else abort Ti ;
11 :     }
12 :     else q에 Ti 삽입 후 FCFS 방법으로 스케줄링 ;
13 :   } //end of while
```

(알고리즘) TransactionManager (Ti, RTDS(Ti), Q, q)

대기 큐에서 기다리던 트랜잭션 T_i가 트랜잭션 관리자로 삽입되면 T_i의 유형을 검사한다(line 2-3). T_i의 유형이 실시간이면 실행여부 검사기, 우선순위 할당기, 스케줄러를 거쳐 디스패처가 실행될 트랜잭션을 결정한다(line 3-9). 그러나, T_i가 실시간 트랜잭션임에도 불구하고 실행여부 검사기에서 종료시한 내에 트랜잭션이 완료가 불가능하다고 판단되었을 때는 취소된다(line 10). 트랜잭션 T_i가 실시간 유형이 아니라면 큐에 삽입 후 FCFS 방법으로 트랜잭션들을 스케줄링한다(line 12).

3.2 실행여부 검사기

실시간 이동 트랜잭션을 수행하기 전에 트랜잭션의 정보를 이용하여 해당 트랜잭션이 종료시한 내에 완료가능하고, 사용되는 데이터 객체가 트랜잭션의 완료 시점까지 유효성을 상실하지 않는지를 검사해야 한다. 따라서, 실행여부 검사기에서는 트랜잭션의 수행여부를 판단하기 위하여 다음의 조건을 만족하는 트랜잭션을 우선적으로 선택하여 수행한다.

조건 1) 실시간 이동 트랜잭션의 종료시한 만족 :
실시간 이동 트랜잭션은 종료시한 이전에 완료되어야 한다.

트랜잭션의 완료 예정시간 < 트랜잭션의 종료시한

조건 2) 실시간 데이터의 제약조건 만족 :
실시간 이동 트랜잭션은 트랜잭션에서 사용되는 실시간 데이터들의 유효시간 이전에 완료되어야 한다.

Min(참조될 실시간 데이터 집합의 유효시간)
> 트랜잭션의 완료 시간

조건 1에서 트랜잭션의 완료 예정시간은 트랜잭션 시작시간에 트랜잭션 예측 실행 시간을 합하여 구하며, 트랜잭션의 예측 실행 시간은 데이터베이스 시스템에서 사용자가 요구한 데이터베이스에 대한 연산을 수행하는 시간, 응용 프로그램에서 데이터베이스 연산의 결과를 처리하는데 소요되는 트랜잭션 연산 시간과 트랜잭션을 스케줄링하는데 소요되는 시간으로 계산한다. 실시간 이동 데이터베이스 시스템에서 발생하는 트랜잭션의 유형은 응용 시스템에 따라 대부분 결정되어 있으며, 데이터베이스에 대한 트랜잭션의 수행시간이 일정함에 따라, 이러한 트랜잭션의 예측 실행 시간은 종료시한과 더불어 메타 데이터베이스에 저장되어 필요할 때마다 참조된다. 조건 2의 Min() 함수는 최소값을 구하는 함수이다. 조건 1과 조건 2를 바탕으로 트랜잭션의 실행 여부를 판단하는 실행여부 검사기 알고리즘은 아래와 같다.

```
                //실행여부 검사기 알고리즘
입력 : 트랜잭션 Ti,
       참조될 실시간 데이터 집합 RTDS(Ti)
출력 : BOOL 값
01 : TransactionCommitTime =
       TransactionArriveTime + TransactionExecutionTime ;
```

```
02 : if (TransactionExecutionTime < TransactionDeadline)
03 : then if (MIN(RTDS(Ti)의 유효시간) >
                    TransactionCommitTime)
04 :      then return True ;
05 :      else return False ;
06 : else return False ;
```

(알고리즘) ExecutionTester(Ti, RTDS(Ti))

실행여부 검사 알고리즘에서는 우선 트랜잭션 T_i의 도착 시간과 실행 예상 시간을 더하여 트랜잭션의 예상 완료 시간을 산출한 후 트랜잭션의 종료시한과 비교해 트랜잭션이 종료시한 내에 수행 가능한지를 검사한다(line 1-2).

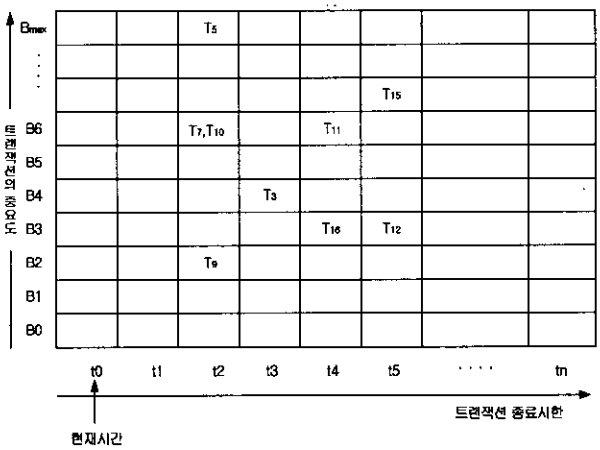
이 검사에서 트랜잭션의 완료 예상 시간이 종료시한보다 작으면 트랜잭션은 종료시한 내에 완료가 가능한 경우이므로 다음 단계로 실시간 데이터의 유효성 제약조건을 검사해야 한다. 트랜잭션 T_i가 사용하는 실시간 데이터들 중에서 가장 빨리 유효시간을 상실하는 데이터 객체의 유효시간과 트랜잭션의 예상 완료 시간을 비교하여 예상 완료 시간이 작다면 실시간 데이터의 유효시간 내에 트랜잭션의 완료가 가능함을 의미한다(line 3). 즉, 조건 1과 조건 2가 동시에 만족되는 경우에만 트랜잭션 T_i는 정상적으로 우선순위 할당기에 있는 우선순위 큐에 삽입되어 트랜잭션이 실행되지만, 하나라도 만족되지 않은 경우에는 트랜잭션을 철회한다.

3.3 우선순위 할당기

우선순위 할당기에서는 실시간 데이터베이스 시스템의 특성인 종료시한 정보를 가장 잘 반영할 수 있는 EDF 기법을 이용하여 실시간 이동 트랜잭션을 처리한다. 또한, 트랜잭션의 중요도를 기준으로 높은 중요도를 지니는 트랜잭션을 우선적으로 종료시한을 만족할 수 있도록 한다. 트랜잭션의 중요도는 트랜잭션이 완료되었을 때 나타나는 결과의 가치이다. 예를 들어, 비행기를 추적하는 트랜잭션과 미사일을 요격하는 트랜잭션이 있을 때 미사일을 요격하는 트랜잭션이 종료시한을 만족시켰을 때의 가치가 더 크므로 중요도가 높다고 할 수 있다. 이러한 트랜잭션들의 중요도는 메타 데이터베이스 저장되어 있어 우선순위 할당기에서 트랜잭션을 우선순위 큐에 배치시킬 때 활용된다.

고정망 시스템의 경우 우선순위 큐를 생성할 때는 하드(hard), 소프트(soft), 펌(firm)으로 구분되는 실시간 트랜잭션의 형태를 고려했지만, 이동 컴퓨팅 환경에서는 상이한 사용자의 요구가 발생하기 때문에 트랜잭션의 종료시한을 만족시킴으로써 얻을 수 있는 중요도가 더 중요한 의미를 가진다. 그러므로, 본 논문에서 사용하는 우선순위 큐는 (그림 3)과 같이 트랜잭션의 중요도와 종료시한을 나타내는 이차원적인 구조를 가지고 있다. 즉, x축은 트랜잭션의 종료시한을 의미하고, y축은 트랜잭션이 지니는 중요도를 나타내며 최대 값은 Bmax이다.

실행여부 검사기에서 보내진 트랜잭션들은 (그림 3)과 같이 우선순위 큐에 배치된다. 우선순위 큐에 트랜잭션을 삽입하는 방법은 트랜잭션의 우선순위와 트랜잭션의 종료시한을 기준으로 해당 셀에 할당하는 것이다. 예를 들어, 트랜잭션 T_3이 B4의 중요도를 가지고 있고, 설정된 종료시한이 t3 값 이라면 셀(t3, B4)를 할당받는다. 트랜잭션의 우선순위를 결정하는 우선순위 할당기를 위한 알고리즘은 다음과 같다.



(그림 3) 우선순위 큐

```
입력 : 우선순위가 부여될 트랜잭션 Ti,
       우선순위 큐 Q
출력 : 트랜잭션 Ti가 삽입된 우선순위 큐 Q'
변수 : 우선순위 큐 내의 트랜잭션 Tj
01 : while(Q 내의 모든 트랜잭션들에 대해) {
02 :    if (Ti의 중요도 > Tj의 중요도) {
03 :       Tj 앞에 Ti 삽입 ; exit ; }
04 :    else if (Ti의 중요도 == Tj의 중요도) {
05 :       if (Ti의 종료시한 < Tj의 종료시한) {
06 :          Tj 앞에 Ti 삽입 ; exit ; }
07 :    }
08 :    Q 내의 다음 우선순위 트랜잭션에 대해
                  위의 단계 반복 ;
09 : } // end of while
```

(알고리즘) PriorityAssignment(Ti, Q)

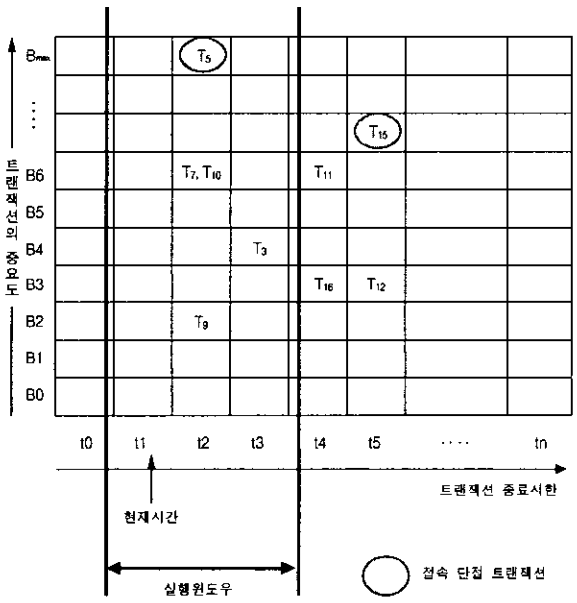
우선순위 할당 알고리즘은 실시간 이동 트랜잭션 T_i가 우선순위 큐에 들어왔을 때 우선순위를 결정하기 위한 알고리즘이다. 우선 T_i의 중요도와 우선순위 큐 Q에 있는 가장 높은 우선순위를 가지는 트랜잭션 T_j의 중요도를 비교한다. T_i의 중요도가 T_j의 중요도 보다 높아지면 T_i의 우선순위가 T_j의 우선순위 보다 높아야 하므로 T_i를 T_j 앞에 삽입한다(line 2-3). T_i의 중요도가 T_j의 중요도가 같으면 종료시한을 고려해야 한다. T_i의 종료시한 이 T_j의 종료시한 보다 빠르면 T_i의 우선순위가 높아야 하므로 T_i를 T_j 앞에 삽입한다(line 4-6). 그 이외의 경우는 T_j가 우선순위가 높은 것이므로 T_i의 우선순위를 찾기 위해 Q 내의 다음 우

선순위를 가진 트랜잭션을 가지고 앞의 단계를 반복 수행한다(line 1, 8).

3.4 스케줄러

스케줄러가 스케줄링할 트랜잭션을 선택하기 위해 본 논문에서는 실행 윈도우 기법을 사용한다. 실행 윈도우는 현재 시간, 트랜잭션의 수행 정보, 중요도, 그리고 종료시한을 고려했을 때 종료시한 내에 수행 가능한 트랜잭션의 집합을 의미하고, 스케줄러는 실행 윈도우 내에 있는 트랜잭션들을 대상으로 스케줄링을 한다.

(그림 4)는 현재 시간이 t1일 때, 실행 윈도우가 설정된 예를 보여 주고 있다. 실행 윈도우 안에 트랜잭션 T_3, T_5, T_7, T_9, T_{10}이 속해있고, 스케줄러는 중요도가 가장 높은 트랜잭션 T_5에 가장 높은 우선순위를 할당한다. 즉, 실행 윈도우 내에서 가장 가중치가 높은 트랜잭션을 우선적으로 처리는 기법으로 (그림 4)의 예에서는 $T_5 \rightarrow T_7 \cdot T_{10} \rightarrow T_3 \rightarrow T_9$ 순서로 우선순위를 할당한다. 이 기법을 사용하면 전체적인 트랜잭션의 중요도의 합을 최대화 할 수 있다. 스케줄링할 트랜잭션 집합을 구성하기 위한 스케줄러 알고리즘은 아래와 같다.



(그림 4) 우선순위 큐에서의 실행 윈도우

```
입력 : 우선순위 큐 Q,
출력 : 실행윈도우 EW
변수 : Q 내의 트랜잭션 Tj
01 : 실행윈도우 EW의 크기를 결정 ;
02 : while(Q 내의 모든 트랜잭션들에 대해) {
03 :     if (Tj의 종료시한 < EW의 크기)
04 :         Tj를 EW에 삽입 ;
05 : }
```

(알고리즘) SchedulingInExecutionWindow(Q)

실행윈도우를 이용한 스케줄러 알고리즘은 스케줄링할 트랜잭션을 우선순위 큐에 있는 모든 트랜잭션을 대상으로 하지 않고 일부분만을 선택하기 위한 알고리즘이다. Q에 실행윈도우를 설정한다(line 1). 실행윈도우 EW는 트랜잭션의 여러 정보를 고려했을 때 종료시한 내에 가능한 트랜잭션의 집합이므로, Q 내의 T_j의 종료시한과 EW의 크기를 비교하여 종료시한이 작은 경우에는 EW에 T_j를 포함시킨다(line 2-4).

3.5 디스패처

디스패처는 두 가지 역할을 담당한다. 첫째, 트랜잭션의 형태 즉, 실시간 트랜잭션과 비 실시간 트랜잭션에 따라 실시간 트랜잭션은 실행여부 검사기를 거쳐 우선순위 큐에 할당하고, 비 실시간 트랜잭션은 큐에 할당하는 역할을 담당한다. 둘째, 수행 중인 트랜잭션이 완료되었을 때, 다음에 수행할 트랜잭션을 스케줄러에서 선택하는 역할을 한다. 두 번째 역할을 수행할 때는 이동 호스트의 특성에 따른 빈번한 접속 단절을 고려해야 한다.

접속 단절이 발생한 트랜잭션의 처리는 이동 호스트가 연결될 때까지 기다리므로 트랜잭션의 지연 현상이 발생하여 동시성을 저하시킬 수 있다. 따라서 실행 윈도우 기반의 스케줄링 기법에서는 접속 단절이 발생한 트랜잭션의 경우 해당 스케줄링 대상에서 제외하고 수행 가능성이 높은 트랜잭션을 우선적으로 실행한다. (그림 4)에서 T_5와 T_{15}는 이동 호스트가 접속 단절인 상태이다. 따라서 스케줄러가 스케줄링한 $T_5 \rightarrow T_7 \cdot T_{10} \rightarrow T_3 \rightarrow T_9$에서 T_5가 가장 높은 중요도를 갖는 트랜잭션이지만 접속 단절이 발생하였으므로 디스패처는 T_5가 아닌 T_7이나 T_{10}을 선택하여 실행한다. 접속 단절을 고려하여 실행 트랜잭션을 선택하기 위한 디스패처 알고리즘은 아래와 같다.

```
입력 : 실행윈도우 EW,
출력 : 실행할 트랜잭션 Tj
변수 : EW 내의 트랜잭션 Tj
01 : 우선순위가 가장 높은 Tj 선택 ;
02 : while(EW 내의 모든 트랜잭션들에 대해) {
03 :     if (Tj가 접속단절 상태) {
04 :         다음 우선순위 트랜잭션 Tj 선택 ; continue ; }
05 :     else {
06 :         Tj 실행 ; exit ; }
07 : } //end of while
```

(알고리즘) Dispatcher(EW)

디스패처 알고리즘은 한 트랜잭션이 종료된 후 다음에 실행할 트랜잭션을 선택하는 알고리즘이다. 실행할 후보 트랜잭션은 스케줄러 알고리즘에서 얻은 EW내에서. EW에서 가장 우선순위가 높은 트랜잭션 T_i를 선택한다(line 1).

선택된 T_j가 접속단절인 경우에는 다음 우선순위의 트랜잭션을 선택하여 앞의 단계를 반복한다(line 2-4). 그러나, 선택된 T_j가 접속단절이 아닌 경우에는 T_j를 실행한다(line 6).

4. 성능 평가

이 장에서는 제안된 스케줄링 기법에 대한 성능을 평가한다. 성능 평가는 [4]에서 사용한 방법을 이용하여 기존의 FCFS 기법, EDF 기법, LSF 기법과 비교, 평가하였다.

4.1 성능 평가 모델

성능 평가를 위한 모델의 주요 구성 요소는 트랜잭션 생성기, 실시간 이동 트랜잭션 관리자, 실행 관리자, 실시간 데이터 관리자로 구성된다. 트랜잭션 생성기에서는 트랜잭션들을 무작위로 발생시키며, 트랜잭션의 스케줄링은 실시간 이동 트랜잭션 관리자에서 담당하며, 우선순위가 가장 높은 트랜잭션을 실행시키는 역할은 실행 관리자가 담당한다. 실시간 데이터 관리자는 트랜잭션의 스케줄링시 실시간 데이터에 관한 정보를 실시간 이동 트랜잭션 관리자에 전달하는 역할을 한다.

모의 실험을 간단히 하기 위해 트랜잭션이 생성될 때 트랜잭션의 종료시한, 중요도, 실행예상시간, 사용하는 실시간 데이터 등의 정보를 설정하였으며, 모의 실험을 위한 가정사항은 다음과 같다.

① 트랜잭션이 발생되었을 때 트랜잭션의 종료시한은 실행예상시간 보다 작을 수 없다. 트랜잭션이 실시간 이동 트랜잭션 관리자에 도착했을 때 종료시한이 실행예상시간 보다 작다면 이 트랜잭션은 언제나 종료시한을 위반하게 된다. 그러므로, 트랜잭션의 정보를 설정할 때 종료시한은 실행예상시간에 비해 크게 지정한다.

② 실제 실시간 이동 응용에서는 실시간 이동 트랜잭션과 비실시간 응용 트랜잭션이 동시에 발생하지만 비실시간 트랜잭션은 실시간 처리에 미치는 영향이 없으므로 본 실험에서는 배제한다.

③ 실행윈도우의 크기는 실시간 이동 응용의 특성에 따라 그 값이 다르게 결정되므로 본 실험에서는 임의로 트랜잭션의 평균수행시간의 배수로 결정한다.

④ 트랜잭션의 중요도는 1부터 10 사이의 값으로 지정했는데, 중요도 등급별 트랜잭션의 발생 비율은 동등하게 하였다.

⑤ 성능 평가를 위해 비교되는 스케줄러들은 종료시한, 중요도, 실시간 데이터의 유효성, 트랜잭션들의 접속 단절을 고려하였다. 다음 <표 1>은 성능 비교를 하기 위해 스케줄러들에서 고려한 사항이다. FCFS 기법은 트랜잭션이 도착한 순서대로 스케줄링 하는 기법이므로 네 가지 고려 사항 모두를 반영하지 않았으며, EDF 기법은 종료시한 정보만을 반영했으며, LSF 기법은 종료시한을 기반으로 슬랙 시간을 계산하여 트랜잭션들을 스케줄링 하였다.

<표 1> 스케줄링시 고려 사항

	종료시한	중요도	실시간 데이터의 유효성	접속 단절
FCFS	×	×	×	×
EDF	○	×	×	×
LSF	○	×	×	×
제 안 기 법	○	○	○	○

4.2 성능 평가 환경

평가 시스템은 고정망의 서버로서 Solaris 5.5.1 운영체제를 사용하는 유닉스 머신을 사용하였고, 이동 호스트를 위한 부분으로써는 Windows NT 4.0 운영체제 하에서 수행하였다. 성능 평가를 위한 시스템의 환경은 <표 2>와 같다.

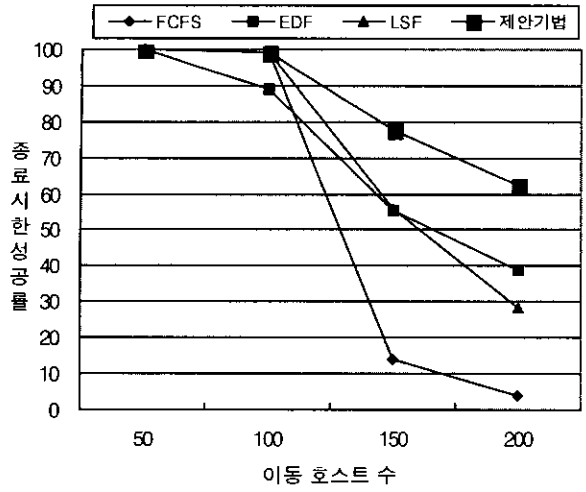
<표 2> 성능 평가 시스템 사양

항 목	P C	Unix
기 종	IBM PC	SUN Workstation
C P U	Pentium II 400Mhz	UltraSPARC 167 Mhz
Main Memory 크기	128 MB	128 MB
Hard Disk 용량	8 GB	10 GB
운 영 체 제	Windows NT 4.0	Solaris 5.5.1
개 발 언 어	Visual C++ 6.0	Sun Workshop Pro C++

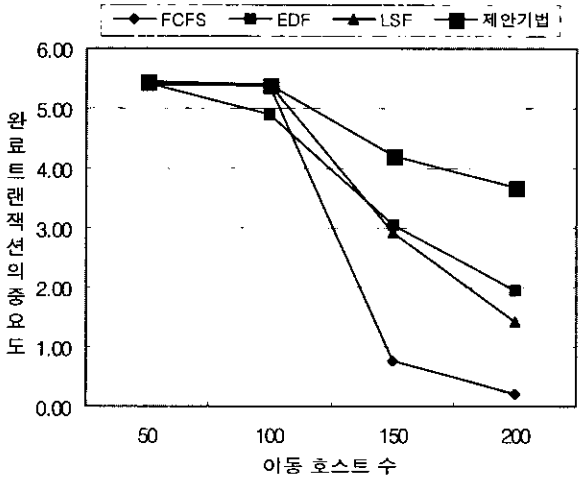
성능평가를 위해서 사용된 인자의 기본 설정 값으로 이동 호스트를 지원하기 위한 이동 지원국인 고정 호스트의 개수는 10개로 설정하였다. 1페이지의 크기는 4096바이트이고, 전체 데이터베이스의 크기는 10000 페이지로 하였으며, 이동 호스트가 가지는 지역 데이터베이스의 크기는 200 페이지로 하였다. 이동 호스트와 고정망과의 데이터 전송속도는 2 Mb.ps로 설정하였고, 고정망에서 고정 호스트간의 데이터 전송속도는 100 Mbps로 설정하였다.

4.3 성능 평가 및 분석

성능 평가의 결과로는 종료시한을 만족하는 즉, 완료된 트랜잭션 개수의 비율과 완료 트랜잭션의 중요도를 기준으로 평가한다. 만약 종료시한의 성공률이 높으면 수행된 전체 트랜잭션 중에서 트랜잭션이 지니는 종료시한을 만족시킨 트랜잭션의 비율이 높은 것을 의미하며, 완료 트랜잭션의 중요도의 평균이 높다는 것은 발생되는 트랜잭션 중에서 중요도가 높은 트랜잭션들이 우선적으로 처리되었음을 의미한다.

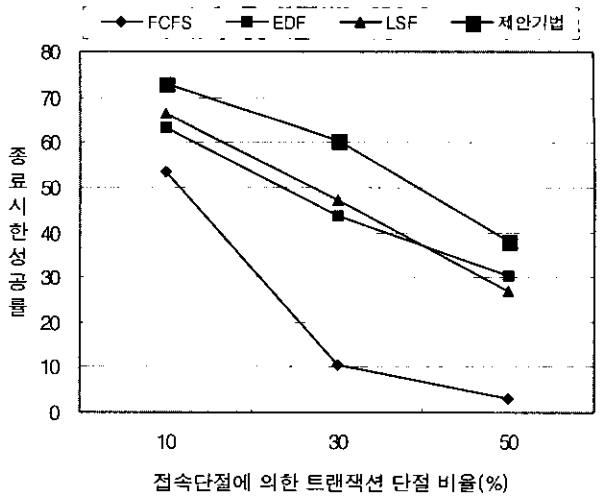


(그림 5) 이동 호스트 수에 따른 스케줄링 기법의 종료시한 성공률

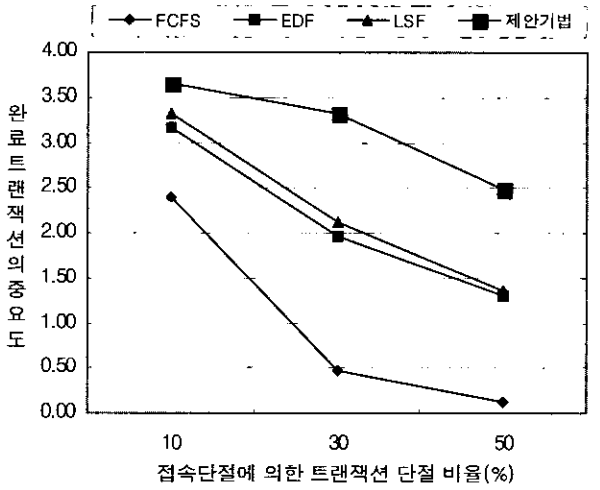


(그림 6) 이동 호스트 수에 따른 완료 트랜잭션의 중요도

(그림 5)와 (그림 6)는 이동 호스트의 수를 50개에서 200개까지 50개씩 증가시켰을 때의 기존의 기법과 제안된 기법의 종료시한 성공률과 중요도를 비교 평가한 그래프이다. 이동 호스트의 수가 증가함에 따라 처리를 요구하는 트랜잭션의 수가 증가되어 전체적으로 종료시한 내에 처리되는 트랜잭션 수가 감소하게 된다. 특히, FCFS 기법은 거의 모든 트랜잭션이 종료시한을 초과한다. 제안된 기법은 트랜잭션을 수행하기 전에 수행여부 검사기에서 데드라인을 초과하는 트랜잭션을 제외하였고, 또한 사용되는 실시간 데이터의 유효성을 검사하여 수행 가능한 트랜잭션들만 스케줄링하였으므로 기존의 기법에 비해 급격하게 성공률이 저하되지 않고 완만한 하강 곡선을 그리고 있으며, 타 기법에 비해 성능이 20%이상 높게 나타난다. 또한, 제안 기법이 트랜잭션의 우선순위를 결정할 때 중요도를 우선 고려함으로 인해 완료 트랜잭션의 중요도도 종료시한 성공률에 비례해 높게 나타난다.



(그림 7) 접속단절에 따른 스케줄링 기법의 종료시한 성공률



(그림 8) 접속단절에 따른 완료 트랜잭션의 중요도

(그림 7)과 (그림 8)은 이동 호스트와 이동 지원국 간의 접속단절의 비율을 10%에서 50%까지 20%씩 증가시킴에 따른 트랜잭션의 종료시한 성공률과 완료 트랜잭션의 중요도를 구한 것이다. 접속 단절이 증가함에 따라 전체적으로 급격한 성공률의 저하가 발생한다. 제안 기법에서는 이동 호스트의 접속 단절의 경우에 트랜잭션의 우선순위를 조절함으로 인해 타 기법이 접속 단절로 인한 트랜잭션의 지연현상이 발생할 때 제안 기법은 지연하지 않고 다음 우선순위의 트랜잭션을 수행함으로 기존의 기법에 비해 20%이상의 높은 성공률을 보인다.

5. 결 론

본 논문에서는 사용자가 자유롭게 이동하는 이동 컴퓨팅 환경에서 발생되는 트랜잭션이 시간적인 제약조건을 지니는 실시간 이동 트랜잭션을 지원하기 위한 트랜잭션의 처리에 관한 연구를 수행하였다. 이동 컴퓨팅 환경에서 이동

호스트의 이동성에 의한 접속 단절과 핸드오버 등에 의해 트랜잭션의 실행에 대한 예측성이 저하됨으로 인해 실시간 특성을 지원하는 것이 매우 어렵다. 또한, 기존의 시간적인 제약조건을 갖는 실시간 트랜잭션 처리를 위한 스케줄링 기법, 동시성 제어 기법 등이 고정망을 기반으로 개발된 것으로 이를 무선통신을 사용하는 이동 컴퓨팅 환경에 적용하는 것은 부적합하다. 따라서 본 논문에서는 이동 컴퓨팅 환경에서 발생하는 실시간 이동 트랜잭션을 효율적으로 처리하기 위한 실시간 이동 트랜잭션 관리자를 위한 트랜잭션 스케줄링 기법을 제안하였다.

제안된 실행 윈도우를 이용한 트랜잭션 스케줄링 기법은 트랜잭션의 종료시한과 실시간 데이터의 유효성을 고려하여 실행여부를 판단하여 예측성을 향상시켰으며, 트랜잭션의 중요도와 종료시한을 기반으로 한 우선순위 큐에 트랜잭션을 배치시켜 우선순위를 할당했다. 또한, 스케줄링을 위해서 우선순위 큐에 최적의 실행 윈도우를 생성하였고, 접속 단절을 고려하였다. 따라서 제안된 기법은 기존에 고려하지 않았던 실시간 데이터의 유효성 제약조건 위반에 따른 트랜잭션의 재시작을 방지할 수 있었으며, 종료시한을 만족하는 트랜잭션의 중요도의 합을 최대로 한다. 또한, 본 논문에서 제안한 실시간 이동 트랜잭션의 처리 기법은 트랜잭션의 시간적인 제약 조건을 최대한 만족시키며, 접속 단절 등으로 인한 트랜잭션 지연이 빈번하게 발생하는 이동 컴퓨팅 환경에서도 기존의 기법에 비해 효율적으로 트랜잭션을 처리함을 성능평가를 통해서 20% 정도 향상됨을 보였다.

향후 연구 방향으로는 본 연구에서 제안한 실행 윈도우의 적정 크기를 구하는 방법에 대한 연구와 실시간 이동 트랜잭션들의 스케줄링에 영향을 미치는 동시성 제어 기법에 대한 연구가 진행되어야 한다.

참 고 문 헌

[1] Rafael Alonso and Henry F. Korth, "Database system issues in nomadic computing," In Proceedings of the ACM International Conference on Management of Data, pp.388-392, 1993.

[2] Tomasz Imielinski and B. R. Badrinath, "Wireless mobile computing : Solutions and challenges in data management," Technical Report DCS-TR-296, Department of Computer Science, Rutgers University, New Brunswick, NJ 08903, 1993.

[3] M. H. Dunham and A. Helal, "Mobile computing and databases : Anything new?," ACM SIGMOD Record, Vol.24, No.4, pp.5-9, 1995.

[4] O. Ulusoy, "Real-Time transaction management for mobile computing" IADT '98, pp.223-240, 1998.

[5] I. Ahn, "Database issues in telecommunications network management," ACM SIGMOD International Conference on Management of Data, pp.37-43, 1994.

[6] E. Pitoura and B. Bhargava, "Maintaining consistency of data in mobile distributed environments," Proceedings of 15th International Conference on Distributed Computing Systems, 1995.

[7] C. Liu and J. Laylabd, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," Journal of the ACM, 1973.

[8] 김경배, 배해영, "분류된 클래스 큐를 이용한 실시간 데이터베이스 시스템의 트랜잭션 관리기", 한국정보처리학회논문지, Vol.5, No.11, 1998.

[9] G. D. Walborn and P. K. Chrysanthis, "Supporting semantics-based transaction processing in mobile database applications," In Proceedings of the 14th IEEE Symposium on Reliable Distributed Systems, 1995.

[10] J. R. Haritsa, M. Livny and M. J. Carey, "Earliest Deadline Scheduling for Real-Time Database Systems," Proceedings 12th Real-Time System Symposium, IEEE, pp.232-242, 1991.

[11] R. Abbott and H. G arcia-Molina, "Scheduling I/O Requests with Deadlines : a Performance Evaluation," Proceedings of 11th Real-Time Systems Symposium, IEEE, pp.113-124, 1990.

[12] P. K. Chrysanthis,"Transaction processing in mobile computing environment," In IEEE Workshop on Advances in Parallel and Distributed Systems, pp.77-82, 1993.

[13] M. H. Dunham, A. Helal, and S. Balakrishnan, "A mobile transaction model that captures both the data and movement behavior," ACM/Baltzer Journal on Special Topics in Mobile Networks and Applications, Vol.2, pp.149-162, 1997.

[14] R. Abbottand H. Garcia-Molina, "Scheduling Real-time Transactions," SIGMOD RECORD, ACM, Vol.17, No.1, March 1988.

조 숙 경

e-mail : skyoe@netsgo.com

1990년 인하대학교 전자계산학과(이학사)

1994년 인하대학교 대학원 전자계산공학과(공학석사)

1997년~현재 인하대학교 대학원 전자계산공학과 박사과정

관심분야 : 데이터베이스, 실시간 데이터베이스 시스템, 이동 데이터베이스 시스템

김 경 배

e-mail : gbkim@etri.re.kr

1992년 인하대학교 전자계산학과(공학사)

1994년 인하대학교 대학원 전자계산공학과(공학석사)

2000년 인하대학교 대학원 전자계산공학과(공학박사)

2000년~현재 한국전자통신연구원, 선임연구원

관심분야 : 자료저장시스템, SAN, 이동 컴퓨팅, 실시간 데이터베이스 시스템

이 순 조

e-mail : sjlee@seowon.ac.kr

1985년 인하대학교 전자계산학과(이학사)

1987년 인하대학교 대학원 전자계산공학과 (이학석사)

1995년 인하대학교 대학원 전자계산 공학과 (공학박사)

1995년~1997년 대림대 전자계산과 교수

1997년~현재 서원대 컴퓨터교육학과 조교수

관심분야 : 데이터베이스, 실시간 데이터베이스 시스템, GIS, 이동 데이터베이스 시스템, 데이터베이스 시스템의 보안

배 해 영

e-mail : hybae@dragon.inha.ac.kr

1974년 인하대학교 응용물리학과(공학사)

1978년 연세대학교 대학원 전자계산학과 (공학석사)

1989년 숭실대학교 대학원 전자계산학과 (공학박사)

1992년~1994년 인하대학교 전자계산소 소장

1982년~현재 인하대학교 전자계산공학과 교수

1999년~현재 정보통신부 국가 GIS기술 개발 분과 자문위원

1999년~현재 지능형 GIS 연구 센터 소장

1999년~현재 연변과학기술 대학 겸직 교수

2000년~현재 중경우전대학 대학원 명예 교수

관심분야 : 데이터베이스, GIS, 실시간 데이터베이스 시스템, 이동 데이터베이스 시스템