# 이동 컴퓨팅 환경에서 비연결 수행을 위한 실시간 알고리즘*

한국희°, 김재훈, 고영배
아주대학교 정보통신전문대학원 정보통신공학부
{justuniq°, jaikim, youngko}@ajou.ac.kr

# Real-Time Scheduling Algorithm for Disconnected Operation

Kook-Hee Han°, Jai-Hoon Kim, Young-Bae Ko
Graduate School of Information and Communication, Ajou University

**요 약**

이동 컴퓨팅 환경에서 통신망 제약을 극복하기 위한 방법으로 비연결 수행을 통한 컴퓨팅이 사용된다. 이동 컴퓨팅 환경은 유비쿼터스 환경으로 발전함에 따라 처리 할 정보의 요구량과 통신망 사용률이 증가 하게 된다. 그러나 노드의 환경 변화에 따라 빈번한 통신망 단절은 시스템의 성능을 저하시킨다. 본 논문에서는 이동 단말 노드에서 여러 개의 태스크가 통신망 자원을 사용하여 수행될 때 통신망 단절에 대비하기 위한 실시간 비연결 수행 스케줄링기법의 성능을 비교하였다. 실시간 알고리즘에 따른 성능 변화를 분석하며, 각 영향을 주는 요소를 파악하여 가장 효율적인 실시간 알고리즘을 적용할 수 있도록 한다.

## 1. 서 론

컴퓨팅 환경의 발달과 무선기술이 나날이 발전함에 따라, 점차 이동컴퓨팅이 컴퓨팅 환경에서 많은 비중을 차지하게 되었다. 또한 이러한 이동컴퓨팅의 발달로 인해 이동컴퓨팅 환경에서의 하드웨어의 성능향상보다 빠르게 처리할 정보의 요구량이 더욱 빠르게 증가하고 있다. 그러나 이런 정보의 처리량과 함께 증대되는 사용자의 만족도에 반하여 이동 컴퓨팅은 아직도 통신망 불안정과 같은 해결해야 할 많은 제약을 갖고 있다.

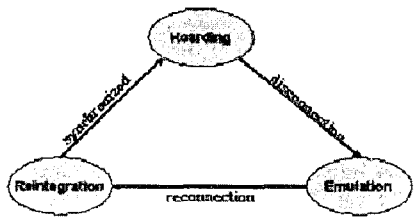
무선 네트워크로 연결된 이동 컴퓨팅은 환경의 특성상 잦은 끊김과 높은 에러율 발생하며, 이로 인한 가용 할 통신 용량의 부족현상과, 이동에 따른 장소의 통신 품질의 변화가 발생한다. 또한 이동 컴퓨팅 시스템은 제한된 전력으로 인해 태스크 수행 시 에너지 효율을 고려가 필요하다. 이러한 이동컴퓨팅환경에서의 통신망 제한에 의한 태스크 수행 기법을 비연결 수행이라 한다. 비연결 수행은 시스템의 제한된 전력과 같은 자원을 보존하기 위해 수행되는 의도적인 비연결 수행과 통신 품질의 변화나 통신의 단절로 인하여 통신망 사용을 할 수 없게 되는 상황에서 수행되는 비의도적인 비연결 수행으로 나눌 수 있다.

이동 컴퓨팅 환경에서의 태스크가 증가하고, 각 태스크의 통신 자원 사용률이 증가하게 됨에 따라, 통신만 단절로 인한 에러율은 증가하게 된다. 실시간 응용에 비연결 수행을 위한 실시간 스케줄링 알고리즘을 적용하여 노드의 수행 성능을 비교 하였다. 성능비교 결과를 통하여 비연결 수행에 적절한 실시간 스케줄링 알고리즘을 선택 사용 할 수 있다.

## 2. 관련 연구

### 2.1 비연결 수행

비연결 상태에서 태스크의 수행을 효과적으로 지원하여 무선 이동 환경에서 응용 어플리케이션의 효율적인 실행을 위해서 많은 기법들이 제안되었다[1]. 이 기법은 대부분 파일 시스템에서 비연결 수행을 지원하기 위해서 사용 할 자원을 미리 예측하여 캐싱 기법을 통해 태스크를 수행하는 기법에 대한 알고리즘이다. 통신 네트워크 리소스를 사용 가능 시간에 미리 자원을 호스트에 저장하고, 통신망이 단절 되었을 때, 이 캐싱 된 자원을 사용하여, 사용자는 통신망의 단절의 영향을 받지 않도록 한다. 통신망이 재연결되어 통신 자원이 사용 가능할 때, 로컬에 변화된 자원과 네트워크를 통한 서버의 데이터를 일치시키게 된다.




**<그림 1> 비연결 수행의 상태 변화**

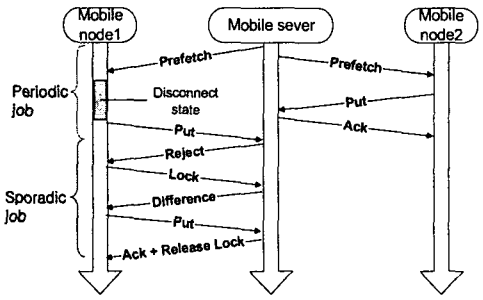
<그림 1>과 같이 비연결 수행은 3가지의 상태로 구분 할 수 있다. 세가지 상태는 다음과 같다.

- Hoarding: Preloading, Prefetch라고도 하며, 통신망에 연결된 상태로, 이 상태에서 현재 사용할 데이터를 통신 네트워크의 서버로부터 사용 가능하며, 통신망을 사용 할 수 없게 될 때를 위해서 사용할 데이터를 예측하여 캐싱한다.
- Emulation: 통신망의 사용이 불가능한 상태로 Hoarding상태에서 캐싱한 데이터를 사용하여 태스크를 지속적으로 수행하는 상태이다. 이 상태에서의 데이터의 수정은 실제 서버의 데이터와 다르게 되며, 이 차이에 대해 일치성 작업이 필요하게 된다.
- Reintegration Emulation상태에서 다시 통신망 사용이 가능하게 된 경우의 상태를 말하며, 이 상태에서는 Emulation상태에서 변화된 데이터를 서버의 실제 데이터와 일치성 작업

* 본 연구는 정보통신연구진흥원의 국제공동연구 지원사업, 한국과학재단의 지원사업(R05-2003-000-10607-0, R012003-00-0794-0), 한국학술진흥재단의 지원사업(KRF-2003-003-D00375)의 지원을 받음.

을 수행하게 된다. 이 일치성 작업의 수행이 끝나면 다시 Hoarding상태가 되어 데이터를 캐쉬 한다.

Hoarding방식에서의 데이터 캐쉬 방법은 두 가지로 나눌 수 있는데, 사용자의 예측이나 정해진 스케줄에 따르는 방법[2]과 시스템에서 자동적으로 이전까지의 사용자의 사용 패턴을 기억하고 분석하여 그에 따라 적응하는 방법[3]이 있다. 또한 Reintegration은 Emulation 상태에서 변경된 데이터의 일치성 작업을 수행하는 것으로 이 두 가지의 수행의 흐름은 <그림 2>와 같이 나타낼 수 있다.




**<그림 2> 비연결 수행의작업 흐름**

위 <그림 2>에서와 같이 비연결 수행은 상태 변화에 따라 작업의 흐름이 나타나게 되는데, 일반적으로 모바일 노드는 <그림 2>의 Mobile node2와 같이 Hoarding상태에서 Prefetch를 하고 통신망 사용이 가능해지면 Reintegration상태로 변화된 데이터에 대한 Put을하고 서버로부터 Ack를 받아 일치성 작업을 종료하고 다시 Hoarding상태로 변하게 된다. 그러나 하나의 Mobile server로부터 데이터를 사용하게 되는 Mobile node가 증가 하게 됨에 따라서 데이터의 쓰기 충돌로 인하여 일치성 작업의 수행은 지연되게 된다. 여러 개의 태스크가 진행될 경우 각 태스크의 일치성 작업이 지연됨에 따라 통신네트워크 자원을 사용 가능한 시간이 줄어들게 되며, 이에 따라서 대기 중인 다른 태스크의 수행에 영향을 미치게 된다. 그러므로 실시간 스케줄을 통해 각 실시간 태스크의 마감시간 준수율(Deadline Meet Ratio)을 최대로 할 수 있는 방법이 요구된다. Hoarding 상태에서의 Prefetch는 각 태스크에 따라 필요한 적정 캐쉬량과 사용량, 그리고 캐쉬를 통한 Emulation 가능시간을 통해서 적절한 Prefetch를 하게 된다. 그러므로 해당 태스크는 Prefetch를 통해 할당될 통신네트워크 자원을 예상 할 수 있다. 그러나 Reintegration상태에서의 일치성 작업 수행에 통신 자원이 태스크 별로 얼마나 할당해야 할지 예상 할 수 없다. 데이터의 변화가 없다면 한번의 Put과 Ack로 일치성 작업 수행이 끝나게 되지만, 데이터의 변화가 심하고, 이 데이터를 사용하는 Mobile node의 수가 증가 할수록 일치성 작업의 지연은 증가하게 된다. <그림 2>의 Mobile node1은 put하는 과정에서 한번의 데이터 변경으로 인하여 다시 변경된 정보를 받아 Put하는 흐름을 보여주게 되는데, 서버로부터 Reject되는 횟수와 Reject에 대한 대기 시간은 데이터 변화율과 Mobile node의 수에 따라 증가하게 된다. 노드 및 네트워크 자원을 절약하기 위해서 의도적인 비연결 수행이 필요한데, 이때 일정주기란 의도적 비연결 수행을 가정할 때 이를 주기적 작업으로 가정 할 수 있다. 반면 Reintegration시 Reject되는 경우 이를 해결하기 위한 일치성 작업을 비주기적인 작업이라 할 수 있다. 이러한 주기적인 작업과 비주기적인 작업을 하게 되는 각 태스크에 대하여 가장 효율적으로 스케줄 할 수 있는 비연결 수행을 위한 실시간 스케줄 알고리즘의 성능을 비교 한다.

### 2.2 실시간 알고리즘

실시간 시스템은 주기적이거나 비주기적인 이벤트에 대해 일정시간 내에 완료해야 하는 시간 제약성을 갖는 시스템으로, 실시간 알고리즘은 여러 개의 태스크가 하나의 프로세스 자원에 의해서 처리 될 때 시간 제약성을 지키며 각 태스크를 효율적으로 스케줄 하는 것을 목적으로 한다. 주기적으로 발생하는 태스크를 정해진 시간 내에 성공적으로 처리하기 위한 대표적인 실시간 스케줄링 알고리즘은 Fixed Priority 위한 스케줄 방법과 Dynamic Priority에 위한 방법으로 나눌 수 있다. 본 논문에서는 Fixed Priority의 방법으로 Rate Monotonic과 Dynamic Priority의 방법으로 Earliest Deadline First와 Least Slack Time First 방법을 사용하여, 비연결 수행에 적합한 실시간 스케줄링 방식에 대해서 알아본다.

- Rate Monotonic(RM)- 고정적인 우선순위에 위한 방법으로 각 태스크 중에서 주기가 짧은 태스크를 우선적으로 수행하게 된다. 그러므로 주기가 짧은 태스크는 항상 우선적으로 실행되게 된다.
- Earliest Deadline First(EDF)- 동적으로 우선순위가 할당되는 스케줄 방식에서 가장 많이 쓰이는 방법으로 수행될 태스크는 마감시간이 가장 짧은 태스크가 먼저 수행되는 방법이다.
- Least Slack Time First(LST)- 동적 우선 순위 할당 방식으로 남는 여유의 slack time이 가장 적은 태스크가 먼저 수행되게 되는 방법이다.

위 3가지 스케줄 방식은 주기적인 태스크에 대한 스케줄 방식으로 비연결 수행에의 주기적인 작업에 대해서 스케줄이 가능하다. 그러나 비주기적인 작업에 대한 스케줄링은 고려되지 않았기 때문에, 이동 단말노드의 특성에 따라 각 실시간 알고리즘의 효율은 달라지게 된다.

## 3. 제안 기법

비연결 수행에서 이동 단말기에 여러 개의 태스크가 수행되는 것은 시스템의 제약 및 통신망의 제약에 의해서 높은 실패율을 일으키게 되는데, 이를 실시간 스케줄 기법을 통해서 실패율을 줄이고, 효율적으로 사용자의 만족도를 증가 시키도록 한다.

<그림 2>과 같이 비연결 수행의 작업의 흐름은 실행시간을 예상 할 수 있는 요소와 예상 할 수 없는 요소로 나눌 수 있다. 먼저 예상 할 수 있는 작업의 실행시간은 Hoarding상태에서의 Prefetch와 Reintegration상태에서의 Put과 Ack이다. 이것은 데이터의 변화율과는 무관하게 일정한 작업시간이 소요되며, 이것은 미리 예상하여 할당하는 것이 가능하다. 그러나 Reintegration작업 중 Put에 대한 Reject의 발생 빈도는 예상이 불가능하다. 이 빈도는 Mobile Server의 data를 Reintegration의 Lock작업을 통해서 변경하려는 node가 많을 수록, Put의 실패에 의한 Reject 빈도는 증가하고, 한 node의 Lock을 통해서 Reintegration 하는 동안 다시 데이터를 변경시키려 하는 다른 노드는 대기 하게 된다. 그러므로, Reintegration작업은 지연되어 소비 시간은 증가하게 된다. 그러므로 실시간 알고리즘을 적용시 예상할 수 있는 할당 시간의 요소를 Periodic Job이라 하고, 다른 Mobile node에 의한 한번의 Lock에 의한 지연된 할당 시간을 하나의 Sporadic Job이라 할 수 있다. Emulation상태에서는 캐쉬한 데이터를 소모하게 되는데, Hoarding상태에서 캐쉬한 데이터를 전부 소모 할 때까지 통신망의 연결이 일어나지 않으면 Emulation상태에서 태스크의 수행은 Block되게 된다. 태스크가 Block상태를 들어가게 되는 것을 실시간 작업에서 마감시간 실패로 본다.

각 태스크에 대해서 위와 같이 Periodic Job과 Sporadic Job으로 구분하여, 주기적인 태스크에 대한 실시간 스케줄 알고리즘을 통해서 Periodic Job을 스케줄 하여 성능을 평가하였다. 이를 이용하면 Sporadic Job의 발생에 대응하여 마감시간 실패율이 가장 적게 발생하며, 효율적으로 대처 할 수 있는 스케줄

방법을 비연결 수행에서 선택 할 수 있다.

## 4. 시뮬레이션 및 성능 분석

비연결 수행에 대한 스케줄 방법으로 실시간 스케줄 방법인 RM, EDF, LST를 각각 적용 시켰을 때의 마감시간 실패율을 알아보고 비연결 수행에 가장 적합한 실시간 스케줄 방법을 결정한다. 시뮬레이션은 비연결 수행에 대해 각각의 알고리즘을 적용한 스케줄러를 만들고, 시뮬레이션을 통해서 각 스케줄러의 실패율을 알아 본다.

각 태스크는 주기와 실행 시간을 갖는데, 주기는 한번 Hoarding작업을 시작할 때부터 다음 Hoarding 작업을 할 때 까지를 나타내며, 실행시간은 Periodic Job이 수행되는 시간을 나타낸다.

각 태스크는 요소들을 <그림 2>과 같이 표시 할 수 있다. 비연결 수행의 각 요소를 실시간 스케줄 알고리즘 요소로서 정의하고, 이 실시간 스케줄 요소로서 스케줄 한다.

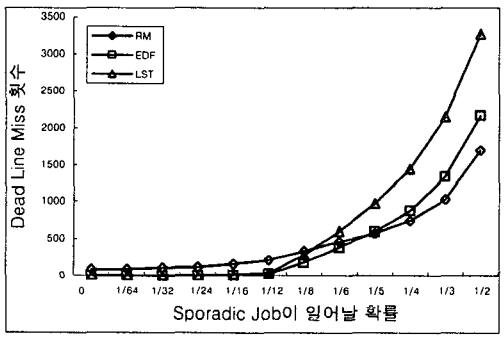
- Periodic Job- Hoarding 상태에서의 Prefetch를 통한 캐싱, Reintegration상태에서의 Put, 그리고 Ack 등 예상 할 수 있는 프로세스 수행 작업을 말한다.
- Sporadic Job- Reintegration 작업 중 Put이 Reject되는 작업부터 일치성 작업이 완료되어서 서버로부터 Ack를 받는 작업까지의 흐름을 말한다.

위와 같이 정의 된 각 태스크는 주기적인 작업의 주기와 실행시간을 갖으며, 각 태스크는 아래 <수식 1>과 같이 나타낼 수 있다.

$$T_i = \{ p_i, e_i \} \cdots \quad \text{<수식 1>}$$

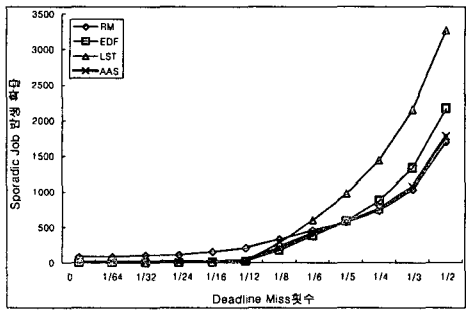
$T_i$=i 번째 태스크, $p_i$=주기, $e_i$=실행시간

시뮬레이션은 우선순위에 의한 스케줄 방식으로 RM, EDF, LST의 성능을 비교 하였으며, 태스크의 Hyper-Period는 120이며, Sporadic Job에 의한 평균적인 영향을 알아보기 위하여 12000시간동안 테스트를 하였다.




**<그림 3>실시간 알고리즘 별 실패 횟수**

<그림 3>은 T1={6, 1}, T2={9, 2}, T3={12, 3}, T4={16, 4}의 4개의 태스크에 대해서 Sporadic Job이 발생할 때 한번의 Sporadic Job의 실행시간을 1 소모하고, 모든 태스크는 같은 Sporadic Job의 발생 확률을 갖는다고 가정하고 실험한 그래프이다. 위 그림에서 보면 확률이 1/6이하에서는 EDF가 가장 좋게 나타나며, 확률이 1/6보다 큰 경우는 RM이 가장 좋은 성능을 나타낸다. RM의 경우 Sporadic Job의 확률이 작을 때부터 가장 큰 마감시간 실패 횟수를 보이지만, 확률이 클 때는 가장 좋은 성능을 보이며, EDF의 경우 확률이 낮을 때는 가장 좋은 성능 보이지만 확률이 높을 때는 중간의 성능을 보인다.




**<그림 4> 실시간 알고리즘별 실패 횟수**

시뮬레이션 결과로 보면 Sporadic Job이 발생할 확률의 변화에 따라 가장 적합한 알고리즘은 다르게 된다. 이동 컴퓨팅 환경의 특성상 통신망의 상태 변화가 심한 것과 같이 Sporadic Job의 발생 확률도 일정하지 않다. 그러나 사용자는 지속적인 데이터의 사용을 요구하게 되며, 이런 환경에서 고정적으로 하나의 실시간 알고리즘을 채택하여 사용하는 것은 전체 시스템 성능에 좋지 못하다. 그러므로, 비연결 수행에서의 실시간 알고리즘은 환경에 적응적으로 선택하여 사용할 수 있도록 하였다.

<그림 4>은 적응적인 알고리즘 선택 기법(Adaptive Algorithm Selection(AAS))을 통한 실패 횟수와 기존 실시간 알고리즘을 비교한 것이다. AAS 방식은 태스크의 Hyper-Period 시간을 하나의 주기로 보고 두 번의 주기 동안 RM과 EDF를 한 번씩 수행한 후 실패율 비교를 통해 세 번째 주기부터는 실패율이 적은 방식으로 스케줄을 하게 되는 방법이다. 이 방식을 사용하면, 대부분의 경우 가장 좋은 알고리즘과 동일한 성능을 보이게 된다. 또한 이 방식은 통신망 환경이 일정할 때는 물론이고 변화하는 통신망 환경에 서도 통신 환경 변화를 감지하여, 환경에 변화에 적응하여 적절한 알고리즘을 선택하게 된다.

## 5. 결 론

비연결 수행에서 여러 실시간 태스크가 수행될 때 가장 적합한 알고리즘은 Sporadic Job의 빈도에 따라 달라진다. 본 논문에서는 이들 알고리즘의 성능을 비교 하였다. 또한 시스템 환경에 가장 적합한 알고리즘을 자동으로 선택할 수 있도록 하였다. 제안된 AAS알고리즘은 환경의 변화에 대해서 발생하는 Sporadic job에 적응하여 RM방식과EDF방식을 비교 선택하여 태스크의 전체 성능과 사용자의 만족도를 높일 수 있다.

## 참고 문헌


[1] T. Ye, H. Jacobsen and R. Katz, "Mobile awareness in a Wide area wireless network of info-stations," *In Proceedings of ACM International Conference on Mobile Computing and Networking*, October 1998.

[2] E. Pitoura and G. Samaras, "Data Management for Mobile Computing," *Kluwer Academic Publishers*, pp. 37-48, 1997.

[3] J. J. Kistler and M. Satyanarayanan, "Disconnected Operations in the Coda File System," *In Proceedings of the ACM Transactions on Computer Systems*, 10(1), pp. 213-225, February 1992.

[4] 가진호, 김재훈, "실시간 시스템을 위한 실시간 메모리 교체 기법," 한국정보과학회 가을 학술 발표 논문집, 2001.

[5] 정승식, 김재훈, "통신망 단절 시 프로세스의 지속적 수행을 위한 모델 및 성능 분석," 한국정보과학회 추계학술발표논문집, 2000.10.