셀룰라 네트워크 환경에서의 이중화 체크포인팅을 이용한 이동 호스트 및 기지국 결함 복구 기법

정회원 변 계 섭*, 김 재 훈*

Replicated Checkpointing Failure Recovery Schemes for Mobile Hosts and Mobile Support Station in Cellular Networks

Kyue-Sup Byun*, Jai-Hoon Kim* ***Regular Members***

요 약

이동 호스트는 무선 통신망의 낮은 대역폭과 호스트들의 이동성, 부족한 저장장치와 배터리 수명 등으로 인하여 결함 발생 가능성이 높다. 이동 호스트의 결함에 효율적으로 대처하기 위한 결함 허용 기법에 관한 연구가 많이 진행되어 왔다. 셀룰라 네트워크에서는 이동 호스트이외에도 이동 호스트를 연결시키는 기지국은 보다 높은 수준의 가용도를 요구하므로 기지국의 결함에 대한 연구도 필수적이다. 본 논문에서는 이동 호스트 결함 복구를 위한 체크포인팅 기법을 기반으로 기지국 결함 복구를 위한 체크포인팅 이중화 기법을 제안하고 성능을 분석하였다. 또한 이동 호스트의 결함 복구를 위해 체크포인트가 존재하는 기지국의 복구를 기다리는 방법과 다른 기지국의 체크포인트를 이용하는 방법의 성능을 비교 분석하였다.

ABSTRACT

A mobile host is prone to failure due to lack of stable storage, low bandwidth of wireless channel, high mobility, and limited battery life on the wireless network. Many researchers have studied to overcome these problems. For high level Availability in the cellular networks, it is necessary to consider recovery from the failures of mobile support stations as well as mobile as mobile hosts.

In this paper, we present modified trickle scheme for recovery from failures of Mobile Support Station based on checkpointing scheme and analyze and compare the performance. We propose and analyze the performance of two schemes : one is waiting recovery scheme for the mobile support station having the last checkpoint and the other is searching the new path to the another mobile support station having the checkpoint.

Ⅰ. 서 론

이동 호스트는 여러 자원의 제약성(즉, 낮은 대역폭과 호스트들의 이동성, 작은 저장장치, 충분하지 못한 배터리 수명)으로 인하여 결함의 발생 가능성이 높다. 이동 호스트 자체의 결함 이외에 네트워크 연결 단절, 무선 링크의 결함 등 기존 유선 네트워크에서의 결함 원인들과는 다른 여러 결함을 포함하고 있다[2]. 그러므로 이동 호스트의 결함 복구에 대한 많은 연구가 진행되어왔다. 이동 호스트의 이동성으로 인해 기존 유선에서의 체크포인팅 기법[8,9]은 이동 컴퓨팅 환경에 적합하도록 변경하여 제안되었다[3,5,6]. 이들 이동 호스트의 결함에 대한 복구 기법은 이동 호스트 자체의 제한적인 자원(작은 저장장치 및 안전하지 않은 저장장치, 낮은 대역폭과 호스트들의 이동성)으로 인해 기지국에 자신들의 정

* 아주대학교 정보통신전문대학원({ksbyun,jaikim}@madang.ajou.ac.kr)
논문번호 : 010246-0911, 접수일자 : 2001년 9월 11일
※ 본 연구는 정보통신부에서 지원하는 대학기초연구지원사업으로 수행되었음(2001-103-3).

보를 저장[1,5,6]하기도 하고 이동 호스트 자신의 로컬 디스크에 저장할 수도 있다[3]. 그러나, 이들 이동 호스트의 결함 복구에 관한 기법들은 관련된 기지국에 결함이 발생했을 때, 기지국이 복구 될 때까지 기다려야 하는 단점을 지니고 있다.

본 논문에서는 이동 호스트에 결함이 발생했을 때 이를 복구하고 기지국 결함 발생 시에도 이를 해결할 수 있는 복구기법을 제안한다. 이동 호스트의 결함을 해결하기 위한 Pradhan[1]의 Trickle 기법을 기반으로하여 기지국의 결함 복구 기법을 제안하였는데 기본 개념은 체크포인트를 복수개의 기지국에 다중화하는 방법으로 기지국의 가용도를 향상시켰다. 또한, 이동 호스트의 결함시 체크포인트가 존재하는 기지국에 동시에 결함 발생시 기지국 복구를 기다리는 방법과 체크포인트가 존재하는 다른 기지국의 정보를 이용하는 방법의 복구 비용을 비교 분석하였다.

Ⅱ. 관련 연구

이동 호스트의 결함에 대비하여 일정한 간격으로 이동 호스트의 프로세스 정보를 기지국에 저장하는 기법이 제안되었다[1,6]. 기지국에 프로세스 정보를 저장하는 이유는 이동 호스트가 가진 로컬 디스크에 프로세스의 정보를 저장하는 것보다 안전하다고 판단되기 때문이다. 이는 유선 환경에서부터 많은 연구가 진행되어 왔다[7,8]. 또한 이동 호스트는 이동성을 갖기 때문에 주기적인 저장 외에 이동한 이동 호스트가 요청한 동작들에 대해 로그를 기록할 필요가 있다. 이런 로깅이나 체크포인팅은 셀룰라 네트워크 기반의 이동 컴퓨팅 환경에서 기지국에 저장되어진다. 이들은 각기 이동 호스트가 안정적이냐 그렇지 않느냐에 따라 메시지 로그나 체크포인트를 이동 호스트에 저장하는 기법과 기지국에 저장하는 기법으로 나눌 수 있다. 이와 같이 이동 컴퓨팅 환경에서 이동 호스트의 결함 복구를 위해서 프로세스 정보를 안전한 저장소에 저장하기 위한 많은 연구가 진행되었다[1,3,5,6,9].

Pradhan[1]은 이동 호스트에서 발생하는 메시지 로그에 대해서 메시지가 발생할 때마다 동기적으로 메시지 로그를 기록하는 Logging 기법과 메시지를 받을 때마다 체크포인팅을 수행하는 No Logging 기법으로 나누어 체크포인팅을 수행하였으며, 이 두 기법을 핸드오프마다 체크포인트와 메시지 로깅을 관리하는 기법에 따라 세 가지로 구분하고 비용을 분석하였다. 이 세 가지 기법 중 (1)첫째 방법(Pessimistic 기법)은 이동 호스트의 정보를 저장한 기지국의 정보를 이동 호스트가 다른 셀 내의 기지국으로 이동하였을 때, 이전 기지국에 있던 이동 호스트의 정보를 모두 가져오는 방법이며, (2)둘째(Lazy 기법)은 첫째의 경우에 발생하는 오버헤드를 최소화하기 위해 이동 호스트가 이동하였더라고 이동 호스트의 로그나 체크포인트 정보를 가져오지 않고 링크만을 갖는 방법을 제안하였으며, (3)마지막(Trickle 기법)으로 이동 호스트가 이동한 기지국의 이전 기지국의 정보를 바로 이전 기지국까지 가져오는 방법이 있다. 그러나 이동 호스트의 결함에 대하여 복구가 진행되는 동안이나 결함이 발생하지 않은 상황에서도 체크포인트가 존재하는 기지국에 결함이 발생하면 이동 호스트는 즉각적인 서비스를 요하는 업무(증권정보, 전자메일 서비스, 티켓예약 서비스 등)를 더 이상 수행할 수 없게 된다. 이런 기지국 결함 복구 기법에 대하여 Alagar[4]에 의하여 연구 되었다. Alagar는 기지국 결함에 대하여 또 다른 기지국에 이동 호스트에 대한 프로세스 정보의 카피를 두어 이를 해결하였다. Alagar는 기지국 결함 복구 기법을 두 가지로 나누었다. 첫번째 결함 복구 기법은 pessimistic 기법으로 이동 호스트의 정보를 기지국에 저장하기 전에 보조 기지국에 이 이동 호스트의 정보가 저장 되어있는지를 확인하고 저장하는 방법이며, 두 번째 결함 복구 기법은 이동 호스트의 정보를 비동기적으로 기지국에 저장하는 방법이다. 또한 Alagar는 이동 호스트들의 프로세스 정보들을 다른 기지국에 중복해야 하는 데 이를 위해 보조 기지국을 선택하는 방법에 대하여 다루었다.

본 논문에서는 관련연구[1]의 Trickle기법을 기반으로 하여 기지국의 이중화 기법을 제안하였다. Trickle기법을 선택한 이유는 Lazy기법은 기지국을 이중화 하였을 때, Trickle기법에 비하여 핸드오프 시에 많은 네트워크 비용이 소요되기 때문이며 관련연구[4]와 비교하여 [4]에서는 이웃한 기지국에서 중복할 기지국을 선택하는 반면 제안한 이중화 기법은 이동 호스트가 거쳐간 이동 경로를 이용 거쳐간 셀 내의 기지국에 정보를 중복한다는 점에서 다르며, 이렇게 함으로써 중복할 기지국을 찾는데 소요되는 추가적인 비용이 소요되지 않으므로 효과적이다.

Ⅲ. Modified Trickle 결함 복구 기법

이동 컴퓨팅 환경에서의 복구 기법은 Pradhan[1]에 의해 제안되었다. 그러나 Pradhan의 논문에서는 기지국에 결함이 발생하였을 때, 발생하는 비용에 대해 고려하지 않았다. 그림 1에서와 같이 Pradhan[1]이 제안한 Trickle 기법은 현재 이동 호스트(MH)가 BS3에 존재할 때, 메시지 로그와 프로세스 상태(체크포인트)를 지역의 기지국(BS3)에 저장한다. 만일 마지막 로그와 체크포인트를 BS1지역에서 저장하고 BS2를 거쳐 BS3지역까지 이동을 했다면 로그와 체크포인트를 한 홉(hop) 떨어진 기지국(BS2)으로 전송한 후 저장하여 결함 복구시 최대한 홉 떨어진 기지국에서 필요한 로그와 체크포인트를 얻을 수 있도록 한다. 이동 호스트(MH)가 이동하여 또 다른 셀로 이동할 때는 핸드오프 중에 컨트롤 메시지를 이전에 속해있던 셀의 기지국(BS2)에 보냄으로써 마지막에 속해있던 기지국(BS2)을 알 수 있다. 새로운 셀의 기지국(BS3)에서 체크포인트가 발생하면 이전의 셀의 기지국(BS2)에 저장되어있던 로그와 체크포인트 정보를 삭제한다. 그러나 이동 호스트가 결함이 발생하여 복구 도중에 현재 속해 있던 기지국(BS3)에서 체크포인트가 수행되지 않았다면 이전에 속해 있던 셀의 기지국(BS2)에서 로그와 체크포인트에 대한 정보를 얻을 수 있다. 이동 호스트에 결함이 발생하였을 때, 이들 정보를 이용하여 결함을 복구할 수 있다. 그러나 이러한 Trickle기법은 이동 호스트에 대한 결함만을 고려하였으므로 기지국이 결함을 발생하였을 때는 복구할 수 없는 단점을 가지고 있다. 이런 상황을 극복하기 위해 본 논문에서는 Trickle기법을 기반으로 체크포인트를 이중화하는 기법을 제안한다. Modified Trickle기법은 이동 호스트와 기지국 결함 모두를 복구하기 위한 방안을 제시한다.

그림 2는 Modified Trickle 기법을 나타낸다. 현재 이동 호스트가 속한 셀 내의 기지국에서 체크포인트가 수행되지 않은 상태에서 이동 호스트가 결함을 발생하였다면, 현재에 속한 셀의 기지국은 이전 셀의 기지국으로 마지막으로 저장된 이동 호스트의 체크포인트 정보를 요청한다. 그러나 이전 셀의 기지국 역시 결함이 발생하였을 때는 이동 호스트의 정보를 요청하여도 이에 응답하지 못할 것이다. 이런 상황에서 그림 2에서와 같이 이동 호스트가 현재의 셀(BS3)로 이동하기 이전 셀의 기지국(BS2)뿐만 아니라 그 이전 셀의 기지국(BS1)에도 이동 호스트의 로그나 체크포인트 정보를 삭제하지 않고 저장하였다가 현재의 셀 바로 이전 셀의 기지국(BS 2)에 결함이 발생하였을 때, 다른 경로(BS4)를 이용하여 복구에 필요한 정보를 얻음으로써 이동 호스트의 결함을 극복할 수 있다. 제안한 Modified Trickle 기법은 이동 호스트와 기지국의 결함에 대처할 수 있는 방안을 제공한다. 기지국에 결함이 발생하면, 이동 호스트에 비하여 상대적으로 긴 시간 그리고 많은 이용자들이 서비스를 받지 못한다. 본 논문에서 제안할 기지국 결함 허용 기법은 이를 해결하기 위하여 유용하게 적용될 수 있다. 현재 이동 호스트가 속한 기지국(지역 기지국)에서의 결함 발생시 이동 호스트는 다른 기지국과도 통신이 단절되므로 지역 기지국이 복구될 때까지 기다려야 한다.

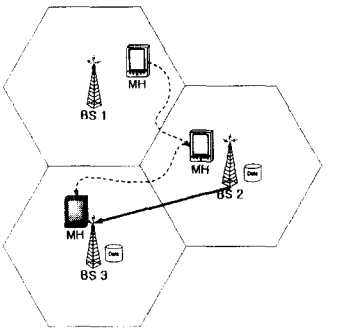


그림 1. Trickle 기법

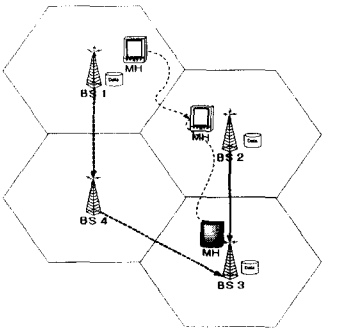


그림 2. Modified Trickle 기법

본 논문에서는 Modified Trickle기법을 Logging과 No Logging기법으로 나누어서 비용을 분석하였으며, 기지국 복구를 기다리는 기법과 새로운 경로를 설정하는 Modified Trickle 기법의 성능을 비교 분석하였다. Logging기법이란 주기적으로 이동 호스트의 프로세스 상태(체크포인트)를 지역 기지국에 저장하고 메시지 또는 쓰기 동작 발생시마다 이를

지역 기지국에 저장하는 기법을 의미하며. No Logging기법이란 메시지 또는 쓰기 동작 발생시마다 이동 호스트의 프로세스 상태를 지역 기지국에 저장하는 방식을 의미한다.

Ⅳ. 성능 분석 및 평가

본 장에서는 제안하는 Modified Trickle기법의 무선 및 유선 네트워크에서 통신 비용을 분석하고 성능을 비교한다.

기지국 사이에서 이동 호스트가 이동하면서 핸드오프 시에 발생하는 통신 비용, 이동 호스트와 기지국의 결함허용을 위한 추가비용, 기지국이나 이동 호스트에 결함이 발생하였을 때 복구에 소요되는 평균비용을 고려하였다. 또한 Loging과 No Logging 기법에 따라 비용을 비교 분석하였다. 로그와 체크포인트가 저장된 기지국 결함 발생시 복구 되기를 기다리는 비용이 새로운 경로를 설정하여 다른 기지국에 저장된 정보를 얻어 결함을 복구하는 비용에 비해 크다면 후자를 선택하는 방안이 필요하다. 물론, 지역 기지국의 결함 발생시 복구될 때까지 기다려야 한다.

이를 위해 본 논문에서는 제안한 Modified Trickle 기법을 기반으로 두 가지 기법의 성능을 비교하여 분석하였다. <표 1>은 성능 분석에 사용된 패러미터의 기호와 설명을 나타낸다.

1. 시스템 모델링(System Modeling)

한 핸드오프 간격을 그림 3과 같은 상태전이도로 표시할 수 있다. 여기에서 핸드오프 간격은 두 핸드오프 사이의 시간 간격으로 정의한다. 상대 0은 핸드오프의 초기 상태를 말하며, 상태 0에서 1로 전이하는 과정은 핸드오프 시까지 이동 호스트가 쓰기 동작 중에 결함이 발생하지 않았을 때를 의미한다. 상태 0에서 상태 2로의 전이는 이동 호스트가 핸드오프 이전에 결함을 발생시킨 경우를 의미한다.

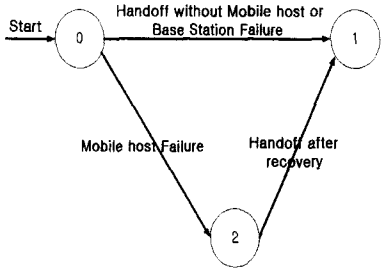


그림 3. Modified Trickle 기법의 Markov Chain

본 논문에서는 이동 호스트의 결함 발생율과 기지국의 결함 발생율을 함께 고려하였다.

이동 호스트의 결함 발생율을 λ_m 로 그리고 기지국의 결함 발생율은 λ_b 정의하였다. 그러므로 상태 0에서 상태 2로 전이할 확률을 P_{02} 라 하면 이 확률은 다음과 같이 정의된다.

$$P_{02} = \frac{\lambda_m}{\lambda_m + \mu}$$

각각의 상태 전이 (a, b)는 Markov에서 상태 a에서 상태 b로의 전이를 말하며 이들의 확률을 P_{ab} 라고 정의하며, 이들의 전이하는 데 소요되는 비용을 C_{ab} 라고 정의한다. 상태 전이 (0, 1)의 비용 C_{01}은 상태 0에서 상태 1로 전이되기 전까지의 예상되는 비용으로 다음과 같이 계산할 수 있다[1].

$$C_{01} = \alpha C_c \times N_c(T) + \alpha C_l \times N_l(T) + C_h \quad (1)$$

여기서 $C_c \times N_c(T)$항은 체크포인트를 무선 링크를 통해 현재 기지국으로 보내는 비용이며, $C_l \times N_l(T)$항은 메시지 로그를 기지국으로 보내는 비용이다. $N_c(T)$와 $N_l(T)$는 t시간 단위의 체크포인트 수와 메시지 로그의 수를 나타낸다. 또한 C_h는 핸드오프 비용을 나타낸다. 이동 호스트가 핸드오프 간격 사이에서 발생시키는 총 비용 C_t는 핸드오프를 수행하는데 소요되는 총 비용 C_{01}과 결함 발생시 복구 비용 C_r 합한 것으로 다음과 같이 계산된다.

$$C_t = C_{01} + P_{02} C_r \quad (2)$$

여기서 C_r은 이동 호스트의 결함 시 복구 비용으로 이동 호스트에서 현재 기지국까지의 무선 네트워크와 기지국들 사이의 유선 네트워크를 모두 사용하며 상세한 분석은 4.2와 4.3에 나타내었다. 본 논문에서는 체크포인트가 존재하는 기지국이 결함 발생시 복구될 때까지 기다리는 기법과 새로운 경로를 설정하여 체크포인트가 존재하는 다른 기지국을 이용하여 결함을 복구하는 기법의 비용을 비교 분석한다.

이동 호스트가 이동하는 상황에서 결함이 발생할 경우에는 이동 호스트의 정보가 저장된 기지국에서 정보를 가져오게 되는데 이 때 소요되는 비용은 아래와 같다.

- 무선 네트워크 환경에서 소요되는 비용 : x

표 1. 파러미터 : 성능분석에 사용된 파러미터들

λ_m	이동 호스트의 결함 발생율	S	기지국 간의 홉 수
λ_b	기지국의 결함 발생율	C_r	이동 호스트의 복구 비용
μ_b	기지국의 결함 복구율	ρ	쓰기 동작에 대한 사용자 입력의 비율
γ	C_l / C_c, 인접한 기지국끼리 체크포인트 전송율대 애플리케이션 메시지 전송비율	k	체크포인트 당 쓰기 동작의 수, Logging에서는 $k = \beta T_c$ 이며, No Logging에서는 k는 항상 1이다.
α	무선 네트워크 요소(wireless network factor)로 무선 네트워크의 한 홉(one hop) 사이에서 메시지를 보내는 비용과 유선 네트워크의 한 홉 사이에서 메시지를 보내는 비용과의 비율로 정의되고 α 값이 작을수록 무선 네트워크의 대역폭이 높아진다. 무선 링크의 사용 비용으로 정의할 수 있다.	C_l	이동 호스트가 핸드오프를 수행하는데 드는 총 비용
		μ	이동 호스트의 핸드오프율
		β	단위 시간 동안의 통신 회수
C_{cx}^{mh}	이동 호스트가 x홉 떨어진 기지국에 체크포인트를 전송하는 평균 비용	C_{mx}^{bs}	x홉 떨어진 기지국들간의 컨트롤 메시지를 전송하는 평균 비용
C_{lx}^{mh}	이동 호스트가 x홉 떨어진 기지국에 애플리케이션 메시지를 전송하는 평균 비용	$C_{cx}^{mh'}$	이동 호스트가 x홉 떨어진 기지국에 결함없이 체크포인트를 전송하는 평균 비용
C_{mx}^{mh}	이동 호스트가 x홉 떨어진 기지국에 컨트롤 메시지를 전송하는 평균 비용	$C_{lx}^{mh'}$	이동 호스트가 x홉 떨어진 기지국에 결함없이 애플리케이션 메시지를 전송하는 평균 비용
C_{cx}^{bs}	x홉 떨어진 기지국들간의 체크포인트를 전송하는 평균 비용	$C_{mx}^{mh'}$	이동 호스트가 x홉 떨어진 기지국에 결함없이 컨트롤 메시지를 전송하는 평균 비용
C_{lx}^{bs}	x홉 떨어진 기지국들간의 애플리케이션 메시지를 전송하는 평균 비용	$C_{cx}^{bs'}$	x홉 떨어진 기지국들간의 결함없이 체크포인트를 전송하는 평균 비용
$C_{lx}^{bs'}$	x홉 떨어진 기지국들간의 결함없이 애플리케이션 메시지를 전송하는 평균 비용	$C_{mx}^{bs'}$	x홉 떨어진 기지국들간의 결함없이 컨트롤 메시지를 전송하는 평균 비용
T_c	체크포인트 주기, No Logging기법을 위해서 T_c 는 $1/\beta$이다.	γ'	통신 이동율, 핸드오프 당 쓰기 동작의 기대값, $1/\mu$과 같다.
N_h'	지역 기지국과 체크포인트가 저장될 기지국간 떨어진 홉(hop) 수	ε	C_m / C_c, 기지국간의 체크포인트를 전송하는 비용에 대한 컨트롤 메시지를 전송하는 비용의 비율

홉 떨어진 기지국과 이동 호스트의 통신 비용($x \geq 0$)으로써 이 때, 무선 통신은 제외하여 이동 호스트와 지역 기지국은 0홉으로 계산한다.

$$C_{cx}^{mh} = (\alpha + x)C_c + \frac{(x+1)\lambda_b}{(x+1)\lambda_b + \mu_b}\frac{1}{\mu_b}$$

$$C_{mx}^{mh} = (\alpha + x)C_m + \frac{(x+1)\lambda_b}{(x+1)\lambda_b + \mu_b}\frac{1}{\mu_b}$$

$$C_{lx}^{mh} = (\alpha + x)C_l + \frac{(x+1)\lambda_b}{(x+1)\lambda_b + \mu_b}\frac{1}{\mu_b}$$

$$C_{cx}^{mh'} = (\alpha + x)C_c$$

$$C_{mx}^{mh'} = (\alpha + x)C_m$$

$$C_{lx}^{mh'} = (\alpha + x)C_l$$

- 유선 네트워크 환경에서 소요되는 비용 : x 홉 떨어진 기지국사이의 통신 비용을 계산한다.($x < 0$)

$$C_{cx}^{bs} = xC_c + \frac{(x+1)\lambda_b}{(x+1)\lambda_b + \mu_b}\frac{1}{\mu_b}$$

$$C_{mx}^{bs} = xC_m + \frac{(x+1)\lambda_b}{(x+1)\lambda_b + \mu_b}\frac{1}{\mu_b}$$

$$C_{lx}^{bs} = xC_l + \frac{(x+1)\lambda_b}{(x+1)\lambda_b + \mu_b}\frac{1}{\mu_b}$$

$$C_{cx}^{bs'} = xC_c$$

$$C_{mx}^{bs'} = xC_m$$

$$C_{lx}^{bs'} = xC_l$$

2. No Logging Modified Trickle 기법

No Logging 기법에서 핸드오프 비용은 현재의 기지국에서 새로운 기지국으로부터 컨트롤 메시지를 보내는 비용으로 계산할 수 있다[1]. Modified Trickle 기법은 기지국의 결함까지 대처하기 때문에 기지국의 결함 발생율도 함께 고려하여야 한다. 이동 호스트가 새로운 기지국의 셀 내로 이동했을 때, 컨트롤 메시지는 이동하기 이전의 기지국에게 이동 호스트의 상태를 요청하기 위한 것이다. 핸드오프 동작 시에 소요되는 비용은 자신이 속한 지역 기지국으로부터 이동 호스트의 상태 정보를 가져오는 비용(C_{c0}^{mh})과 바로 앞과 전 기지국의 정보를 가져오는 데 소요되는 비용의 합($C_{m1}^{bs}+N_h'(C_{m1}^{bs}+C_{c1}^{bs})$)으로 계산할 수 있다. 체크포인트가 현재의 기지국에 처음 저장했을 때, 컨트롤 메시지는 마지막으로 체크포인트를 수행한 기지국에 전송되어서 그 이전 기지국의 체크포인트 정보를 제거해야 하므로 $N_h'C_{m2}^{bs}$로 계산할 수 있다.

No Logging 기법을 사용하기 때문에 $N_c(T)=r$, $N_f(T)=0$, $C_h=C_{c0}^{mh}+2C_{m1}^{bs}$이 되고 이것을 식 (1)에 대입하게 되면 비용 C_{01}은 다음과 같이 계산된다.

$$C_{01}=rC_{c0}^{mh}+C_{m1}^{bs}+N_h'(C_{m1}^{bs}+C_{c1}^{bs})+N_h'C_{m2}^{bs}$$

이동 단말기에 결함이 발생되면 이를 복구하는데 소요되는 비용 C_r은 다음과 같이 두 가지 방법으로 나눌 수 있다. (a)결함이 체크포인트가 존재하는 기지국에 결함이 발생하고 그 기지국이 지역 기지국이 아닌 경우 이를 해결하기 위하여 기지국의 복구를 기다리는 방법과 (b)다른 기지국을 통해 경로를 선택한 방법이 있다.

$$C_r = a.\ N_h'(C_{c1}^{mh}+C_{m1}^{mh})+(1-N_h')(C_{c0}^{mh}+C_{m0}^{mh})$$

$$b.\ N_h'[(\frac{\mu_b}{2\lambda_b+\mu_b})(C_{c1}^{mh'}+C_{m1}^{mh'})+(\frac{\lambda_b}{2\lambda_b+\mu_b})\{(C_{c1}^{mh}+C_{m1}^{mh})+(C_{cs}^{mh}+C_{ms}^{mh})\}]+(1-N_h')(C_{c0}^{mh}+C_{m0}^{mh})+N_h'^2(\frac{\lambda_b}{\lambda_b+\mu_b})T_c$$

식 a는 복구를 기다리는 비용을 나타내며, N_h'은 현재의 기지국으로부터 이동 호스트의 정보가 저장되어 있는 기지국과 떨어진 홉 수의 기대치를 나타내며 $N_h'=(1-e^{-\mu Tc})$로 정의한다. 현재 기지국에서 마지막 체크포인팅이 일어나지 않았을 확률로 이전 기지국에 존재하는 체크포인트에 대한 정보를 얻기 위해서 소요되는 비용을 계산한 것이다.

식에서 b는 체크포인트가 존재하는 기지국에 결함에 발생하였을 때, 그리고 그 기지국이 지역 기지국이 아닐 때, 복구를 기다리지않고 Modified Trickle 기법을 통해 다른 기지국에서 체크포인트 정보를 얻어오는데 소요되는 비용을 뜻한다. $(1-N_h')(C_{c0}^{mh}+C_{m0}^{mh})$는 지역 기지국에 체크포인트가 있는 경우, 이를 수신하는 비용이고,

$$N_h'\{(\frac{\mu_b}{2\lambda_b+\mu_b})(C_{c1}^{mh'}+C_{m1}^{mh'})+(\frac{\lambda_b}{2\lambda_b+\mu_b})\{(C_{c1}^{mh}+C_{m1}^{mh})+(C_{cs}^{mh}+C_{ms}^{mh})\}$$

은 1홉 떨어진 기지국에 체크포인트가 있는 경우 그 기지국이 정상이면 정보를 얻어오는데 소요되는 비용을 계산하고, 결함이 발생했으면 다른 s홉 떨어진 기지국에서 체크포인팅 정보를 얻어오는데 소요되는 비용을 계산한 것이다.

또한 $N_h'^2(\frac{\lambda_b}{\lambda_b+\mu_b})T_c$은 s홉 떨어진 기지국에서 가져온 체크포인팅 정보가 마지막 체크포인팅 정보가 아닌 그 이전의 옛 체크포인팅 정보인 경우 T_c 만큼 계산 손실 비용이 발생한 것을 계산한 것이다. 그러므로 No Logging Modified Trickle 기법의 총 비용은 식(2)에서 구할 수 있다.

즉, $C_{tNLMT}=C_{01}+P_{02}C_r$ 이 된다. 총비용은 두 가지 경우를 나누어 계산할 수 있다. 기지국의 복구를 기다렸다가 서비스를 받는 경우와 다른 경로로 설정해 정보를 얻어오는 경우가 있다.

기지국의 복구를 기다리는 경우 :

$$C_{tNLMT}=(rC_{c0}^{mh}+C_{m1}^{bs}+N_h'(C_{m1}^{bs}+C_{c1}^{bs})+N_h'C_{m2}^{bs})+\frac{\lambda_m}{\lambda_m+\mu}\{N_h'(C_{c1}^{mh}+C_{m1}^{mh})+(1-N_h')(C_{c0}^{mh}+C_{m0}^{mh})\}$$

다른 기지국을 통한 경로를 선택하는 경우 :

$$C_{tNLMT}=(rC_{c0}^{mh}+C_{m1}^{bs}+N_h'(C_{m1}^{bs}+C_{c1}^{bs})+N_h'C_{m2}^{bs})+\frac{\lambda_m}{\lambda_m+\mu}[N_h'\{(\frac{\mu_b}{2\lambda_b+\mu_b})(C_{c1}^{mh'}+C_{m1}^{mh'})+(\frac{\lambda_b}{2\lambda_b+\mu_b})\{(C_{c1}^{mh}+C_{m1}^{mh})+(C_{cs}^{mh}+C_{ms}^{mh})\}\}+(1-N_h')(C_{c0}^{mh}+C_{m0}^{mh})+N_h'^2(\frac{\lambda_b}{\lambda_b+\mu_b})T_c]$$

3. Logging Modified Trickle 기법

No Logging기법과 달리 Logging기법은 쓰기 동작에 의해 생성되는 로그와 체크포인트 모두 관리하여야 한다. 그러므로 Logging Modified Trickle 기법의 핸드오프 동작의 비용 $C_h = \nu C_{l2}^{bs} + C_{c2}^{bs} + 3C_{m2}^{bs}$ 이 된다. Logging 기법이기 때문에 $N_c(T) = r/k$, $N_l(T) = \rho r$이 되며 ρr는 두 기지국간의 핸드오프 사이에서 발생하는 사용자의 입력수를 나타낸다. 여기에서 ρ는 쓰기 동작 중에서 사용자 입력에 의한 비율을 의미한다. 다른 이동 호스트로부터의 메시지 전송은 기지국에서 저장한다고 가정하였기 때문에 로깅할 필요가 없다. 위의 값들을 식 (1)에 대입하여 C_{01}을 계산할 수 있다.

$$C_{01} = \frac{r}{k}C_{c0}^{mh} + \rho r C_{l0}^{mh} + \rho r C_{m0}^{mh} + \nu C_{l2}^{bs} + C_{c2}^{bs} + 3C_{m2}^{bs} + N_h' C_{m2}^{bs}$$

이동 단말기에 결함이 발생되면 이를 복구하는데 소요되는 비용 C_r은 다음과 같이 두 가지 방법으로 나눌 수 있다. (a)결함이 체크포인트가 존재하는 기지국에 결함이 발생하고 그 기지국이 지역 기지국이 아닌 경우 이를 해결하기 위하여 기지국의 복구를 기다리는 방법과 (b)다른 기지국을 통해 경로를 선택한 방법이 있다. 기지국의 복구를 기다리는 경우는 두 홉 떨어진 기지국에서 이전에 저장된 체크포인트에 대한 정보를 제거하는 데 소요되는 비용을 고려해야 하며, 다른 기지국을 통해 경로를 선택한 경우는 s홉 떨어진 기지국에서 체크포인팅 정보를 받아야 한다. 여기에서 V는 핸드오프 중에 발생하는 평균적인 로그의 크기를 나타내며, 포아송 프로세스라고 가정하면 $V=(k-1)/2$라 정의 한다. 복구 비용 C_r은 4.2절에서 구한 비용에 로그를 전송 받는 비용만을 추가로 고려하면 다음과 같이 계산할 수 있다.

No Logging 기법과는 달리 로그에 대한 정보를 기록하는데 추가 비용이 소요된다. 여기에서 핸드오프 동작이 포아송 분포를 따른다고 가정하면 $V'=(k-1)/2$이 되며 V'은 결함 발생시의 로그 크기의 기대값이다. No Logging 기법에서의 비용에 로그를 전송받는데 소요되는 추가비용만을 고려하면 된다.

$$C_r = a.\,N_h'(\nu' C_{l1}^{mh} + C_{c1}^{mh} + C_{m1}^{mh}) + (1 - N_h')(\nu' C_{l0}^{mh} + C_{c0}^{mh} + C_{m0}^{mh})$$
$$b.\,N_h'[\frac{\mu_b}{2\lambda_b + \mu_b}(\nu' C_{l1}^{mh'} + C_{c1}^{mh'} + C_{m1}^{mh'}) +$$
$$\frac{\lambda_b}{2\lambda_b + \mu_b}\{(\nu' C_{l1}^{mh} + C_{c1}^{mh} + C_{m1}^{mh}) + (\nu' C_{ls}^{mh} + C_{cs}^{mh} + C_{ms}^{mh})\}] +$$
$$(1 - N_h')(\nu' C_{l0}^{mh} + C_{c0}^{mh} + C_{m0}^{mh}) + N_h'^2(\frac{\lambda_b}{\lambda_b + \mu_b})T_c$$

그러므로 Logging Modified Trickle 기법의 총 비용은 $C_{tLMT} = C_{01} + P_{02}C_r$ 이 된다.

결과적으로 Logging modified Trickle 기법의 총 비용은 두 가지 경우에 다음과 같다.

기지국의 복구를 기다리는 경우 :

$$C_{tLMT} = (\frac{r}{k}C_{c0}^{mh} + \rho r C_{l0}^{mh} + \rho r C_{m0}^{mh} + \nu C_{l2}^{bs} + C_{c2}^{bs} + 3C_{m2}^{bs} + N_h' C_{m2}^{bs}) +$$
$$\frac{\lambda_m}{\lambda_m + \mu}\{N_h'(\nu' C_{l1}^{mh} + C_{c1}^{mh} + C_{m1}^{mh}) + (1 - N_h')(\nu' C_{l0}^{mh} + C_{c0}^{mh} + C_{m0}^{mh})\}$$

다른 기지국을 통한 경로를 선택하는 경우 :

$$C_{tLMT} = (\frac{r}{k}C_{c0}^{mh} + \rho r C_{l0}^{mh} + \rho r C_{m0}^{mh} + \nu C_{l2}^{bs} + C_{c2}^{bs} + 3C_{m2}^{bs} + N_h' C_{m2}^{bs}) +$$
$$\frac{\lambda_m}{\lambda_m + \mu}[N_h'\{\frac{\mu_b}{2\lambda_b + \mu_b}(\nu' C_{l1}^{mh'} + C_{c1}^{mh'} + C_{m1}^{mh'}) +$$
$$\frac{\lambda_b}{2\lambda_b + \mu_b}\{(\nu' C_{l1}^{mh} + C_{c1}^{mh} + C_{m1}^{mh}) + (\nu' C_{ls}^{mh} + C_{cs}^{mh} + C_{ms}^{mh})\}\} +$$
$$(1 - N_h')(\nu' C_{l0}^{mh} + C_{c0}^{mh} + C_{m0}^{mh}) + N_h'^2(\frac{\lambda_b}{\lambda_b + \mu_b})T_c]$$

V. 비용 분석 및 성능 평가

1. 최적의 체크포인트 간격

제안한 Modified Trickle기법에서 logging기법에 대한 비용을 분석하기 위해 최적의 체크포인트 간격이 요구된다. No Logging기법에서는 매 쓰기동작이 발생할 때마다 체크포인트가 발생하는 반면, Logging기법에서는 매 체크포인트 주기인 T_c 마다 주기적으로 체크포인트를 저장한다. 쓰기 동작율인 β 을 1로 정의하고, T_c 는 체크포인트당 쓰기 동작의 수인 k와 같다. Logging기법에서 최적의 k는 전체 비용을 최소화할 수 있는 값을 의미한다. 최적의 k는 기지국 결함율인 λ_b와 이동 호스트의 결함율인 λ_m, 상대적인 로깅 비용인 γ, 무선 통신 비용인 α, 핸드오프당 쓰기비용의 기대값인 γ 의 함수들로 정의할 수 있다. Logging기법에서 전체 비용을 최소화할 수 있는 최적의 k 값을 구하기 위해 아래와 같은 방법을 정의한다. (아래에서 $k_{opt_Recovery}$는 복구비용을 최소화하는 최적의 값을, $k_{opt_Recovery}$는 새로운 경로를 설정할 때 소요되는 전체비용을 최소화할 수 있는 k 값을 의미한다.)

$$\frac{\partial C_{tLMT\_Recovery}}{\partial k} = 0,\ \frac{\partial^2 C_{tLMT\_Recovery}}{\partial^2 k} << 0$$
$$\frac{\partial C_{tLMT\_NewPath}}{\partial k} = 0,\ \frac{\partial^2 C_{tLMT\_NewPath}}{\partial^2 k} << 0$$

위의 조건을 만족하는 최적의 k 값을 구하면 아래와

같다.

$$k_{opt\_Recovery} = \sqrt{\frac{r \cdot C_{c0}^{mh}}{1/2 \cdot (N_h' \cdot P_{02} \cdot C_{l1}^{mh} + (1 - N_h') \cdot P_{02} \cdot C_{c0}^{mh} + C_{l2}^{bs})}}$$

$$k_{opt\_NewPath} = \sqrt{\frac{r \cdot C_{c0}^{mh} \cdot}{1/2 \cdot (N_h' \cdot P_{02} \cdot C_{l1}^{bs'} (\frac{\mu_b}{2\lambda_b + \mu_b}) + N_h' \cdot P_{02} \cdot C_{l1}^{bs} (\frac{\lambda_b}{2\lambda_b + \mu_b}) + N_h' \cdot P_{02} \cdot C_{ls}^{bs} (\frac{\lambda_b}{2\lambda_b + \mu_b}) + (1 - N_h') \cdot P_{02} \cdot C_{l0}^{bs}}}$$

2. 비용 분석 및 성능 평가

비용 분석을 위해 패러미터 값을 C_c에 대해서 정규화하였다. γ 는 상대적인 로깅 비용으로 C_l / C_c와 같다. 또한 ε 는 상대적인 컨트롤 메시지 비용으로 C_l / C_c와 같다. 그러므로 $C_l = \gamma C_c, C_m = \varepsilon C_c$, $C_c = 1$ 이다. 실제적으로 $C_m << C_c$ 라고 가정하였다. 결국 위의 식으로부터 $C_l = \gamma, C_m = \varepsilon$ 를 얻을 수 있다. 본 논문의 비용 분석에서는 컨트롤 메시지 $C_m = \varepsilon = 10^{-4}$, 로깅 비용을 $\gamma = C_l = 10^{-1}$, 쓰기 동작에 대한 사용자 입력의 비율을 $\rho = 0.5$, 무선 네트워크 요소 $\alpha = 10$이라고 가정하였다. 최적의 체크포인트 간격은 Logging 기법을 위해 필요하므로 최적의 체크포인트 간격을 결정하여 실제 비용분석에 이용한다. Logging 기법은 주기적인 체크포인트마다 발생한다. 본 논문에서는 Logging 기법에 대하여 $T_c = \sqrt{\frac{2C}{\lambda}}$ 을 이용하여 최적의 체크포인트 간격을 정의하였으며, No logging 기법일 경우 $T_c = 1/\beta$이므로 $T_c = 1$로 정의 된다.

그림 4에서 $\lambda_m = 10^{-2}, \lambda_b = 10^{-4}, r = 10$ 로 정의하였으며, 기지국 결함 발생시 기지국의 복구율에 따른 복구비용을 나타낸다. 그림에서 기지국의 복구율 μ_b가 증가할수록 기지국의 복구를 기다리는 비용이 상대적으로 적게 소요됨을 알 수 있다. 즉, 이동 호스트의 결함 발생시(transient fault) 지역 기지국에 있던 정보를 얻어오는 상황에서 마지막으로 저장된 이동 호스트의 정보가 지역 기지국에 없다면, 그 이전의 기지국으로 정보를 요청할 것이다. 그러나 이전의 기지국에 결함이 발생하였을 경우, 기지국의 복구율이 크다면 복구를 기다리는 경우가 유리하기 때문이다. 즉, 이동 호스트의 정보를 중복하여 저장한 기지국에서 정보를 가져오기 위해 새로운 경로를 설정하는 방법보다 기지국의 복구를 기다렸다가 정보를 가져오는 비용이 적게 소요된다. Logging과 No Logging기법에 비교해보면, 공통적으로 $\mu_b = 10^{-1}$ 과 $\mu_b = 10^{-2}$ 사이에서 기지국 복구율이 증가할수록 복구를 기다리는 것이 유리하며, 반면 감소할수록 새로운 경로를 설정하는 것이 유리함을 알 수 있다.

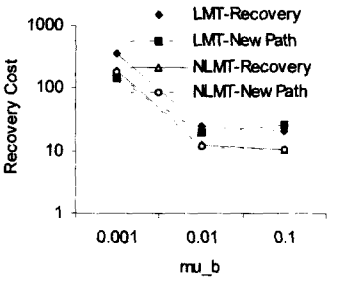


그림 5. 이동성에 따른 복구비용(μ_b = 0.1)

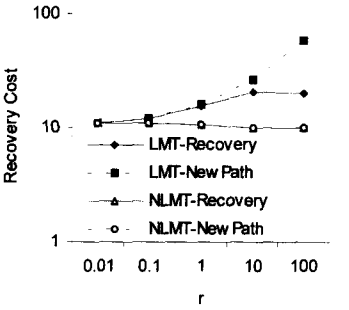


그림 6. 이동성에 따른 복구비용(μ_b = 0.01)

그림 5와 6 그리고 7에서는 기지국 복구율 μ_b와 핸드오프당 쓰기 동작의 기대값 r에 따른 복구 비용을 나타낸 것이다. 여기에서 $\lambda_m = 10^{-2}, \lambda_b = 10^{-4}$ 로 정의하였다. 그림 5에서 $\mu_b = 0.1$ 일 때 Logging 기법에서 r이 증가할수록(핸드오프율 감소) 결함이 발생한 기지국에 대한 복구를 기다리는 것이 유리함을 알 수 있다. 즉, 기지국 복구율이 높으면서 핸드오프당 쓰기 동작의 기대값이 증가할 때, 기지국의 복구를 기다리는 것이 유리하다. 그러나, 그림 6과 그림 7에서는 기지국의 결함복구율이 감소하면서 γ 값이 증가할 때, 새로운 경로를 설정하는 방법이 유리하다. 그림 8과 9는 $\lambda_m = 10^{-2}, \lambda_b = 10^{-4}$ 에서의 Logging과 No Logging기법에서의 γ 에 따른 전체 비용을 나타내었다. Logging 기법의 경우 $\mu_b = 0.1$

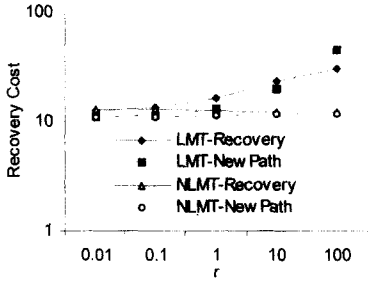


그림 6. 이동성에 따른 복구비용($\mu_b = 0.01$)

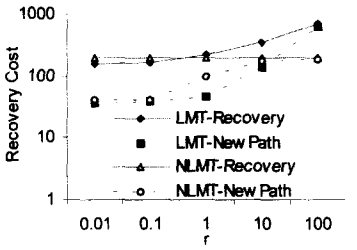


그림 7. 이동성에 따른 복구비용($\mu_b = 0.001$)

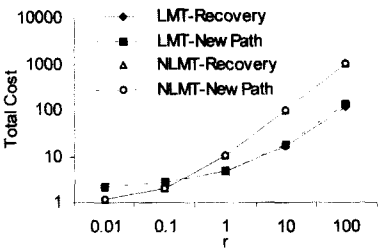


그림 8. 전체 비용($\mu_b = 0.1$)

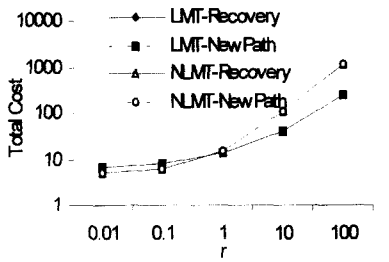


그림 9. 전체 비용($\mu_b = 0.01$)

일 때, $r = 0.1$, $\mu_b = 0.01$일 때, $r = 1$을 기준으로 γ 값이 감소할수록 새로운 경로를 선택하는 방법이, 증가할수록 복구를 기다리는 방법이 유리함을 알 수 있다. 결함이 발생했을 때, 복구를 기다리는 시간이 새로운 경로를 설정하는 방법에 비해 상대적으로 길어진다면 새로운 경로를 설정하여 복구하는 것이 효율적이다. 실제적으로 그림에서 새로운 경로를 설정하는 것과 기지국 복구를 기다리는 것의 전체 비용의 차이는 크지 않지만 복구 비용은 많은 차이를 보인다. 이동 호스트가 결함이 발생한 상황에서 정상적인 서비스를 위한 기지국 복구 시간을 단축시키는 것은 매우 중요하다. 그림 8과 9에서 핸드오프당 쓰기 동작의 기대값인 γ 이 증가할수록 Logging기법에서 주기적으로 이동 호스트의 프로세스 상태(체크포인트)를 지역 기지국에 저장하고 메시지 또는 쓰기 동작 발생시 이를 지역 기지국에 저장하기 때문에 메시지 또는 쓰기 동작 발생시마다 이동 호스트의 프로세스 상태를 저장하는 No Logging기법에 비해 비용이 적게 소요됨을 알 수 있다. 그 이유는 No Logging기법에서 이동 호스트의 이동성이 적을 때, 사용자의 쓰기 동작이 발생할 때마다 체크포인트를 수행하기 때문에 이에 소요되는 비용은 Logging기법에서의 주기적인 체크포인트를 저장하는 데 소요되는 비용보다 크기 때문이다.

표 2. Modified Trickle에서 각 기법별 복구 비용 비교

이동성, 기지국 복구율	Logging 기법	No Logging 기법
이동성이 작으면서 기지국 복구율이 큰 경우	R < N	R≈N
이동성이 크면서 기지국 복구율이 작은 경우	R > N	R > N

*기지국 복구를 기다림(R), 새로운 경로를 설정(N)

표 2는 각 기법별 복구 비용의 비교를 정리하여 나타내었다. 위의 분석결과의 공통적인 사항은 기지국 결함에 대한 복구를 기다리기 때문에 평균 $1/\mu_b$의 시간이 지연되며, 이동성이 많은 이동 호스트는 핸드오프 발생 빈도가 많으며 기지국 결함 발생시 이를 복구하는 복구율이 크면 클수록 Logging 기법의 경우 기지국 복구를 기다리는 것이 유리하며, 반대의 경우 본 논문에서 제안한 새로운 경로를 설정하여 결함을 복구하는 Modified Trickle 기법이 유리함을 알 수 있다.

Ⅵ. 결 론

이동 호스트의 결함에 대한 복구 기법은 이동 호스트 자체의 제한적인 자원으로 인해 많은 이동 컴퓨팅 환경에서 제약을 받는다. 본 논문에서는 이동 호스트에 결함이 발생했을 때, 이를 복구하고 또한

기지국 결함 발생 시에도 해결할 수 있는 복구기법을 제안하였다. 제안한 기지국 결함 복구 기법은 Pradhan[11]이 제안한 이동 호스트의 결함 복구 기법 중 Trickle 기법을 변형한 Modified Trickle기법이다. 본 논문은 기지국 결함에 대하여 고려하였다. 기지국에 결함이 발생하였을 때, 이 기지국 결함에 대한 복구 기법으로는 크게 두 가지로 나눌 수 있다. (1)기지국에서 복구를 하는 동안 이동 호스트는 기다리는 기법과 (2)새로운 경로를 설정하여서 다른 기지국에 저장되어 있는 이동 호스트들에 대한 체크포인트나 로그에 대한 정보를 가져오게 하는 것이다. 그러나 기지국의 일시적인 결함(transient failure)에 대해서는 이동 호스트가 이를 위해서 기다리는 시간이 상대적으로 짧기 때문에 새로운 경로를 설정하는 것보다 기지국의 복구를 기다리는 것이 더 효율적이며, 반대로 기지국의 복구가 느릴 때는 새로운 경로를 통한 기법이 이동 호스트의 결함 복구에 더 효율적이다.

본 논문에서는 기지국 결함에 효율적으로 대처할 수 있는 기법을 제안하였으며 이를 토대로 이동 호스트에 결함이 발생되었을 때, 효율적이고 빠르게 대처할 수 있는 방안을 제시하였다.

참 고 문 헌

[1] D. K. Pradhan, P. Krishna, and N. H. Vaidya, "Recovery in Mobile Environments: Design and Tradeoff Analysis," Proceedings of the 26th International Symposium on Fault Tolerant Computing, pp. 16-25, June 1996.

[2] P. Krishna, N. H Vaidya, and D. K. Pradhan. "Recovery in Distrubuted Mobile Environments", *IEEE* Workshop on Advances in Parallel and Distributed System, October 1993.

[3] R. Prakash and M. Singhal, "Low-Cost Checkpointing and Failure Recovery in Mobile Computing Systems," IEEE Transaction on Parallel and Distributed Systems, pp. 1035-1048, October 1996.

[4] S. Alagar, R. Rajagopalan and S. Venkatesan, "Tolerating Mobile Support Station Failures," Technical Report UTDCS-16-93, University of Texas, Dallas, 1993.

[5] A. Acharya and B. Badrinath, "Checkpointing Distributed Applications on Mobile Computers," Proceedings of the 3rd International Conference on Parallel and Distributed Information Systems, pp. 73-80, September 1994.

[6] B. Yao, K. Ssu and W.K. Fuchs, "Message Logging in Mobile Computing," Proceedings of the 29th International Symposium on Fault Tolerant Computing, pp. 294-301, June 1999.

[7] K. M. Chandy and L. Lamport, "Distributed Snapshots: Determining Global States of Distributed Systems," ACM Transactions on Computer Systems, Vol. 3, No. 1, pp. 63-75, February 1985.

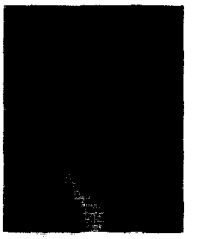
[8] E. Elnozahy, D. Johnson and Y. M. Wang, "A Survey of Rollback-Recovery Protocols in Message-Passing Systems," Technical Report CMU-CS-96-181, School of Computer Science, Carnegie Mellon University, October 1996.

[9] N. Neves and W. K. Fuchs, "Adaptive Recovery for Mobile Environments," Communication of the ACM, Vol. 40, No. 1, pp. 68-74, January 1997.

변 계 섭(Kyue-Sup Byun) 학생회원

1999년 : 아주대학교 정보 및 컴퓨터공학부 졸업(학사)
2000년 : 현재 아주대학교 정보통신전문대학원 석사과정
<주관심 분야> 실시간 시스템, 이동컴퓨팅, 결함허용 시스템

김 재 훈(Jai-Hoon Kim) 정회원



1984년 : 서울대학교 제어계측공학과(학사)
1993년 : Indiana University, Computer Science(석사)
1997년 : Texas A&M University, Computer Science (공학박사)
1984년~1991년 : 대우통신㈜ 컴퓨터연구실, 팀장
1995년~1997년 : Texas A&M University, Graduate Research Assistant

1997년 ~ 1998년 : 삼성전자㈜ 컴퓨터시스템팀,
수석연구원
1998년 ~ 현재 : 아주대학교 정보통신전문대학원,
조교수
<주관심 분야> 분산시스템, 이동컴퓨팅, 실시간시스템