이동 호스트의 효율적 결함 복구를 위한 Lazy 기법의 성능 개선

권 원 석[†] · 김 성 수[††] · 김 재 훈[††]

요 약

이동 호스트는 이동 호스트 자체의 결함, 네트워크 연결 단절, 무선 링크의 결함 등 기존 유선 네트워크에서는 찾아볼 수 없는 새로운 결함 원인들을 포함하고 있다. 현재 이동 컴퓨팅 연구 중에서 이동 호스트의 결함을 효율적으로 대처하는 결함 허용 기법에 관한 연구는 미미한 실정이다. 이동 호스트의 결함 복구 기법인 Lazy 기법은 비용 면에서는 효율적이지만 이동 호스트의 체크포인트를 소유하고 있는 기지국의 결함 발생 시 이동 호스트가 결함으로부터 복구될 수 없다는 단점을 가지고 있다. 본 논문에서는 위와 같은 Lazy 기법의 성능을 개선할 수 있는 새로운 Redundant Lazy 기법을 제안하고 성능을 분석한다.

Performance Improvement of Lazy Scheme for an Efficient Failure Recovery of Mobile Host

Wonseok Kwon[†] · Sungsoo Kim[††] · Jai-Hoon Kim[††]

ABSTRACT

A mobile host has failure causes such as failure of the mobile host, disconnection of the mobile host, and wireless link failure that have not been seen in traditional computing environments. So far there have been few studies on fault tolerance of a mobile host in mobile computing environments. The Lazy scheme, a failure recovery technique of the mobile host, is a cost-effective one. However, this scheme has a defect that the mobile host cannot be recovered from failure of the base station with acheckpoint of the mobile host. In this paper, we propose and evaluate the Redundant Lazy scheme for performance improvement of the Lazy scheme.

1. 서 론

휴대형 컴퓨터와 무선 통신 기술의 급속한 발달은 사용자에게 언제, 어디서나 필요한 컴퓨팅 환경을 제공할 수 있게 하였고, 현재 다양한 사용자의 요구를 수용하기 위하여 이동 컴퓨팅에 관한 연구가 세계적으로 활발히 진행되고 있다.

이동 호스트는 이동 호스트 자체의 결함, 네트워크 연결 단절(disconnection), 무선 링크의 결함 등의 기존 유선 네트워크에서는 찾아볼 수 없는 새로운 결함 원인들을 포함하고 있다[1]. 그러나 현재 이동 컴퓨팅 연구 중에서 이러한 이동 호스트의 결함을 효율적으로 대처하는 결함 허용 기법에 관한 연구는 미미한 실정이다.

메시지 로깅(message logging)과 체크포인팅(checkpointing)은 호스트가 결함이 발생했을 때 효율적으로

※ This work is supported in part by the Ministry of Information & Communication of Korea("Support Project of University Foundation Research<2000>" supervised by IITA).
† 준 회 원 : 아주대학교 BK21 정보통신전문대학원 석사과정
†† 정 회 원 : 아주대학교 BK21 정보통신전문대학원 교수
논문접수 : 2000년 4월 4일, 심사완료 : 2000년 8월 24일

복구할 수 있는 결함 허용 기법 중 하나이다. 메시지 로깅이란 호스트의 프로세스 상태(state)가 변할 때마다 프로세스 상태 정보를 기록하는 기법이고 체크포인팅은 주기적으로 호스트의 프로세스 상태를 기록하는 기법이다. 또한 메시지 로깅과 체크포인팅은 프로세스 상태 정보를 기록하기 위하여 디스크와 같은 안정적인 저장 장치를 필요로 한다. 만약 호스트가 작업 도중에 결함이 발생했을 경우 기록된 정보를 저장 장치로부터 추출하여 호스트가 마지막으로 메시지 로깅이나 체크포인팅을 수행한 시점으로 되돌아가 작업을 다시 수행할 수 있도록 한다.

기존의 유선 네트워크 상에서 고정된 호스트들에 대한 메시지 로깅과 체크포인팅에 관한 연구는 활발히 진행되고 있으나 이동 컴퓨팅 환경처럼 호스트에 이동성이 추가 될 경우에는 기존의 메시지 로깅이나 체크포인팅 기법을 그대로 적용하기에는 어려움이 따른다. 그 이유로는 첫 번째 이동 호스트는 로그나 체크포인트를 저장하는 안정적인 저장 장치를 소유하기 어렵다는 것이다. 이러한 문제점을 해결하기 위해 로그나 체크포인트를 기지국에 저장하는 다수의 방법들이 제안되었다[1, 2, 3]. 두 번째로 이동 호스트가 하나의 셀에 고정되어 위치하고 있다면 기존의 기법들을 그대로 수정 없이 사용할 수 있으나 이동 호스트는 그 자체의 이동성 때문에 여러 개의 셀들을 옮겨 다니는 핸드오프(handoff)를 수행함으로 기지국에 저장되어 있는 로그나 체크포인트를 관리하기가 어렵게 된다.

Pradhan[1]은 이동 컴퓨팅 환경에서의 메시지 로깅과 체크포인팅에 관한 연구에서 호스트의 로그나 체크포인트를 저장하는 기지국의 결함 발생을 고려하지 않았다. Pradhan의 기법 중 비용 측면에서 가장 효율적인 Lazy 기법은 기지국의 결함이 발생했을 경우 호스트는 기지국으로부터 디스크에 기록된 프로세스의 상태 정보를 가져올 수 없게 되어 복구를 수행할 수 없게 된다.

본 논문에서는 Lazy 기법을 보완하여 기지국의 결함이 발생하여도 호스트가 복구 가능하도록 하는 Redundant Lazy 기법을 제안하고 성능을 분석한다. 구체적으로 2장에서는 이에 관련된 기존 연구들을 살펴보고, 3장에서 이동 호스트가 수행하는 프로세스의 상태 저장 기법을 설명하고 4장에서는 이동 호스트의 결함 복구 기법을 제안한다. 5장에서는 제안된 기법의 성능을 분석하고, 마지막으로 6장에서 본 논문의 결론과 향후 연구 방향에 대해 논의한다.

2. 관련 연구

정적(static) 시스템에 대한 체크포인팅 알고리즘들은 수십년에 걸쳐 광범위하게 연구되어 왔다[4, 5, 6, 7, 8]. 이러한 연구들은 크게 두 가지의 카테고리로 분류할 수 있다. Coordinate 프로토콜은 분산 어플리케이션에 참여하는 호스트들이 자신의 로컬 체크포인트가 전역 체크포인트(global checkpoint)의 일관성(consistency)과 복구가능성(recoverability)을 보증하도록 하는 방법이고 반대로 uncoordinated 프로토콜은 호스트가 각기 독립적으로 체크포인트를 수행할 수 있도록 하는 방법이다.

이동 컴퓨팅 환경에서 체크포인팅을 수행하기 위해서는 셀과 셀 사이를 이동하는 호스트의 이동성(mobility), 이동 호스트의 안정적인 저장 장치의 가용성 문제, 무선 링크의 대역폭 등의 문제를 고려하여 체크포인팅의 빈도를 결정하여야 한다.

이동 컴퓨팅 환경에서의 메시지 로깅과 체크포인팅에 관한 기법은 로그와 체크포인트를 저장하는 저장 장치의 안정성 여부에 따라 분류할 수 있다. 첫 번째는 이동 호스트의 저장 장치를 안정적이라고 가정하는 방법이고[9, 12, 13] 두 번째 방법은 이동 호스트의 저장 장치는 안정적이지 못하다고 가정하고 로그나 체크포인트를 안정적인 기지국에 저장하는 방법이다[1, 3]. 마지막은 앞의 두 가지 방법을 혼합하는 경우로 체크포인트를 중요도에 따라 나누어 상대적으로 덜 중요한 체크포인트는 이동 호스트의 저장 장치에 저장하고 중요한 체크포인트는 기지국의 저장 장치에 각기 따로 저장하는 방법이다[2].

Achaya[2]는 로그와 체크포인트를 기지국에 저장하는 것을 가정하고 있으며 전역적으로 일관성 있는 체크포인팅을 수행하기 위해 2단계(two-phase) 체크포인팅 기법을 제안하였다.

Pradhan[1]이 제안한 두 가지의 체크포인팅 프로토콜 중 첫 번째 프로토콜은 프로세스가 메시지를 받을 때마다 새로운 체크포인트를 생성하는 방법이고 두 번째는 주기적으로 프로세스를 체크포인팅 하는 것과 동시에 메시지를 받을 때마다 그 메시지를 로깅하는 방법이다. Pradhan은 이 두 가지의 체크포인팅 프로토콜을 세 가지의 서로 다른 핸드오프 기법과 조합하여 총

여섯 가지의 기법을 제안하고 각 제안 기법들의 비용 분석을 수행하였다.

Neves[9]는 이동 컴퓨팅을 위한 적응적(adaptive) 체크포인팅 기법을 제안하였다. 이 기법에서는 체크포인팅을 하드(hard) 체크포인팅과 소프트(soft) 체크포인팅으로 나누어 하드 체크포인트는 기지국으로 전송하고 소프트 체크포인트는 이동 호스트의 로컬 디스크에 저장한다.

Yao[3]는 이동 IP(mobile IP) 환경에서의 메시지 로깅에 관한 연구를 수행하였다. 이 논문에서는 이동 호스트가 셀 사이를 이동하면서 체크포인팅을 수행하면 해당 셀의 MSS(Mobile Support Station)에 체크포인트를 저장하고 HA(Home Agent)에게 체크포인팅 수행 사실을 알리는 기법을 제안하였다. 또한 이 논문에서는 여러 셀에 저장되어 있는 체크포인트에 대한 쓰레기 값(garbage)을 처리하는 연구도 수행하였다.

3. 프로세스 상태 저장 기법

이동 컴퓨팅 환경에서 이동 호스트의 프로세스 상태 저장 기법은 기존의 메시지 로깅과 체크포인팅에서의 상태 저장 기법을 기반으로 한다. 즉 이동 호스트는 주기적으로 프로세스의 상태를 저장하는 것이다. 만약 이동 호스트의 결함이 발생했을 때, 프로세스는 마지막으로 저장된 체크포인트로부터 복구되어 다시 수행된다.

이동 호스트는 로그나 체크포인트를 저장하기 위해서 안정적인 저장 장치를 필요로 한다. 그러나 일반적으로 이동 호스트의 디스크는 안정적이지 못한 저장 장치로 인식되고 있다[1, 3]. 그러므로 이동 호스트의 결함 발생 시 체크포인트로부터 안정적으로 복구되기 위해서는 안정적인 저장 장치가 필요하고 이동 호스트는 항상 통신을 하기 위해 기지국과 무선 링크로 연결이 되어 있기 때문에 기지국의 저장 장치를 로그나 체크포인트의 저장 공간으로 사용하면 효과적일 수 있다. 따라서 본 논문에서는 기지국을 로그나 체크포인트를 저장하는 장치로 사용하는 것을 가정한다.

본 논문에서 사용하는 프로세스의 상태를 저장하는 두 가지의 기법은 다음과 같다. 첫 번째는 No Logging 기법으로 이동 호스트는 따로 일정한 주기에 맞혀 체크포인팅을 수행하지 않고 프로세스의 상태가 변하는 쓰기 동작이 일어날 때마다 체크포인팅을 수행하는 기법이다. 프로세스의 상태를 변화시키는 쓰기 동작의 유형은 사용자의 입력과 다른 이동 호스트로부터 들어오는 메시지로 구분할 수 있다. 이 두 가지 유형의 차이점은 이동 호스트가 기지국과의 무선 링크의 사용 여부에 따라 구분할 수 있고 사용자의 입력은 무선 링크를 사용하지 않으며 다른 이동 호스트의 메시지는 무선 링크를 사용한다. Logging 기법은 이동 호스트가 주기적으로 프로세스를 체크포인팅하고 쓰기 동작이 발생할 때마다 해당 동작을 로깅하는 기법이다. 쓰기 동작이 사용자 입력의 경우에는 먼저 입력 동작의 정보를 기지국에 보내 로깅을 수행한 후 기지국으로부터의 응답이 왔을 경우에 작업 수행을 계속하게 된다. 쓰기 동작이 다른 이동 호스트로부터의 메시지일 경우에는 메시지가 기지국에 먼저 도착하고 무선 링크를 통해 이동 호스트로 전송되기 때문에 바로 기지국으로 로깅을 수행하면 된다.

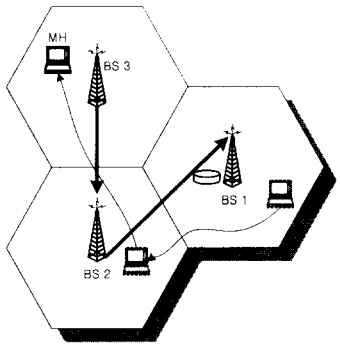
본 논문에서의 체크포인팅 프로토콜은 이동 호스트의 계산(computation)이 분산되었을지라도 결함 발생 시 또 다른 이동 호스트와는 독립적으로 복구되는 uncoordinated 프로토콜로 가정한다.

4. Redundant Lazy 결함 복구 기법

이동 호스트는 셀과 셀 사이를 움직이며 작업을 수행하기 위해 핸드오프 과정을 거치게 된다. 이동 호스트가 하나의 셀에 고정되어 움직이지 않는다면 메시지 로깅과 체크포인팅은 기존의 기법들을 그대로 사용하여도 문제가 발생하지 않지만 이동 호스트의 이동성에 의한 핸드오프 때문에 로그와 체크포인트의 관리 문제가 발생한다. 만약 이동 호스트가 메시지 로깅과 체크포인팅을 현재 위치하고 있는 기지국에서 수행한 후 다른 기지국의 셀 영역으로 핸드오프 하여 작업을 계속 수행하다 결함이 발생하면 현재의 기지국에는 자신의 로그와 체크포인트가 저장되어 있지 않기 때문에 결함에 대한 복구를 할 수 없게 된다. 그러므로 이동 컴퓨팅 환경에서 메시지 로깅과 체크포인팅 기법을 적용하기 위해서는 이동 호스트의 핸드오프에 의해 발생하는 로그와 체크포인트 관리 문제를 해결하여야 한다.

본 논문에서 제안하는 Redundant Lazy 결함 복구 기법은 Lazy 기법의 단점을 보완한 결함 복구 기법이다[16]. Lazy 기법은 (그림 1)과 같이 이동 호스트가 쓰기 동작이나 체크포인팅 주기 사이에 핸드오프를 수

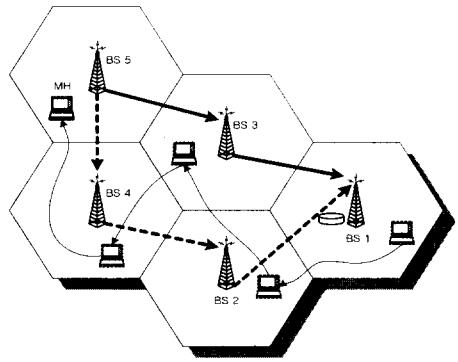
행하면 방문하였던 기지국의 연결 리스트를 생성하여 자신의 로그나 체크포인트가 저장된 기지국을 확인하는 기법이다. 그러나 Lazy 기법은 연결 리스트로 관리되던 기지국 중 하나의 기지국에 결함이 발생하면 연결 리스트의 가장 끝에 위치한 기지국에 저장되어 있던 자신의 체크포인트를 전송 받을 수 없는 단점을 가지고 있다.



(그림 1) Lazy 기법

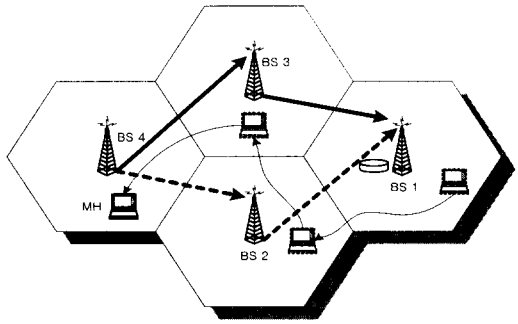
Redundant Lazy 기법의 기본 아이디어는 연결 리스트를 중복하여 만드는 것이다. 연결 리스트의 중복화로 인해 하나의 연결 리스트에 속한 기지국의 결함이 발생하여도 나머지 또 다른 연결 리스트를 통해 이동 호스트의 결함이 발생했을 경우 로그와 체크포인트를 전송 받을 수 있다. (그림 2)는 Redundant Lazy 기법의 예를 보여주고 있다. 그림에서 BS1에서 BS5까지 총 다섯 개의 기지국을 거친 이동 호스트는 로그와 체크포인트가 저장되어 있는 BS1까지의 연결 리스트 두 개를 만들고 기지국은 중복되어 두 개의 연결 리스트에 동시에 삽입되지 않는다. 그러므로 이동 호스트는 하나의 연결 리스트가 결함이 발생하더라도 즉, (기지국의 결함이 발생하더라도) 또 다른 연결 리스트를 통해 로그와 체크포인트를 접근할 수 있다.

연결 리스트를 중복하여 설정하는 방법은 두 가지의 경우로 나누어 볼 수 있다. 다음 장에서 평가할 각 기법의 비용 분석에서 서로 인접해 있는 기지국들간의 네트워크의 사용 비용이 전체 각 기법의 비용 분석에 포함되는데 서로 인접해 있지 않은 기지국들은, 예를 들어 (그림 2)에서 BS5와 BS1사이의 네트워크 사용 비용은 바로 계산될 수 없고 BS3을 거쳐 계산하여야 한다. 즉 BS5의 인접 기지국 BS3과의 네트워크 사용 비용과 BS3과 BS1과의 네트워크 사용 비용의 합이 BS5와 BS1 사이의 네트워크 사용 비용이 된다. 이와 같은 제한에 의해 연결 리스트의 포인터는 서로 인접해 있는 기지국들 사이에만 설정할 수 있다.



(그림 2) Redundant Lazy 기법

연결 리스트를 설정하는 첫 번째 경우는 (그림 3)과 같이 이동 호스트가 이동한 셀의 인접 셀 중 이전에 방문하였던 셀이 두 개인 경우이다. 이와 같은 경우에는 현재 셀의 기지국이 이전 두 셀의 기지국에 각 포인터를 설정함으로써 중복된 연결 리스트를 생성할 수 있다.

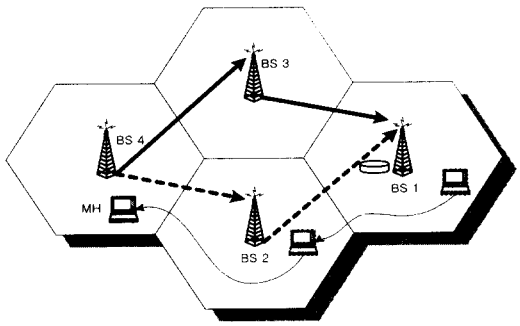


(그림 3) 이동 호스트가 이동한 셀의 인접 셀 중 이전에 방문하였던 셀이 두 개일 경우

두 번째 경우는 이동 호스트가 이동한 셀의 인접 셀 중 이전에 방문하였던 셀이 하나일 경우이다. 이와 같은 경우는 포인터를 두 개 이상의 셀을 거쳐 설정할 수 없기 때문에 (그림 4)와 같이 현재 셀과 바로 이전에 방문하였던 셀에 둘 다 인접해 있는 하나의 셀을 선택한다. 그리고 첫 번째 연결 리스트의 포인터는 이전에 방문하였던 셀의 기지국으로 설정하고 또 다른

연결 리스트의 포인터는 앞의 조건을 만족하는 셀의 기지국으로 설정한다.

이와 같이 연결 리스트를 중복화 함으로서 이동 호스트는 하나의 연결 리스트가 결함이 발생하더라도 자신의 결함 발생 시 로그와 체크포인트를 통해 복구할 수 있게 된다.



(그림 4) 이동 호스트가 이동한 셀의 인접 셀 중 이전에 방문하였던 셀이 한 개일 경우

5. 성능평가

이 장에서는 4장에서 제안된 Redundant Lazy 결함 복구 기법을 Lazy 결함 복구 기법과 비교하여 비용 분석과 신뢰도를 평가한다. 여기에서의 비용 분석이란 각 기법에 따라 이동 호스트가 핸드오프와 핸드오프 사이에 메시지 로깅과 체크포인팅을 하는 데 소요되는 비용과 이동 호스트의 결함 발생 시 소요되는 복구 비용의 합을 계산하는 과정을 의미한다. 그리고 비용이란 이동 호스트의 네트워크(무선 네트워크 + 유선 네트워크) 사용량을 의미한다.

본 논문에서는 두 가지의 결함 복구 기법을 3장의 프로세스 상태 저장 기법과 조합하여 다음과 같이 총 네 가지 기법의 비용 분석을 수행한다.

(1) No logging Redundant Lazy 기법
(2) No logging Lazy 기법
(3) Logging Redundant Lazy 기법
(4) Logging Lazy 기법

네 가지 기법의 비용 분석을 하기 위해 <표 1>과 같이 용어와 기호들을 정의한다.

〈표 1〉 용어와 기호

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.		
λ_m	이동 호스트의 결함 발생율	λ_b	기지국의 결함 발생율
μ	이동 호스트의 핸드오프율	r	핸드오프 당 쓰기 동작의 기대 값으로 $1/\mu$과 같다.
ρ	쓰기 동작 중 사용자 입력의 비율	T_c	체크포인트 주기
k	체크포인트 당 쓰기 동작의 수	$N_c(T)$	시간 t에서 체크포인트의 수
$N_l(T)$	시간 t에서 로깅된 메시지의 수	C_c	서로 인접해 있는 기지국들끼리 체크포인트를 전송하는 평균 비용
C_l	서로 인접해 있는 기지국들끼리 어플리케이션 메시지를 전송하는 평균 비용	γ	C_l/C_c, 서로 인접해 있는 기지국들끼리 체크포인트를 전송하는 비용에 대한 어플리케이션 메시지를 전송하는 비용의 비율
C_m	서로 인접해 있는 기지국들끼리 컨트롤 메시지를 전송하는 평균 비용	ε	C_m/C_c, 서로 인접해 있는 기지국들끼리 체크포인트를 전송하는 비용에 대한 컨트롤 메시지를 전송하는 비용의 비율
C_h	핸드오프 동작의 평균 비용	C_r	이동 호스트의 복구 비용
C_r^f	Redundant Lazy 기법에서 두 번째 연결 리스트를 사용하여 복구하는 비용	C_t	이동 호스트가 핸드오프를 수행하는데 드는 총 비용

5.1 모델링

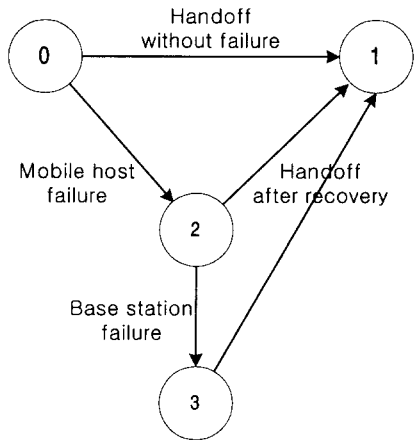
비용을 분석하기 위해 두 개의 연속되는 핸드오프 사이의 시간 간격을 핸드오프 간격이라고 정의를 하고 핸드오프 간격은 (그림 5)와 같은 상태 전이도(state transition diagram)로 표현할 수 있다.

(그림 5)에서 핸드오프 간격 중 이동 호스트는 쓰기 동작을 받을 수도 있고 일정 주기마다 체크포인트를 수행할 수도 있다. 상태 0에서 상태 1로 가는 전이는 핸드오프 간격 중에 이동 호스트의 결함이 발생하지

않은 경우를 표현한 것이다. 만약 핸드오프 간격 중 이동 호스트의 결함이 발생한다면 상태 0에서 상태 2로 상태 전이가 발생한다. 상태 2에서 이동 호스트의 복구를 수행하고 핸드오프가 일어나면 상태 1로 다시 전이된다.

Lazy 기법에서는 연결 리스트에 포함된 기지국의 결함이 발생하면 상태 2에서 이동 호스트의 복구를 수행할 수 없기 때문에 핸드오프가 수행될 때 상태 1로 전이할 수가 없게 된다. 그러나 Redundant Lazy 기법은 첫 번째 연결 리스트에 포함된 기지국의 결함이 발생하면 상태 2에서 상태 3으로 전이되어 두 번째 연결 리스트를 사용하여 이동 호스트의 복구를 수행하게 되고 핸드오프가 일어나면 상태 1로 전이된다.

상태 a에서 상태 b로의 상태 전이 (a, b)는 전이 확률 P_{ab} 및 비용 C_{ab}와 관련이 있다. C_{ab}는 상태 b로 전이가 일어나기 전까지 상태 a에서 머무른 시간동안 발생한 동작의 예상되는 총 비용을 의미한다.



(그림 5) Redundant Lazy 기법의 상태 전이도

전이 확률 P_{02}는 핸드오프 간격 내에서 이동 호스트의 결함이 발생할 확률로 다음과 같이 정의된다.

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

그리고 체크포인트 간격에서 체크포인트가 일어난 시점부터 다음 체크포인트가 일어나기 전에 이동 호스트의 결함이 발생할 때까지 예상되는 존속 시간 T_{cexp}는 다음과 같이 계산할 수 있다[1].

$$T_{cexp}=\int_0^{Tc}\frac{t\lambda e^{-\lambda t}}{1-e^{-\lambda T_c}}dt=\frac{1}{\lambda}-\frac{T_c e^{-\lambda T_c}}{1-e^{-\lambda T_c}}$$

상태 전이 (0, 1)의 비용 C_{01}은 상태 0에서 상태 1로 가기 전의 상태 0에서 머무른 시간동안 발생한 동작의 예상되는 총 비용으로 다음과 같이 계산할 수 있다[1].

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

여기서 $\alpha C_c\times N_c(T)$ 항은 체크포인트를 무선 링크를 통해 현재 기지국으로 보내는 비용이며, $\alpha C_l\times N_l(T)$ 항은 메시지 로그를 기지국으로 보내는 비용이다.

그러므로 이동 호스트가 핸드오프 간격 사이에서 발생시키는 총 비용 C_t는 핸드오프를 수행하는데 드는 총 비용 C_{01}과 결함 발생시 복구 비용 C_r를 합한 것으로 다음과 같이 계산된다.

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

본 논문은 네 가지 기법의 비용 분석으로 각 기법의 총 비용 C_t를 구함으로써 각 기법을 비교 평가한다.

5.2 No Logging Redundant Lazy 기법

No Logging Redundant Lazy 기법은 연결 리스트에 포함된 기지국의 결함까지 대처하기 때문에 기지국의 결함 발생율도 함께 고려하여야 한다. (그림 5)에서 상태 2에서 상태 3으로 가는 기지국의 결함 확률 P_{23}은 연결 리스트에 포함된 기지국의 결함 발생율을 고려하여 다음과 같이 계산할 수 있다.

$$P_{23}=\frac{N_h\lambda_b}{N_h\lambda_b+\mu}$$

여기서 N_h는 마지막 체크포인트가 일어나고 이동 호스트의 결함이 발생하기 전까지 발생한 평균 핸드오프의 수로 다음과 같이 계산할 수 있다.

$$N_h=\mu T_{c\exp}$$

핸드오프가 일어나면 기존 연결 리스트를 수정하거나 새로이 생성하여야 하는데 이것은 두 개의 연결 리스트에 현재의 기지국을 추가하는 것으로 단순히 계산할 수 있으며, 이것은 현재 기지국과 이전 두 기지국

들 사이에 컨트롤 메시지만을 보내」는 것으로 계산할 수 있기 때문에 핸드오프 비용 C_h=$2C_m$이 된다.

또한 체크포인트가 일어났을 경우 이전 체크포인트를 삭제하는데 소요되는 비용은 이동한 셀의 개수만큼 컨트롤 메시지를 두 개의 연결리스트에 속한 기지국으로 보내면 되기 때문에 $2N_hC_m$로 계산할 수 있다.

No Logging 기법을 사용하기 때문에 $N_c(t)=r$, $N_l(t)=0$, $C_h=2C_m$이 되고 이것을 식 (1)에 대입하게 되면 비용 C_{01}은 다음과 같이 계산된다.

$$C_{01} = r\alpha C_c + 2N_hC_m + 2C_m$$

이동 호스트의 복구 비용은 체크포인트를 가지고 있는 기지국에 체크포인트 전송 요구를 위한 컨트롤 메시지 전송 비용과 체크포인트 전송 비용으로 계산될 수 있다. 기지국 결함을 고려한 전체 복구 비용 C_R은 다음과 같이 계산할 수 있다.

$$C_R = (1-P_{23})C_r + P_{23}C_r^f$$

여기서 C_r은 기지국의 결함 없이 첫 번째 연결 리스트를 통해 복구되는 경우로 이동 호스트에서 현재 기지국까지의 무선 네트워크와 기지국들 사이의 유선 네트워크를 모두 사용하기 때문에 다음과 같이 계산할 수 있다.

$$C_r = (a+N_h)(C_c + C_m)$$

첫 번째 연결 리스트의 결함이 발생하였을 때만 두 번째 연결 리스트를 통해 이동 호스트가 복구되기 때문에 복구 비용 C_r^f 은 첫 번째 연결 리스트가 결함이 발생했는지를 알아야 하는 비용 δ에 C_r만을 더해 주어 다음과 계산할 수 있다.

$$C_r^f = C_r + \delta$$

여기서 연결 리스트에 포함된 기지국의 결함이 일양분포(uniform distribution)를 가지고 발생한다고 가정하면 δ는 다음과 같이 계산할 수 있다.

$$\delta = \frac{1}{2}N_hC_m$$

그러므로 No Logging Redundant Lazy 기법의 총 비용은 $C_{NoLoggingRedundantLazy}$=$C_{01}$+$P_{02}C_R$이 되고 다음과 같이 계산할 수 있다.

$$C_{NoLoggingRedundantLazy} = \left\{ r\alpha + (\alpha+N_h)\frac{\lambda_m}{\lambda_m+\mu} \right\} C_c +$$

$$\left\{ 2N_h + (\alpha+N_h)\frac{\lambda_m}{\lambda_m+\mu} + \frac{1}{2}\frac{\lambda_m}{\lambda_m+\mu}\frac{N_h\lambda_b}{N_h\lambda_b+\mu}N_h + 2 \right\} C_m$$

5.3 No Logging Lazy 기법

No Logging Lazy 기법의 총 비용 $C_{NoLoggingLazy}$은 다음과 같다[1].

$$C_{NoLoggingLazy} = \left\{ r\alpha + (\alpha+N_h)\frac{\lambda_m}{\lambda_m+\mu} \right\} C_c +$$

$$+ \left\{ N_h + (\alpha+N_h)\frac{\lambda_m}{\lambda_m+\mu} + 1 \right\} C_m$$

5.4 Logging Redundant Lazy 기법

Logging 기법은 체크포인트와 쓰기 동작에 의해 생성되는 로그를 모두 관리하여야 한다. 쓰기 동작의 로깅 비용은 무선 네트워크를 통해 이동 호스트에서 기지국까지의 전송하는 비용만 고려하면 된다. 쓰기 동작의 로깅 중 사용자 입력만이 무선 네트워크를 사용하기 때문에 비용 계산에 포함되고 다른 호스트로부터의 메시지는 먼저 기지국을 거쳐 로깅이 된 후 이동 호스트에게 전달되기 때문에 네트워크 사용 비용엔 포함되지 않는다. 사용자 입력 로깅의 경우 기지국에 로깅을 시도하고 그에 대한 응답을 받아야 하기 때문에 하나의 메시지에 대한 응답 비용은 αC_m이 되고 로깅 비용은 αC_l이 된다. 그리고 쓰기 동작 중 사용자 입력에 대한 비율이 ρ이므로 핸드오프 간격 사이의 사용자 입력의 수는 ρr이 된다.

Logging Redundant Lazy 기법의 핸드오프 동작의 비용 C_h는 5.2절에서의 핸드오프 동작 비용과 같다. 즉, 연결 리스트를 설정하는 비용은 $2C_m$이 된다. Logging 기법이기 때문에 $N_c(T)=k/r$, $N_l(T)=\rho r$이 되고 비용 C_{01}은 다음과 같이 앞의 값을 식 (1)에 대입하여 계산할 수 있다.

$$C_{01} = \frac{r}{k}\alpha C_c + \rho r\alpha C_l + \rho r\alpha C_m + 2N_hC_m + 2C_m$$

이동 호스트의 결함으로부터 소요되는 복구 비용도 4.2절에서 계산한 것과 같이 $C_R = (1-P_{23})C_r + P_{23}C_r^f$ 로 계산할 수 있다. 첫 번째 연결 리스트를 사용하여 복구하는 복구 비용 C_r은 4.2절에서 구한 비용에 로그를 전송 받는 비용만을 추가로 고려하면 다음과 같이 계산할 수 있다.

$$C_r = (\alpha + N_h)(C_c + \nu C_l + C_m)$$

여기서 ν는 핸드오프 동작이 포아송 분포를 따른다고 가정하면 $\nu=(k-1)/2$이 된다[1].

그러므로 Logging Redundant Lazy 기법의 총 비용은 $C_{LoggingRedundantLazy}=C_{01}+P_{02}C_R$이 되고 다음과 같이 계산할 수 있다.

$$C_{LoggingRedundantLazy} = \left\{\frac{r}{k}\alpha + (\alpha+N_h)\frac{\lambda_m}{\lambda_m+\mu}\right\}C_c + \left\{\rho r\alpha + \nu(\alpha+N_h)\frac{\lambda_m}{\lambda_m+\mu}\right\}C_l + \left\{\rho r\alpha + 2N_h + (\alpha+N_h)\frac{\lambda_m}{\lambda_m+\mu} + \frac{1}{2}\frac{\lambda_m}{\lambda_m+\mu}\frac{N_h\lambda_b}{N_h\lambda_b+\mu}N_h + 2\right\}C_m$$

5.5 Logging Lazy 기법

Logging Lazy 기법의 총 비용 $C_{LoggingLazy}$는 다음과 같다[1].

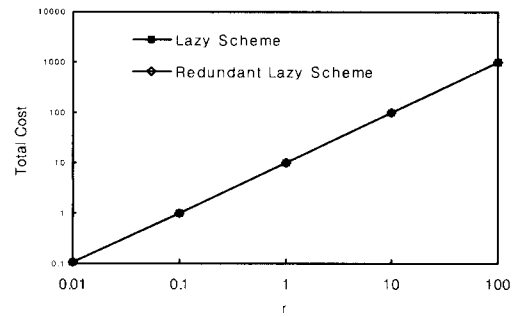
$$C_{LoggingLazy} = \left\{\frac{r}{k}\alpha + (\alpha+N_h)\frac{\lambda_m}{\lambda_m+\mu}\right\}C_c \cdot + \left\{\rho r\alpha + \nu(\alpha+N_h)\frac{\lambda_m}{\lambda_m+\mu}\right\}C_l \cdot + \left\{\rho r\alpha + N_h + (\alpha+N_h)\frac{\lambda_m}{\lambda_m+\mu} + 1\right\}C_m$$

5.6 비용 분석 결과

각 기법의 비용 공식을 비교하기 위해 먼저 각 기법의 총 비용을 C_c에 대해 정규화(normalization)를 한다. $C_l=\gamma C_c$, $C_m=\varepsilon C_c$, $C_c=1$이기 때문에 $C_l=\gamma$, $C_m=\varepsilon$이 된다. 비용 분석을 위해 각 파라미터를 다음과 같이 가정하였다.

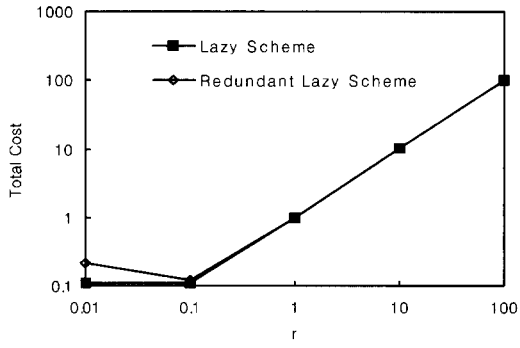
$$\gamma = C_l = 0.1,\ \varepsilon = C_m = 10^{-4},\ \rho = 0.5,\ \alpha = 10$$

(그림 6)은 이동 호스트의 결함 발생율 $\lambda_m=10^{-5}$이고 기지국의 결함 발생율 $\lambda_b=10^{-7}$일 때 r을 0.01에서 100까지 변화 시켰을 때의 No Logging 기법을 사용한 두 가지 기법들의 총 비용을 보여주고 있다. 여기서 $r=1/\mu$이기 때문에 r이 작을수록 이동 호스트의 이동성이 커지게 되고 핸드오프 비율이 많아지게 된다.



(그림 6) No Logging 기법을 사용하는 두 가지 기법의 총 비용

(그림 7)은 이동 호스트의 결함 발생율 $\lambda_m=10^{-5}$이고 기지국의 결함 발생율 $\lambda_b=10^{-7}$일 때 r을 0.01에서 100까지 변화 시켰을 때의 Logging 기법을 사용한 두 가지 기법들의 총 비용을 보여주고 있다. Logging 기법은 일정한 주기로 체크포인트를 수행하는데 여기서는 그 주기 T_c=20으로 설정하였다.



(그림 7) Logging 기법을 사용하는 두 가지 기법의 총 비용

(그림 6)과 (그림 7)에서 볼 수 있듯이 r의 값이 커질수록 총 비용이 늘어나는 것을 볼 수 있다. 그리고 기지국의 결함 발생을 고려한 Redundant Lazy 기법은 Lazy 기법과 비교하여 거의 차이가 없다는 것을 알 수 있다. 즉 기지국의 결함을 허용하더라도 비용면에서는 큰 차이가 없으므로 Redundant Lazy 기법이 Lazy 기법과 비교해서 효율적이라는 것을 알 수 있다.

5.7 신뢰도 분석

본 절에서는 기지국의 결함 발생에 따른 Lazy 기법과 Redundant Lazy 기법의 신뢰도를 비교한다. 신뢰도란 시스템이 t_0에서 정확하게 동작한다고 가정할 때, (t_0, t) 시간동안에 시스템이 정확하게 동작할 확률을 말한다[10, 11]. 신뢰도 모델링을 위해 몇 가지 가정이 필요한데 본 논문에서는 다음과 같이 가정한다.

(1) 결함 복구는 고려하지 않는다.
(2) 한 번에 단지 하나의 결함이 발생한다.
(3) 시스템은 완벽한 상태에서 시작한다.

조합 모델(combinatorial model)을 이용한 결함 허용 기능이 없는 단일 시스템의 신뢰도는 $R(t)=e^{\lambda t}$로 구할 수 있고 여기서 λ는 결함 발생율이다. 그러므로 두 가지 기법의 신뢰도는 연결 리스트의 결함 발생율로부터 계산할 수 있으며 연결 리스트의 결함 발생율 τ는 기지국의 결함 발생율 λ_b에 의해 다음과 같이 결정할 수 있다.

$$\tau = N_h \lambda_b$$

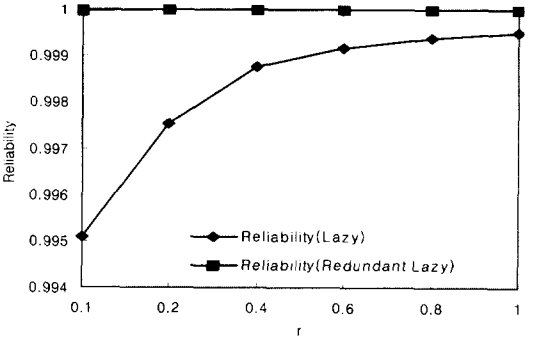
Lazy 기법은 결함 허용 기법을 사용하지 않은 단일 모듈로 볼 수 있으며 신뢰도 R_{Lazy}는 다음과 같이 계산할 수 있다[10, 11].

$$R_{Lazy} = e^{-\pi}$$

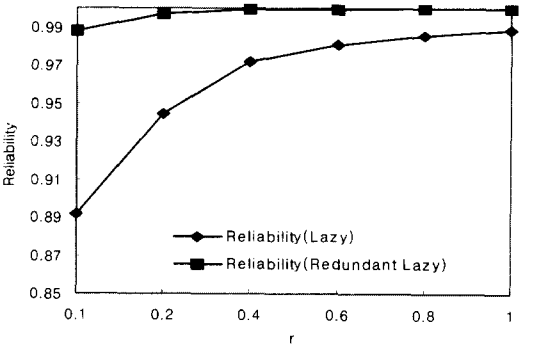
Redundant Lazy 기법은 2스페어 모듈로 볼 수 있으며 신뢰도 $R_{RedundantLazy}$는 다음과 같이 계산할 수 있다[10, 11].

$$R_{RedundantLazy} = 2e^{-\pi} - e^{-2\pi}$$

체크포인트 간격이 $T_c=10$일 경우와 $T_c=50$일 경우 r값의 변화에 따른 두 기법의 신뢰도변화를 (그림 8)와 (그림 9)에서 보여주고 있다. r값이 작을수록 이동 호스트의 이동성이 커지기 때문에 N_h값이 커지고 연결 리스트의 결함 발생율이 높아져 신뢰도는 떨어지는 것을 볼 수 있다. 마찬가지로 체크포인트 간격이 클수록 N_h의 값이 커지게 됨으로 신뢰도가 떨어지는 것을 볼 수 있다.



(그림 8) 신뢰도 ($T_c=10$)



(그림 9) 신뢰도 ($T_c=50$)

6. 결 론

컴퓨터 기술과 통신 기술의 발달은 이동 컴퓨팅을 점차 일반화시키고 있으나 이동 컴퓨팅은 정적 컴퓨팅 환경에서 볼 수 없는 새로운 결함 유형들을 가지고 있어 기존의 결함 허용 기법들을 그대로 적용하기에는 무리가 따르므로 이동 컴퓨팅에서의 새로운 결함 허용 기법들이 연구되어지고 있다.

본 논문에서 제안한 Redundant Lazy 기법은 이동 호스트의 로그나 체크포인트를 기지국에 저장하여 이

동 호스트의 결함 발생시 이 것을 통해 복구할 수 있고 연결 리스트에 포함되는 기지국의 결함이 발생하더라도 이동 호스트는 결함 발생으로부터 복구를 수행할 수 있다. 따라서 이전에 Pradhan에 의해 제안되었던 Lazy 기법과 비교해 비용면에서는 큰 차이를 보이지 않으면서 Lazy 기법의 약점을 극복하였으며 신뢰도 분석을 통하여 Redundant Lazy 기법이 Lazy 기법보다 더 높은 신뢰도를 가진다는 사실을 검증하였다. 그러나 본 논문에서는 단순히 연결 리스트의 결함 발생율을 통해 신뢰도를 분석하였으므로 향후 좀 더 세밀한 신뢰도 분석이 필요하다고 본다.

참 고 문 헌

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

[2] 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.

[3] 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.

[4] 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.

[5] E.N. Elmagarmid and W. Zwuenepoel, and W. Manetho, "Transparent Rollback Recovery with Low Overhead, Limited Rollback and Fast Output Commit," IEEE Computer, Vol.27, No.6, pp.38-47, April 1994.

[6] D.B. Johnson and W. Zwuenepoel, "Recovery in Distributed Systems using Optimistic Message Logging and Checkpointing," Journal of Algorithms, Vol.11, No.3, pp.462-491, September 1990.

[7] R.E. Strom and S. Yemini, "Optimistic Recovery in Distributed Systems, ACM Transactions on Computer Systems," Vol.3, No.3, 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.

[10] D.K. Pradhan, 'Fault-Tolerant Computer System Design,' Prentice-Hall Press, 1996.

[11] B.W. Johnson, 'Design and Analysis of Fault-Tolerant Digital Systems,' Addison-Wesley Press, 1989.

[12] 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.

[13] H. Higaki and M. Takizawa, "Checkpoint-Recovery Protocol for Reliable Mobile Systems," Proceedings of the 17th Symposium on Reliable distributed Systems, pp.93-99, October 1998.

[14] G. Cao and M. Singhal, "Low-Cost Checkpointing with Mutable Checkpoints in Mobile Computing Systems," Proceedings of International Conference on Distributed Computing System, pp.462-471, May 1998.

[15] E. Pitoura and G. Samaras, 'Data Management for Mobile Computing,' Kluwer Academic Publishers, 1998.

[16] W. Kwon and S. Kim, "Failure Recovery of Mobile Host in Mobile Environments," Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications, pp.2072-2077, June 2000.

권 원 석

e-mail : wonseok@madang.ajou.ac.kr

1999년 아주대학교 정보 및 컴퓨터공학부(공학사)

1999년~현재 아주대학교 BK21 정보통신전문대학원 석사과정

관심분야 : 이동컴퓨팅, 결함허용, 스토리지 시스템 등

김 성 수

e-mail : sskim@madang.ajou.ac.kr

1982년 서강대학교 전자공학과 (공학사)

1984년 서강대학교 전자공학과 (공학석사)

1995년 Texas A&M University, Computer Science Dept. (공학박사)

1983년~1986년 삼성전자(주) 종합연구소 컴퓨터연구실 (주임연구원)

1986년~1996년 삼성종합기술원(수석연구원)

1991년~1992년 Texas Transportation Institute(연구원)

1993년~1995년 Texas A&M University, Computer Science Dept.(T.A. & R.A.)

1997년~1998년 한국정보처리학회, 한국정보과학회 논문지 편집위원

1996년~현재 아주대학교 BK21 정보통신전문대학원 부교수

관심분야 : 클러스터 시스템, 결함허용, 성능 평가, 이동 컴퓨팅, 멀티미디어 등

김 재 훈

e-mail : jaikim@madang.ajou.ac.kr

1984년 서울대학교 제어계측공학과 (공학사)

1993년 Indiana University 전산학과(공학석사)

1997년 Texas A&M University, Computer Science Dept. (공학박사)

1995년~1997년 Texas A&M University, Computer Science Dept.(R.A.)

1984년~1991년 대우통신(주) 컴퓨터 연구실

1997년~1998년 삼성전자(주) 컴퓨터시스템팀

1998년~현재 아주대학교 BK21 정보통신전문대학원 조교수

관심분야 : 분산시스템, 실시간시스템, 운영체제 등