# 기지국 결함 발생시 효율적인 이동 호스트의 결함 복구 기법

권원석° 김성수
아주대학교 정보통신전문대학원
{wonseok, sskim}@madang.ajou.ac.kr

# An Efficient Recovery Scheme for Mobile Host under Base Station Failure

Wonseok Kwon° Sungsoo Kim
Professional Graduate School of Information and Communication Technology,
Ajou University

## 요 약

이동 컴퓨팅이란 시간과 장소에 구애 받지 않으면서 컴퓨팅을 수행할 수 있는 것을 말한다. 현재 다양한 사용자의 요구를 수용하기 위하여 이동 컴퓨팅에 관한 연구가 세계적으로 활발히 진행되고 있다. 이동 컴퓨팅은 이동 호스트 자체의 결함, 네트워크 연결 단절, 무선 링크의 결함 등의 기존 유선 네트워크에서는 찾아볼 수 없는 새로운 결함 원인들을 포함하고 있다. 그러나 현재 이동 컴퓨팅 연구 중에서 이러한 이동 호스트의 결함을 효율적으로 대처하는 결함 허용 기법에 관한 연구는 미미한 실정이다. 이에 본 논문에서는 기지국의 결함이 발생하여도 이동 호스트가 복구 가능하도록 하는 Redundant Lazy 기법을 제안하고 성능을 분석한다.

## 1. 서론

이동 컴퓨팅 환경은 일반적으로 네트워크 연결을 유지하면서 자유롭게 이동할 수 있는 이동 호스트(mobile host)와 이동 호스트가 유선 네트워크(wire network)에 접근할 수 있도록 무선 네트워크(wireless network)와 유선 네트워크 사이의 무선 인터페이스를 제공하는 기지국(base station) 등으로 구성되어 있다. 이동 호스트의 송수신기는 지리적으로 제한된 범위 내에서 데이터를 주고 받을 수 있기 때문에 기지국이 이러한 범위 내에서 이동 호스트에게 통신을 제공할 수 있도록 하는 영역을 셀(cell)이라고 하며 이동 호스트는 셀 내에서만 통신이 가능하고 호스트의 이동성 때문에 시간에 따라 여러 개의 서로 다른 셀 사이를 이동할 수 있다. 이렇게 두 셀 사이를 이동하는 것을 핸드오프(handoff)라고 하며 핸드오프 과정은 사용자가 서비스를 받는 기지국이 변경되었다는 것을 인식하지 못하도록 하는 투명성을 제공하여야 한다.

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 복구할 수 있는 결함 허용 기법 중 하나이다. 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]. 두 번째로 이동 호스트가 하나의 셀에 고정되어 위치하고 있다면 기존의 기법들을 그대로 수정없이 사용할 수 있으나 이동 호스트는 그 자체의 이동성 때문에 여러 개의 셀들을 옮겨 다니는 핸드오프를 수행함으로 기지국에 저장되어 있는 로그나 체크포인트를 관리하기가 어렵게 된다.

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

This work is supported in part by the Ministry of Education of Korea (Brain Korea 21 Project supervised by Korea Research Foundation).

## 2. 관련 연구

Pradhan[1]은 두 가지의 체크포인팅 프로토콜을 제안하였다. Pradhan은 두 가지의 체크포인팅 프로토콜을 세 가지의 서로 다른 핸드오프 기법과 조합하여 총 여섯 가지의 기법을 제안하고 각 제안 기법들의 비용 분석을 수행하였다.

Neves[3]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[2]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.

## 3. 이동 호스트의 결함 복구 기법

본 장에서는 Pradhan이 이동 컴퓨팅 환경에서 이동 호스트의 결함 허용을 위해 제안한 두 가지의 프로세스 상태 저장 기법과 Lazy 핸드오프 기법을 소개하고 문제점을 설명한다. 그리고 Lazy 기법의 문제점을 보완하는 새로운 핸드오프 기법인 Redundant Lazy 기법을 제안한다.

### 3.1 프로세스 상태 저장 기법

본 논문에서 사용하는 프로세스의 상태를 저장하는 두 가지의 기법은 다음과 같다. 첫 번째는 No Logging 기법으로 이동 호스트는 따로 일정한 주기에 맞춰 체크포인팅을 수행하지 않고 프로세스의 상태가 변하는 쓰기 동작이 일어날 때 마다 체크포인팅을 수행하는 기법이다. 프로세스의 상태를 변화시키는 쓰기 동작의 유형은 사용자의 입력과 다른 이동 호스트로부터 들어오는 메시지로 구분할 수 있다. Logging 기법은 이동 호스트가 주기적으로 프로세스를 체크포인팅하고 쓰기 동작이 발생할 때 마다 해당 동작을 로깅하는 기법이다.

### 3.2 핸드오프 기법

#### 3.2.1 Lazy 기법[1]

Lazy 기법은 이동 호스트가 쓰기 동작이나 체크포인팅 주기 사이에 핸드오프를 수행하면 방문하였던 기지국의 연결 리스트를 생성하여 자신의 로그와 체크포인트가 저장된 기지국을 확인하는 방법이다.

그러나 이 기법은 불필요하게 현재 이동 호스트 자신이 위치하고 있지 않은 셀의 기지국에 자신의 로그와 체크포인트가 저장되어 있는 단점이 있다. 이 문제는 이동 호스트의 쓰기 동작이나 체크포인팅이 일어날 때 마다 현재 셀의 기지국에 로그와 체크포인트를 저장하고 이전까지 연결 리스트로 관리하던 자신의 로그와 체크포인트는 해당 기지국에 메시지를 보내 삭제시켜 해결할 수 있다.

이 기법은 연결 리스트로 관리되던 기지국 중 하나의 기지국에 결

함이 발생하면 연결 리스트의 가장 끝에 위치한 기지국에 저장되어 있던 자신의 체크포인트를 전송 받을 수 없는 단점을 가지고 있다.

*3.2.2 Redundant Lazy 기법*

Redundant Lazy 기법은 Lazy 기법의 치명적인 약점인 연결 리스트로 관리되던 로그와 체크포인트를 연결 리스트에 속한 하나의 기지국에서 결함이 발생했을 경우 사용할 수 없다는 것을 보완한 방법이다. 이 기법의 기본 아이디어는 연결 리스트를 중복하여 만드는 것이다. 연결 리스트의 중복으로 인해 하나의 연결 리스트에 속한 기지국의 결함이 발생하여도 나머지 또 다른 연결 리스트를 통해 이동 호스트의 결함이 발생했을 경우에도 로그와 체크포인트를 전송 받을 수 있다.

연결 리스트를 중복하여 설정하는 방법은 두 가지의 경우로 나누어 볼 수 있다.

연결 리스트를 설정하는 첫 번째 경우는 이동 호스트가 이동한 셀의 인접 셀 중 이전에 방문하였던 셀이 두 개가 있을 경우이다. 이와 같은 경우에는 현재 셀의 기지국이 이전 두 셀의 기지국에 각 포인터를 설정함으로써 중복된 연결 리스트를 생성할 수 있다.

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 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 함으로 이동 호스트는 하나의 연결 리스트가 결함이 발생하더라도 자신의 결함 발생 시 로그와 체크포인트를 통해 복구할 수 있게 된다.

## 4. 성능평가

이 장에서는 3장에서 제안된 각 핸드오프 기법들의 비용 분석과 신뢰도를 평가한다. 여기에서의 비용 분석이란 이동 호스트가 핸드오프와 핸드오프 사이에 메시지 로깅과 체크포인팅을 하는데 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 핸드오프 기법을 3.1절의 프로세스 상태 저장 기법과 조합하여 총 네 가지 기법의 비용 분석을 수행한다.

네 가지 기법의 비용 분석을 하기 위해 다음과 같은 용어와 기호들을 정의한다.

- $\alpha$ : 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 비율
- $\lambda_m$ : 이동 호스트의 결함 발생율, $\lambda_b$ : 기지국의 결함 발생율
- $\mu$ : 이동 호스트의 핸드오프율
- $r$ : 핸드오프 당 쓰기 동작의 기대 값으로 $1/\mu$과 같다.
- $\rho$ : 쓰기 동작 중 사용자 입력의 비율, $T_c$ : 체크포인트 주기
- $k$ : 체크포인트 당 쓰기 동작의 수
- $N_c(T)$ : 시간 $t$에서 체크포인트의 수
- $N_l(T)$ : 시간 $t$에서 로그된 메시지의 수
- $C_c$ : 기지국들끼리 체크포인트를 전송하는 평균 비용
- $C_l$ : 기지국들끼리 어플리케이션 메시지를 전송하는 평균 비용
- $\gamma$ : $C_l/C_c$, 기지국들끼리 체크포인트를 전송하는 비용에 대한 어플리케이션 메시지를 전송하는 비용의 비율
- $C_m$ : 기지국들끼리 컨트롤 메시지를 전송하는 평균 비용
- $\varepsilon$ : $C_m/C_c$, 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

### 4.1 모델링

비용을 분석하기 위해 두 개의 연속되는 핸드오프 사이의 시간 간격을 핸드오프 간격이라고 정의하고 핸드오프 간격은 그림 1과 같은 상태 전이도(state transition diagram)로 표현할 수 있다.

그림 1은 Lazy 기법에서 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 상태 2에서 기지국의 결함이 발생하면 Redundant Lazy 기법은 상태 3으로 전이되어 두 번째 연결 리스트를 사용하여 이동 호스트의 복구를 수행하게 되고 핸드오프가 일어나면 상태 1로 전이된다.

전이 확률 $p_{02}$는 핸드오프 간격 내에서 이동 호스트의 결함이 발생할 확률로 다음과 같이 정의 된다[1].

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

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

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

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

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

그러므로 이동 호스트의 결함에 관계없이 핸드오프 간격 사이에서 발생할 수 있는 총 비용은 다음과 같이 계산된다.

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

본 논문은 네 가지 기법의 비용 분석으로 각 기법의 총 비용 $C_t$를 구함으로써 각 기법을 비교 평가한다.

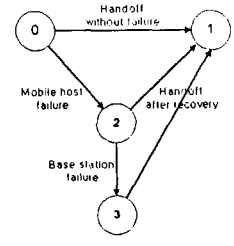


그림 1 상태 전이도

### 4.2 No Logging Lazy 기법[1]

No Logging Lazy 기법의 총 비용 $C_{NoLoggingLazy}$은 다음과 같이 계산할 수 있다.

$$C_{NoLoggingLazy} = \left\{ r\alpha + (\alpha + N_h)\frac{\lambda_m}{\lambda_m + \mu} \right\} C_c + \left\{ N_h + (\alpha + N_c)\frac{\lambda_m}{\lambda_m + \mu} + 1 \right\} C_m$$

### 4.3 No Logging Redundant Lazy 기법

No Logging Redundant Lazy 기법은 연결 리스트에 포함된 기지국의 결함까지 대처하기 때문에 기지국의 결함 발생율도 함께 고려하여야 한다. 그림 1의 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 $N_h$는 마지막 체크포인트가 일어나고 이동 호스트의 결함이 발생하기 전까지 발생한 평균 핸드오프의 수행 회수로 이동 호스트가 결함이 발생하기 전까지 이동한 셀의 개수를 의미하고 다음과 같이 계산된다[1].

$$N_h = \mu T_{cexp}$$

비용 $C_{01}$은 No Logging Lazy 기법의 비용과 거의 유사하다[1]. 단지 두 개의 연결 리스트를 설정하는 비용과 체크포인트가 새로 일어났을 경우 이전 체크포인트와 두 개의 연결리스트를 삭제하는 비용만을 고려하면 된다. 비용 $C_{01}$은 다음과 같이 계산될 수 있다.

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

이동 호스트의 복구 비용은 기지국의 결함에 의해 두 번째 연결 리스트를 통해 복구하는 비용도 포함하여야 한다. 전체 복구 비용 $C_R$은 다음과 같이 계산할 수 있다.

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

첫 번째 연결 리스트의 결함이 발생하였을 때만 두 번째 연결 리스트를 통해 이동 호스트가 복구되기 때문에 복구 비용 $C_r^f$는 첫 번째 연결 리스트가 결함이 발생했는지를 알아야 하는 비용 $\delta$에 $C_r$만을 더해 주어 다음과 계산할 수 있다[1].

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

여기서 연결 리스트에 포함된 기지국의 결함이 일양 분포(uniform distribution)를 가지고 발생한다고 가정하면 $\delta$는 다음과 같이 계산할 수 있다.

$$\delta = N_h C_m / 2$$

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

$$C_{NoLoggingRedundantLazy} = \left\{ r\alpha + (\alpha + N_h)\frac{\lambda_m}{\lambda_m + \mu} \right\} C_c + \left\{ 2N_h + (\alpha + N_h)\frac{\lambda_m}{\lambda_m + \mu} + \frac{1}{2}\frac{\lambda_m}{\lambda_m + \mu}\frac{N_h\lambda_b}{N_h\lambda_b + \mu} N_h + 2 \right\} C_m$$

4.4 Logging Lazy 기법[1]

Logging Lazy 기법의 총 비용 $C_{LoggingLazy}$는 다음과 같이 계산할 수 있다.

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

4.5 Logging Redundant Lazy 기법

Logging Redundant Lazy 기법의 비용 $C_{01}$은 두 개의 연결 리스트 설정과 메시지 로깅 비용을 고려하면 다음과 같이 계산할 수 있다.

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

이동 호스트의 결함으로부터 소요되는 복구 비용 $C_r = (\alpha + N_h)(C_c + \nu C_l + C_m)$이고 $C_r' = C_r + \delta$이 된다. 여기서 $\nu'$은 저장되는 로그의 평균 크기로 핸드오프가 포아송(Poisson) 분포를 따른다고 가정하면 $\nu' = (k-1)/2$가 된다.

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

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

4.6 비용 분석 결과

각 기법의 총 비용을 비교하기 위해 먼저 각 기법의 총 비용을 $C_c$에 대해 정규화(normalization)를 한다. 비용 분석을 위해 각 파라미터를 다음과 같이 가정하였다.

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

그림 2는 이동 호스트의 결함 발생율 $\lambda_m = 10^{-5}$이고 기지국의 결함 발생율 $\lambda_b = 10^{-7}$일 때 $r$을 0.01에서 100까지 변화 시켰을 경우의 네 가지 기법들의 총 비용을 보여주고 있다. Logging 기법은 일정한 주기로 체크포인트를 수행하는데 여기서는 그 주기 $T_c = 20$으로 설정하였다.

그림 2에서 볼 수 있듯이 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 비교해 효율적이라는 것을 알 수 있다.

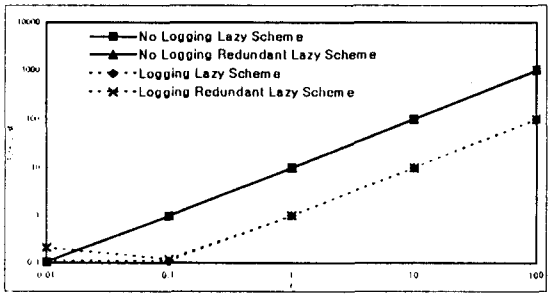


그림 2 네 가지 기법의 총 비용($\lambda_m = 10^{-5}, \lambda_b = 10^{-7}$)

4.7 신뢰도 분석

본 절에서는 기지국의 결함 발생에 따른 Lazy 기법과 Redundant Lazy 기법의 신뢰도를 비교한다. 신뢰도란 시스템이 $t_0$라는 시간에 정확하게 동작한다고 할 때, $(t_0, t)$ 시간동안에 시스템이 정확하게 동작할 확률을 말한다[5].

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

$$\tau = N_h\lambda_b$$

Lazy 기법은 결함 허용 기법을 사용하지 않은 단일 모듈로 볼 수 있으므로 신뢰도 $R_{Lazy}$는 다음과 같이 계산할 수 있다[4].

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

Redundant Lazy 기법은 2스페어 모듈로 볼 수 있으며 신뢰도 $R_{RedundantLazy}$는 다음과 같이 계산할 수 있다[4].

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

기지국 결함 발생율 $\lambda_b = 10^{-5}$이고 체크포인트 간격이 $T_c = 10$일 경우와 $T_c = 50$일 경우, $r$ 값의 변화에 따른 두 기법의 신뢰도 변화를 그림 3에서 보여주고 있다. $r$ 값이 작을수록 이동 호스트의 이동성이 커지기 때문에 $N_h$ 값이 커지고 따라서 연결 리스트의 결함 발생율이 높아져 신뢰도는 떨어지는 것을 볼 수 있다. 마찬가지로 체크포인트 간격이 클수록 $N_h$의 값이 커지게 됨으로 신뢰도가 떨어지는 것을 볼 수 있다.

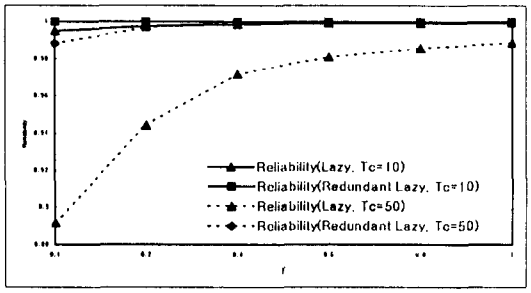


그림 3 Lazy 기법과 Redundant Lazy 기법의 신뢰도

5. 결론

본 논문에서는 결함 허용 기법의 하나인 메시지 로깅과 체크포인팅 기법을 이동 컴퓨팅 환경에서 효율적으로 사용할 수 있는 기법을 제안하였다. 구체적으로 본 논문에서 제안한 Redundant Lazy 기법은 이동 호스트의 로그나 체크포인트를 기지국에 저장하여 이동 호스트의 결함 발생시 이 것을 통해 복구할 수 있다. 제안된 기법에서 사용되는 연결 리스트에 포함되는 기지국의 결함이 발생하더라도 이동 호스트는 결함 발생으로부터 복구를 수행할 수 있다. 그리고 이전에 Pradhan에 의해 제안되었던 Lazy 기법과 비교해 비용면에서는 큰 차이를 보이지 않으면서 Lazy 기법의 약점을 극복하였다. 또한 신뢰도 분석을 통하여 Redundant Lazy 기법이 Lazy 기법보다 더 높은 신뢰도를 가짐을 검증하였다. 그러나 본 논문에서는 단순히 연결 리스트의 결함 발생율을 통해 신뢰도를 분석하였으므로 향후 좀 더 세밀한 신뢰도 분석이 필요하다고 본다.

6. 참고문헌

[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] 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.

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

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