HRing 오버레이 네트워크에서 실패 탐지 및 회복

Failure Detection and Resilience in HRing Overlay Network

구 태 완* 이 광 모**
Tae-Wan Gu Kwang-Mo Lee

요 약

오버레이 네트워크는 물리적 네트워크를 기반으로 하는 가상의 컴퓨터 네트워크로서, 오버레이 네트워크의 노드들은 가상의 링크 또는 논리적 링크로 상호 연결된 것으로 각 링크는 물리적 링크상의 경로와 일치한다. 이러한 오버레이 네트워크는 분산 환경에서 이질적인 자원의 공유에 적합하다. 그러나 오버레이 네트워크는 신뢰적인 통신에 제한이 많으므로 오버레이 네트워크에서의 실패 탐지는 네트워크의 신뢰성을 향상시키는데 중요한 요소이다. 본 논문에서는 분산 환경에서 전형적인 실패 탐지가 가져야 하는 조건들을 살펴보고, HRing 오버레이 네트워크(Hierarchical Ring Overlay Network)에서 수행할 수 있는 실패 탐지와 회복 기법을 제안한다. 제안된 방법은 크게 실패 탐지 단계와 실패 회복 단계로 구분되며, HRing 오버레이가 갖는 특징들을 활용하여 실패 탐지를 수행하기 때문에 불필요한 네트워크 트래픽을 감소시킬 수 있으며 네트워크의 확장성과 유연성을 보장할 수 있다. 본 논문에서는 또한 제안된 실패 탐지 및 회복 기법에 대한 분석과 시뮬레이션을 통한 실험적 평가를 통해 성능을 평가하였다.

Abstract

An overlay network is a virtual network which is constructed on top of a physical computer network. A node in the overlay network is connected through virtual or logical links, where each link corresponds to a path of the links in the underlying physical network. Overlay networks are suitable for sharing heterogeneous resources in distributed environments. However, overlay networks are limited for achieving reliable communication that failure detection in overlay networks is a very important issue. In this paper, we review conditions of conventional failure detection and propose a new approach to failure detection and resilience which can be applied to HRing (Hierarchical Ring) overlay networks. The proposed method consists of the failure detection and the failure resilience phases. Because it utilizes the characteristics of the HRing overlay network for failure detection, it can reduce unnecessary network traffic and provide better scalability and flexibility. We also analyzed and evaluated the performance of the proposed approach through simulations.

☞ keywords : failure detection, resilience, overlay network, Hierarchical ring

1. 서 론

오버레이 네트워크(overlay network)는 물리적 네트워크를 기반으로 하는 네트워크로서, 오버레이 네트워크의 노드들은 가상의 논리적 링크로 상호 연결되어있다[18]. 이러한 오버레이 네트워크는 이질적이고 동적인 대규모 분산 자원 공유에 있어 매우 적합한 환경이다. 그러나 오버레이 네트워크는 신뢰적인 통신을 수행하는데 많은 어려움이 있으므로 오버레이 네트워크에서의 실패 탐지는 네트워크의 신뢰성을 향상시키는데 중요한 요소로 실패 탐지에 관한 많은 연구들이 진행되었다[2, 3, 13]. 오버레이 네트워크에서 사용하는 실패 탐지 기법에는 주로 probing 기법[1, 3, 6, 11, 12, 14]이 사용되며, 이는 heartbeat 메시지를 이용하는 방법이다. Probing 기법은 모든 노드들이 주기적으로 heartbeat 메시지를 전송하여 해당 노드들이 네트워크에 정상적으로 참여하고 있음을 알리는 방법이다. 이러한 heartbeat 메시지 기

* 준 회 원 : 대한상공회의소 강원인력개발원 정보기술과 교수 gutaewan@korcham.net
** 종신회원 : 한림대학교 정보통신공학부 컴퓨터공학 전공 교수 kmlee@hallym.ac.kr
[2007/04/02 투고 - 2007/04/19 심사 - 2007/07/30 심사완료]

반의 실패 탐지는 그룹기반의 오버레이 네트워크에서 모든 노드들은 자신이 소속된 그룹의 리더에게 heartbeat 메시지를 전송하게 되며, 이때 리더 노드는 heartbeat 메시지 수신 여부에 따라 노드의 실패 여부를 결정하게 된다.

또한 실패 탐지 기법은 분산 시스템에서 고장 허용에 관한 문제 중의 하나로 볼 수 있기 때문에 리더의 고장에 따른 문제점을 해결하기 위해 대부분 기본 리더와 백업 리더를 함께 유지하게 된다[8]. 그러나 이러한 리더 기반의 실패 탐지 방식은 주기적으로 송/수신되는 heartbeat 방식으로 인해 리더 노드에서 트래픽 병목 현상이 발생할 수 있을 뿐만 아니라, 그룹의 규모가 커질 경우 리더 노드에서 처리해야 하는 연산의 증가로 인한 오버헤드가 발생하게 된다[9, 13].

본 논문에서는 probing 기반의 실패 탐지 기법의 단점을 보완하고, 실패 탐지 및 회복의 확장성과 신속성을 보장하기 위한 논의를 한다. 이를 위해 Gu 등[4]이 제안한 계층형 링을 기반으로 하는 HRing 오버레이 네트워크(Hierarchical Ring Overlay Network)를 이용한다. HRing오버레이 네트워크는 실패 탐지를 수행함에 있어 고려해야 할 사항에 대한 적절한 대응방법을 제시할 수 있다. 예를 들면 HRing 오버레이 네트워크에서 각 노드는 자신과 인접한 한 개의 노드에게만 heartbeat 메시지를 전송하게 되므로 리더 노드를 따로 선출할 필요가 없으며, 여러 노드로부터 동시에 heartbeat 메시지를 수신하지 않기 때문에 트래픽 병목 현상도 발생하지 않기 때문에 대규모 분산 환경에서 신속하게 노드의 실패에 대응할 수 있다.

그러나 일반적으로 링을 기반으로 하는 실패 탐지의 경우 동시에 발생하는 실패에 대해 탐지가 불가능하다는 단점이 있다. 그러나 본 논문에서 사용되는 HRing의 경우 기본적으로는 heartbeat 메시지를 이용하여 실패 탐지를 처리하지만 링의 어느 한 곳에서 실패가 발생하였을 경우 지역 링에서 주고받는 MCM(Membership Control Message)을 이용하여 여러 노드에서 발생되는 실패에 대해서도 탐지가 가능하게 된다.

본 논문의 구성은 다음과 같다. 1장에서는 오버레이 네트워크의 통신환경이 갖는 특징과 기존의 실패 탐지 기법의 특징에 대해 기술한다. 2장에서는 본 연구에서 사용된 HRing 오버레이 네트워크와 실패 탐지 기법에서 고려하는 사항들에 대해 알아보고, 3장에서는 본 논문에서 제안하는 실패 탐지 및 회복기법의 절차에 대해 기술한다. 그리고 4장에서는 제안된 실패 탐지 및 회복 기법의 성능평가를 위해 고려해야하는 요소들을 3가지로 구분하여 살펴보고, 5장에서는 제안된 실패 탐지 및 회복 기법의 성능 평가를 위한 성능 평가 매트릭을 정의하고, 이를 시뮬레이션한다. 6장에서는 결론 및 본 연구에서 고려해야 하는 향후 연구 과제에 대하여 기술하였다.

2. 관련 연구

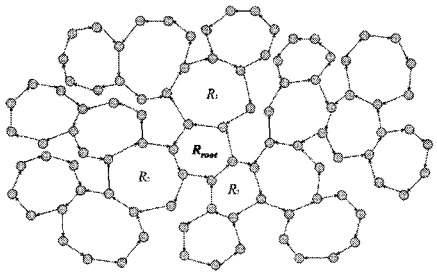
2.1 HRing 오버레이 네트워크

HRing 오버레이 네트워크(hierarchical ring overlay network)는 $G = <V,E>$로 표현하고, 이때 $V = \{v_i | i \in N\}$와 같이 정의하며, N은 오버레이 네트워크에 참여하는 전체 노드의 수가 된다. 그리고 $E = \{<v_i, v_j> | v_i, v_j \in V\}$로 정의한다. 오버레이에 참여하는 노드는 인접한 이웃노드들의 정보를 유지하기 위해 out_{main}, in_{main}, out_{sub}, in_{sub} 연결을 갖는다[4]. 또한 각 노드에서 발생되는 연결 부하에 대한 상한을 설정하기 위해 기본 입출력 연결(out_{main}, in_{main})에 부가적인 연결(out_{sub} 또는 in_{sub})을 포함하여 최대 차수는 3을 넘지 않는다.

HRing 오버레이 네트워크는 링을 기반으로 하는 계층구조를 갖도록 설계되었다. 그림 1은 HRing 오버레이 네트워크의 개념을 보여주고 있다.

계층 구조를 이루기 위한 링의 분할은 분할 임계값 k에 의해 결정되며, 분할 임계값 k는 각 링

에서 허용할 수 있는 최대 RTT(Round-Trip Time)을 의미한다. 이를 통해 각 링에서 전송되는 메시지 전송 시간에 대한 상한을 두어 메시지 전송의 신뢰성을 향상시키는 장점을 갖는다.



(그림 1) HRing 오버레이 네트워크의 개념

각 노드에서는 MCM(Membership Copntrol Message)이라 불리는 메시지를 자신의 out_{main}에게 전송하며, 만일 out_{sub}가 활성화된 경우라면 MCM_{sub}를 생성하여 out_{sub}으로 전송한다. 이것은 각 링에서 자신의 존재를 알리는 heartbeat 메시지와 같은 역할을 수행하게 된다. 이때 각 링에 속한 노드들은 이 MCM을 통해 링에 속한 노드들의 정보를 공유할 수 있으며 만일 노드의 실패가 발생하였을 경우 이를 통해 실패를 탐지할 수 있다.

MCM에 기록된 노드의 정보는 실패 탐지 시 불필요한 메시지 전송을 방지하기 위해 MCM이 전달되는 순서, 즉 링을 구성하고 있는 노드의 순서대로 정렬된다. 그리고 HRing 오버레이 네트워크는 참여하는 모든 노드가 전체 네트워크 정보를 유지하지 않아도 필요한 경우엔 멤버쉽의 변화 즉, 멤버쉽에서 발생하는 이벤트를 인식할 수 있기 때문에 변경된 정보 전송 시간이 작다는 것이 특징이다.

2.2 실패 탐지 시 고려 사항

Hayashibara 등[5]과 Hwang 등[7]은 대규모 그리드 환경에서 실패 탐지를 수행함에 있어 고려해야 할 문제점들에 대해 다음과 같이 논의하였다.

- 메시지 급증*(Message Explosion)* : 실패 탐지수행에서 발생하는 메시지들이 기하급수적으로 증가하지 말아야 한다.
- 확장성*(Scalability)* : 그리드에서 동작하는 응용프로그램들의 수가 증가함에 따라 응용프로그램들에서 사용하는 자원들의 수도 증가하게 된다. 그러므로 대규모 자원들에 대한 실패 탐지도 원활하게 이루어져야 한다.
- 메시지 손실*(Message Loss)* : 네트워크에서 발생하는 메시지들은 네트워크의 상태에 영향을 많이 받게된다. 그러므로 실패 탐지를 수행함에 있어 전송 실패, 네트워크 지연, 케이블에서 발생할 수 있는 잡음 등을 통해 발생할 수 있는 거짓 실패를 구분할 수 있어야 한다.
- 유연성*(Flexibility)* : 시스템에서 발생하는 실패를 탐지함에 있어 사용자의 다양한 요구사항과 서로 다른 응용 프로그램에 대해 유연한 대처를 할 수 있어야 한다.
- 역동성*(Dynamism)* : 일반적으로 그리드 시스템은 매우 동적인 속성을 가진다. 그러므로 이러한 동적인 속성에는 노드의 참여/탈퇴 연산, 네트워크 위상의 변화 등을 신속하게 대응할 수 있어야 한다.
- 보안*(Security)* : 실패 탐지기는 공격자로 오인될 수 있고 최악의 시간에 항상 실패를 의심할 수도 있기 때문에 DoS(Denial of Service) 공격에 사용될 수 있다. 결국 실패 탐지기는 자신이 이와 같은 공격자로 오인받지 말아야 하며, 적절한 시간에 실패를 의심할 수 있는 기술이 필요하게 된다.

2.3 기존의 실패 탐지 기법

네트워크에서 전통적인 실패 탐지 기법은 2가

지 방법으로 구분한다. 첫째 Pull 모델은 실패를 탐지하는 모니터 프로그램이 대상 노드에게 생존(liveness) 메시지를 전송하여 실패가 발생하였는지를 감지하는 모델이며, 둘째 Push 모델은 리더(leader) 기반의 실패 탐지로서 네트워크에 참여하는 노드가 리더 노드에게 heartbeat 메시지를 전송하여 노드에 실패가 발생하지 않았음을 알리는 방식이다[2]. Push 모델의 대표적인 예로서 Globus Toolkit의 실패 탐지 서비스가 있다[13]. Stelling 등[13]은 중앙 집중형 모델로서 실패를 탐지하는 시간을 개선한다는 장점은 있으나 각 노드들이 각 리더에게 heartbeat 메시지를 브로드캐스트하기 때문에 네트워크에서 발생하는 메시지가 기하급수적으로 증가한다는 단점에 따라 확장성에 문제가 있다[5].

Renesse 등[10], Gupta 등[3], Sistla 등[15]은 Gossip 기반의 실패 탐지 기법을 제안하였다. Renesse 등[10]이 제안한 실패 탐지 기법은 multi-level gossiping을 사용하여 확장성 문제를 해결하였으며 네트워크 구조가 동적으로 변화하는 환경에 유연하게 대응할 수 있다는 장점을 가진다. 또한 메시지 손실률에 따른 실패 탐지 시간도 완만하게 증가함을 보여주고 있다. Multi-level gossiping에서는 인터넷 도메인과 서브넷을 이용하여 이들을 각기 서로 다른 레벨로 매핑하는 구조를 갖는다. 또한 서브넷에서 대부분의 goissip 메시지들은 일반적인 gossip 형식으로 전송되고 다른 서브넷으로는 거의 메시지가 전송되지 않는다. 하지만 gossip 메시지들에 의해 노드의 실패를 탐지 하는데 많은 시간이 소비 된다는 단점을 가진다. 또한 트리 구조와 같은 계층 구조에서는 비효율적인 메시지 전송과 실패 탐지 성능을 갖는다[5].

3. 제안하는 실패 탐지 및 회복 기법

앞에서 기술한 바와 같이 기존의 리더기반의 실패 탐지는 리더 선출에 따른 부하와 모든 heartbeat 메시지가 리더에게 집중됨으로써 리더의 부하가 증가하게 된다. 그러나 HRing 오버레이 네트워크에서는 서로 인접한 노드에게만 heartbeat 메시지를 전송하기 때문에 리더에게서 발생할 수 있는 이러한 문제를 해결할 수 있다. 다시 말해 실패 탐지를 위한 메시지 전송에 있어 불필요한 메시지의 증가를 방지 할 수 있다. 이것은 2.2절에서 언급된 Message Explosion문제를 해결하게 된다.

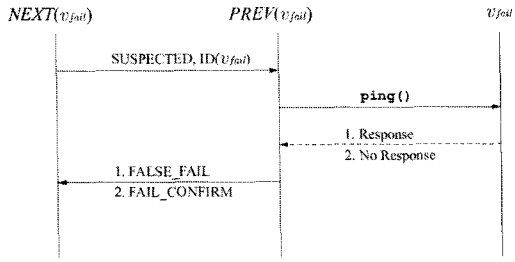
그러나 Gu 등[4]의 연구는 MCM을 이용한 실패 탐지의 가능성만을 제시할 뿐, 구체적인 실패 탐지 및 회복에 대한 언급이 없다. 그러므로 본 논문에서는 HRing 오버레이 네트워크에서 활용 가능한 실패 탐지 및 회복 기법을 제안한다. 본 연구에서는 우선 실패 탐지 단계와 이에 대한 회복 단계로 구분한다. 실패 탐지 단계에서는 MCM에 포함되어 있는 지역 링에 참여하는 노드들의 리스트에 근거하여 노드의 실패를 탐지하고, 회복 단계에서는 전체적인 오버레이 구조를 그대로 유지하면서 실패에 대한 회복을 수행한다. 이때 각 노드의 회복은 지역 링에서 먼저 수행되며, 실패가 의심되는 노드의 이전 위치에 연결된 노드에서 시작하여 상위 링에서 하위 링의 순서로 이루어진다.

실패가 의심되는 노드를 v_{fail}라고 하고, v_{fail}의 in_{main}으로 연결된 노드를 PREV(v_{fail}), out_{main} 노드를 NEXT(v_{fail}) 라고 한다. 또한 임의의 노드 n에 대해 deg(n)은 n의 차수(degree)를 의미하고, indeg(n)는 n의 진입차수(indegree)를, outdeg(n)는 n의 진출차수(outdegree)라고 할 때, indeg(v_{fail}) = 2인 경우, v_{fail}의 in_{sub}로 연결된 노드를 $PREV_{sub}(v_{fail})$라고 정의한다. 그리고 outdeg(v_{fail}) = 2인 경우, v_{fail}의 out_{sub}로 연결된 노드를 $NEXT_{sub}(v_{fail})$이라고 정의한다. 또한 임의의 노드 v_i에 대해 같은 링에 속한 노드의 집합을 NEIGHBORS(v_i)라고 한다. 그리고 HRing 오버레이에서 최상위 링(root ring)을 R_{root}라고 하며, 단말 링(terminal ring)을 $R_{terminal}$이라고 부른다. 그리고 $\forall v_i$가 속해 있는 현재 링을 R_c

라고 할 때, R_c의 상위 링을 R_{c-1}, 하위 링을 R_{c+1}로 표기한다.

3.1 실패 탐지 단계

본 논문에서 제안된 실패 탐지 단계는 Jain 등 [8]의 연구와 유사하다. 그러나 실패가 의심되었을 경우 모든 노드에게 브로드캐스트를 하는 것이 아니라 이미 정렬된 MCM을 이용하기 때문에 실패 의심 노드를 보다 정확하게 파악할 수 있다. 또한 연속적인 노드의 실패에 대해서도 탐지가 가능하다는 특징을 가진다. 그림 2는 HRing 오버레이에서 동시에 발생하는 실패가 없는 경우, 즉 하나의 노드에 대해서만 실패가 발생한 경우, 실패 탐지 단계를 수행하는 절차를 나타낸다.



(그림 2) 실패 탐지 절차

우선 HRing 오버레이에서 주기적으로 전송되는 MCM을 NEXT(v_{fail})가 일정 시간 동안 수신하지 못했다면 NEXT(v_{fail})는 v_{fail}의 실패를 의심하게 된다. 그리고 이를 확인하기 위해 MCM을 참조하여 PREV(v_{fail})에게 *SUSPECTED* 메시지를 전송한다. *SUSPECTED* 메시지를 수신한 PREV(v_{fail})는 v_{fail}이 자신의 out_{main}(또는 out_{sub})과 같은지를 확인하게 되고 만약 일치한다면, 고장이 의심되는 노드에 대해 실패 확정 절차를 거치게 된다. 이러한 실패 확정 절차는 PREV(v_{fail})에 의해 v_{fail}에게 ping()을 수행한다. 이때 ping()은 전통적인 방식의 ping()이 아니라 노드에서 발생한 실패를 식별하기 위한 메시지를 포함하게 된다. 그런 다음 v_{fail}로부터 수신되는 정보에 의해 v_{fail}에서 발생한 실패의 종류를 판단하게 되고, 그에 따른 결과 즉, *FALSE_FAIL* 또는 *FAIL_CONFIRM* 메시지를 NEXT(v_{fail})에게 전송하게 된다.

그러나 노드의 실패가 연속적으로 일어난 경우를 고려해야 하므로 다음과 같이 일반화된 수행 절차를 갖는다.

1. 주어진 시간(*MCM_TIMEOUT*)동안 MCM을 수신하지 못하면, 자신의 MCM을 참조하여 PREV(v_{fail})에게 *SUSPECTED*를 전송한다. 그러나 PREV(v_{fail})이 *SUSPECTED*를 수신하지 못하면 그 이전 노드(PREV(PREV(v_{fail})))에게 전송하게 된다. 이를 통해 여러 노드가 동시에 실패한 사실을 인식하게 된다. 그러므로 실패가 의심되는 각 노드별로 다음의 과정을 수행한다.
   - 이때 이전 노드(PREV...(PREV(v_{fail})))가 NEXT(v_{fail})라면 현재 링의 모든 노드에 실패가 발생한 경우이므로 따로 *SUSPECTED*를 전송하지 않고 NEXT(v_{fail}) 자신이 직접 다음 절차를 수행한다.
2. *SUSPECTED* 메시지를 수신한 노드는 참실패(true failure)인지 거짓 실패(false failure)인지를 판단하기 위한 절차를 수행한다. 이 판단 절차는 다음 절에서 논의하도록 한다. 그 다음 PREV(v_{fail})에 의해 노드의 실패가 확정되면 *FALSE_FAIL* 메시지 또는 *FAIL_CONFIRM*를 전송한다.
3. NEXT(v_{fail})이 *FAIL_CONFIRM*을 수신하면, 실패 회복 단계를 시작하고, MCM에서 해당 노드를 삭제한다. 그리고 MCM에 기록된 R_c의 정보를 이용하여 $v_i \in R_c$들에게 멀티캐스트 한다.
4. 절차 1에서 여러 노드에서 연속적인 실패가 발생하였다면, 실패가 보고된 v_{fail}의 PREV(v_{fail})을 v_{fail}로 놓고 다시 절차 1을 반복해서 수행한다.

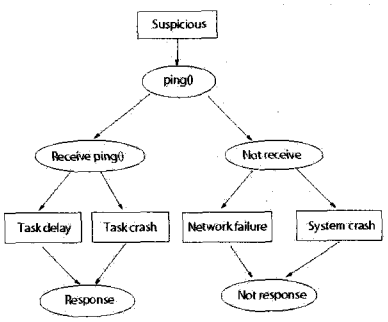
3.2 노드 실패 결정

앞 절에서 노드의 실패가 의심되는 경우 우선 PREV(v_{fail})에게 *SUSPECTED* 메시지를 전송하고, PREV(v_{fail})은 v_{fail}에게 ping()을 수행하여 노드 실패를 확인하게 된다. 이때 2.2절에서 언급한 유연성(*flexibility*) 문제를 해결하기 위해서는 노드 실패를 확정하기 하기 전에 각 노드에서 발생한 실패에 대한 분석이 선행되어야 한다. 본 연구에서는 시스템의 고장(host crash), 네트워크 실패(network failure), 그리고 작업 실패 또는 지연(task crash or delay)과 같이 3가지로 구분하도록 한다. 그리고 노드의 실패가 의심되는 경우 이러한 실패들의 유형을 식별하는 방법과 각 실패들에 대한 해결책을 제시하여 시스템의 유연성(*flexibility*)를 보장하도록 한다.

3.2.1 실패의 식별

SUSPECTED 메시지를 수신한 PREV(v_{fail})가 ping()을 수행하기 전에 전통적인 기능의 ping()만으로는 노드에서 발생한 실패를 식별하기 어렵기 때문에 본 연구에서 사용하는 ping()은 수정된 형태의 ping()을 사용해야 한다.

우선 PREV(v_{fail})가 ping()을 호출하였을 때 v_{fail}이 이를 수신하였다면 이는 작업 실패 또는 지연에 따른 실패로서 거짓 실패(false failure)로 처리 되거나 혹은 네트워크 지연으로 인한 MCM의 전달 지연으로 판단할 수 있기 때문에 그에 대한 응답을 PREV(v_{fail})에게 전송하고 PREV(v_{fail})은 NEXT(v_{fail})에게 *FALSE_FAILURE*를 전송한다. 그러나 v_{fail}이 ping()을 수신하지 못한 경우에는 시스템의 고장(system crash) 또는 네트워크 실패로 인한 문제로 판단할 수 있다. 즉, v_{fail}이 ping() 메시지를 수신하지 못한 경우, 시스템 고장과 네트워크 실패를 판단하기 위해 추가적인 작업을 수행해야 한다. 그림 3은 v_{fail}에서 수행되는 실패의 식별 절차를 나타내며, 별다른 실패가 발생하지 않은 경우는 제외하였다. 그리고 사각형은 v_{fail}의 현재 상태를 표현하며 타원은 상태 변화를 위한 행위를 나타낸다. 또한 ping() 메시지를 수신한 경우와 수신하지 못한 경우의 자세한 상태 변화의 조건은 그림 4와 그림 5와 같다.



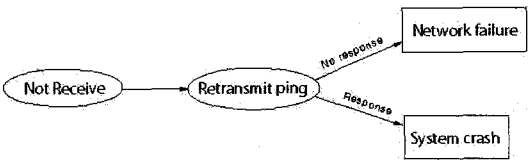
(그림 3) Vfail에서의 실패 식별

3.2.2 시스템의 고장

시스템 자체의 고장으로 인한 실패는 PREV(v_{fail})가 *FAIL_CONFIRM* 메시지를 생성하도록 한다. 하지만 v_{fail}에서 발생한 실패가 시스템 자체의 고장인지 네트워크 실패에 의한 것인지 식별할 수 있어야 한다. 그러므로 PREV(v_{fail})에서는 전통적인 방식의 ping()을 v_{fail}의 기본 게이트웨이로 재전송한다. 그래서 이에 대한 응답이 있는 경우에는 해당 게이트웨이의 서브넷에 문제가 없는 것으로 판단하여 시스템 고장으로 인한 실패로 확정하게 된다.

3.2.3 네트워크 실패

시스템의 고장과 마찬가지로 네트워크 실패의 경우에도 v_{fail}로부터 응답을 수신할 수 없다. 그러므로 위의 경우에서처럼 전통적인 방식의 ping()을 v_{fail}의 기본 게이트웨이로 재전송한다.



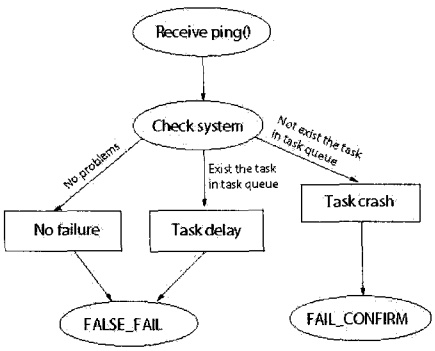
(그림 4) PREV(v_{fail})의 실패 처리

만일 전통적인 방식의 ping()에 대해 응답이 없다

면 이것은 네트워크 실패로 인한 노드의 실패로 인식하게 된다. 그림 4는 PREV(v_{fail})에서 시스템 고장과 네트워크 실패 시 처리되는 과정을 보여준다.

3.2.4 작업 실패 또는 지연

각 노드에서 수행중인 작업들이 실패하거나 혹은 작업들의 수가 증가하여 미처 작업들을 처리 하지 못하는 경우, MCM의 전송이 지연될 수 있으므로, NEXT(v_{fail})로부터 실패가 의심될 수 있다. 이것은 실제로 노드에 실패가 발생하여 더 이상 오버레이에 참여하지 못하는 것은 아니다. 뿐만 아니라 실제로는 실패 자체가 발생하지 않았을 수도 있다. 그래서 PREV(v_{fail})로부터 수신된 ping() 메시지에 의해 응답 메시지를 전송할 수 있게 된다.



(그림 5) ping 메시지 수신한 경우 V_{fail}에서의 실패 식별

그러나 이때 작업의 폭주로 인한 일시적인 중지 상태 즉, 작업 지연(task delay)인 경우에는 크게 문제 되지 않지만, 작업 자체의 실패인 경우에는 해당 노드에서 더 이상 작업을 수행할 수 없게 되므로 PREV(v_{fail})에게 이러한 사실을 알려주어야 한다. 그림 5는 작업 실패 또는 지연의 경우 v_{fail}에서 처리하는 과정을 보여준다.

3.3 실패 회복 단계

실패 회복단계는 실패가 탐지된 경우, 즉 NEXT(v_{fail})이 *FAIL_CONFIRM* 메시지를 수신한 경우 실행된다. 실패 회복 단계의 기본적인 아이디어는 현재 구조를 유지하기 위해 $\forall v_i \in R_{terminal}$에 대해 $\deg(v_i) = 2$인 v_i를 $v_{candidate}$로 선출하여 이를 실패가 발생한 노드의 위치로 참여(join) 시키는 것이다. 여기서 $\deg(v_i) = 2$인 v_i를 $v_{candidate}$로 선출하는 이유는 v_i는 $R_{terminal}$에서 간편하게 탈퇴(leave) 연산을 수행할 수 있기 때문이다. 또한 $|R_{terminal}| = 3$이라면 $R_{terminal}$이 사라지게 되므로, 전체 오버레이 계층의 레벨(level)이 줄어들어 더 나은 성능을 가질 수 있는 요건이 되기 때문이다. 이를 위해 $v_j \in R_{root}$는 $v_{candidate}$를 선출하기 위해 $\forall v_i \in R_{terminal}$의 정보를 유지하게 된다.

다음은 실패 회복의 일반적인 절차다.

1. NEXT(v_{fail})이 *FAIL_CONFIRM*을 수신하면, $v_{candidate}$의 정보가 필요하게 된다. 그러므로 NEXT(v_{fail})은 R_{root}의 노드에게 $v_{candidate}$의 정보를 요청한다.
   - $v_{candidate}$는 $\deg(v_{candidate}) = 2$를 만족해야 한다. 그렇지 않고 임의의 노드를 $v_{candidate}$로 선출하게 되면 선출에 소요되는 비용과 이를 이용한 회복에 대한 비용이 크기 때문에 $\deg(v_i) = 2$인 $v_i \in R_{terminal}$를 $v_{candidate}$로 선출하도록 한다.
2. $v_{candidate}$의 정보를 수신한 NEXT(v_{fail})는 $v_{candidate}$에게 연결 요청을 하여 out_{main}과 in_{main}에 대한 연결을 수행한다.
3. 연결이 완료되면 NEXT(v_{fail})로부터 MCM_{main} 정보를 수신하여 자신의 MCM_{main}으로 설정하여 회복 단계를 완료한다.
   - $\text{indeg}(v_{fail}) = 2$ 또는 $\text{outdeg}(v_{fail}) = 2$인 경우, $NEXT_{sub}(v_{fail})$ 또는 $PREV_{sub}(v_{fail})$에 대한 회복 단계는 완료된 것이 아니다. 그러므로 NEXT(v_{fail})에 대한 연결이 완료되면 $NEXT_{sub}(v_{fail})$ 또는 $PREV_{sub}(v_{fail})$은 앞의 1, 2, 그리고 3의 과정을 동일하게 수행하되, $v_{candidate}$의 out_{main}이 아닌 out_{sub}으로, 또는 in_{main}이 아닌 in_{sub}으로 설정된다.

- 연속적인 노드의 실패가 발생한 경우에도 회복 단계가 완료된 것이 아니다. 그러므로 $v_{candidate}$의 설정이 완료된 후, 연속적인 노드의 실패가 있는 경우에는 $v_{candidate}$를 NEXT(v_{fail})로 놓고, 앞의 1, 2, 그리고 3의 과정을 반복하여 수행한다.

4. 분석 및 평가

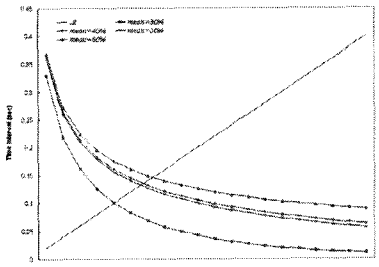
4.1 최적의 Δt 결정

제안된 실패 탐지는 MCM 수신에 근거하여 실패 의심 연산이 수행되므로 MCM 수신 시간 간격(interval)을 결정하는 것은 실패 탐지 성능에 큰 영향을 미치게 된다. 그러므로 MCM 수신 시간 간격을 Δt라고 할 때, Δt가 고정 값을 가지는 경우와 변화하는 경우 대해 알아보도록 한다.

우선 Δt가 고정 값을 가지게 되면, Δt는 노드의 부하에 직접적인 연관을 갖게 되므로 노드의 부하 $L=\{L_i|i\in N\}$에 대해 $L\propto\frac{1}{\Delta t}$인 관계가 있다. 그러므로 Δt와 L의 접점을 근사한 최적의 Δt(approximate optimal Δt)라고 정의한다. 이때 각 노드가 가지고 있는 자체 부하를 L_{local}이라고 하고, L_{local}은 네트워크에 참여하는 노드에서 임의의 시간에 자유롭게 발생하는 부하를 가정하기 때문에 포아송 분포(poisson diatribution)를 따른다고 가정한다. 그리고 heartbeat 메시지 처리에 따른 부하를 c라고 할 때, c는 단위 시간 Δt에 대한 처리 부하이므로 상수값을 가진다. 그러므로 노드 n_i에서의 부하 L, 즉 $L(n_i)$는 다음 식 (1)과 같다.

$$L(n_i)=L_{local}+c=\frac{e^{-\lambda\Delta t}(\lambda\Delta t)^k}{k!}+c \qquad (1)$$
$$(k=0,1,2,\ldots,i\in N)$$

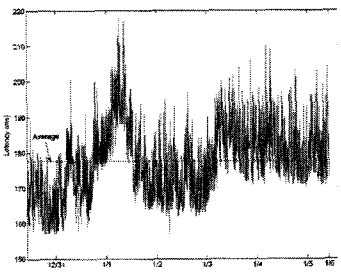
그림 6은 위의 식 (1)에 대해 CPU 처리 속도가 비슷한 노드에서 $L(n_i)$에 대한 최적의 Δt를 나타낸다. 즉, 최적의 Δt는 평균 부하가 30-60% 일 때 Δt와의 접점을 최적의 Δt(optimal Δt)로 결정한다. 이때 메시지 전송에 따른 네트워크 지연은 고려하지 않았으며 네트워크 대역폭은 거의 동일한 값을 가진다고 가정한다.



(그림 6) 최적의 Δt의 근사값

그러므로 Δt를 고정 값으로 한 경우, 가질 수 있는 적정한 Δt는 근사적으로 100ms $\leq \Delta t \leq$ 200ms의 간격에 존재하게 된다. 그림 7은 네트워크의 평균 지연 시간을 나타내며 여기에서 알 수 있듯이 평균 지연시간이 약 170ms이므로 계산된 Δt의 값은 적절한 값이라 할 수 있다[17].

그러나 Shi 등[16]의 연구에서는 Δt를 고정 값으로 하지 않고 임의로 변화하는 값으로 결정하였다. 이는 송신자와 수신자 간의 상태를 고려하여 각 상태에 따라 heartbeat 메시지의 전송 시간을 변화시킨다는 의미로 heartbeat 메시지를 전송하는 시간이 가우시안 분포(gaussian distribu- tion)를 따른다고 가정하였고, 이를 이용하여 최적화된 Δt를 얻기 위해 다음 식 (2)을 이용하여 Δt를 얻을 수 있다[16].



(그림 7) 네트워크 평균 지연

$$f(\Delta_{interval}) = \Delta_{interval} \prod_{j=1}^{\frac{T_D^U}{\Delta_{interval}}} \frac{\Delta_{tr} + (T_D^U - j\Delta_{interval})^2}{\Delta_{tr} + P_L(T_D^U - j\Delta_{interval})^2} \quad (2)$$

이때 P_L는 메시지가 소실될 확률이며, T_D^U는 Δt가 가질 수 있는 최대값이며, $\Delta tr \leq T_D^U - \Delta t$가 된다. 그러나 $\Delta tr < \Delta_{interval}$이 되면 $x = \Delta_{interval}$인 $f(x)$에 대해 $f'(x) < 1$이 되기 때문에, $0 \leq \Delta t \leq \infty$에서 최대 $\Delta_{interval}$ 보다 큰 Δt가 존재한다. 이때에는 위의 식 (2)가 항상 만족하지 않게 된다. 그러므로 이 경우에는 Δtr의 값을 반드시 유지해 주어야 한다.

그러나 HRing 오버레이 네트워크에서는 heartbeat 메시지를 다수의 노드에게 전송하지 않고, 오직 자신의 out_{main} 또는 out_{sub}에게만 전송하게 되므로 heartbeat 메시지 전송에 필요한 Δtr은 불필요하게 된다. 그러면 $\Delta t \simeq \Delta_{interval}$이 된다. 다시 말해 Δt는 식 (2)에 의해 결정된 다고 볼 수 있으므로 Δt에 대한 시간 복잡도는 O(n)이 된다.

4.2 후-처리에 대한 평가

제안된 실패 탐지 방법이 MCM에 의해 최초 실패가 의심되었더라도, 실패 탐지의 완료는 후-처리(Post-Processing)가 수행되어야한다. 이때 후-처리(Post-Processing)란 *SUSPECTED* 메시지를 수신한 PREV(v_{fail})이 NEXT(v_{fail})에게 *FALSE_FAIL* 또는 *FAIL_CONFIRM* 메시지를 전송할 때까지의 과정을 의미한다.

*SUSPECTED*를 수신한 PREV(v_{fail})은 먼저 v_{fail}에게 ping()을 수행하게 된다. 그런데 ping()은 기본 대기 시간, *MAX_WAIT*를 가지므로 후-처리는 *MAX_WAIT*를 상한으로 수행된다. 하지만 전통적인 ping()에서의 기본 대기 시간은 기존의 다른 실패 탐지 절차에 비해 매우 높은 대기 시간을 갖기 때문에 ping()의 기본 대기 시간을 조정해 주어야 한다.

그러므로 n을 지역 링에서 실패한 노드의 개수라고 하고, NEXT(v_{fail})에게 *FALSE_FAIL* 또는 *FAIL_CONFIRM*를 응답하기까지의 시간을 r이라고 할 때, 후-처리가 종료되기까지의 시간은 n×(*MAX_WAIT* + r)이 된다. 그리고 Δt에 따른 실패 의심 단계를 함께 고려 할 경우 실패 탐지 단계는 *SUSPECTED*를 전송하는데 소요되는 시간을 s라고 할 때, n×(s + *MAX_WAIT* + r)이 되고 또한 실패 탐지는 R_c에서만 영향을 미치게 되므로 전체 실패 탐지 단계에 대한 시간 복잡도는 O(n)이 된다.

4.3 실패 탐지의 인식 및 회복

기존의 리더 기반의 실패 탐지기들은 실패가 발생하면, 실패 메시지를 우선 리더에게 전송한다. 그런 다음 리더들은 연결된 다른 리더들에게 전송하게 된다. 이것은 네트워크에 참여하는 전체 노드들이 동일한 노드들의 정보를 가지고 있기 때문이다. 이에 따른 네트워크 대역폭 소비는 전체 노드의 수 N에 대해 적어도 O(log$N$)이 되며, 최악의 경우 O(N^2)가 된다.

그러나 HRing에서는 각 링에서 동작하는 연산들이 서로 독립적이므로, 실패가 발생한 노드가 포함된 링에서만 이를 인식하면 된다. 그러므로 최적의 Δt에 의해 실패탐지 시간이 결정된다. 하지만 HRing에서는 Δt가 상수값을 가질 수 있으므로, 실패 탐지에 따른 실패 메시지의 전송은 $O(1)$이 된다. 또한 제안된 기법은 실패가 인식 되었을 때 항상 R_{root}의 노드에 저장된 정보를 참조하여 회복이 수행되므로 O(1)이 된다.

5. 실험 및 평가

본 연구에서는 실패 탐지와 회복에 대한 성능을 평가하기 위해 클러스터 단위의 리더를 기반으로 하는 Push 모델과 지역 정보(partial view)를 이용으로 하는 gossip 메시지 기반의 실패 탐지

모델과 비교한다. 리더 기반의 Push 모델은 heartbeat 메시지를 사용한다는 점에서 본 연구에서 제안된 방법과 유사하다고 할 수 있으나, 우선 일정한 클러스터(또는 그룹) 내에서 리더가 선출되어야 하며, 모든 heartbeat 메시지는 리더에 의해 처리되는 가장 일반적이고 보편적인 실패 탐지 및 회복 기법이다. 그리고 지역 정보를 이용하는 gossip 메시지 기반의 실패 탐지 모델은 완전한 분산 실패 탐지 기법으로 일정한 수의 gossip 메시지만을 이용하여 노드의 실패를 탐지하기 때문에 기존 방법이 가질 수 있는 문제점을 해결하였다는 의미에서 본 연구에서 제안된 방법과 비교하기에 적합하다고 판단된다.

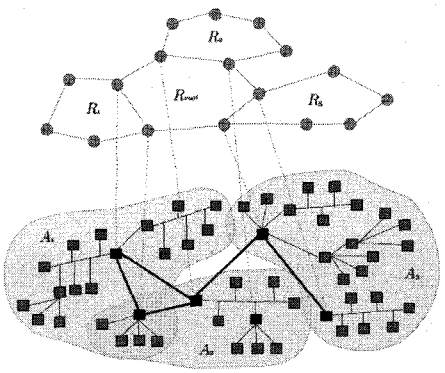
제안된 실패 탐지 및 회복 기법의 성능을 평가하기 위해 우선 성능 매트릭을 정의한다. 정의된 성능 매트릭은 2.2절에서 언급되었다시피 실패 탐지 시 고려해야 할 사항을 근거로 작성 하였다.

(표 1) 성능평가 매트릭

측정값	설 명
대역폭 소비율	네트워크 대역폭의 가용성 또는 실패 탐지 및 회복 연산이 수행되는 동안의 대역폭 소비율을 의미한다.이를 이용하여 네트워크에서 발생하는 메시지 폭주 정도를 알 수 있다.
관리 비용	네트워크의 규모가 증가함에 따라 대규모 노드들에 대한 실패 탐지도 원활하게 이루어져야 한다. 그러므로 대규모 노드들의 실패 탐지시 발생할 수 있는 탐지 오버헤드를 의미한다. 이를 이용하여 실패 탐지 및 회복 연산에 대한 확장성과 역동성을 평가할 수 있다.
신속성	네트워크에서 발생하는 실패들을 얼마나 빨리 인식할 수 있는지를 말한다. 이것은 heartbeat이 전송되는 주기에 따라 달라 질 수 있으나 본 연구에서는 heartbeat 주기는 모두 동일한 값으로 하여 평가한다.

5.1 실험 환경

본 연구에서는 시뮬레이션을 통해 리더를 갖는 heartbeat 기반의 Push 모델(Leader-based), Gossip 기반의 실패 탐지 모델, 그리고 제안된 HRing을 이용한 모델을 비교하였다. 시뮬레이션 코드는 Java언어를 이용하여 작성하였으며 최대 100,000개의 노드까지 생성하여 시뮬레이션을 수행하였다. 실험을 위해 우선 임의의 물리 네트워크를 구성한 다음, 그림 8과 같이 물리 네트워크 기반의 HRing 오버레이 네트워크를 구성한다. 이때 R_1의 노드들은 물리 네트워크에서의 A_1에 존재하는 노드들로 구성되며, R_2의 노드들은 A_2의 노드들, 그리고 R_3의 노드들은 A_3의 노드들로 구성된다. 또한 R_{root}의 노드들은 물리 네트워크의 영역 A_1, A_2, 그리고 A_3의 임의의 노드 또는 라우터 역할을 수행하는 노드들로 구성된다.



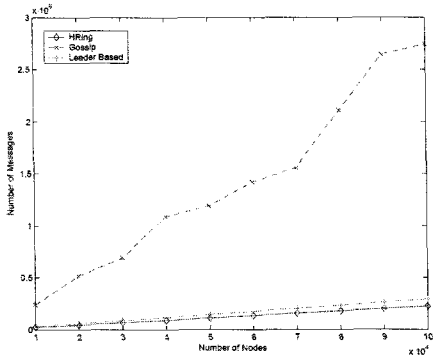
(그림 8) 실험을 위한 HRing 오버레이 네트워크

마지막으로 앞에서 기술한 바와 같이 메시지 송/수신에 따른 네트워크 트래픽을 제외 하고 다른 트래픽은 없다고 가정하였으며, 메시지 손실과 실패 탐지 및 회복 수행에 따른 보안 관련 사항은 고려하지 않았다.

5.1 실험 결과

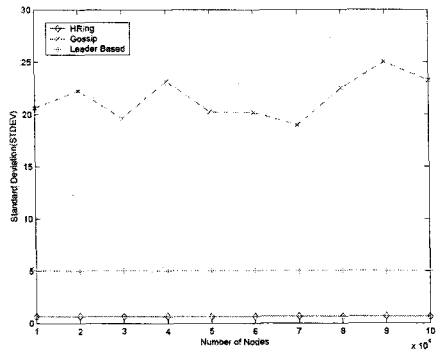
그림 9는 실패 탐지 및 회복을 수행함에 있어 각 노드에서 발생되는 메시지의 양을 나타낸다. 이것은 성능 평가 매트릭에서 메시지 발생에 따른 대역폭 소비율을 평가하는 것이다. 실패 탐지 시 메시지의 전달은 리더 기반과 HRing은 각각

지역 클러스터와 지역 링에서만 수행되므로 리더 기반과 HRing은 큰 차이를 보이지 않는다. 그러나 gossip의 경우 지역 정보(partial view)를 업데이트하기위한 gossip 메시지 전달에 따른 메시지 전송량이 많아짐을 알 수 있다.



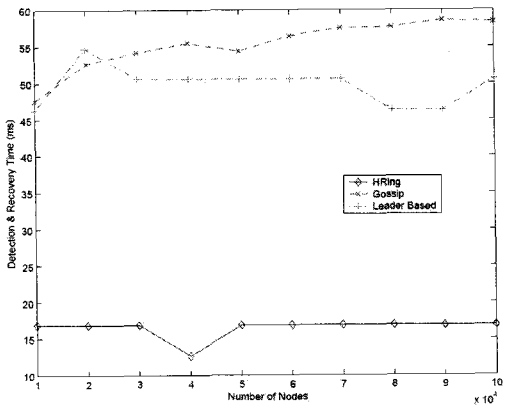
(그림 10) 실패 탐지에 따른 메시지 발생

그림 10은 성능 평가 매트릭의 관리 비용을 평가한 결과를 보여준다. HRing은 단지 자신의 out_{main}(또는 $outsub$)과 in_{main}(또는 in_{sub})에 대한 통신을 수행하기 때문에 각 노드에서의 부하가 낮아지게 된다. 반면 리더 기반과 gossip은 여러 노드와 통신을 수행해야 하며 특히 리더 기반의 경우 리더가 실패한 경우에는 새로운 리더를 선출해야 하는 부하가 추가로 발생 하였을 수 있기 때문에 HRing이 다른 방법에 비해 낮은 관리 비용을 가짐을 알 수 있다.



(그림 9) 실패 탐지에 따른 노드들의 부하

마지막 성능 평가 요소로 실패 탐지 및 회복에 대한 신속성이 있다. 실제로 리더 기반의 경우 HRing에 비해 빠른 탐지 성능을 보였다. 이것은 실패가 발생할 때 리더 기반은 리더가 직접 실패를 탐지하여 전체 노드에게 멀티캐스트 하게 되나, HRing의 경우 전체 노드에 의한 실패 탐지는 실패가 의심되는 경우 실패의 종류를 파악하기 위한 추가 연산을 수행하게 되므로 실패 탐지 성능은 다소 낮아지는 것으로 판단된다.



(그림 11) 실패 탐지 및 회복 시간

그러나 실패 탐지 시간과 그에 대한 회복 시간을 함께 고려하였을 경우, 그림 11에서 보는 바와 같이 HRing이 리더 기반과 gossip에 비해 나은 성능을 보임을 알 수 있다. 리더 기반의 경우 리더에서 실패가 발생한 경우 새롭게 리더를 선출하여 회복이 이루어져야 하며 또한 이러한 정보가 전체 노드에게 모두 전송되어야 한다. Gossip의 경우도 네트워크 회복에 관한 정보를 지역 정보에 근거하여 전체 노드에게 gossip하기 때문에 HRing에 비해 회복 시간이 길어진다. 반면 HRing의 경우 실패 탐지와 회복에 관한 정보가 전체 노드에게 전달되지 않고 단지 PREV(v_{fail})과 NEXT(v_{fail})의 정보만 갱신되므로 리더 기반과 gossip에 비해 빠른 회복 시간을 가지게 된다.

6. 결론 및 향후 연구과제

오버레이 네트워크의 단점은 신뢰적인 통신을 보장하지 못한다는 것이다. 이것은 오버레이 네트워크가 물리적인 네트워크가 아니라 가상의 링크를 기반으로 하는 네트워크이므로 노드 실패에 따른 신뢰성을 보장하기 어렵다는 의미이다. 본 논문에서는 HRing 오버레이 네트워크의 신뢰적인 통신을 보장하기 위한 실패 탐지 및 회복 기법을 제안하였다. 제안된 실패 탐지 및 회복 기법은 기존의 리더 기반의 실패 탐지기법과 gossip 기반의 실패 탐지 기법에 비해 확장성, 신속성의 측면에서 나은 성능을 가짐을 시뮬레이션을 통해 보였다. 또한 리더 기반 기법과 gossip 기법에서 수행하지 않았던 실패 탐지 시, 야기된 실패들의 종류를 판단하는 기능이 추가되었기 때문에 실패 발생 시 적절한 대응책을 마련할 수 있도록 해 준다는 것이 장점이라 할 수 있다.

향후 연구과제로는 본 논문에서 가정한 노드의 지역 부하를 가정함에 있어 자기 유사성(self-similarity) 모델로 가정하여 실험 할 예정이다. 이는 현실적인 지역 부하를 가정하여 보다 근접한 최적의 Δt를 예측할 수 있을 것으로 판단된다. 또한 제안된 실패 탐지 및 회복 기법의 생존성(liveness)와 안정성(safety) 검증을 위한 논의도 추가할 예정이다.

참 고 문 헌

[1] Andersen D., H. Balakrishnan, F. Kaashoek and R. Morris, 'Resilient overlay networks.', Proceedings of SOSP01, pp.131-145, 2001.

[2] Felber P., X. Defago, R. Guerraoui and P. Oser, 'Failure detectors as first class objects.', Proceedings of DOA'99, pp.132-141, 1999.

[3] Gupta I., T. D. Chandra and G. S. Goldszmidt, 'On scalable and efficient distributed failure detectors.', Proceedings of PODC'01, pp.170-179, 2001.

[4] Gu T., S. Hong, S. Uhmn and K. M. Lee, 'Membership management based on hierarchical ring for large grid environments.', International Journal of Information Processing Systems(IJIPS) accepted, 2007.

[5] Hayashibara N., A. Cherif and T. Katayama, 'Failure detectors for large-scale distributed systems.', Proceedings of SRDS'02, pp.404-409, 2002.

[6] Hildrum K., J. D. Kubiatowicz, S. Rao and B. Y. Zhao, 'Distributed object location in a dynamic network.', Proceedings of SPAA'02, pp.41-42, 2002.

[7] Hwang S. and C. Kesselman, 'A flexible framework for fault tolerance in the grid.', Journal of Grid Computing, 1(3):251-272, 2003.

[8] Jain A. and R. K. Shyamasundar, 'Failure detection and membership management in grid environments.', Proceedings of GRID'04, pp.44-52, 2004.

[9] Lian Q., W. Chen, Z. Zhang, S. Wu and B. Y. Zhao, 'Z-ring: Fast prefix routing via a low maintenance membership protocol.', Proceedings of ICNP'05, pp.132-146, 2005.

[10] Renesse R., Y. Minsky and M. Hayden, 'A gossip-style failure detection service.', Proceedings of Middleware'98, pp.55-60, 1998.

[11] Ratnasamy S., P. Francis, M. Handley, R. Karp and S. Schenker, 'A scalable content-addressable network.', Proceedings of SIGCOMM'01, pp.161-172, 2001.

[12] Rowstron A. I. T. and P. Druschel, 'Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems.', Proceedings of Middleware'01, pp.329-350, 2001.

[13] Stelling P., C. Lee, I. Foster, G. von Laszewski and C. Kesselman, 'A fault detection service for wide area distributed computations.', Proceedings of HPDC-7'98, page 268, 1998.

[14] Stoica I., R. Morris, D. Karger, M. F. Kaashoek and H. Balakrishnan, 'Chord: A scalable peer-to-peer lookup service for internet applications.', Proceedings of SIGCOMM'01, 2001.

[15] Sistla K., A. D. George and R. W. Todd, 'Experimental analysis of a gossip-based service for scalable, distributed failure detection and consensus.', Cluster Computing, 6(3):237-251, 2003.

[16] Shi X., H. Jin and W. Qiang, 'Alter: first step towards dependable grids.', Proceedings of SAC'06, pp.805-806, 2006.

[17] Internet traffic report: http://www.internettrafficreport.com.

[18] Wikipedia, the free encyclopedia: http://www.wikipedia.org/.

◐ 저 자 소 개 ◑

구 태 완(Tae-Wan Gu)
2000년 한림대학교 수학과, 컴퓨터공학과 졸업(학사)
2002년 한림대학교 대학원 컴퓨터공학과 졸업(석사)
2007년 한림대학교 대학원 컴퓨터공학과 졸업(박사)
2007~현재 대한상공회의소 강원인력개발원 정보기술과 교수
관심분야 : 분산처리, 오버레이 네트워크, 그리드/P2P 네트워크 등
E-mail : gutaewan@korcham.net

이 광 모(Kwang-Mo Lee)
1975년 서울대학교 응용수학과 졸업(학사)
1984년 서울대학교 대학원 계산통계학과 졸업(석사)
1992년 서울대학교 대학원 계산통계학과 졸업(박사)
1985~현재 한림대학교 정보통신공학부 컴퓨터공학 전공 교수
관심분야 : 병렬처리, 분산처리, Cluster Computing 등
E-mail : kmlee@hallym.ac.kr