클라우드에서 효율적인 메시지 전파를 위한 그룹 기반 가쉽 프로토콜

임종범[†] · 이종혁[††] · 진성호[†††] · 유헌창[††††] · 이화민[†††††]

요 약

클라우드 컴퓨팅은 공유 가능한 여러 자원들이 모여 가상화된 형태로 서비스를 제공하는 인터넷 기반의 컴퓨팅 패러다임이다. 이러한 클라우드 컴퓨팅 환경에서 무수한 자원들에 대한 상태 정보를 신속히 전달하기 위한 방법으로 가쉽 프로토콜이 주목받고 있다. 가쉽 프로토콜은 견고하고 확장적인 멀티캐스트를 제공하는 반면, 100%의 도달가능성을 만족하기 위해서는 여분의 중복 메시지를 요구한다는 문제점이 있다. 본 연구에서는 자원들의 상태정보를 효율적으로 전달하면서 발생하는 메시지 오버헤드를 줄이기 위해 그룹 기반의 가쉽 멀티캐스트 프로토콜(GGMP)을 제안한다. 또한, 제안한 프로토콜의 성능을 실험을 통해 검증하였다.

주제어 : 가쉽 프로토콜, 클라우드 컴퓨팅, 멀티캐스트

Group-based Gossip Protocol for Efficient Message Dissemination in Clouds

Jong-Beom Lim[†] · Jong-Hyuk Lee[††] · Sung-Ho Chin[†††] · Heon-Chang Yu[††††] · Hwa-Min Lee[†††††]

ABSTRACT

Cloud computing is an Internet-based computing paradigm that provides services in a virtualized form composed of plenty of resources sharable. In Cloud computing environments, gossip protocols are engaged as a method to rapidly disseminate the state information for innumerable resources. Although gossip protocols provide a robust and scalable multicast, there is a drawback that requires redundant messages in satisfying 100% of reachability. In our study, we propose a Group-based Gossip Multicast Protocol in order to reduce the message overhead while delivering the state information efficiently. Furthermore, we verified the performance of the proposed protocol through experiments.

Keywords : Gossip Protocol, Cloud Computing, Multicast

† 준 회 원: 고려대학교 컴퓨터교육과 석사과정
†† 종신회원: 고려대학교 컴퓨터교육과 박사과정
††† 종신회원: 고려대학교 정보창의교육연구소 연구교수
†††† 종신회원: 고려대학교 컴퓨터교육과 교수
††††† 종신회원: 순천향대학교 컴퓨터학부(교신저자)
논문접수: 2010년 08월 03일, 심사완료: 2010년 09월 18일
* 본 논문은 2009년 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(2009-0070138).

1. 서 론

당대의 IT 패러다임의 하나로 클라우드 컴퓨팅이 각광받고 있다. 클라우드 컴퓨팅은 서비스 제공자가 제시하는 서비스 수준 협약(Service Level Agreement)에 따라 사용자가 서비스를 사용한 만큼의 금액을 지불하는 형태로, 네트워크를 통해 분산되어 있는 자원들을 사용자에게 하나의 거대한 자원으로 결합된 추상 레벨을 제공한다[1].

예를 들어, 클라우드 컴퓨팅에서 스토리지 시스템을 구성하는 데이터 노드들은 쉽게 추가 또는 제거 될 수 있는 유연하게 연결된 모델로, 각각의 데이터 노드들은 인터네트워킹을 통해 하나의 가상 파일 시스템(Virtual File System)으로 표현될 수 있다. 클라우드 서비스 제공자 입장에서 높은 가용성, 신뢰성을 제공하기 위해서는 각 자원 노드들의 상태정보 관리가 특히 중요하다[2]. 즉, 특정 데이터 노드의 실패 정보 등을 감지하였을 때, 이를 신속히 전달하여 해당 노드에 대한 대처를 해야 한다.

이에, 대규모의 분산 컴퓨팅 환경에서 견고하면서 확장적인 메시지 전달 메커니즘으로 응용단(Application-level)의 멀티캐스트 기법인 가쉽(Gossip) 프로토콜이 사용되어 왔다. 가쉽 프로토콜은 주기적으로 대상 노드를 임의적으로 선택하여 메시지를 주고받는 형태로, 시스템을 구성하는 각 노드가 전체 노드 정보를 알지 않더라도 높은 확률로 모든 노드가 메시지를 전달 받는다는 것을 보장한다.

가쉽 프로토콜은 노드가 무수히 많은 환경에서도 확장성을 제공하고 천(Churn)-노드의 가입과 탈퇴-과 실패가 잦은 환경에서도 회복력이 뛰어난 특성을 갖는 반면, 100%의 도달가능성을 만족하기 위해서는 여분의 중복 메시지를 요구한다.

가용한 자원들의 상태가 매우 동적인 클라우드 컴퓨팅 환경에서 서비스 수준 협약에 따른 가용성, 신뢰성 등의 서비스 품질(QoS)을 만족하기 위해서는, 자원 노드의 실패 정보와 같은 상태 정보를 신속하게 전달하는 것이 중요하다. 예를 들어, 가용할 수 없는 자원 노드의 상태를 전달 받지 못한 상태에서 해당 자원 노드를 포함하여 사용자의 요구에 따라 가상 자원으로 공급하였을 때, 사용자는 클라우드 서비스의 실패를 경험함으로써 더 이상 해당 클라우드 제공자를 신뢰하지 않을 것이다.

가쉽 프로토콜이 내재하고 있는 특성, 즉 모든 노드가 메시지 전파를 받기 위해 요구되는 전파 지연과 발생하는 중복 메시지를 줄이기 위해 본 연구에서는 그룹 기반 가쉽 멀티캐스트 프로토콜을 제안한다. 제안하는 프로토콜은 그룹을 생성하고 생성된 그룹 리스트를 유지하기 위한 자가 조직(Self-Organization) 단계와 메시지 전파(Dissemination) 단계로 나누어진다.

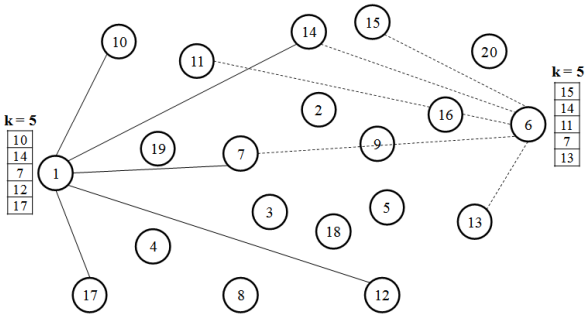
본 논문의 구성은 다음과 같다. 2장에서는 가쉽 프로토콜 모델과 관련 연구를 살펴보고, 3장에서는 제안하는 그룹 기반 가쉽 프로토콜 및 알고리즘을 설명한다. 4장에서는 그룹 기반 가쉽 프로토콜의 성능 실험 결과를 보여주고, 5장의 결론을 통해 본 논문을 마친다.

2. 관련 연구

2.1 가쉽 프로토콜 모델

가쉽 또는 전염성(Epidemic) 프로토콜에서는 각 노드가 시스템 내의 전체 노드 정보를 유지하기보다는 부분 뷰(Partial view)라는 시스템 내의 전체 노드 정보 중 일부만을 가지는 목록을 관리함으로써 확장성의 문제를 극복할 수 있다.

가쉽 프로토콜에서 각 노드는 특정 크기의 노드 정보를 유지할 수 있는 부분 뷰를 가지게 되며, 부분 뷰의 크기는 시스템 매개변수 k에 의해 결정된다. 각 노드는 이렇게 만들어진 부분 뷰내에서 가쉽 대상을 임의적으로 선택하여 메시지를 주고받을 수 있다.



<그림 1> 부분 뷰를 가지는 가쉽 프로토콜 모델

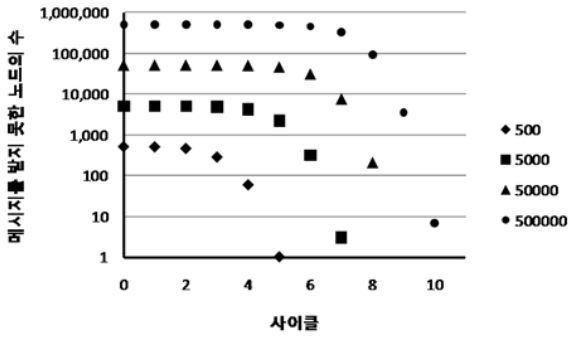
<그림 1>은 일부 노드가 5개의 노드 정보를 지닐 수 있는 부분 뷰를 가졌을 때의 모습을 표현한 것으로, 모든 노드의 부분 뷰 정보는 균일한 확률에 의해 임의적으로 결정된다. 여기서 각 노드가 가지는 부분 뷰의 정보는 하위단의 피어 샘플링 서비스(Peer Sampling Service)[3]에 의해 유지될 수 있으며, 다음과 같은 대표적인 두 가지의 메소드 인터페이스를 제공한다:

- ***init()***: 서비스 시작 초기에 호출되는 메소드로, 피어 샘플링 서비스를 제공할 수 있도록 초기화 시킨다.
- ***getPeer()***: 시스템에 존재하는 특정 노드의 식별자를 반환하는 메소드로, 반환되는 노드의 식별자는 *init()* 메소드의 구현에 따라 임의적으로 선택된다.

또한, 가쉽 프로토콜 운용상의 제어를 위해 다음과 같은 시스템 매개변수가 사용 된다:

- ***k***: 앞에서 언급한 것과 같이 각 노드가 관리하는 이웃 노드 정보(부분 뷰) 목록의 개수로, 노드 정보는 노드의 식별자와 타임스탬프(Timestamp)가 포함될 수 있다.
- ***Cycle***: 가쉽 메시지가 발생하는 라운드(Round)의 수로 초기에는 0부터 시작하여 가쉽 메시지가 발생할 때마다 1씩 증가하며, 설정한 사이클의 수보다 작을 때까지 주기적으로 가쉽 메시지 교환이 이루어진다.
- ***Fanout***: 한 사이클마다 부분 뷰에서 선택되는 가쉽 대상의 수로, 이 매개변수의 증가에 따라 모든 노드가 메시지를 전달 받는다는 신뢰성이 높아진다. 하지만 그에 따른 중복 메시지의 발생으로 인해 메시지 오버헤드 또한 증가한다.

가쉽 프로토콜은 가쉽 메시지를 보내는 송신자와 메시지를 받는 수신자의 메시지를 전달하는 방식에 따라 푸시(Push) 모드, 풀(Pull) 모드, 푸시-풀(Push-Pull) 모드로 나누어질 수 있다. 푸시 모드는 송신자가 가쉽 대상을 선정하여 상대방에게 메시지를 전달하는 방식이다. 반면, 풀 모드는 노드 자신이 질의(Query)를 통해 메시지를 받지 못하였다는 것을 인지하였을 때, 가쉽 대상을 선정하여 상대방으로부터 메시지를 가져오는 방법이다. 푸시-풀 모드는 푸시 모드와 풀 모드가 혼합된 방법으로, 가쉽 대상과 메시지를 서로 주고받을 수 있다.



<그림 2> 사이클에 따른 정보를 받지 못한 노드의 수(Fanout=1)

<그림 2>는 푸시-풀 모드를 사용하는 경우 노드의 수가 각각 500(◇), 5,000(□), 50,000(△), 500,000(○) 일 때, 사이클의 변화에 따른 메시지를 받지 못한 노드(Virgin)의 수를 나타내는 그래프로, 플랫형(Flat) 가쉽 프로토콜을 사용하였을 때 모든 노드가 메시지를 전달 받기 위해서는 각각 6, 8, 9, 11 사이클이 필요하다.

2.2 관련 연구 내용

Scamp[4]는 전체 구성원 정보를 알지 않더라도 각 노드가 가지는 부분 뷰 정보만을 이용한 멤버십 통신 기법을 제안하였다. Scamp에서는 부분 뷰의 크기가 전체 노드 수(N)에 따라 변하는 특징을 가진다. 다시 말해, 부분 뷰의 크기는 $(c+1)\log(N)$으로 전체 노드수가 증가함에 따라 부분 뷰의 크기도 증가하는 형태이다. Scamp는 노드가 가지는 일부의 정보만으로 가쉽 멤버십 프로토콜의 실효성을 검증한 반면, 발생하는 중복 메시지를 줄이기 위한 고려가 없다.

HiScamp[5]는 Scamp를 기반으로 진행된 연구로, 각 노드간의 거리 값을 측정하여 거리가 가까운 노드들을 계층적인 클러스터로 구성한 가쉽

프로토콜을 제안하였다. 이 프로토콜은 각 계층에 따라 두 가지의 부분 뷰를 가지며, 이 정보를 기반으로 구성원 정보를 유지한다. 하지만 계층적인 구조가 갖는 단점인 단일 실패점이 존재하게 되며, 이를 위한 추가적인 기법의 사용이 요구된다.

Cyclon[6]은 셔플(Shuffle)이라는 개념을 도입하여 노드들의 잦은 가입과 탈퇴에 대비한다. 셔플은 가쉽 메시지 발생 시 두 노드간의 부분 뷰의 내용을 오래됨 정도를 기준으로 부분 뷰의 노드 정보를 서로 교환하고 갱신하는 연산으로, 오래된 정보일수록 탈퇴했을 가능성이 높다는 가정을 가진다. 이 셔플 연산을 통해 노드들의 가입과 탈퇴에 의해 동적으로 변하는 환경에 대처하는 프로토콜을 제시하였다.

HyParView[7]는 적극적인(Active) 부분 뷰와 피동적인(Passive) 부분 뷰라는 두 개의 별개적인 부분 뷰를 가진다. 이 프로토콜에서는 사이클마다 셔플 연산에 의해 교환되는 대량의 노드 정보를 피동적인 부분 뷰에 넣어 보관하고, 적극적인 부분 뷰의 특정 노드가 실패하였을 때 피동적인 부분 뷰 정보를 이용하여 적극적인 부분 뷰를 재구성하는 결함 포용 기법을 제공한다.

CLON[8]은 WAN 규모의 환경을 가지는 클라우드 환경에서 지역성을 고려한 가쉽 프로토콜을 제안하였다. 다시 말해, 지리적으로 분산되어 있는 대규모 네트워크 환경에서 멀리 떨어져 있는 노드들과 통신하기보다는 클러스터내의 가까운 노드들과 통신함으로써 발생할 수 있는 통신 비용을 줄인다. 하지만, 지리적으로 가까운 노드들과 클러스터를 구성하기 위한 지식이 사전에 요구된다.

HiScamp, Cyclon, HyparView, CLON 연구에서는 노드들의 가입과 탈퇴 비율 및 실패 비율에 따른 가쉽 프로토콜에 대한 연구가 이루어졌으나, 본 연구에서 제안하는 100%의 메시지 도달가능성을 만족하기 위해 발생하는 메시지 오버헤드를 줄이기 위한 방법을 제시하지 못하였다.

3. 그룹 기반 가쉽 알고리즘

기존의 가쉽 프로토콜에서는 시스템 내에서 노드를 식별하기 위해 유일한 값으로 지정되는 노드 식별자와 노드의 생성시간을 나타내는 타임스탬프가 노드 정보로 표현되었다. 본 연구에서 제안하는 그룹 기반의 가쉽 프로토콜에서는 노드 식별자와 타임스탬프 외에 그룹 식별자(Group ID)와 해당 그룹의 노드들의 수를 나타내는 그룹 크기 정보(Group Size)를 포함할 수 있는 필드를 <그림 3>과 같이 추가하였다.

Node ID	Timestamp	Group ID	Group Size

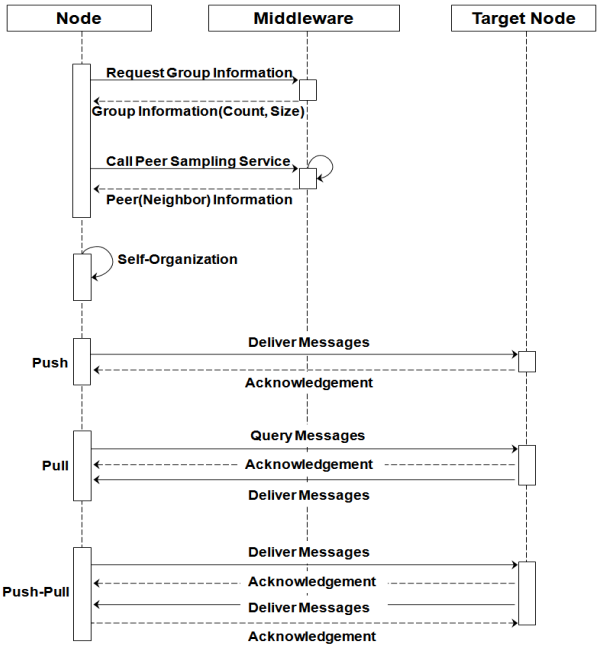
<그림 3> 그룹 기반 가쉽 프로토콜에서의 노드 정보 필드

그룹 기반 가쉽 프로토콜에서는 <그림 3>에서 추가된 그룹 정보를 이용하여 특정 노드가 그룹에 속해있는지 여부를 파악할 수 있으며, 그룹 리스트를 비교하지 않고도 노드간의 같은 그룹인지 여부를 쉽게 파악할 수 있다.

그룹 기반 가쉽 프로토콜의 흐름도는 <그림 4>와 같으며, 그 절차는 다음과 같다.

1. 시스템의 초기화가 완료되면, 모든 노드는 시스템의 정보를 가지고 있는 미들웨어에 자가 조직에 필요한 *k*, *Cycle*, *Fanout*과 같은 시스템 정보를 요청하면, 미들웨어는 해당 노드에 관련 정보를 전달한다.
2. 그룹을 구성하고 그룹 리스트를 유지하기 위한 정보를 가지기 위해 노드는 미들웨어에 그룹 정보를 요청한다. 이에, 미들웨어는 그룹의 수와 한 그룹의 구성원의 수의 정보가 포함된 메시지를 해당 노드에게 전달한다.
3. 노드는 전달받은 그룹 정보를 가지고 그룹 리스트를 생성하기 위해 피어 샘플링 서비스를 호출한다. 호출의 결과로 시스템 내에 존재하는 노드들 중 임의로 선택된 노드의 식별자가 반환된다.
4. 미들웨어를 호출함으로써 얻어지는 노드의 식별자를 가지고 실제로 자가 조직을 수행한다. 자가 조직 절차에서는 부분 뷰 정보를 포함하여 그룹 정보 리스트를 구성하는 단계가 수행된다.
5. 자가 조직 단계를 수행한 이후에는 구현된 응

용에 따라 메시지 전파가 이루어진다. 이 때, 메시지 전달 모드는 임의적으로 선택된 노드와 푸시, 풀, 또는 푸시-풀 모드로 수행될 수 있으며, 모든 노드는 동일한 모드를 사용하여 메시지 교환이 이루어진다.



<그림 4> 그룹 기반 가쉽 프로토콜의 흐름도

그룹 기반 가쉽 프로토콜의 알고리즘은 시스템 내 운용상의 효율성을 위해 부분 뷰와 그룹을 구성하기 위한 자가 조직 단계와 메시지 전파 단계로 구성된다.

자가 조직 단계는 시스템 초기에 한 번 이루어지는 단계로, 메시지 전파가 이루어지기 전에 부분 뷰를 생성하고 그룹의 수와 그룹의 크기에 따라 그룹을 생성하는 단계이다. 이후, 자가 조직 단계를 거쳐 생성된 그룹 정보를 가지고 메시지 전파 단계가 수행된다.

<알고리즘 1>은 자가 조직 단계의 의사코드로, 조직할 그룹의 개수와 각 그룹이 가지는 노드들의 크기가 설정된 시스템 매개변수의 값을 불러온다(줄:2~3).

다음 절차로 피어 샘플링 서비스에 의해 임의적으로 선택된 노드를 부분 뷰에 할당한다(줄:4~6). 이어 동일한 방법으로 불러온 매개변수를 기반으로 반복문을 통해 노드들은 자신의 그룹에 포함된 노드들의 정보 목록을 가지게 된다.

만약 피어 샘플링 서비스에 의해 선택된 노드가 어떤 그룹에 속해 있다면(줄:10), 노드 정보를 다시 가져온다. 또, 가져온 노드 정보가 이미 그룹 목록에 존재한다면, 해당 노드 정보를 그룹 목록에서 제거하고 다시 피어 샘플링 서비스를 호출하여 노드를 가져온다(줄:14~17). 이렇게 추가된 그룹 정보에 의해 각 노드의 그룹 식별자와 그룹 크기 필드에 해당 정보가 설정된다(줄:19~22). 마지막으로, 그룹의 대표자를 지정하고 그룹 구성원 목록 설정에 사용된 임시변수를 비운다(줄:23~24).

<알고리즘 1> 자가 조직 알고리즘

```
1: do Self-organization
2:     groupCount ← call getGroupCount()
3:     groupSize ← call getGroupSize()
4:     for each n ∈ partialView
5:         n ← getPeer()
6:     end for
7:     for i < groupCount
8:         for j < groupSize
9:             peer ← getPeer()
10:            if peerGroupID != ∅ then
11:                continue
12:            end if
13:            call addGroupList(peer)
14:            if checkDuplicate(groupList) then
15:                call removeGroupList(peer)
16:                continue
17:            end if
18:        end for
19:        for each n ∈ groupList
20:            call setGroupList(groupList)
21:            call setGroupProperty(groupID, groupSize)
22:        end for
23:        call setGroupRepresantative()
24:        call clearGroupList()
25:    end for
26: end do
```

이러한 단계를 통해, 각 노드들은 자신이 속해 있는 그룹의 식별자를 포함한 그룹에 속한 노드들의 정보를 관리한다. 각 노드는 최대 하나의 그룹에만 속할 수 있으며, 만약 전체 노드 수가

|Group|×|{n:g_i∈Group}| 보다 크다면 어떠한 그룹에도 속하지 않는 노드가 존재할 수 있다.

<알고리즘 2> 메시지 전파 알고리즘

```
1: do Dissemination
2:     if myself == groupRepresentative then
3:         for each n ∈ groupView
4:             SendMessages(n, message, myself)
5:         end for
6:     end if
7:     peer ← n ∈ partialView
8:     while myGroupID == peerGroupID
9:         peer ← n ∈ partialView
10:        if myGroupID == peer.GroupID then
11:            continue
12:        end if
13:    end while
14:    SendMessages(peer, message, myself)
15: end do
```

<알고리즘 2>는 메시지 전파 단계의 의사코드로써, 부분 뷰에 포함된 노드뿐만 아니라 그룹 목록에 존재하는 노드들에게 메시지를 전파하는 과정이 추가된 형태이다. 이 단계에서 사용하는 메시지 전달 방식은 모드에 따라 푸시, 풀, 푸시-풀 방식이 사용될 수 있다.

만약 자가 조직 단계를 거쳐 생성된 그룹 구성원 목록을 가지고 있으면서 자신이 그룹의 대표자라면, 그룹 구성원들에게 메시지를 전파한다(줄:2~6). 이 절차의 효과로 그룹 구성원들과 발생할 수 있는 중복 메시지를 제한할 수 있다.

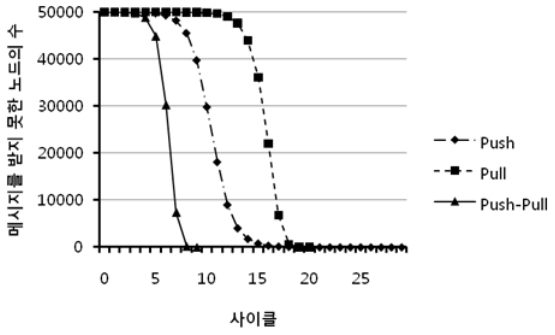
다음으로, 부분 뷰에 속한 구성원 중 하나의 노드를 임의적으로 선택하여(줄:7) 메시지를 전달하게 된다. 이 때, 부분 뷰에서 선택한 가쉽 대상이 자신의 그룹에 속해 있는 노드라면, 자신의 그룹이 아닌 노드를 부분 뷰에서 선택하도록 한다(줄:8~13). 이러한 과정을 거친 다음에 실질적으로 부분 뷰에서 선택한 노드에게 메시지를 전달한다(줄:14).

4. 성능 평가

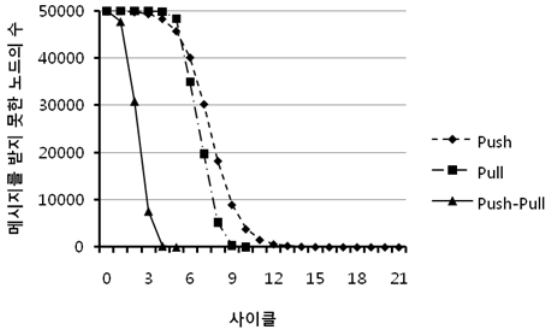
그룹 기반 가쉽 프로토콜의 유효성을 평가하기 위해 Peersim[9]을 사용하여 실험 환경을 구성하였다. Peersim은 분산 환경에서 다양한 프로토콜의 확장적인 성능 실험을 위해 자바(Java) 언어로 구현된 시뮬레이터로, 본 연구에서 사용한 환경 설정은 <표1>과 같다.

<표 1> 실험에 사용한 환경 변수 및 설정 값

시스템 매개변수	설정 값
N (전체 노드의 수)	50,000
k (부분뷰가 가지는 노드의 수)	20
Cycle	30
Fanout	1
G (그룹의 수)	20, 40, 60, 80
M (한 그룹의 구성원의 수)	100, 200, 300, 400

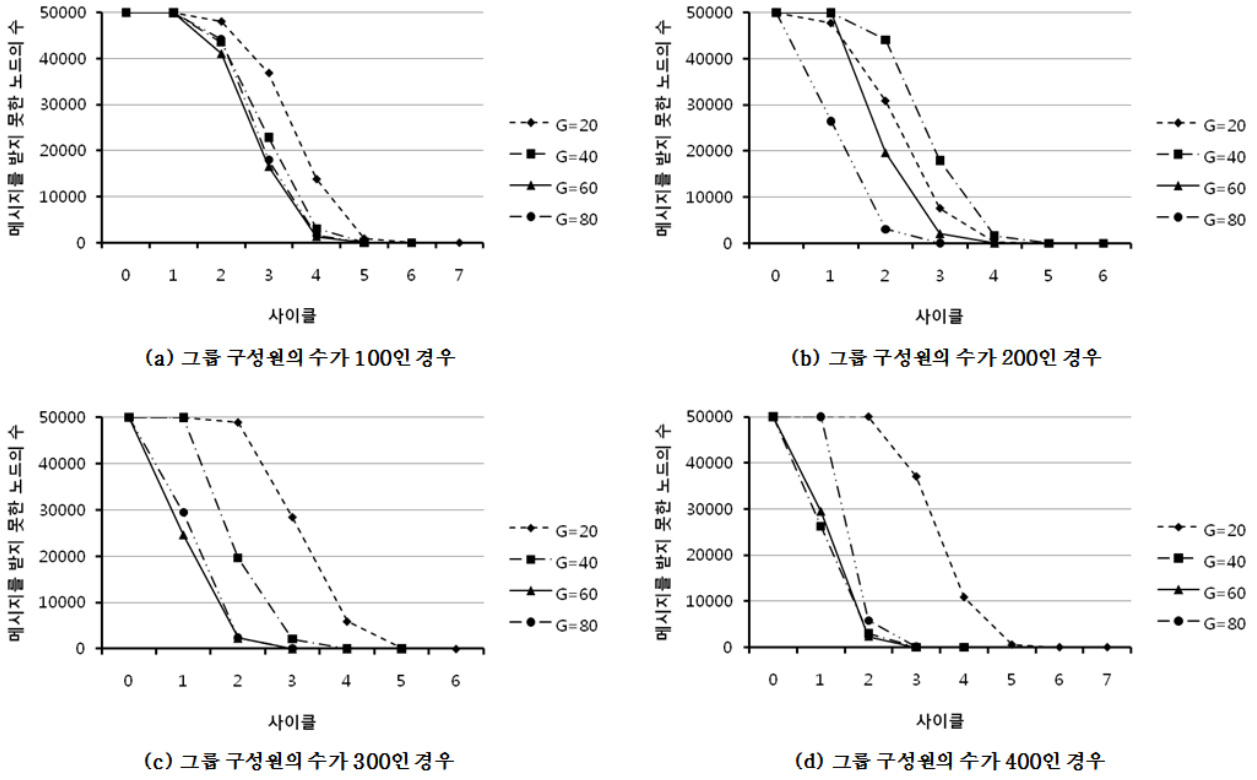


<그림 5> 플랫형 프로토콜의 푸시, 풀, 푸시-풀 모드 비교



<그림 6> G=40, M=200인 경우 그룹 기반 가쉽 프로토콜의 푸시, 풀, 푸시-풀 모드 비교

실험 초기화 과정에서 각 노드들은 시스템 내에서 유일한 식별자를 가지게 되며, 전체 노드 중



<그림 7> 그룹 수와 그룹 구성원의 수에 따라 메시지를 받지 못한 노드의 수(*G*: 그룹의 수)

정보를 가진 하나의 노드에서 메시지 개시가 이루어지도록 구성하였다.

또한, 그룹 기반 가쉽 프로토콜의 성능 실험을 위해 시스템 매개변수로 *G*(그룹의 수)와 *M*(한 그룹이 가지는 구성원의 수)을 추가하였으며, *G*와 *M*의 설정 값의 조합에 따라 메시지를 전달 받지 못한 노드의 수를 측정하였다.

<그림 5>는 플랫형 가쉽 프로토콜의 푸시 모드, 풀 모드, 푸시-풀 모드를 비교한 그림이다. 푸시, 풀, 푸시-풀 모드에서 모든 노드가 메시지를 전달 받기 위해 필요한 사이클의 수는 각각 28, 20, 9이다. 이 성능 평가로 알 수 있는 점은 푸시 모드가 풀 모드에 비해 사이클 초반에는 메시지 전파가 빠르게 이루어지지만 모든 노드가 메시지 전달을 받기 위해 더 많은 사이클의 수를 요구한다는 것이다.

이는 푸시 모드가 풀 모드에 비해 정보를 가지고 있는 노드의 수가 적을 때 메시지의 전파율이 좋은 반면, 풀 모드는 푸시 모드에 비해 정보를 가지고 있는 노드의 수가 많을 때 메시지를 가져올 확률이 높음을 말해준다.

<그림 6>은 그룹 기반 가쉽 프로토콜에서 시스템 매개변수 *G*의 값을 40, *M*의 값을 200으로 설정하였을 때의 푸시, 풀, 푸시-풀 모드를 비교한 그림으로, 플랫형 프로토콜에 비해 100%의 전달가능성을 만족하기 위한 사이클의 수가 각각 21, 10, 5임을 나타낸다.

또한, 플랫형, 그룹 기반 프로토콜 모두 푸시-풀 모드를 사용하였을 경우, 푸시 또는 풀 모드를 사용한 경우보다 모든 노드가 메시지를 전달받기 위해 필요한 사이클의 수가 더 적은 것을 알 수 있다. 따라서, 이후에 나오는 실험에는 모두 푸시-풀 모드를 사용한다.

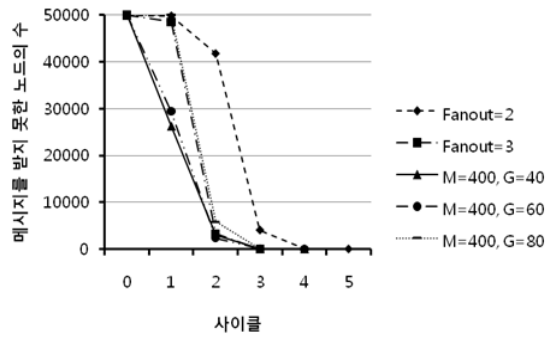
<그림 7>은 노드의 수가 50,000인 경우, 그룹의 수와 그룹 구성원의 수에 따른 실험 결과를 보여준다. <그림 7(a)>는 그룹 구성원의 수가 100인 경우 그룹의 수가 20, 40, 60, 80일 때, 각각 7, 6, 6, 6의 사이클이 요구됨을 보여준다. 이는 전체 노드 중에서 그룹을 가지는 노드의 비율이 4%인 경우에도 플랫형 프로토콜과 비교했을 경우, 성능

상의 이점이 있음을 보여준다.

또한, 그룹의 수와 그룹 구성원의 수가 점차적으로 증가함에 따라 100%의 메시지 도달가능성을 만족하기 위한 사이클의 수가 줄어드는 것을 확인할 수 있다. 특히, <그림 7(d)>는 그룹 구성원의 수가 400인 경우, 그룹의 수를 20씩 80까지 증가하였을 때, 각각 7, 4, 4, 4의 사이클이 필요함을 보여준다. 이는 그룹 구성원의 수가 400이고 그룹의 수가 40인 경우와 플랫형 가쉽 프로토콜에서 요구되는 사이클의 수를 비교했을 때 약 55%의 성능 이점이 있음을 보여준다.

그룹 기반 가쉽 멀티캐스트 프로토콜도 기존의 가쉽 멀티캐스트 프로토콜과 마찬가지로 확률적 모델에 기반을 두고 있어, 그룹의 수와 그룹의 크기가 크다고 해서 항상 좋은 성능을 보여주진 않는다. 한 예로, <그림 7(d)>의 그룹의 수가 80인 경우를 보면 사이클 1에서 메시지를 받지 못한 노드의 수의 감소가 거의 없는 것으로 나타난다. 이러한 현상은 초기에 메시지를 가지고 있는 하나의 노드가 그룹에 속하지 않았거나 그룹 내에서 메시지 교환 순서가 나중일 때 나타날 수 있다.

한편, *Fanout*이 1인 경우 사이클마다 발생하는 메시지 페이로드(Payload)는 플랫형 가쉽 프로토콜이 N개인 반면, 그룹 기반 가쉽 프로토콜을 사용했을 경우 $N+(|\text{Group}|\times|\{n:g_i\in\text{Group}\}|-|\text{Group}|)$개로 그룹 기반 가쉽 프로토콜이 더 많은 메시지 페이로드를 발생시킴을 알 수 있다.



<그림 8> 플랫형 프로토콜과 그룹 기반 프로토콜의 비교

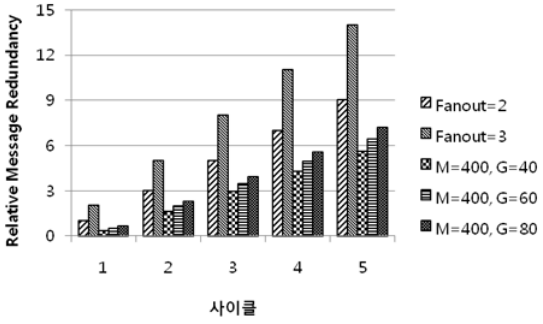
<그림 8>은 전체 노드의 수가 50,000인 경우, *Fanout*이 2와 3일 때 플랫형 가쉽 프로토콜과 *Fanout*이 1이고 그룹 구성원의 수가 400이고 그룹의 수가 40, 60, 80인 경우를 비교했을 때, 요구되는 사이클의 수는 각각 5, 3, 4, 4, 4임을 보여준다. 각 사이클마다 발생하는 메시지 페이로드의 수는 각각 100,000, 150,000, 65,960, 73,940, 81,920으로, *Fanout*이 2일 때의 플랫형 가쉽 프로토콜을 사용했을 때보다 *Fanout*이 1인 경우 그룹의 수가 40이고, 그룹 구성원의 수가 400일 때의 그룹 기반 가쉽 프로토콜이 더 적은 양의 메시지 페이로드를 발생시키면서도 요구되는 사이클의 수가 적다는 것을 실증적으로 보여준다.

추가적으로, 발생하는 중복 메시지를 측정하기 위한 지표로 [10]에서 제안한 RMR(Relative Message Redundancy)을 사용하였으며 정의는 아래와 같다:

$$\left(\frac{M}{N-1}\right)-1$$

여기서 N은 전체 노드의 수를 나타내고, M은 한 사이클당 발생하는 메시지 페이로드의 수를 의미한다.

RMR의 값이 0에 가까울수록 페이로드당 각 노드가 한 번의 메시지를 주고 받는다는 것을 나타내고, 값이 높을수록 중복 메시지가 많다는 것을 나타낸다.



<그림 9> 사이클에 따라 누적된 RMR의 값

<그림 9>는 사이클이 진행됨에 따라 누적된 메시지 페이로드에 대한 RMR 값을 보여준다. 플랫형 가쉽 프로토콜에서 *Fanout*이 2와 3일 경우와 *Fanout*이 1인 경우 그룹 구성원의 수가 400이고 그룹의 수가 40, 60, 80인 경우를 비교했을 때, 사이클의 수가 증가할수록 RMR 값의 편차가 커지는 것을 볼 수 있다.

특히 100%의 메시지 도달가능성을 만족하기 위한 사이클의 수에 대한 RMR의 값을 비교했을 때, *Fanout*이 3인 플랫형 가쉽 프로토콜의 RMR 값은 약 8(사이클 3)인데 반해 그룹의 수가 40이고 그룹 구성원의 수가 400인 경우의 그룹 기반 가쉽 프로토콜의 RMR 값은 약 4.27(사이클 4)인 것을 확인할 수 있다. 이는 RMR 값을 비교했을 때 그룹 기반 가쉽 프로토콜이 약 46% 정도 앞선 결과를 도출한다는 것을 보여준다.

5. 결 론

클라우드 컴퓨팅 환경에서는 자원 노드들의 상태 정보를 기반으로 사용자의 요구에 따라 즉각적인 프로비전(Provision)이 요구된다. 대규모, 동적인 자원의 특성을 가진 클라우드 컴퓨팅 환경에서 각 자원들의 상태 정보를 신속히 파악하는 일은 클라우드 서비스 제공자의 입장에서 중요하다.

본 연구에서는 가쉽 프로토콜에서 100%의 도달가능성을 만족하기 위해 발생하는 중복 메시지와 전파 지연을 줄이기 위한 그룹 기반 가쉽 프로토콜을 제안하였다. 그룹 기반 가쉽 프로토콜은 부분 뷰와 그룹을 생성하기 위한 자가 조직 단계와 이 단계를 거쳐 생성된 정보를 기반으로 메시지 전달이 이루어지는 메시지 전파 단계로 구성된다.

제안한 그룹 기반 가쉽 프로토콜은 그룹을 생성하기 위해 필요한 어떠한 지식을 요구하지 않으며, 기존 가쉽 프로토콜이 가지는 효율성과 확장성의 특성을 지니면서 보다 더 적은 메시지의 수로 원자적 메시지 전달을 가능하게 하였다.

참 고 문 헌

[1] Buyya, R., Yeo, C. S., & Venugopal, S. (2008). *Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities.* Los Alamitos: Ieee Computer Soc.

[2] DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., et al. (2007). Dynamo: amazon's highly available key-value store. *SIGOPS Oper. Syst. Rev.*, *41*(6), 205-220.

[3] Jelasity, M., Guerraoui, R., Kermarrec, A.-M., & Steen, M. v. (2004). *The peer sampling service: experimental evaluation of unstructured gossip-based implementations.* Paper presented at the Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware, Toronto, Canada.

[4] Ganesh, A. J., Kermarrec, A.-M., & Massoulié, L. (2001). *SCAMP: Peer-to-Peer Lightweight Membership Service for Large-Scale Group Communication.* Paper presented at the Proceedings of the Third International COST264 Workshop on Networked Group Communication.

[5] Ganesh, A. J., Kermarrec, A.-M., & Massoulié, L. (2002). *HiScamp: self-organizing hierarchical membership protocol.* Paper presented at the Proceedings of the 10th workshop on ACM SIGOPS European workshop, Saint-Emilion, France.

[6] Voulgaris, S., Gavidia, D., & van Steen, M. (2005). CYCLON: Inexpensive Membership Management for Unstructured P2P Overlays. *Journal of Network and Systems Management*, *13*(2), 197-217.

[7] Leitao, J., Pereira, J., & Rodrigues, L. (2007). *HyParView: A Membership Protocol for Reliable Gossip-Based Broadcast.* Paper presented at the Dependable Systems and Networks, 2007. DSN '07. 37th Annual IEEE/IFIP International Conference on.

[8] Matos, M., Sousa, A., Pereira, J., Oliveira, R. C., Deliot, E., & Murray, P. (2009). *CLON: Overlay Networks and Gossip Protocols for Cloud Environments.* Paper presented at the Proceedings of the Confederated International Conferences, CoopIS, DOA, IS, and ODBASE 2009 on On the Move to Meaningful Internet

Systems: Part I, Vilamoura, Portugal.
[9] Jelasity, M., Montresor, A., Jesi, G. P., & Voulgaris, S. The Peersim simulator, from http://peersim.sf.net
[10] Leitao, J., Pereira, J., & Rodrigues, L. (2007). *Epidemic Broadcast Trees.* Paper presented at the Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems.

진 성 호

2002 고려대학교 컴퓨터교육과(이학사)
2004 고려대학교 컴퓨터교육과(교육학석사)
2010 고려대학교 컴퓨터교육과(이학박사)
2010 ~ 현재 고려대학교 정보창의교육연구소 연구교수
관심분야: 그리드 컴퓨팅, 클라우드 컴퓨팅, 분산 시스템, 컴퓨터교육
e-Mail: wingtop@korea.ac.kr

임 종 범

2009 백석대학교 정보통신학부(공학사)
2009 ~ 현재 고려대학교 컴퓨터교육과 석사과정
관심분야: 분산시스템, 클라우드컴퓨팅
e-Mail: jblim@korea.ac.kr

유 헌 창

1989 고려대학교 이과대학 전산과학과(이학사)
1991 고려대학교 대학원 전산과학과(이학석사)
1994 고려대학교 대학원 전산과학과(이학박사)
1998 ~ 현재 고려대학교 컴퓨터교육과 교수
2006 ~ 2010 한국컴퓨터교육학회 부회장
관심분야: 그리드 컴퓨팅, 분산시스템, 결함포용시스템, u-learning
e-Mail: yuhc@korea.ac.kr

이 종 혁

2004 고려대학교 컴퓨터교육과 (이학사)
2006 고려대학교 컴퓨터교육과 (이학석사)
2006 ~ 현재 고려대학교 컴퓨터교육과 박사과정
관심분야: 클라우드 컴퓨팅, 그리드 컴퓨팅, 분산 시스템, 컴퓨터 교육
e-Mail: spurt@korea.ac.kr

이 화 민

2000 고려대학교 컴퓨터교육과 (이학사)
2002 고려대학교 컴퓨터교육과 (교육학석사)
2006 고려대학교 컴퓨터교육과(이학박사)
2006 ~ 2007 특허청 전자상거래심사팀 통신사무관
2007 ~ 현재 순천향대학교 컴퓨터학부 조교수
관심분야: 그리드 컴퓨팅, 분산시스템, 결함포용시스템, 자원 스케줄링
e-Mail: leehm@sch.ac.kr