계층적 Ring 기반의 그리드 멤버쉽 프로토콜의 설계

구태완° 홍성준 엄상용 이광모
한림대학교 컴퓨터공학과
{taewani°, teferi, suhmn, kmlee}@hallym.ac.kr

Design of Grid Membership Protocol based on Hierarchical Ring

Taewan Gu°, S.J. Hong, S. Uhmn, K.M. Lee
Dept. of Computer Engineering, Hallym University

요 약

그리드 환경은 원격의 이질적인 자원들을 서로 공유하며 상호 접근이 가능하도록 하여 사용자의 작업을 처리할 수 있는 메커니즘을 제공한다. 이때 그리드에 참여하는 각 노드들은 그리드에 자유롭게 참여 및 탈퇴가 가능하다는 점에서 P2P(Peer-to-Peer) 네트워크 구조와 유사하다. 또한 그리드 정보 서비스의 배치(deployment)와 특정한 자원 검색 서비스의 측면을 고려할 때 P2P 구조는 그리드 환경에서 유용하게 활용 될 수 있다. 따라서 그리드를 설계함에 있어 기존의 방법과는 달리 P2P 형식으로 그리드를 구성하는 것은 시스템의 확장성과 이에 따른 효율성을 개선하는 중요한 요소가 된다. 본 논문에서는 P2P 형식의 그리드를 구성하는데 필요한 계층형 Ring 기반의 멤버쉽 프로토콜을 제안한다. 계층형 Ring 기반의 멤버쉽 프로토콜은 제한된 네트워크 연결만을 가지게 되므로 멤버쉽을 위한 통신 비용을 감소 시킬 수 있게 되며, 각 노드에서 유지해야 하는 멤버쉽 정보들이 지역적인 정보들만을 대상으로 하기 때문에 멤버쉽 정보 유지에 따른 관리 비용 또한 감소하게 되는 장점을 갖게 된다.

1. 서 론

그리드 환경은 원격의 이질적인 자원들을 서로 공유하고, 상호 접근토록 하여 사용자의 작업을 처리 할 수 있는 메커니즘을 제공한다[1]. 복잡한 응용 프로그램들을 사용하는 그리드에서 호스트들이 증가함에 따라 발생 할 수 있는 기능들의 병목 현상(bottleneck)을 회피하기 위해 그리드는 분산된 형태로 구성되어야 한다. 이러한 관점에서 볼 때 P2P(Peer-to-Peer) 모델은 그리드의 규모에 대한 유연성을 제공할 수 있다. 즉, P2P의 본질적인 개념과 비계층적 분산 구조를 이용하면 기존의 그리드에서의 문제시되는 확장성 문제를 해결 할 수 있게 된다. 그리고 정보서비스의 배치(deployment)와 특정한 자원검색 서비스의 측면을 고려할 때 P2P 구조는 그리드 환경에서 매우 유용하게 활용될 수 있다[1]. 또한 그리드에 참여하는 자원의 규모와 사용자의 요구가 지속적으로 증가함에 따라 그리드 환경은 P2P가 지향하는 방향으로 발전하게 된다. 하지만 P2P 기술의 경우 크게 네트워크 구성에 있어 크게 구조적인 속성을 가지는 기술(예를 들어 Chord[2], Tapestry[3], CAN[4] 등)과 비구조적인 속성을 가지는 기술(예를 들어 Gnutella[5], KaZaA[6] 등)로 구분 할 수 있다. 이것은 그리드에 참여하는 각 노드들이 자유롭게 참여/탈퇴가 가능하다는 특성을 고려한다면 비구조적인 구성을 갖는 P2P 네트워크와 유사하다[7].

그러나 현재까지의 그리드 시스템에서 대상이 되는 자원들은 연구소나 공공 기관에서 소유하고 있기 때문에 매우 안정적이므로 멤버쉽 문제들은 OGSA(Open Grid Service Architecture)와 WSRF(Web Service Resource Framework)에서와 같이 중앙 집중형으로 계층적인 구조를 가지고 효율적으로 관리된다. 그러나 P2P에서는 노드와 자원들의 연결이 불규칙하게 구성될 수 있기 때문에 P2P 스타일의 분산 접근 방법은 기존의 중앙 집중형 계층 구조에 비해 매우 효율적이며 고장 허용의 측면에서 뛰어난 성능을 가지게 된다. 그러므로 이러한 접근법에 유사한 P2P 기술은 그리드에 참여하는 자원의 규모와 사용자의 요구가 지속적으로 증가함에 따라 그리드를 구성함에 있어 P2P 기술과의 접목이 이루어 지고 있는 실정이다[11].

본 논문에서는 HRing 프로토콜이라는 계층형 Ring 구조를 제안한다. 이 HRing 프로토콜은 기존의 계층 구조를 갖는 멤버쉽 서비스의 장점을 그대로 유지하지만 여기서 발생할 수 있는 각 노드의 멤버쉽 연산 오버헤드, 이웃 노드 정보의 유지에 따른 오버헤드를 감소시키고, 멤버들의 변화에 유연하게 대처하면서 확장성을 갖는다는 장점을 갖는다.

2. 관련 연구

일반적인 그리드 시스템은 중앙 집중형 모델을 사용하고 있다. 예를 들면 GT3(Globus Toolkit 3)에서 사용되는 정보 모델은 그리드 서비스의 특수한 형태인 인덱스 서비스를 사용한다[8]. 이러한 인덱스 서비스는 그리드 호스트에서 제공하는 자원들의 메타 데이터와 서비스 데이터를 통합하는 역할을 하며 가상 조직(Virtual Organization, VO)마다 한 개씩 인덱스 서비스를 갖게 되는데 대규모 그리드 시스템의 경우 여러 개의 인덱스 서비스가 계층적인 구조를 이루며 구성되기도 한다. 그러나 기존의 연구에 따르면 중앙 집중형 모델의 경우 그리드의 규모에 대한 유연성이 적기 때문에 대규모 그리드에서는 적절한 성

능을 제공하지 못하게 될 수 있다. 또한 빈번하게 발생하는 질의를 처리함에 있어 인덱스 서비스에 상당한 크기의 부하가 발생하게 되며 해당 인덱스 서비스에서의 병목현상이 문제가 될 수 있다[8].

최근의 그리드 시스템들은 대규모 분산형 모델을 기반으로 구성되고 있기 때문에 [9]에서는 P2P 시스템과 그리드 시스템의 유사성에 대해 언급하고 있으며 그 활용 가능성에 대해서도 비교 분석하였다.

P2P의 경우 네트워크를 구성함에 있어 크게 2개의 서로 다른 접근법을 가지게 된다. 우선 비구조적 속성을 가지는 기술[5,6]이 있는데 이것은 전체 네트워크의 정보를 따로 유지하지 않으면서 자유롭게 가입/탈퇴를 수행할 수 있다. 두번째로 구조적인 속성을 가지는 기술[2,3,4]이 있는데 이것은 매우 정형적인 논리 네트워크를 가지고 네트워크 상에서 사용자의 요구를 전달하기 위해 DHT(Distributed Hash Table)을 사용하게 된다. 그러나 구조적인 속서을 가지는 기술은 중앙 집중형의 인덱스를 유지해야 하므로 확장성에 문제가 발생하게 된다[12].

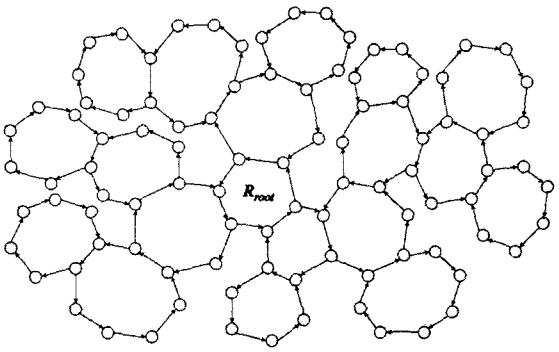
[10]에서 사용된 비구조적 속성의 접근법은 모든 노드들이 주기적으로 심장발진(heartbeat) 메시지를 사용하며 해당 메시지는 모든 노드에서 주기적으로 송수신이 일어난다. 이때 심장발진 메시지에는 각 노드들의 멤버쉽 정보들이 포함되어 있으며 각 노드에서는 전체 노드들의 멤버쉽 정보를 유지하기 위해 멤버쉽 디렉토리를 유지하게 되고, 각 멤버쉽 디렉토리는 서로 독립적인 특성을 가지게 된다. 그러므로 이러한 기술은 A2A(All-to-All) 형식의 접근 기술이라 할 수 있다. 하지만 A2A의 경우 전체 노드의 정보를 유지해야 하기 때문에 연결 비용과 각 노드에서 독립적으로 유지되는 멤버쉽 디렉토리 유지에 따른 비용이 상당하다는 문제가 있다[13]. 이와 유사하게 [12]에서는 Gossip 형식의 멤버쉽 서비스를 이용하였다. 이는 각 노드가 임의의 이웃 노드 집합(partial view)을 선택하여 해당 집합의 노드들에게 현재의 멤버쉽 정보를 전송하면 해당 멤버쉽 정보를 수신하는 노드들은 수신 정보를 바탕으로 자신의 디렉토리에 있는 멤버쉽 정보를 업데이트하게 된다. 그러나 이 방식은 낮은 지연 비율과 높은 대역폭을 갖는 네트워크에서는 효과적일 수 있으나 각 노드에서 유지하게 되는 이웃 노드 집합이 안정화 되는데 소요되는 시간이 많아지게 된다는 단점을 가지고 있다[13].

[13]에서는 위의 두 가지 접근법이 가지는 문제점에 대해 계층적 구조를 제안하였다. 이 계층적 멤버쉽 서비스는 가상의 트리 구조를 이용하여 멤버쉽 정보의 계층을 구성하게 되며 상위 계층에서 하위 계층으로 멤버쉽 정보의 전달이 가능하도록 설계되어 있다. 하지만 이러한 계층 구조가 가지는 문제 중 하나는 단일점 실패 문제(single point of failure)로써 최상위 노드의 탈퇴 시 발생할 수 있는 문제에 대해 따로 고려해야 한다는 점이다.

3. 계층형 Ring 기반의 멤버쉽 프로토콜(HRing Protocol)

멤버쉽을 위한 그래프의 구성은 계층적 구조를 갖는 Ring 형식의 그래프 $G=(V, E)$로 구성된다. 그리고 멤버들의 정보 전달을 위해 MCM(Membership Control Message)을 정의하고 노드의 간선(E)를 통해 이웃한 멤버들에게 전송하게 된다. 이때 그래프를 구성하는 정점 $v \in V$는 $\deg(v)=4$의 특징을 가지며 임의의 정점 $v \in V$에서의 간선 $e_v \in E$는 $e_v=\{out_{main}, out_{sub}, in_{main}, in_{sub}\}$으로 정의된다. 또한 MCM의 전송은 시간주기 t 에서 out_{main}과 out_{sub}를 이용하여 이루어지며 MCM은 전송할 정점 즉, 자신의 정보를 포함하여 전송하게 된다. 그리고 in_{main}과 in_{sub}는 자신으로 송신되는 정점의 정보를 갖게 된다.

멤버쉽 구조는 Ring을 기반으로 계층적 구조를 가지므로 각 Ring은 고유한 식별자($RngID$)를 가지게 된다. 이때 최상의 계층의 Ring을 R_{root}라고 하고 하위 계층의 Ring을 R_{child}, 상위 계층의 Ring을 R_{parent}, 계층상의 내부 Ring을 $R_{internal}$, 그리고 단말 Ring은 R_{leaf}라고 정의한다. 다음의 [그림 1]은 계층적 Ring 기반의 그래프를 구조를 나타낸 것이다.



[그림 1] HRing의 구조

또한 Ring을 구성하는데 사용되는 요소로서 분할 임계값(split threshold) k를 정의한다. 이것은 Ring이 계층구조를 갖게 하는데 사용되는 요소이다.

이렇게 구성되는 멤버쉽 그래프의 구성은 전체 멤버들의 정보를 모두 알고 있지 않으며, 멤버쉽 유지를 위한 연산도 일부 정점을 제외하고는 하나의 간선만을 활용하게 되므로 그래프 구성에 따른 부하도 줄어 들 수 있다. 또한 자신이 속한 계층 내부의 정보만을 유지하게 되므로 지역적인 그래프의 변화에 쉽게 대응할 수 있다는 장점을 가지게 된다.

3.1 멤버쉽 제어 메시지(Membership Control Message, MCM)

MCM은 Ring에 포함되는 노드의 out 간선을 따라 시간주기 t에 동시에 전송되어 멤버쉽 그래프를 유지하는 기능을 담당하며 비동기 통신을 수행한다. MCM에 포함되는 정보에는 Ring에 포함되는 노드의 개수, 각 노드의 IP 주소, 현재 Ring에서의 k값, 그리고 R_{leaf}까지의 깊이가 있다.

시간 t에 out_{main}을 통해 전송하고자 하는 MCM을 $OutMsg_t$ 라 하고, in_{main}을 통해 수신되는 MCM을 $InMsg_t$ 라고 할 때, $OutMsg_t = InMsg_t$이면 해당 Ring에서 멤버쉽 정보의 변화가 없음으로 간주하고 현재 멤버쉽 정보를 그래도 유지하게 되고 그렇지 않으면 멤버쉽 정보가 변경되었음을 의미하므로 $OutMsg_t$를 $InMsg_t$로 변경하게 된다.

3.2 MCM 의 흐름

MCM은 정점간의 연결이 완료 된 다음 각각의 노드에서 생성된다. 이렇게 생성된 MCM은일정한 시간 주기에 따라 노드의 out_{main}과 out_{sub}를 따라서 전송되며 자신이 속한 Ring에 있는 다른 정점들과 서로 공유하게 되며 소속된 Ring의 외부로는 전송이 되지 않는다. 그러므로 계층구조를 갖는 Ring의 수가 증가하더라도 유지해야 할 정보의 양은 크게 늘어나지 않는다. 이런 경우 부모 Ring에 속한 노드들은 자식 Ring에 속한 노드들의 정보를 알 수 없게 되므로 멤버쉽 정보의 순환을 위해 각 Ring에서는 outdeg(v)≠2인 임의의 노드 $v \in R_{leaf}$가 임의의 정점 $v' \in R_{root}$에 대한 연결을 가지도록 한다.

또한 MCM이 전송될 때 Ring의 상태 정보 즉, k값의 변화를 인식하기 위해 out_{main}에 있는 노드로 임의의 패킷을 전송하여 RTT(Round Trip Time)를 MCM에 기록하게 된다. 이렇게 하여 그래프에서 발생되는 k값을 판단하게 된다. 이렇게 MCM에 기록된 k는 MCM이 전송될 때 해당 Ring에 포함되어 있는 모든 노드에게 전송되며 전송이 완료되면 Ring에 대한 k가 설정되며 이를 이용하여 노드 참여시 계층을 생성할 것인지에 대한 여부를 판단하게 된다.

3.3 초기 멤버쉽 Ring 구성

초기 멤버쉽을 구성하는 노드의 집합을 $M=\{n_1, n_2, \cdots, n_i\}$로 정의하며, 이때 i는 전체 노드의 개수이다. 이때 각 노드들은 연결 가능한 노드, 즉 집합 M의 원소들을 이미 알고 있음을 가정한다. 또한 초기 Ring을 구성하기 위한 링크 $Link$는 다음과 같이 정의하며 각 링크는 두 노드간의 가용한 대역폭을 나타내는 값(cost)을 가지게 된다. 이때 임의의 노드 n_1, n_2에 대해 n_1에서 n_2로의 연결은 n_1의 out_{main}에 n_2의 정보를 추가하고, n_2의 in_{main}에는 n_1의 정보를 추가하게 된다. 이렇게 생성된 링크들은 각기 가용한 대역폭을 가지며 다음과 같이 표현 될 수 있다.

$$\begin{aligned} Link_{1,2} &= n_1 \rightarrow n_2 = c_1 \\ Link_{2,3} &= n_2 \rightarrow n_2 = c_2 \\ &\vdots \\ Link_{i-1,i} &= n_{i-1} \rightarrow n_i = c_{i-1} \\ Link_{i,1} &= n_i \rightarrow n_1 = c_i \end{aligned}$$

그러므로 Ring에서 전체 가용한 대역폭($Link_{total}$)은 다음과 같이 정의된다. 이때 각 링크의 개수가 증가하면 그만큼 메시지 전달에 따른 비용이 증가하기 때문에 이러한 사항이 반드시 반영이 되어야 한다.

$$\begin{aligned} Link_{total} &= \prod_{j=2}^{i} Link_{j-1,j} \times Link_{i,1} \\ &= c_1 \times c_2 \times \cdots \times c_i \\ &= \prod_{j=1}^{i} c_j \end{aligned}$$

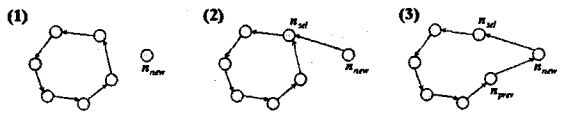
이 값을 이용하여 Ring에 대해 존재하는 분할 임계값(split threshold) k는 다음과 같이 정의 한다.

$$k = \sqrt{Link_{total}}$$

그리고 이때 k가 갖는 의미는 임의의 링크에 대한 c_i가 $c_i \geq k$ 가 되면 아래에 정의되는 멤버쉽 그래프의 분할 방법에 의해 Ring이 분할되어 계층구조를 가지게 되는 기준이 된다. 또한 k는 전체 노드의 개수에 독립적으로 계산될 수 있기 때문에 전체 네트워크에서 발생할 수 있는 불필요한 대역폭의 낭비를 막을 수 있다.

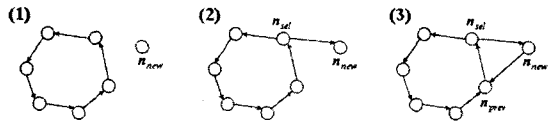
3.4 새로운 노드의 참여

새로운 노드(n_{new})가 멤버쉽에 참여하는 것은 기존의 Ring에 포함되는 경우와 새로운 Ring을 생성하여 분할하는 경우로 나누어 진다. 우선 먼저 새로운 노드가 기존의 Ring으로 포함되는 경우를 살펴보면 다음과 같다. 새로운 노드(n_{new})는 기존에 존재하는 Ring의 노드들 중에서 임의의 한 노드를 선택하게 되는데 이렇게 선택된 노드(n_{sel})에 참여 요청 REQ_{join}을 전송한다. 이때 REQ_{join}를 수신하는 노드, 즉 선택된 노드(n_{sel})에서 이미 outdeg(n_{sel})=2 이면 차선의 노드를 n_{sel}로 선택하여 REQ_{join} 을 전송하게 된다. 그러면 이렇게 선택된 노드 n_{sel}은 새로운 노드(n_{new})에게 REQ_OK_{join}을 전송한다. 그런 다음 n_{new}에서는 $Link_{new,sel}$가 k보다 작은 값을 가지고, RES_OK_{join}을 정상적으로 수신한다면 Ring이 분할되지 않는 경우이므로 n_{sel}와의 링크를 생성하고 자신의 out_{main}에 n_{sel}를 저장한다. 그리고 n_{sel}은 자신의 in_{main}의 노드(n_{prev})에게 n_{new}의 정보를 알려 주어 새로운 링크를 생성하게 된다. 그런 다음 새로운 MCM을 생성하면 새로운 노드의 참여가 완료된다. 다음의 [그림 2]은 위의 연산을 도식화 시켜 놓은 것이다.



[그림 2] Join Operation Steps when $Link_{new,sel} < k$

두 번째로 n_{new}가 새로운 Ring을 구성하여 분할 하는 경우는 다음과 같다. 먼저 n_{sel}이 포함되어 있는 Ring의 k에 대해 $Link_{new,sel} \geq k$이면 멤버쉽 분할 방법에 의해 Ring이 나눠지도록 참여 연산을 수행하게 된다. 이때 멤버쉽에 새로운 노드가 참여하는 알고리즘은 앞에서 언급된 경우와 마찬가지로 Ring의 노드들 중에서 임의의 한 노드를 선택하여 해당 노드(n_{sel})에 REQ_{join}를 전송한다. 이때 REQ_{join}을 수신하는 노드, 즉 선택된 노드(n_{sel})에서 outdeg(n_{sel})=2이면 차선의 노드에 REQ_{join}를 전송하게 된다. 그러면 n_{sel}은 자신의 out_{sub}에 새로운 노드를 추가한다. 그런 다음 새로운 노드에서는 자신의 in_{main}에 n_{sel}를 추가하고, n_{sel}의 in_{main}의 노드, 즉 n_{sel}의 이전 노드 n_{prev}를 자신의 out_{main}으로 저장 하게 된다. 이러한 과정이 종료되면 새로운 노드는 MCM 을 생성하게 된다. 다음의 [그림 3]는 위의 연산을 도식화 시켜 놓은 것이다.



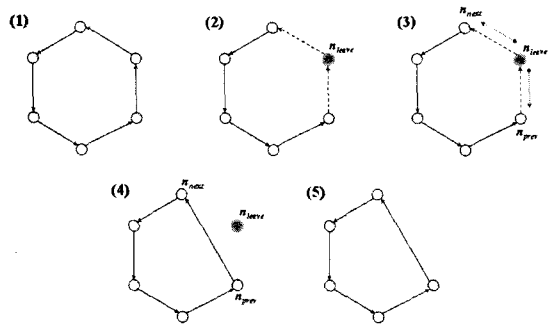
[그림 3] Join Operation Steps when $Link_{new,sel} \geq k$

3.5 노드의 탈퇴

멤버쉽 그래프로부터 임의의 노드가 탈퇴하고자 할 때에는 기본적으로 자신이 가지고 있는 모든 간선으로 자신의 탈퇴 사실을 알리게 된다. 즉 Ring에 참여하는 모든 노드에게 탈퇴 메시지를 멀티캐스팅 한다. 이때 기존의 그래프를 그대로 유지해야 하므로 탈퇴 사실을 알릴 때 탈퇴하는 노드의 연결 정보를 비롯한 몇 가지 정보를 추가로 전송하게 된다.

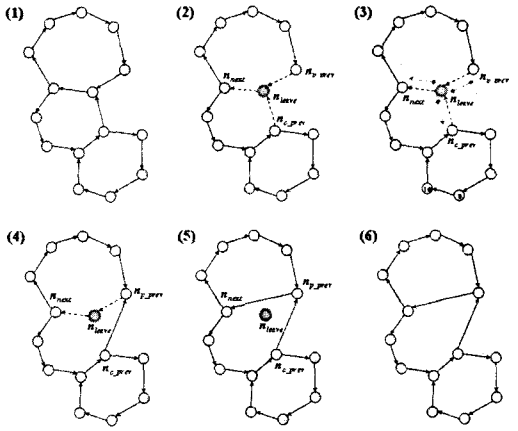
그리고 Ring으로부터 노드의 탈퇴는 3가지로 구분하여 동작하게 된다. 첫째, 임의의 Ring에 포함되어 있으며 in_{main}과 out_{main}만이 활성화 되어 있는 노드가 탈퇴하는 경우이다. 본 논문에서는 이것을 일반적인 경우의 탈퇴라고 부르기로 한다. 둘째, 탈퇴하고자 하는 노드를 n_{leave}라 할 때, indeg(n_{leave})=2인 경우이며 마지막으로 셋째, outdeg(n_{leave})=2인 경우이다. 이 경우에는 탈퇴하고자 하는 노드가 탈퇴 메시지를 전송하는 경우 여기에 포함되는 정보가 달라지게 된다.

우선 일반적인 경우의 탈퇴 연산을 살펴보면 n_{leave}를 기준으로 자신이 속한 상위 Ring에서 이전 노드를 n_{prev}, 이후의 노드를 n_{next}라고 하면 이 경우 탈퇴하고자 하는 노드 n_{leave}에 활성화 되어 있는 간선은 out_{main}과 in_{main}이다. 그러므로 n_{leave}는 n_{prev}와 n_{next}에 기록되어 있는 노드에게 자신의 탈퇴 사실을 알리게 된다. 이때 기존의 Ring을 그대로 유지 할 수 있어야 하므로 n_{leave}는 n_{next}에게는 자신의 in_{main}을 알려 주고 n_{prev}에게는 자신의 out_{main}을 알려 주게 된다. 그런 다음 n_{prev}는 n_{leave}로부터 수신한 노드를 자신의 out_{main}에 등록하고, n_{next}는 n_{leave}로부터 수신한 노드를 자신의 in_{main}으로 등록 하게 된다. 그런 다음 n_{leave}는 멤버쉽 그래프에서 삭제 된다. 다음의 [그림 4]은 일반적인 경우에 탈퇴 연산을 수행하는 경우이다.



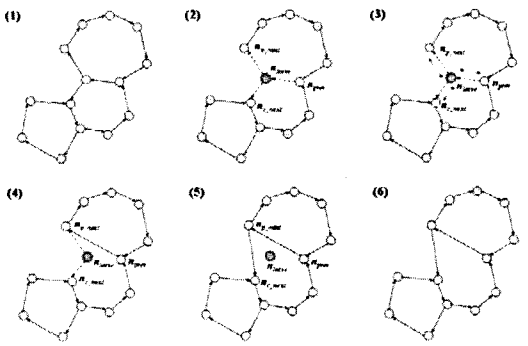
[그림 4] Leave Operation in General Case

두 번째로 indeg(n_{leave})=2인 경우에는 다음의 [그림 5]와 같이 탈퇴 메시지를 전송함에 있어 자신의 Ring에 속한 이전 노드(n_{p_prev})와 하위 Ring에 속한 이전 노드(n_{c_prev})로 탈퇴 메시지를 전송하게 된다.



[그림 5] Leave Operation when indeg(n_{leave})=2

마지막으로 outdeg(n)=2 인 경우에는 [그림 6]에서와 같이 탈퇴 메시지를 전송함에 있어 자신이 속한 Ring의 이전 노드(n_{prev})와 다음 노드(n_{p_next})와 하위 Ring의 다음 노드(n_{c_next})에게 탈퇴 사실을 전송하게 된다.



[그림 6] Leave Operation when outdeg(n_{leave})=2

3.6 멤버쉽 그래프의 분할

멤버십 그래프의 분할(splitting)은 새로운 노드가 참여하고자 할 때 분할 임계값 k에 근거하여 임의의 두 노드 n_a와 n_b에 대한 링크 $Link_{a,b}$에 대해 $Link_{a,b} \geq k$일 때 분할이 발생하게 된다. 단, 이미 Ring을 분할하여 구성하고 있는 노드들에 대해서는 더 이상 분할이 발생하지 않는다. 그래서 새롭게 참여하는 노드에서 $Link_{a,b} \geq k$이 되면 해당 요청을 처리 하는 것이 비효율적이 된다는 의미이다. 그러므로 새로운 Ring을 구성하여 독립적인 MCM을 이용하여 멤버쉽을 관리하는 것이 보다 효율적이라 할 수 있다. 멤버쉽 그래프의 분할이 발생하는 경우 새로운

Ring을 구성하는 방법은 다시 말해 Ring을 분할하게 되는 것은 참여 연산에서 노드를 선택하는 방법과 동일하다.

4. 분석 및 평가

멤버쉽 프로토콜의 성능을 측정하기 위해 기존의 몇 가지 연구와 비교하여 평가하도록 한다. 첫 번째 전체 노드가 멤버들의 정보를 모두 가지고 있는 접근 방법인 A2A(All-to-All)과 두 번째 각 멤버들은 부분 정보만을 가지고 있으며 이를 바탕으로 Gossip 형식의 메시지 전송을 수행하는 방법, 그리고 본 논문에서 제안된 HRing 프로토콜을 비교하도록 하며 각 프로토콜의 성능 비교를 위해 [표 1]과 같이 성능 평가 매트릭을 정의하도록 한다.

[표 1] 성능 평가 매트릭

통신 비용 (Communication Cost)	프로토콜을 수행함에 있어 통신 대역폭의 활용도
관리 비용 (Management Cost)	각 노드에서 멤버쉽 유지를 위해 필요로 하는 저장 공간의 크기와 정보의 갱신에 따른 연산 비율
안정화 시간 (Convergence Time)	멤버의 정보가 변경 되었을 때 해당 사실이 전체 노드에게 알려지기까지의 시간

4.1 통신 비용(Communication Cost)

[13]에 따르면 A2A의 경우 각 노드는 고정된 멀티캐스트 전송 대역폭을 가지며 이것은 노드의 개수에 대해 독립적이기 때문에 전체 네트워크 대역폭의 활용도는 $O(n^2)$이 된다. 또한 Gossip 기반의 프로토콜의 경우 Gossip 메시지에는 전체 멤버들에 대한 부분 정보들만을 포함하며 전체 멤버의 정보를 구성하기 위해 임의의 노드를 선택하여 메시지를 전달하게 된다. 그러므로 노드에서 유지하게 되는 부분 정보의 크기는 멤버쉽 정보에 대한 각 메시지의 크기 m과 노드의 개수 n에 대해 $m \times n$이 되기 때문에 대역폭 활용도는 $O(n^2)$ 이 된다[13]. 그러나 HRing 프로토콜의 경우 멤버 정보를 구성하기 위해 임의의 노드를 선택하지 않고 k에 근거하여 멤버를 구성하기 때문에 노드에서 유지하게 되는 멤버쉽 정보의 크기는 멤버쉽 정보에 대한 각 메시지의 크기 m과 임계값 k, 그리고 노드의 개수 n에 대해 $m \times k$가 된다. 그러나 앞 절에서 언급한 바와 같이 가입 및 탈퇴 시 가정하고 있는 사항으로 인해 통신 비용의 정확한 분석은 불가능 하다. 하지만 HRing에서의 노드 증가에 따른 통신비용은 선형적(linear)인 비용을 가짐을 예측할 수 있다.

4.2 관리 비용(Management Cost)

관리비용 $C_{management}$는 다음과 같이 정의 한다.

$$C_{management} = \{C_{storage}, C_{operation}\}$$

이때 $C_{storage}$는 멤버쉽 정보에 대한 저장비용이며 $C_{operation}$은 정보의 갱신에 따른 연산 비율, 즉 이벤트를 처리하기 위해 수행하는 연산 수행의 비율을 의미한다. 그러므로 A2A의 경우 각 노드는 멤버쉽에 참여하는 모든 노드에 대한 정보를 유지하고 있어야 하므로 $C_{storage}$는 노드의 개수 n에 대해 $n \times (n-1)$이 되고, $C_{operation}$은 $n-1$이 된다. 또한 Gossip의 경우 각 노드가 가질 수 있는 이웃 노드들의 집합의 크기가 정해져 있고, 또한 각 노드는 멤버쉽 정보에 대해 미리 정의된 복사본의 수가 지정되어 있으므로 이웃 노드 집합의 크기를 v라고 하고, 멤버쉽 정보에 대해 미리 정의된 복사본의 수를 c라고 하면, $C_{storage} = cn+v$가 되고 $C_{operation} = vn$이 된다.

하지만 HRing의 경우 각 노드에서 유지되는 정보는 Gossip과 마찬가지로 자신이 속한 Ring의 정보만 유지하게 된다. 그러나 따로 멤버쉽 정보의 복사본을 유지 및 포워딩 하지 않기 때문에 $C_{storage}$는 상수 값을 갖게 된다. 그리고 $C_{operation}$ 역시 노드 자신이 가지고 있는 $e_i \in E$만을 대상으로 하기 때문에 상수 값을 갖게 된다. 그러므로 시스템에서 소요되는 관리 비용의 요구는 다른 접근법에 비해 HRing 프로토콜에서 더 낮아 짐을 예측할 수 있다.

4.3 안정화 시간(Convergence Time)

멤버의 정보가 변경 되었을 때 해당 사실이 전체 노드에게 알려 지기까지의 시간을 안정화 시간이라고 할 때 A2A와 Gossip의 경우 각각 $O(n)$ 과 $O(\log n)$ 이 된다[13]. 그러나 HRing의 경우 하위 계층에서 멤버 정보의 변경이 발생하면 일단 최상위 계층으로 전송되며 최 상위 계층에서 하위 계층으로 멀티캐스팅 된다. 그러므로 부모 Ring이 갖는 자식 Ring의 수를 s라고 하고 전체 노드의 수를 n, 그리고 HRing이 갖는 계층을 h라고 할 때 HRing에서 변경되는 노드 정보의 전달에 소요되는 시간은 $\log_{m_h} n$ 이 된다. 그러므로 HRing의 안정화 시간은 $O(\log n)$ 이다.

5. 결론 및 향후 연구

본 논문에서는 P2P 형식의 그리드를 구성하는데 필요한 계층형 Ring 기반의 멤버쉽 프로토콜을 제안하였다. 그리고 이 프로토콜의 이름을 "HRing 프로토콜"이라고 명명한다. HRing 프로토콜은 제한된 네트워크 연결만을 가지게 되므로 멤버쉽을 위한 통신 비용을 감소 시킬 수 있게 되며, 각 노드에서 유지해야 하는 멤버쉽 정보들이 지역적인 정보들만을 대상으로 하기 때문에 멤버쉽 정보 유지에 따른 관리 비용 또한 감소하게 되는 장점을 갖게 된다. 이러한 장점으로 인해 각 노드들은 네트워크의 변화에 유연하게 대처할 수 있도록 설계되었다. 그리고 이러한 성능을 통신비용, 관리비용, 그리고 안정화 시간으로 나누어 평가하였을 때 기존의 A2A와 Gossip 기반의 프로토콜에 비해 나은 성능을 가짐을 보이고 있다.

하지만 본 논문에서는 네트워크에서 발생 할 수 있는 여러 변수들에 대한 고려가 없었기 때문에 이를 향후 연구 과제로 남겨 둔다.

참 고 문 헌

[1]Foster I., Kesselman C. Eds. "The Grid: Blueprint for a New Computing Infrastructure", Morgan Kaufmann,

1999.
[2] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications", In Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer commuinications(SIGCOMM), P. 149~160. ACM Press, 2001.
[3] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz, "Tapestry: A Resilient Global-Scale Overlay for Service Deployment", IEEE Journal on Selected Areas in Communications, 22(1):41-53, 2004.
[4] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Contents-Addressable Network", In Proceedings of the 2001 ACM SIGCOMM, P. 161~172, ACM Press
[5] Gnutella. http://gnutella.wego.com/
[6] KaZaA. http://www.kazaa.com
[7] Abhishek Gupta, Divyakant Agrawal, and Amr El Abbadi, "Distributed Resource Discovery in Large Scale Computing Systems", In Proceedings of the 2005 Symposium on Applications and the Internet(SAINT'05), 2005.
[8] The Globus Alliance, "Information Services in the Globus Toolkit 3.0 Release", http://www-unix.globus.org/ toolkit/mds.
[9] Adriana Lamnitchi, Ian Foster, and Daniel C. Nurmi, "A Peer-to-Peer Approach to Resource Location in Grid Environments, Proceedings of the 11 th IEEE International Symposium on High Performance Distributed Computing HPDC-11 20002 (HPDC'02) Page: 419 , 2002.
[10] K. Shen, T. Yang, L. Chu, J. L. Holliday, D. A. Kuschner, and H. Zhu. Neptune: Scalable Replication Managementand Programming Support for Cluster-based Network Services. In Proc. of 3rd USENIX Symposium on Internet Technologies and Systems, San Francisco, CA, Mar. 2001.
[11] A. Gupta, D. Agrawal, and A. El Abbadi, "Distributed Resource Discovery in Large Scale Computing Systems", Proc. Of the 2005 SAINT.
[12] A. J. Ganesh, A. Kermarrec, and L. Massoulie, "Peer-to-Peer Membership Management for Gossip-Based Protocols", IEEE Transactions on Computers, Vol. 52, No. 2, February 2003.
[13] Jingyu Zhou, Lingkun Chu, and Tao Yang, "An Efficient Topology-Adaptive Membership Protocol for Large-Scale Cluster-Based Services", Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium(IPDPS'05), 2005.