한국산학기술학회논문지
Vol. 11, No. 11 pp. 4543-4552, 2010

모바일 ad hoc 네트워크에서 동적인 토폴로지 변화에 따른 리더 결정

김영란[1*], 한현구[1], 박성훈[2]
[1]한국외국어대학교 컴퓨터공학과
[2]충북대학교 전기전자공학부

Leader Decision Protocol for Dynamic Changing Topology in Mobile Ad hoc Networks

Younglan Kim[1*], Hyungoo Han[1] and Sung-Hoon Park[2]
[1]**Dept. of Computer Science & Engineering, Hankuk University of Foreign Studies**
[2]**Dept. of Computer Engineering, Chungbuk National Univiversity**

요 약 리더는 그룹 내의 여러 프로세스 중에서 조정자 역할을 하는 특별한 프로세스이다. 분산 시스템에서 하나의 조정자를 선출하는 과정을 리더선출이라 하며 이는 결함허용 분산시스템을 구축하기 위한 매우 중요한 문제이다. 현재 정상 작동하고 있는 두 개의 모바일 ad hoc 네트워크 그룹들이 병합되면 두 개의 리더가 존재하게 된다. 이것은 안전성 속성에 위배되므로 이를 감지하고 처리할 수 있는 메커니즘이 필요하다. 본 논문에서는 모바일 ad hoc 분산 컴퓨팅 시스템에서 그룹 멤버십 탐지 기반의 리더 경쟁 프로토콜을 제안하고 이를 시제논리로 검증하고자 한다.

Abstract A leader is a special process who roles as the coordinator within multiple processes of a group. In a distributed system, leader election is the procedure of electing a coordinator. This is a very important issue for building fault-tolerant distributed systems. When two normal mobile ad hoc networks are merged, there are two leaders. This violates the safety property, so a mechanism to detect and handle are required. In mobile ad hoc distributed computing system, we propose a leader competition protocol and to prove the temporal logic to it. This solution is based on the group membership detection algorithm.

Key Words : Group Membership Detection, Leader Election, Mobile Ad hoc Network, Network Group Merge

1. 서론

모바일 ad hoc 네트워크는 기지국 또는 액세스 포인트를 가진 기반 망이 없이 모바일 노드들이 자율적으로 연결을 설정하고 임시 또는 즉흥적인 망의 구성이 가능한 네트워크이다. 네트워크를 구성하는 노드들은 대등하고 능동적인 망의 주체가 되며 이동시점과 이동방향 등을 네트워크의 다른 구성요소들에게 사전에 공지하지 않고 이동 할 수 있다[1]. 이러한 특성으로 인해 네트워크의 토폴로지가 동적으로 변경될 수 있으며, 그 결과가 네트워크 그룹의 성능에 심각한 영향을 주어서는 안 된다. 모바일 ad hoc 네트워크는 긴급 구조작전, 긴급한 정보공유가 필요한 경우, 군사 네트워크, 착용할 수 있는 컴퓨터 및 통신장비 등에 적용되고 있다. 이러한 네트워크 그룹은 협동 작업으로 목적을 성취하기 위하여 조정자 역할을 하는 리더를 필요로 하며, 리더를 선출하는 프로토콜은 그룹간 통신[2]과 매우 밀접한 관계를 가지고 있다.

리더는 그룹 내의 모든 구성요소들 중에서 조정자의 역할을 하므로 유일하여야 한다. 리더에게 장애가 발생하면 시스템의 신뢰성을 보장하기 위하여 네트워크 그룹 내의 모든 구성요소들이 자율적으로 새로운 리더를 선출하여야 한다[3]. 또한 현재 원활히 작동하고 있는 두 개의

*교신저자 : 김영란(ylkim@hufs.ac.kr)
접수일 10년 08월 19일 수정일 (1차 10년 09월 24일, 2차 10년 10월 22일, 3차 10년 11월 15일) 게재확정일 10년 11월 19일

그룹이 병합된다면 이를 감지하고 그에 따른 행동을 수행할 수 있는 메커니즘이 필요하다. 즉, 현재 독립적으로 수행하던 두 개의 네트워크 그룹이 병합된다면 두 개의 리더가 존재한다는 것을 의미하며 리더 선출 문제의 속성 중 하나인 안전성 속성에 위배된다. 그러므로 이를 검출하고 유일한 리더를 결정할 수 있어야 한다.

[4]은 스패닝 트리를 구축하여 그룹의 멤버십을 검출하고 이를 리더선출에 이용하는 알고리즘을 제안했다. 두 개의 그룹들이 병합되는 경우에 리더를 선출하기 위하여 제안한 기법은 두 가지가 있다. 첫 번째는 리더가 존재하는 두 개의 그룹들이 병합되면 링크가 새로 연결된 노드들이 서로의 리더에 대한 정보를 교환하여 리더를 결정한다. 그리고 우선순위가 낮은 리더를 가졌던 노드가 새로운 리더를 자신이 속했던 그룹의 모든 노드들에게 역전파한다. 이 기법의 문제점은 서로 다른 그룹에 속한 노드 사이에 링크가 연결되었다는 것을 감지할 수 있는 기법이 필요하다. 왜냐하면 어떤 두 개의 노드 사이에 링크가 연결된다는 것이 항상 다른 그룹의 노드와 연결된다는 것을 의미하지 않기 때문이다. 또한 새로운 리더를 자신이 속하였던 그룹에만 역전파하기 위해서는 병합되기 바로 이전의 구성 노드들에 대한 정보를 유지하여야 하는 기법이 필요하다. 두 번째는 어떤 하나의 그룹은 리더가 존재하고 다른 하나의 그룹은 리더를 선출 중이라면 새로 선출한 리더를 무조건 전파한다. 만약에 병합되는 시점이 이제 막 리더의 부재를 감지하고 선출이 시작되는 초기단계이면 링크가 새로 연결된 노드들은 리더를 역전파하는 메시지인 *Leader* 메시지를 그리고 리더선출 모드에 속한 노드들은 리더를 선출하는 *Election* 메시지를 전송하게 되므로 병합된 하나의 그룹 내에 리더선출을 하기 위한 메시지가 혼재하게 된다.

본 논문에서는 병합되기 바로 이전에 네트워크 그룹 내에서 리더의 역할을 하였던 노드들의 정보만을 수집하여 병합된 네트워크 그룹의 새로운 리더로 결정하는 프로토콜을 제안한다. 이것은 병합되기 바로 이전에 리더의 역할을 하지 않은 노드들은 단지 *Competition* 메시지를 전파하기만 하고 소스노드에게 응답을 보내지 않으므로 메시지 수는 리더를 새로 선출하는 프로토콜보다 현저히 적게 된다.

본 논문에서는 특정한 ad hoc 모바일 컴퓨팅 환경에서 그룹 멤버십 탐지(Group Membership Detection: GMD) 알고리즘을 기반으로 한다. 본 논문의 구성은 다음과 같다. 2장에 본 논문의 기반 환경인 모바일 ad hoc 네트워크 시스템의 모델과 제약조건을 기술한다. 3장에 네트워크 그룹들이 병합되었을 때 리더를 결정하는 프로토콜을 상세히 기술한다. 4장에서 메시지수와 응답속도를 분석하고 5장에는 제안한 프로토콜의 정확성을 시제논리를 사용하여 검증한다. 그리고 6장에 결론 및 향후 연구방향을 제시한다.

2. 시스템 모델

본 논문에서 제안하는 프로토콜을 적용할 시스템 모델을 정의하고 모바일 ad hoc 네트워크의 구성요소들의 제약조건을 반영하기 위한 가정들을 기술한다. 그리고 리더 선출문제를 정의하고 이를 정형화한다.

2.1 시스템 모델

본 논문에서 모바일 ad hoc 네트워크 그룹들의 집합은 $U=\{G_1, G_2, \cdots, G_p\}(p \geq 1)$로 정의한다. 각 네트워크 그룹은 유일한 식별자와 유일한 리더를 가지며 동등 계층그룹을 형성한다. 네트워크 그룹들의 집합 U내의 노드의 개수는 유한 집합 $\Omega=\{1,2,\cdots,n\}$이고 각 노드는 유일한 식별자를 갖는다.

모바일 ad hoc 네트워크 모델은 무향 그래프 $G=(V,E)$로 정의한다. 정점들의 집합 $V=\{v_1, v_2, \cdots, v_n\}(n \geq 1)$는 모바일 노드들의 집합으로 각각의 정점은 모바일 노드를 나타내고 유일한 식별자를 갖는다. V에 속하는 두 개의 노드가 서로 상대방의 전송 반경 내에 있어 서로 직접 통신이 가능한 경우에 에지(edge)가 존재하며 두 개의 노드는 서로 연결되었다라고 한다. 여기서 각각의 통신 링크는 양방향성이라고 가정한다. 노드 j가 노드 i의 변수 n_i의 원소이고 노드 i가 노드 j의 변수 n_j의 원소일 때 통신 링크는 양방향성이다. 즉, $j \in n_i$ iff $i \in n_j$이다. 그러므로 네트워크 $G=(V,E)$에서 에지들의 집합 E는 모든 노드 i, j가 정점집합 V의 원소일 때, 에지 (i, j)가 에지 집합 E의 원소이고 j는 노드 i의 변수 n_i의 원소인 것을 나타낸다. 즉, $\forall i,j \in V$: $(i,j) \in E$ iff $j \in n_i$ 이다.

네트워크 시스템의 구조와 모바일 노드의 제약조건을 다음과 같이 한다.

- 각각의 노드들은 유일한 식별자를 갖는다. 이것은 노드를 식별하는 데 사용되고 동일한 가중치(weighted value)를 가진 노드들이 있다면 식별자를 사용하여 그들의 우선순위를 결정한다. 가장 작은 값의 식별자를 가진 노드가 가장 높은 우선순위를 갖는다라고 가정한다.
- 각각의 노드들은 가중치를 가지며, 노드의 배터리

잔여 수명, 다른 노드들과의 최소 평균 거리, 그리고 계산능력등과 같은 수행능력과 관련된 속성으로 산출한다. 가중치의 우선순위는 네트워크에서 리더가 될 수 있는 지표를 나타낸다. 가장 작은 가중치를 가진 노드가 가장 높은 우선순위를 갖는다라고 가정한다.

- 각각의 노드들은 자신의 전송 반경 내에 위치하고 있어서 직접 통신이 가능한 이웃한 노드들의 목록을 변수 n_i에 갖는다. 이것은 링크의 형성에 따라 변경될 수 있다.
- 노드간의 정보교환은 비동기 메시지 패싱을 사용하며 통신은 신뢰성이 있다. 통신 링크는 양방향성으로 FIFO이다. 즉, 메시지는 이웃한 노드들 간의 링크를 통해 순차적으로 전달된다. 그리고 메시지 전달은 메시지가 전송되는 전체 기간 동안에 송신자와 수신자가 연결 상태에 있어야 보장된다.
- 각 노드들의 수신버퍼 용량은 충분한 크기를 가지며, 노드의 생존 기간 동안에는 수신버퍼의 오버플로우는 발생하지 않는다.
- 네트워크 그룹의 분리(partition)와 네트워크 그룹들의 병합(merge)을 포함하여 노드의 이동성은 네트워크의 토폴로지를 임의로 변경시키게 된다.

2.2 리더선출 문제 정의

리더는 그룹 내의 여러 프로세서 중에서 조정자의 역할을 하는 특별한 프로세서이다. 분산 시스템에서 그룹 내의 프로세서들이 하나의 조정자를 결정하는 과정을 선출이라 하며 이는 결함 허용(fault tolerant) 분산 시스템을 구축하기 위해 매우 중요한 문제이다. 리더선출 알고리즘은 상호배제를 보장하기 위한 공유자원의 할당과 회수, 프로세스 모니터링과 회복 등 조정자가 필요한 어느 곳이든지 유용하게 이용될 수 있다[3]. 리더선출 해법은 안전성(safety) 속성과 필연성(liveness) 속성이 충족되어야 한다. 일반적으로, 안전성 속성은 전형적으로 시스템이 지속적으로 유지하여야 할 요구사항을 표현한다. 반면, 필연성 속성은 지속적으로 유지할 필요가 없는 요구사항을 표현하고 있으나, 언젠가는 실현되는 것이 보장되어야 한다. 그러므로 안전성 속성은 종료를 보장하지는 않으나 컴퓨테이션이 정확한 결과를 제출하고 필연성 속성은 종료를 보장한다. 이 두 가지 속성은 분리관계(disjoint)이다[5,6].

본 논문에서 제안하는 프로토콜은 시스템을 구성하는 각각의 노드는 지역변수 ldr_i에 현재 자신의 리더가 어느 노드인지를 설정하고 있다. 시스템내의 모든 노드들이 물리적으로 동일한 시간에 지역변수 ldr_i의 값을 동시에 변경하는 것은 불가능하다. 그래서 리더를 변경하는 과정 동안, 각각의 노드는 시스템의 상태를 변수 $status_i$ 에서 관리한다. 어떤 노드 i의 $status_i$가 Norm 이면 동작 모드는 정상이고 그 노드의 ldr_i값은 유효하다. 반면 $status_i$ 가 Norm이 아니면 그 노드의 ldr_i 값은 유효하지 않다. 이러한 지역변수를 사용하여 본 논문에서 제안하는 리더 결정 프로토콜의 안전성 속성과 필연성 속성을 다음과 같이 명세한다.

안전성 속성 : 시스템에 연결된 모든 노드들이 정상 동작 상태일 때, 리더는 언제나(henceforth: $\Box$) 유일해야 한다.

$$\Box\big(\forall i,j\in G_k \subseteq \Omega : i \neq j :$$
$$\big(status_i = Norm \wedge status_j = Norm\big) \Rightarrow \big(ldr_i = ldr_j\big)\big)$$

필연성 속성 : 언젠가는 반드시(eventually: $\Diamond$) 시스템에 연결된 모든 노드들이 정상 동작 상태에 도달하고 시스템에는 새로운 리더가 선출되어야 한다.

$$\neg ldr-Elected \Rightarrow \Diamond ldr-Elected$$

정의 1. (술어 $ldr-Elected$)

$$ldr-Elected \equiv$$
$$\big(\forall i, \exists l \in G_k \subseteq \Omega : \big(status_i = Norm\big) \wedge \big(ldr_i = l\big)\big)$$

정의 1은 필연성 속성을 기술하기 위해 사용한 술어로 "어떤 노드 i는 정상동작 상태이고 리더는 l이다" 라는 의미이다.

3. 리더 결정 프로토콜

모바일 ad hoc 네트워크 그룹들의 집합 $U=\{G_1, G_2, \cdots, G_p\}$에서 임의의 네트워크 그룹 두 개가 병합된다면 병합된 하나의 그룹에 두 개의 리더가 존재하게 된다. 병합되기 바로 이전에 어떤 네트워크 그룹에 속하였든 서로 다른 두 개의 *Heartbeat* 메시지를 받는 어떤 노드 i가 존재할 것이다. 이것은 노드 i에게 두 개의 리더가 존재한다는 것을 의미한다. 이는 리더선출 문제의 속성 중 하나인 안전성 속성에 위배된다.

본 논문에서는 임의의 네트워크 그룹들의 병합을 다음과 같이 정의하고 병합된 네트워크 그룹에 연결된 모든 노드들이 오직 하나의 리더만을 가질 수 있는 리더 경쟁(leader competition)프로토콜을 제안한다.

정의 2. (두 개의 네트워크 그룹들의 병합)

$Merge(G_n, G_m) \triangleq G_n \cup G_m$

정의 2는 임의의 두 개의 네트워크 그룹들의 병합은 n개의 노드로 구성된 네트워크 그룹 G_n과 m개의 노드로 구성된 네트워크 그룹 G_m가 한 개의 네트워크 그룹 G_p가 되는 것이다.

$\forall t, t' : \exists i, j : \big((t <_{im} t') \wedge (i \in G_n \wedge j \in G_m) \wedge$
$\big((t.ldr_i \neq t'.ldr_i) \vee (t.ldr_j \neq t'.ldr_j)\big)\big) \Rightarrow G_n \cup G_m$

어떤 노드 i가 어떤 시각 t에 수신한 *Heartbeat* 메시지와 즉시 후속되는 관계에 있는 시각 t'에 수신한 *Heartbeat* 메시지가 서로 다르다면 노드 i의 리더는 두 개이다. 그러므로 네트워크 그룹 G_n 또는 네트워크 그룹 G_m에 속한 어떤 노드 i가 그룹들이 병합되었다는 것을 인지하는 것은 자신의 리더가 두 개라는 것을 감지한 결과이다. 여기서 t와 t'은 각 노드에서 운영하는 논리 시간이고, $<_{im}$은 즉시 후속되는 관계(immediately precedes relation)를 의미한다.

3.1 Outline

리더 경쟁 프로토콜은 병합되기 바로 이전에 어느 네트워크 그룹에 속한 노드이든지 자신의 리더가 두 개 이상이라는 것을 감지한 노드가 *연산 확산(diffusing computation)*을 시작한다. 연산 확산을 시작한 노드를 *소스노드(source node)*라 한다. 여기서 노드의 가중치는 해당 노드의 식별자로 가정하며 이것은 일반성을 훼손하지 않는 범위에서 알고리즘을 가급적 단순하게 표현하기 위한 것이다. 이 프로토콜의 결과는 병합되기 바로 이전에 각 네트워크 그룹에서 리더의 역할을 한 노드들 중에서 우선순위가 가장 높은 노드를 병합된 네트워크 그룹 G_p의 새로운 리더로 결정한다.

```
if (two_Heartbeat)
{
    send Competition Message to neighboring
                                        processes;
    while(# of leader list is not equals to two)
      receive Response Message from two leaders;
    choose Leader;
    send Leader Message to neighboring processes;
}
```

[그림 1] 소스노드의 behavior

그림 1에 소스노드의 프로토콜을 의사코드로 기술하고 그림 2와 그림 3에 임의의 노드 i가 수행하는 프로토콜을 상세히 기술한다. 이 프로토콜은 리더 경쟁을 확산시키는 *Competition* 메시지, 병합되기 바로 이전에 리더의 역할을 하던 노드가 자신의 식별자를 소스노드에게 전송하는 *Response* 메시지, 그리고 결정된 리더를 전달하는 *Leader* 메시지를 사용한다.

```
Pi::

Var
1.1  ldr_i := current leader;
1.2  status_i := one of state set in
                          {Norm, Compete, Wait};
1.3  n_i := {set of all neighboring processes};
1.4  ll_i := {};          // leader list
1.5  cnt := 0;            // # of merged group;
1.6  s_tag_i := null;     // source tag
1.7  t_stamp_i := 0;      // time stamp

On status_i = Norm:

2.1  if (two_Hearbeat) then
2.2    status_i := Compete;
2.3    s_tag_i := i;
2.4    t_stamp_i := 1;
2.5    send competition(s_tag_i, t_stamp_i)
                          to each process of n_i ;
2.6    if (ldr_i = i) then
2.7      ll_i := ll_i ∪ { i };
2.8    end-if
2.9  end-if

3.1  Upon received competition(s_tag, t_stamp)
                                  from process j:
3.2    status_i  := Wait;
3.3    s_tag_i := s_tag;
3.4    t_stamp_i := t_stamp + 1;
3.5    send competition(s_tag_i, time_stamp_i)
                    to each process of n_i except j;
3.6    if (ldr_i = i) then
3.7      send response(ldr_i) to process s_tag;
3.8    end-if
```

[그림 2] 리더 경쟁 프로토콜 (1/2)

```
On status_i = Compete:

4.1  Upon received competition(s_tag, t_stamp)
                                    from process j:
4.2    if ((s_tag_i,t_stamp_i)<(s_tag,t_stamp)) then
4.3      status_i := Wait;
4.4      s_tag_i := s_tag;
4.5      t_stamp_i := t_stamp + 1;
4.6      send competition(s_tag_i, t_stamp_i)
                    to each process of n_i except j;
4.7    end-if

5.1  Upon received response(ldr_i) from process j:
5.2    ll_i := ll_i ∪ { j };
5.3    if (cnt = # of ll_i)  then
5.4      chooseLeader();
5.5    end-if

On status_i = Wait:

6.1  Upon received competition(s_tag_i, t_stamp_i)
                                    from process j:
6.2    if ((s_tag_i,t_stamp_i)<(s_tag,t_stamp)) then
6.3      s_tag_i := s_tag;
6.4      t_stamp_i := t_stamp + 1;
6.5      send competition(s_tag_i, t_stamp_i)
                    to each process of n_i except j;
6.6    end-if

7.1  Upon received  leader(l) from process j:
7.2    status_i := Norm;
7.3    ldr_i := l ;
7.4    send leader(l)
                    to each process of n_i except j;

chooseLeader():
8.1  ldr_i := max{ll_i};
8.2  send leader(ldr_i) to each process of n_i;
8.3  status_i := Norm;
```

[그림 3] 리더 경쟁 프로토콜(2/2)

3.2 Bootstrapping

각 노드의 *LeaderCompetition* 모듈은 그 노드가 생존하는 기간 동안에는 끊임없이 반복된다. 만약에, LeaderCompetition 모듈이 반복되는 각각의 회전(round)마다 알고리즘 명세의 어떤 행동을 동작하는 것이 가능하다면, 반복되는 각 회전에서 최소한 어떤 하나의 실행 가능한 행동이 존재하고 그것이 실행되는 것을 확인할 수 있다. LeaderCompetition 모듈의 bootstrapping은 LeaderCompetition 모듈의 초기화 부분에 명세한 변수들의 값을 할당하는 과정을 포함한다. (lines 1.1-1.7).

3.3 소스노드의 역할

병합되기 바로 이전에 어느 네트워크 그룹에 속하든 노드 i가 서로 다른 *Heartbeat* 메시지를 받는다면 자신과 이웃한 모든 노드들에게 *Competition* 메시지를 전송하는 것으로 연산 확산을 시작하고 자신의 *status*변수를 Compete로 설정한다.(lines 2.1-2.5). 그리고 병합되기 바로 이전에 리더의 역할을 한 노드들로부터 *Response* 메시지가 전송되기를 기다린다.(lines 5.1-5.2) 소스노드는 몇 개의 네트워크 그룹이 병합되었는지 알기 위하여 자신이 수신한 *Heartbeat* 메시지의 개수를 병합된 네트워크 그룹의 개수로 설정한다. 병합된 네트워크 그룹의 개수와 수신한 *Response* 메시지의 개수가 같으면 병합되기 바로 이전에 리더의 역할을 하던 노드들로부터 그들의 식별자를 모두 받게 된 것이므로(lines 5.3-5.4), 수신한 *Response* 메시지들의 목록인 ll_i의 원소들중에서 우선순위가 가장 높은 노드를 병합된 이후의 리더로 결정한다. 그리고 이것을 이웃한 모든 노드들에게 *Leader* 메시지로 전송하고 자신의 *status* 변수를 Norm으로 한다.(lines 8.1-8.3).

3.4 그 외 노드들의 역할

현재 어떤 노드 i가 Norm *status*이고 자신의 리더가 하나일 때 *Competition* 메시지를 수신한 경우, 현재 자신이 리더가 아니라면 자신의 *status* 변수를 Wait로 설정하는 것만 한다. 그러나 만약자신이 리더의 역할을 하고 있는 노드라면, 소스노드에게 자신이 리더이었다는 것을 통지하는 *Response* 메시지를 전송하고 *status* 변수를 Wait로 설정한다. (lines 3.1-3.8).

어떤 노드 i가 Wait *status*일 때, 노드 i가*Leader* 메시지를 수신하면 자신의 리더를 수신한 값으로 설정하고 *status* 변수는 Norm으로 설정한다. 그리고 자신과 이웃한 노드들에게 수신한 메시지를 전송한다. (lines 7.1-7.4).

3.5 동시에 다중으로 수행되는 연산확산

두 개의 네트워크 그룹이 병합되었다는 것을 두 개의 이상의 노드가 감지할 수 있으므로 리더경쟁 프로토콜도 리더 선출 프로토콜[7]처럼 *동시에 다중으로 수행되는 연산확산* (*multiple- concurrent diffusing computation*)이 될

것이다. 이 경우에 다수의 연산확산이 동시에 진행되는 것을 방지하여야 하므로 선출 프로토콜과 같이 $\langle s_tag,\ t_stamp\rangle$튜플로 구성된 *연산색인*(*computation-index*)을 사용하여 이를 조정한다. *s_tag*는 리더 경쟁을 시작한 노드의 식별자이고 *t_stamp*는 그 연산확산에 참여하는 노드가 증가시키는 정수형 값으로 연산확산이 진행된 논리 시간을 의미한다.

정의 3. (연산색인 우선순위)

$\langle s\_tag_1, t\_stamp_1\rangle > \langle s\_tag_2, t\_stamp_2\rangle \Leftrightarrow$
$\left((t\_stamp_1 > t\_stamp_2) \wedge (s\_tag_1 \neq s\_tag_2)\right) \vee$
$\left((t\_stamp_1 = t\_stamp_2) \wedge (s\_tag_1 > s\_tag_2)\right)$

연산색인은 $computation-index_1 > computation-index_2$ 의 경우에 한하여 연산확산-1이 연산확산-2보다 우선순위가 높다.

현재 어떤 노드 i의 *status*가 Compete 또는 Norm일 때 새로운 *Competition* 메시지를 수신하면 그 노드는 현재 참여하고 있는 연산확산과 새로운 연산확산의 우선순위를 확인한다. 새로운 연산확산의 우선순위가 현재 참여하고 있는 연산확산보다 높으면 새로운 연산확산에 참여하고 아니면 새로운 연산확산에 참여하지 않는다. 연산확산의 우선순위 중에서 첫 번째 경우는 현재의 연산확산의 *t_stamp*보다 새로운 연산확산의 *t_stamp*가 크고 *s_tag*가 서로 같지 않을 때이다. *t_stamp*의 값이 크다는 것은 *t_stamp*의 값이 작은 것 보다 연산확산을 시작한 논리시간이 오래 되었다는 것을 의미하기 때문이다. 두 번째의 경우는 현재 참여하고 있는 연산확산의 *t_stamp*와 새로운 연산확산의 *t_stamp*가 같고 현재 참여하고 있는 연산확산의 *s_tag*보다 새로운 연산확산의 *s_tag*의 우선순위가 높을 때이다. 이 경우는 자신보다 우선순위가 높은 노드의 연산확산에 동의하는 의미에서 현재 참여하고 있는 연산확산을 중지하고 새로운 연산확산에 참여한다. (lines 4.1-4.7, 6.1-6.6).

3.6 리더 경쟁 프로토콜 예

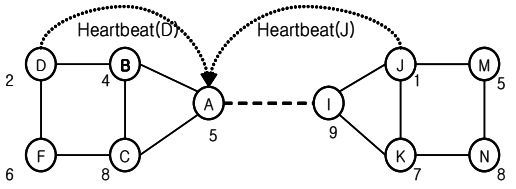
그림 4에 리더 경쟁이 수행되는 과정을 예를 들어 설명한다. 네트워크 그룹 G_n의 구성 요소는 노드 A, B, C, D, F이고 리더는 D이다. 네트워크 그룹 G_m의 구성요소는 노드 I, J, K, M, N이고 리더는 J이다. 노드 식별자 옆에 있는 정수값은 가중치이다.

(a)는 노드 A와 노드 I 사이의 점선은 그룹 G_n과 그룹 G_m이 병합되어 링크가 형성되었음을 나타낸다. 두 개의 *Heartbeat* 메시지를 수신한 노드 A는 리더가 두 개라는 것을 감지한다.

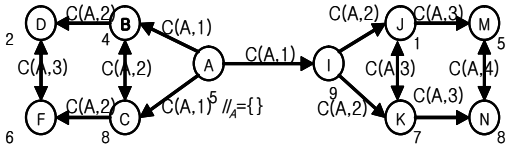
(b)는 소스노드 A가 *LeaderCompetition* 모듈을 시작하여 자신과 이웃한 노드들에게 *Competition* 메시지를 전송한다. 이 메시지를 받은 노드 B, C와 노드 I는 자신에게 *Competition* 메시지를 전송한 노드를 제외하고 *t_stamp*를 증가시켜 *Competition* 메시지를 자신과 이웃한 노드들에게 전송한다. 즉, 노드 B는 D와 C에게, 노드 C는 B와 F에게, 노드 I는 J와 K에게 *Competition* 메시지를 전송한다. 그리고 노드 B, C, I는 리더의 역할을 하던 노드가 아니므로 *status*를 Wait로 설정하고 리더가 결정되기를 기다린다.

(c)에서 노드 D와 J는 병합되기 바로 이전에 리더의 역할을 하던 노드이므로 *Response* 메시지를 소스노드에게 전송한다. 소스노드 A는 수신한 *Response* 메시지를 ll_A 목록에 추가한다.

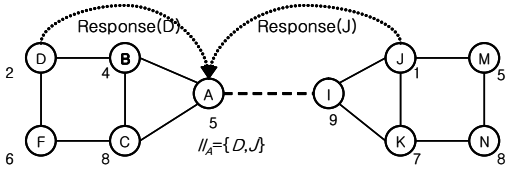
(d)에서 소스노드 A는 병합된 네트워크 그룹의 수와 ll_A 목록의 원소수가 같으므로 ll_A에서 우선순위가 가장 높은 노드를 새로운 리더로 결정한다. 그리고 그것을 *Leader* 메시지로 전송한다. 예제에서는 노드 D와 노드 J가 리더 경쟁을 하여 노드 J가 새로운 리더로 결정된다.



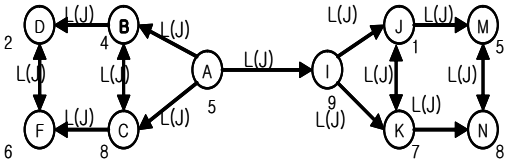
(a) 서로 다른 두 개의 *Heartbeat* 메시지를 수신한 노드 A



(b) 노드 A가 *CompetitionModule*을 시작하여 *Competition* 메시지가 확산되는 과정



(c) 병합되기 바로 이전에 리더역할을 한 노드들로부터 Response 메시지를 수신하여 리더 결정



(d) *Leader* 메시지가 확산되는 과정

[그림 4] 리더 경쟁 프로토콜의 예

3.7 Norm 모드인 네트워크 그룹과 Elect 모드인 네트워크 그룹의 병합

어떤 네트워크 그룹 G_n의 모든 구성 요소들의 *status*는 Norm이고 그 노드들은 모두 리더를 가지고 정상적으로 작동하고 어떤 다른 네트워크 그룹 G_m은 리더 선출 모드이다. 이 두 개의 네트워크 그룹이 병합된다면, 네트워크 그룹 G_m에 리더가 존재하지 않으므로 병합된 네트워크 그룹에서 리더 경쟁이 발생하지 않는다.

이 경우에는 G_m의 리더 선출 연산확산이 G_n까지 확산하게 되므로 병합된 네트워크 그룹의 모든 노드들이 리더선출[7]에 참여하게 된다. 그 결과는 병합된 그룹에서 리더가 새로 선출되므로 오직 하나의 리더만이 존재하게 된다. 이는 안전성 속성을 충족시킨다.

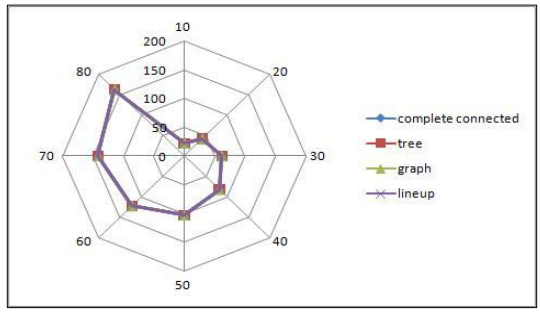
4. 시뮬레이션 결과

병합된 이후에 리더를 결정하는 데 필요한 메시지의 수와 응답시간의 속도의 성능분석을 위하여 [7]에서 사용한 시뮬레이션 모델을 사용한다. 메시지의 수는 *Competition* 메시지, *Response* 메시지, *Leader* 메시지의 수를 산출한다. 병합된 그룹 내에 두 개의 리더가 존재하는 시점부터 리더결정과정이 종료되어 모든 프로세스들이 유일한 리더를 갖게 되기까지의 시간의 지연을 분석한다. 2.1에서 네트워크 그룹들의 집합 U내의 노드의 개수는 유한 집합 $\Omega=\{1,2,\cdots,n\}$이고 각 노드는 유일한 식별자를 갖는다고 가정하였다. 리더결정의 결과는 그룹 내의 모든 노드들 중에서 리더로써 우선순위를 갖는 노드로 결정되기 때문에 시스템 내에서 소스노드가 생성하는 연산색인 우선순위와 상관없으므로 어떤 노드라도 소스노드의 역할이 허용된다. 어떤 노드가 리더경쟁 연산확산을 한 번 수행하면 현재의 리더경쟁 모듈이 종료되기 전에는, 즉 유일한 리더가 결정되기 이전에는, 그 노드는 연산확산을 다시 발생시킬 수가 없으므로 그룹 내의 연산확산의 수도 유한한다.

[4]는 각각 유일한 리더를 갖고 있는 노드가 새로 연결되면 새로 연결된 링크를 통하여 리더의 식별자를 교환한다. 그리고 낮은 순위의 리더를 가진 노드가 나머지 구성원들에게 상대방의 리더를 역전파한다. 이 때 어떤 노드 두 개가 새로 연결될 때마다 서로 다른 그룹인지, 즉 리더가 서로 다른지, 비교하기 위한 메시지가 필요하다. 이는 새로운 노드 연결이 발생할 때마다 서로 전송하게 된다. 반면에 본 논문은 두 개의 그룹이 병합되는 시점을 제시하였고 이를 감지한 노드가 병합된 그 이후에 유일한 리더를 결정하는 프로토콜이다. 따라서 본 논문에서 제안한 프로토콜과 [4]는 성능을 산출하는 시점이 다르므로 이를 단순히 비교하는 것은 타당성이 결여된다.

4.1 메시지의 수

단일 소스노드의 경우에 메시지 수는 다음과 같다. 그룹 내의 노드의 수를 n개라 하였을 때, 소스노드를 포함한 모든 노드들이 *Competition* 메시지를 전송하므로 *Competition* 메시지의 수는 n개이다. 병합되기 바로 이전에 리더의 역할을 하였던 노드만이 소스노드에게 *Response* 메시지를 전송하므로 *Response* 메시지의 수는 2개다. 그리고 *Leader* 메시지의 수는 *Competition* 메시지와 같은 방식으로 전송되므로 *Leader* 메시지의 수는 n개이다. 그러므로 병합된 그룹이 유일한 리더를 갖게 되는데 필요한 총 메시지의 수는 $2n+2$개다.



[그림 5] 메시지의 수

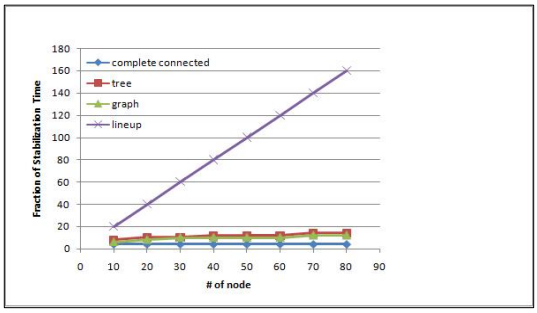
그림 5와 같이 메시지의 수는 토폴로지에 의해 영향을 받지 않고 노드의 수에 비례하여 증가한다. 즉, 노드의 수에 따라 *Competition* 메시지의 수와 *Leader* 메시지의 수가 결정되므로 어떠한 토폴로지이든 노드의 수가 같으면 동일한 메시지의 수가 산출된다.

동시에 수행되는 다중 연산확산일 때 연산색인 우선순위에 의해 연산확산을 중지한 노드의 수가 가장 많은 경우에 메시지의 수가 가장 많다. 이 때 *Competition* 메시지

의 수는 각각의 연산확산에서 *Competition* 메시지가 전파된 타임스탬프의 수가 메시지의 수로 산출된다. 만약에 그룹 내의 모든 노드 n개가 연산확산을 수행하면 종료탐지에 성공한 한 개의 노드를 제외한 나머지 $(n-1)$개의 노드는 종료탐지에 실패한다. 각 노드 i가 시작한 연산확산이 소스노드로부터 h홉 떨어져 위치한 노드들에게까지 *Competition* 메시지가 전송되고 종료탐지에 실패하였을 때 각 노드의 *Competition* 메시지의 수를 h_i라고 하자. 그리고 종료탐지에 성공한 *Competition* 메시지의 수를 n이라고 하면 *Competition* 메시지는 $\sum_{i=1}^{k} h_i + n,\ k \le (n-1)$ 이다. 단 여기서 h_i는 각 노드별로 연산확산이 진행된 단계의 메시지의 수이므로 각 소스노드별로 다른 값을 갖는다. 나머지 메시지의 수는 단일 노드와 동일하게 산출된다. *Response* 메시지의 수는 2개이고 *Leader* 메시지의 수는 n개다. 그러므로 동시에 수행되는 다중 리더경쟁 연산확산에서 병합된 그룹이 유일한 리더를 갖게 되는 데 필요한 총 메시지의 수는 $\sum_{i=1}^{k} h_i + 2n + 2,\ k \le (n-1)$개다.

4.2 처리속도의 성능평가

리더결정 프로토콜의 응답시간은 소스노드의 수와 상관없이 네트워크 위상에서 소스노드의 위치에 의해 영향을 받는다. 즉, 소스노드로부터 단말노드들에 도달하는 홉 h수의 편차가 가장 적은 경우가 응답시간이 가장 좋은 경우이고, 가장 큰 경우가 응답시간이 가장 나쁜 경우이다. 각 노드들 사이의 메시지 전송시간 및 전파지연은 동일하다고 가정한다. 분석을 간단히 하기 위하여 메시지 전송시간(Tm)은 메시지 종류와 관계없이 동일하고 노드가 알고리즘을 실행하는 데 따르는 처리지연(Tp)를 포함한다고 가정한다. 즉, $Tm = Tp + Tt$이고 Tt는 순수한 전송시간이다. 단일 소스노드의 경우에 처리속도는 다음과 같다. 소스노드의 이웃한 노드목록에 있는 노드들에게 *Competition* 메시지가 전송되는 시간은 Tm이다. 그러므로 소스노드로부터 정확히 h홉 떨어져 존재하는 노드들에게 *Competition*메시지가 전송되는 소요시간 $h \times Tm$이다. 이때 h를 네트워크 그룹 내에서 소스노드로부터 가장 멀리 떨어져 있는 노드들에 도달하는 홉의 수로 하면 가장 멀리 떨어져 있는 노드들에게 *Competition* 메시지가 전송되는 시간이 소스노드로부터 $h-1$홉 떨어져 위치한 노드들에게 *Competition* 메시지가 전송되는 시간을 포함한다. *Response* 메시지의 전송시간은 리더의 역할을 하던 노드가 $h-1$홉 내에 위치한다면 *Competition* 메시지가 전송되는 동안에 발생한다. 그렇지 않고 어느 하나의 노드라도 소스노드로부터 가장 멀리 떨어진 h홉에 위치한 단말 노드이면 Tm이 소요된다. *Leader* 메시지는 *Competition* 메시지의 경우와 동일하게 산출된다. 그러므로 리더가 결정되는 데 필요한 소요시간은 $To + 2(h \times Tm) + Tm$이다. 타임아웃 To는 리더가 두 개라는 것을 감지하는 데 필요한 소요시간이다.



[그림 6] 처리속도

그림 6과 같이 네트워크내의 모든 노드들이 리더를 갖게 되는 데 소요되는 시간은 토폴로지에 의해 크게 영향을 받는다. 완전연결 그래프는 홉의 수가 항상 1이므로 노드의 수와 상관없이 처리속도가 언제나 동일하다. 그래프는 소스노드는 정중앙에 위치하고 노드의 분포가 균등하다. 트리는 이진트리로 한다. 처리속도는 그래프와 이진트리의 경우는 노드의 수가 증가하여도 매우 안정적이며 이진트리일 경우에 그래프보다 처리속도가 조금 낮은 것을 확인할 수 있다. 그러나 노드들이 일렬로 된 토폴로지에서 순차적으로 메시지를 전달받는 경우는 노드의 수와 처리속도가 비례하여 증가한다.

소스노드가 다중일 경우, 종료탐지에 성공한 연산확산의 전송시간이 종료탐지에 실패한 연산확산들의 전송시간을 포함하게 되므로 단일 소스노드와 리더를 결정하는 데 소요되는 시간이 동일하다.

5. 정확성 검증

본 논문에서 제안하는 프로토콜의 정확성을 검증하기 위하여 2.2절에서 명세한 안전성 속성을 시제 논리를 사용하여 검증한다.

안전성 속성은 시스템에 연결된 모든 노드들이 정상동

작 상태에 있을 때 리더는 언제나 유일하여야 한다는 것을 의미한다. 모순에 의한 증명을 하기 위하여 다음과 같은 경우를 가정한다.

(모순에 의한 증명)정상적으로 작동하고 있는 두 개의 그룹 G_n과 G_m이 병합되어 어떤 노드 i가 자신의 리더가 두 개라는 것을 감지하였다. 그리고 이는 참이다. 이를 다음과 같이 기술한다.

$(\exists i, l_1, l_2 \in G_k \subseteq \Omega : i \neq l_1 \neq l_2 :$
$status_i = Norm \wedge ldr_i = l_1 \wedge ldr_i = l_2)$

병합된 네트워크 그룹에서 자신의 리더가 두 개라는 것을 감지한 노드가 존재하는 것은 명백한 사실이다. 이 시점에서 각각의 노드들이 리더 경쟁 연산확산을 시작하는 것은 그림 2 lines 2.1-2.5에 의해 참이다. 2.1절에서 정의한 바와 같이 그룹 G_k 내에서 노드의 개수는 유한하고 모든 노드의 식별자는 유일하기 때문에 우선순위가 가장 높은 노드는 하나만이 존재한다. 그러므로 어떤 하나의 리더 경쟁 과정에서 우선순위가 높은 오직 하나의 노드만이 리더로 결정되므로(그림 3 lines 5.1-5.5) 병합된 네트워크 그룹 내에는 유일한 리더가 전파되게 된다.(그림 3 lines 7.1-7.4). 따라서 이는 모순이다. □

필연성속성은 시스템에 연결된 모든 노드들이 반드시 정상 동작 상태에 도달하고 시스템에는 새로운 리더가 선출되어야 한다는 것을 의미한다. 이는 리더경쟁 프로토콜에서는 근본적으로 충족된다. 왜냐하면 그룹 G_n과 G_m은 병합되기 바로 이전에 각각 정상 작동 모드이고 리더를 가지고 있다. 병합이 되어 현재 리더가 두 개라는 것을 감지하였음에도 불구하고 리더경쟁 프로토콜이 수행되지 않는다는 것은 결국은 모든 노드들에게 두개의 리더가 존재한다는 것을 의미한다. 이는 단지 안전성 속성만을 위배할 뿐이기 때문이다.

6. 결론 및 향후 연구방향

본 논문에서는 모바일 ad hoc 네트워크 환경에서 수행되는 분산 리더 경쟁 프로토콜을 제안하였다. 이는 현재 정상 동작 상태인 두 개의 그룹들이 병합되면 안정된 상태에 진입할 수 있는 프로토콜이다. 또한 정상 동작 상태인 그룹과 선출모드[8]에 있는 그룹이 병합되면 선출결과인 유일한 노드가 병합된 그룹의 모든 노드들에게 전파되므로 제안한 프로토콜은 안전성속성과 필연성 속성을 보장한다. 제안한 리더 경쟁 프로토콜의 속성들을 시제논리를 사용하여 형식언어로 명세하고 프로토콜의 정확성을 검증하였다. 향후 본 논문에서 고려하지 않은 리더경쟁 과정 중에서 소스노드의 고장 또는 이탈과 같은 결함 허용에 대한 연구와 분석이 필요하다.

참고문헌

[1] Zygmunt J. Haas, et. al., Wireless Ad Hoc Networks, Perkins, Charlies Ed., AD HOC NETWORKING, pp. 221-225, Addison Wesley, 2001.

[2] David Powell, guest editor, “Special section on group communication,” Communications of the ACM, Vol.39, No. 4, pp.50-97, April 1996.

[3] Vijay K. Garg, Elements of Distributed Computing, Wiley Interscience, 2002.

[4] S. Vasudevan, et..al., “Design and Analysis of a Leader Election Algorithm for Mobile Ad hoc Networks,” Proceeding of the IEEE 12th International Conference on Network Procotols, pp.350-360, 2004.

[5] Zohar Manna and Amir Puneli, Temporal Verification of Reactive Systems: Safety, Springer-Verlag, 1995.

[6] Edward Chang, Zohar Manna, Amir Puneli, "The Safety-Progress Classification," In Logic, Algebra and Computation, NATO ASI Series, Subseries F: Computer and System Sciences, Springer-Verlag, 1992.

[7] Orhan Dagdeviren, Kayhan Erciyes, "A Hierarchical Leader Election Protocol for Mobile Ad hoc Networks," Proceeding of the 8th International Conference on Computational Science, 2008.

[8] 김영란, et al., "모바일 ad hoc 네트워크 시스템하에서 선출 알고리즘의 명세 및 증명,” 멀티미디어학회 논문지 제13권 제7호, pp.950-959, 2010.

김 영 란(Younglan Kim) **[정회원]**

- 1986년 2월 : 서울과학기술대학교 전자계산학(공학사)
- 1989년 2월 : 한양대학교 산업대학원 전자계산학(공학석사)
- 2010년 8월 : 한국외국어대학교 대학원 컴퓨터공학(공학박사)

<관심분야>
Planning, Distributed System

한 현 구(Hyungoo Han) **[정회원]**

- 1980년 2월 : 서강대학교 수학과 (이학사)
- 1986년 5월 : Univ. of South Florida 전산학과 (공학석사)
- 1990년 12월 : Auburn University 전산학과(공학박사)
- 1991년 3월 ~ 현재 : 한국외국어대학교 컴퓨터공학과 교수

<관심분야>
Robot Planning, Wireless Networks

박 성 훈(Sung-Hoon Park) **[정회원]**

- 1982년 2월 : 고려대학교 통계학과(경제학사)
- 1993년 2월 : 인디애나대학교 대학원 컴퓨터학과(공학석사)
- 1997년 2월 : 고려대학교 대학원 컴퓨터공학과(공학박사)
- 2004년 9월 ~ 현재 : 충북대학교 전기전자공학부 교수

<관심분야>
분산, 모바일, 유비쿼터스 시스템, 알고리즘