Contributory 멀티캐스트에서 그룹키 재분배를 위한 효율적인 일괄처리 알고리즘

서혜영*, 김상진**, 오희국*

*한양대학교 컴퓨터공학과, **한국기술교육대학교 인터넷미디어공학부

Efficient Batch Rekeying Algorithm for Contributory Multicasting Environment

Hyeyoung Seo*, Sangjin Kim**, Heekuck Oh*

*Department of Computer Science and Engineering Hanyang Univ.
**School of Internet-Media Engineering, Korea Univ. of Technology and Education

요 약

안전한 멀티캐스트란 동적으로 그룹 멤버가 변하는 환경에서는 현재의 그룹 멤버만 데이터를 얻을 수 있도록 멀티캐스트하는 방법을 말한다. 이를 위해 그룹 멤버간에 그룹키를 공유하며, 이 키로 암호화하여 데이터를 멀티캐스트한다. 전방향 안전성(forward secrecy)과 후방향 안전성(backward secrecy)을 제공하기 위해 멤버가 가입하고 탈퇴할 때마다 공유키를 변경해야 한다. 이 때 확장성을 위해 그룹키의 변경이 그룹 전체에 미치는 영향은 최소화되어야 한다. 지금까지의 연구는 확장성 문제를 해결하기 위해 플랫한 그룹키 공유 구조에서 계층 구조로 변화해 왔으며, 그룹의 파티션을 용이하게 하고 중앙집중 방식의 문제를 극복하기 위해 중앙 키 서버를 사용하지 않고 그룹 멤버가 생성한 값을 계산을 통하여 그룹키를 생성하는 프로토콜로 변화해 오고 있다. 하지만 지금까지 제안된 안전한 멀티캐스트 방식은 멤버의 가입은 확장성을 갖추고 있지만 멤버의 탈퇴는 그렇지 못하며, 성능 측면에서 많은 개선이 있었지만 실제 응용에 사용되기에는 아직도 연산 측면에서 적절하지 못하다. 이 때문에 이 논문에서는 실제 응용에서 안전한 멀티캐스트를 효율적으로 사용할 수 있도록 그룹키 분배를 위한 중앙 서버를 사용하지 않는 환경에서 가입과 탈퇴가 일어날 때마다 개별적으로 처리하지 않고 일괄처리하는 여러 알고리즘을 제안하고 그 성능을 분석한다.

I. 서론

현재 많은 인터넷 서비스가 전송자의 전송 부하를 줄이고 통신 대역폭을 줄이기 위해 멀티캐스트 기반의 그룹 통신을 사용하고 있다. 그러나 이런 서비스가 유료화되기 위해서는 안전한 멀티캐스트가 제공되어야 한다. 안전한 멀티캐스트란 그룹 빈번하게 동적으로 변하는 환경에서 현재의 그룹 멤버만 볼 수 있도록 멀티캐스트하는 방법을 말한다. 이를 위해 그룹 멤버간에 대칭키를 공유하여 이 키로 데이터를 암호화하여 멀티캐스트한다.

안전한 멀티캐스트 기법이 갖추어야 하는 조건은 크게 안전성, 확장성, 신뢰성이다. 안전성이 충족되기 위해서는 우선 기본적으로 그룹 멤버가 아닌 사람은 그룹키를 알 수 없어야 한다. 뿐만 아니라 전방향 안전성과 후방향 안전성까지 만족해야 한다. 전방향 안전성이란 그룹의 과거 멤버가 현재의 그룹키를 알 수 없어야 한다는 것을 말하며, 후방향 안전성이란 새로 가입한 멤버가 과거의 그룹키를 알 수 없어야 한다는 것을 말한다. 이런 안전성을 충족시키기 위해서는 멤버가 가입하고 탈퇴할 때마다 그룹키를 변경해야 한다.

멀티캐스트 기법이 확장성을 충족하기 위해서는 기본적으로 한 멤버의 행동이 그룹 전체에 영향을 주지 않아야 한다. 특히 멤버의 가입과 탈퇴에 따른 그룹키가 변경될 때 그것의 영향이 최소화되거나 항상 일정해야 확정성이 충족된다고 한다. 이런 확장성을 높이기 위해 그룹키 공유 구조는 플랫한 구조에서 계층구조로의 변화하였으며, 이러한 변화로 가입의 복잡성은 O(1)까지 줄였고,

탈퇴의 복잡성은 O(logN)까지 줄였다[1]. 그러나 실제 응용에 적용하기에는 아직도 계산량이 많다. 특히 그룹의 파티션을 용이하게 해주고, 중앙집중 방식의 문제를 극복한 contributory 방식[2]에서는 가입의 복잡성이 O(logN)이다.

이 논문에서는 현실적인 환경에서 안전한 멀티캐스트 서비스를 제공하기 위해 가입과 탈퇴가 일어날 때마다 그룹키를 변경하지 않고, 주기적으로 일괄처리하는 방법을 제안한다. 이 논문에서는 중앙 키서버를 사용하지 않고, 그룹의 멤버가 기여한 값으로 그룹키를 생성하는 contributory 방식의 TGDH 프로토콜[2]을 기반으로 일괄처리 알고리즘을 제안한다.

이 논문의 구성은 다음과 같다. 2장에서는 이 논문에서 기반으로하는 TGDH 프로토콜을 설명한다. 3장에서는 제안하는 세 가지 일괄처리 알고리즘을 자세히 설명하고 4장에서는 이 알고리즘들의 성능을 비교 분석한다. 끝으로 5장에서는 결론과 향후 연구 방향에 대해 서술한다.

II. 관련연구

1. 멀티캐스트 기법의 분류

멀티캐스트는 그룹 멤버에 대한 키를 생성하고 관리하는 방식에 따라 다음과 같이 나눌 수 있다.

- **Centralized key server**[1,3,4]: 하나의 중앙 키 서버를 사용하는 방식으로서, 이 키서버는 그룹키를 생성하여 멤버들에게 분배시켜주고 키를 관리하는 역할을 한다. 서버만이 접근통제, 키 분산을 수행하기 때문에, 신뢰를 가지고 동기적으로 키 분배를 수행할 수 있다. 서버의 병목현상이 생길 수 있으며, 서버가 공격당할 경우 전체 그룹에 영향을 미치므로 공격에 대한 취약점이 될 수 있다.
- **Distributed subgroup**[5, 6]: 큰 그룹을 여러 개의 작은 서브그룹으로 나눈다. 각 서브그룹에는 관리자가 존재하며 서브그룹내의 키 생성과 키 분배, 메시지 전송에 대해서 관리한다. 서버가 여러 개 존재하기 때문에 작업이 분산되고, 서버의 오류는 전체 그룹이 아닌 서브그룹에만 영향을 미치게 된다.
- **Decentralized key server**[7]: 고정된 서버가 아닌 그룹의 멤버 중에서 키 관리와 분배를 해줄 수 있는 서버를 임의로 선택한다. 서버가 동적으로 선택되기 때문에 모든 멤버가 서버가 될 수 있는 능력을 갖추어야 한다.
- **Contributory**[2, 8, 9]: 그룹의 모든 멤버가 참여해서 그룹키를 생성한다. 동적인 서버가 존재할 수 있으며, 자신의 값 이외에 다른 멤버들의 기여값에 대한 정보를 알아야 하므로 다른 프로토콜보다 통신량이나 계산량은 많아진다.

2. TGDH 프로토콜

TGDH 프로토콜[2]은 이진 트리를 사용하는 contributory 방식의 그룹키 동의 프로토콜이다. 이 프로토콜에서 그룹키는 각 그룹 멤버의 기여값(share)을 이용하여 Diffie-Hellman 키 동의 방식으로 그룹키를 생성한다. 따라서 TGDH 프로토콜의 안전성은 Diffie-Hellman 문제에 기반한다. 이 프로토콜에서 이진 트리의 각 노드에는 키 값과 은닉키(blind key) 값이 할당된다. 이 때 키 값이 a이면 은닉키 값은 g^a가 된다. 두 자식 노드의 키 값이 a 와 b이면 이 노드의 키

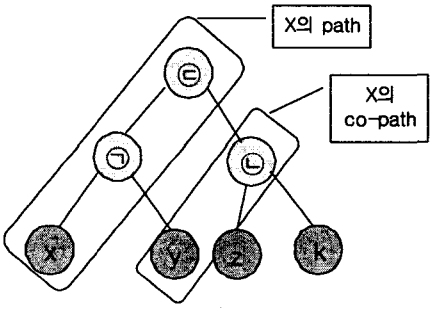

그림 1: TGDH 기본 트리

표 1: 노드값

노드	키	은닉키
x	x	g^x
㉠	g^{xy}	$g^{g^{xy}}$
㉡	g^{zk}	$g^{g^{zk}}$
㉢(그룹키)	$g^{g^{zk}\,g^{xy}}$	

값은 g^{ab}가 되며, 은닉키 값은 $g^{g^{ab}}$가 된다. 이런 방식으로 계산하여 이진 트리의 루트 노드의 키 값이 그룹키가 된다. 따라서 중간(intermediate) 노드들은 그룹키를 계산해 내기 위한 경로가 된다. 그림 1에서 기술된 것처럼 자신의 노드부터 루트까지의 각 노드들의 집합을 경로(path)라고 하고, 경로에서의 각 노드들의 형제 노드(sibling)의 집합을 보조경로(co-path)라 한다. 이진 트리의 leaf 노드에는 각 멤버의 키 값과 은닉키 값이 할당되며, 각 멤버는 자신의 키 값은 비밀로 유지하고, 다른

멤버들에게 은닉키 값은 공개한다. 그림 1에서 각 노드의 키 값과 은닉키 값은 표 1과 같다.

모든 멤버는 다른 모든 멤버의 은닉키 값은 알고 있으며 트리의 현재 모습도 알고 있다고 가정한다. 루트 노드의 키 값이 그룹키가 되므로 각 멤버는 경로와 보조경로에 있는 은닉키 값을 알면 그룹키를 계산할 수 있다. TGDH 프로토콜에서 멤버 중에 오른쪽 leaf 노드에 할당되는 멤버는 모두 sponsor가 되며, 전체 트리에서 가장 오른쪽 노드에 할당된 멤버가 최종 sponsor가 된다. 그림 1에서 y는 x의 은닉키 값을 이용하여 (ㄱ)에 해당하는 은닉키 값을 계산하여 유니캐스트로 k에게 전달한다. k는 z의 은닉키 값을 이용하여 (ㄴ)을 계산하고 y로부터 받은 (ㄱ)의 은닉키 값을 이용하여 그룹키를 계산한다. 그 다음에 모든 중간 노드의 은닉키 값과 트리 구조를 모든 멤버에게 브로드캐스팅한다. 이것을 수신한 다른 모든 노드들은 각자 그룹키를 계산하게 된다.

그림 1에 기술된 트리에서 새 노드가 가입하면 그림 2와 같이 트리가 변경된다. 즉, 레벨이 가장 높은 leaf 노드들 중에서 가장 오른쪽에 있는 leaf 노드와 형제 노드가 되어 트리에 추가된다. 새 멤버가 가입하면 그 멤버의 새 기여값이 트리에 추가되므로 자동적으로 그룹키가 갱신되며 전방향 안전성이 보장된다.

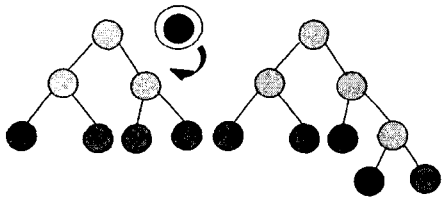

그림 2: join 트리

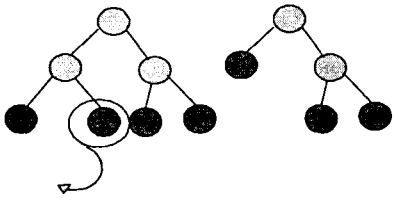

그림 3: leave 트리

그림 1에서 y 노드가 탈퇴하면 그림 3과 같이 트리가 변경된다. 즉, 탈퇴한 노드의 형제 노드가 부모 노드로 옮긴다. 형제 노드가 없는 경우에는 트리 레벨이 하나 감소한다. 멤버가 탈퇴할 경우에는 최종 sponsor가 자신의 기여값을 변경하여 그룹키를 변경한다. 따라서 후방향 안전성이 보장되며, 가입이나 탈퇴는 같은 수준의 복잡성이 요구된다.

III. 제안한 일괄처리 알고리즘

제안한 세 개의 알고리즘에서 각 멤버는 가입하는 순서대로 번호표를 받는다고 가정하며, 일괄처리하여 트리를 재조정하게 되면 왼쪽에 있는 노드부터 다시 번호표가 할당된다고 가정한다.

1. 알고리즘1

표 3 알고리즘 1의 절차

알고리즘1
1. 기존의 멤버로 트리를 구성한다. ① 쌍이 있는 멤버부터 배치+남은 멤버 배치 ② 쌍이 있는 멤버부터 배치+new 멤버를 배치 2. 새로운 멤버로 완전 이진 트리를 구성한다. 3. 기존과 새로운 멤버로 이루어진 완전 이진 트리를 루트로 결합한다.

알고리즘 1은 기존의 멤버와 새로 들어오게 되는 멤버의 트리를 각각 구성한 후에 하나의 트리로 만든다. 기존의 멤버들로 구성이 된 트리는 중간 노드의 키값들을 다시 계산하지 않아도 되므로 지수 계산량이 줄어든다. 기존의 멤버로 트리를 만드는 방법은 크게 두 가지가 있다. 첫 번째는 기존의 멤버들 중에서 형제가 함께 남는 멤버들은 형제를 유지한 채로 먼저 트리에 할당한 후에 혼자 남은 노드들은 그 노드들끼리 연결시켜 트리를 만든다. 두 번째는 연결된 노드끼리 트리에 할당하고 나머지 노드들에게는 새로운 멤버를 할당하는 방법이다. 단점은 두개의 level이 다른 트리를 결합하는 것이기 때문에 비균형 트리가 될 것이고, 가입자가 많은 경우 비례적으로 계산량은 증가한다.

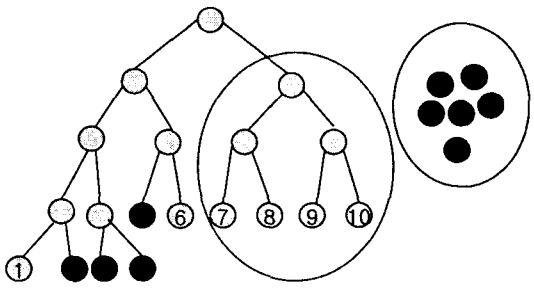

그림 4: 배치 전 프로토콜

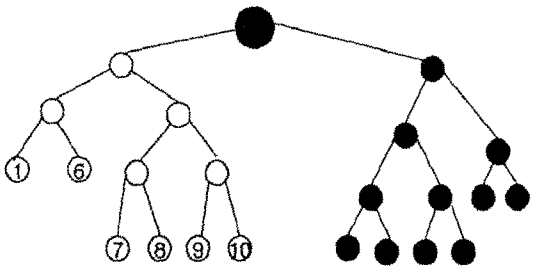

그림 5: 알고리즘1 트리구조

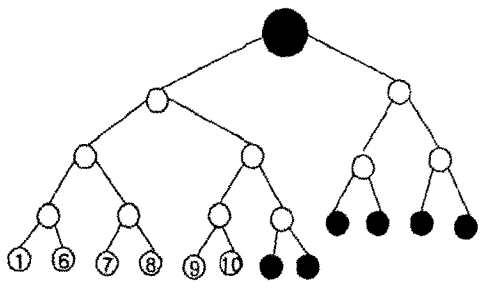

그림 6 알고리즘2 트리구조

2. 알고리즘2

표 4 알고리즘 2의 절차

알고리즘2
1. 진행하게 될 멤버수에 맞는 완전이진 트리 구성을 찾는다. 2. 기존의 멤버부터 왼쪽으로 순서표대로 할당을 한다. 3. 새로운 멤버를 할당한다.

알고리즘 2는 남는 멤버의 수와 새로 가입하는 멤버의 수를 합한 수에 가장 알맞은 최적의 트리 모양을 만든 다음에 남은 멤버와 가입한 멤버를 번호표 순으로 할당한다.

3. 알고리즘3

알고리즘 3은 기존의 트리에서 탈퇴한 멤버들을 제거하고, 새로 가입한 멤버들을 탈퇴한 노드에 할당하는 방식으로 가입한 수가 탈퇴한 수보다 많을 경우에는 가입한 노드들을 탈퇴한 수만큼의 서브트리를 만든 후에 할당한다.

표 5 알고리즘 3의 절차

알고리즘3
1. 기존 트리에서 탈퇴하는 노드들을 멤버만 제거하고 트리 모습은 그대로 유지한다. 2. 새로 가입한 멤버를 할당한다. 이 때 가입한 멤버를 탈퇴한 수만큼의 서브트리를 만들어 탈퇴한 자리에 할당한다. 3. 서브트리를 만드는 방법: 가입한 멤버수는 n, 탈퇴한 멤버수는 d라고 하자 ① join>leave인 경우: n/d, $n-d/d-1$의 나머지가 0이 될 때까지 반복을 한 후, 부분 트리를 만들어 탈퇴한 노드에 할당하여 붙인다. 이때에 두가지 방식으로 가장 낮은 높이에 먼저 삽입을 하는 경우와 순서표대로 삽입을 하는 경우가 있다. ② join=leave인 경우: 그대로 할당하면 된다. ③ join<leave인 경우: 가장 낮은 level의 노드에 할당 한 후에 다시 트리를 조정한다.

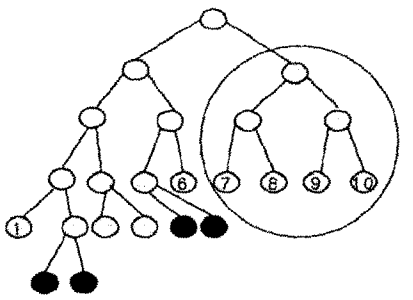

그림 7: 알고리즘3 트리구조

IV. 분석

이 장에서는 3장에서 제안한 세 가지 일괄처리 알고리즘을 비교 분석한다. 이 때 표기는 표6과 같다. 제안한 알고리즘을 분석하기 전에 기존 프로토콜의 복잡성은 표 7과 같다.

알고리즘1.

최악의 경우: 기존의 모든 멤버가 새로이 모여서 기존 멤버로 구성을 할 경우. 한 사람도 같은 쌍을 공유하는 멤버가 없는 경우가 된다. 이 알고리즘의 복잡성은 표 7과 같다.

표 6: 표기법

m	기존 멤버수
j	가입자 수
l	탈퇴자 수
$n(=m-l+j)$	현재 멤버수
$s(i)$	i 쌍이 나오는 경우의 수

표 7: 기존 프로토콜 분석

진행 횟수	$l+j$
멤버수	$l+j$
지수계산량	$\log_2(m+j)\times j+\log_2(m+j-l)\times l$
Unicast수	$\log_2(m+j)\times j+\log_2(m+j-l)\times l$
Broadcast수	$l+j$

표 8: 트리 계산량

멤버수	m
level(l)	$\lceil \log_2 m \rceil$
l에서의 멤버수	$2m-2^l$
$l-1$에서의 멤버수	2^l-m
지수 계산량	$m-1$
unicast	$m-1$
broadcast	1번
sponsor수	$\frac{m}{2}$

새로운 멤버 m명이 완전 이진 트리로 알고리즘을 구성하였을 경우 표7과 같은 계산량을 가진다. 알고리즘1은 두 개의 완전이진 트리를 루트로 결합하는 방식이다. 그러므로 복잡도를 계산할 경우 두 개의 완전 이진트리의 계산량을 한 번의 지수계산량과 한 번의 유니캐스트를 첨가시켜주면 된다. j명의 가입자와 $\frac{m}{2}$ 명의 탈퇴자가 있을 때, 탈퇴자에는 형제 노드가 없다고 한다면, 두 개의 트리에서의 지수계산량은 새롭게 구성하는 완전이진 트리의 계산량과 같고 유니캐스트는 기존 완전 이진 트리에서 절반으로 줄어든다.

지수계산량

$$m-l-s(i)+j-1+1 \quad (1)$$

유니캐스트 수

$$\frac{(m-1)}{2}-s(i)+j-1+1 \quad (2)$$

알고리즘2

멤버수에 맞는 완전 이진트리의 모양이 있다고 가정한다. 그러므로 균형 트리로 프로토콜을 진행할 수 있다. 하지만 최악의 경우로 멤버가 빠지는 경우, 예를들어 순서표대로 멤버를 넣을 경우 짝이 하나도 맞지 않은 구성이 될 수 있다. 표 8의 제시된 복잡성을 이용하여 새로운 트리에 대한 복잡성을 계산할 수 있다. 평균적으로는 기존 멤버와 새로운 멤버로 이루어진 최적화된 트리의 복잡성을 구한 후에, 기존의 멤버에 의해서 구하지 않아도 되는 복잡성을 빼주면 된다.

지수계산량

$$m-l+j-s(i)-1 \quad (1)$$

유니캐스트 수

$$\frac{(m-l)}{2}+j-s(i)-1 \quad (1)$$

알고리즘 3

기존 완전 트리에서 탈퇴한 노드를 제거한 후 그 자리에 가입한 멤버를 할당함으로써 트리를 구성하게 된다. 일반적으로 변화된 공유 노드를 모두 계산을 하게 되므로 기존 멤버의 위치에 따라 복잡성이 달라진다. 변화된 노드 중에서 가장 높은 level을 가지는 노드를 선택한다. 나머지 노드들과의 관계를 살핀 후, 지수 계산량을 계산해 본다. 다른 노드들과의 겹치는 지수계산을 하지 않아도 되므로 만나는 지점을 체크해 준다. 그 체크된 노드와 가입하게 되어서 새롭게 구성이 되는 노드를 모두 합하면 그 노드가 지수 계산을 다시 해야 하는 개수가 된다.

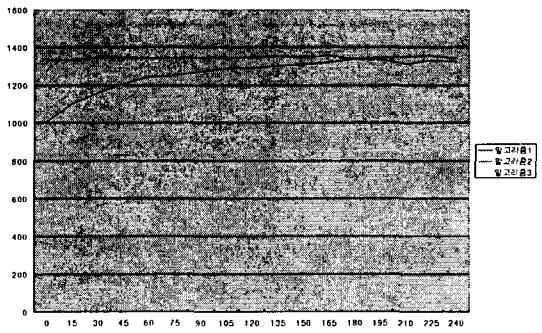

그림 8: leave>join

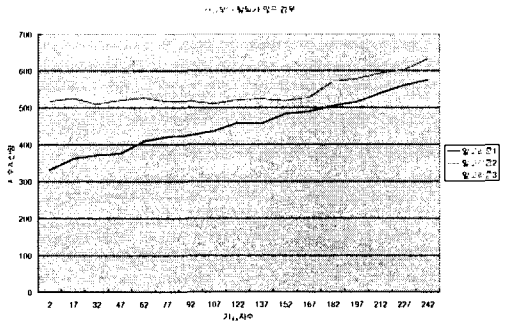

그림 9: join > leave

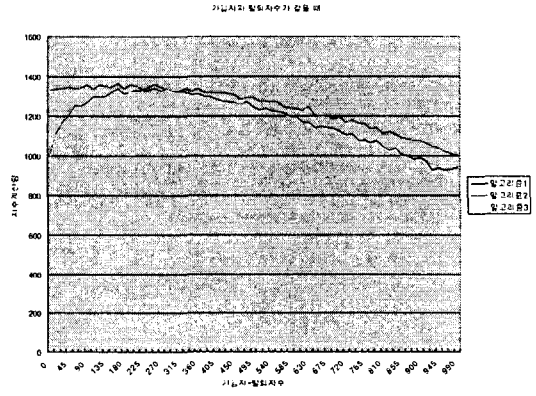

그림 10: 가입자와 탈퇴자수가 같을 때

V. 결론

이 논문에서는 기존 TGDH 프로토콜과 달리 가입과 탈퇴가 발생할 때마다 그룹키를 변경하지 않고 주기적으로 일괄처리 하는 방식을 제안하고 있다. TGDH 프로토콜은 구조상 그룹 전체 멤버 중의 한 사람만 바뀌어도 전방향성과 후방향성 안전성이 만족된다는 장점이 있다. 그러나 이 프로토콜은 키 분배 서버를 사용하지 않는 방식이므로 한 멤버가 자신의 기여값을 바꾸면 모든 멤버들이 공유키를 재계산해야 하는 문제가 있어 기존 방식에 비해 효율이 떨어진다. 이러한 문제를 이 논문에서는 일괄처리하는 방식으로 극복하고자 한다. 일괄처리를 하면 가입과 탈퇴자들이 대기해야 하는 문제점이 있지만 실제 응용에 사용할 수 있는 정도의 효율성을 제공할 수 있다.

이 논문에서는 세 가지 일괄처리 알고리즘을 제안하고 있으며, 각 알고리즘의 성능을 $\frac{m}{2}$ 명의 탈퇴자에서의 최악의 경우에 대해 수학적으로 분석하였다. TGDH 프로토콜에서는 sponsor라는 개념을 두어, 멤버 중 일부가 스스로 키 분배 서버와 같은 역할을 수행한다. 따라서 일괄처리를 할 경우에도 어떤 멤버가 sponsor 역할을 할지 알 수 있도록 일괄처리 후에 트리 모습이 가입하는 멤버의 수와 탈퇴하는 멤버의 수와 상관없이 누구나 자신의 위치를 알 수 있어야 한다. 이 논문에서 제안하고 있는 첫 번째 알고리즘은 기존의 멤버들의 위치의 변화가 적을 수록 성능이 좋지만 기존 트리에서 레벨이 낮은 멤버수보다 가입한 수가 더 적다면 다른 알고리즘보다 트리의 전체 레벨이 하나 증가하는 단점이 있다. 또한 그림 8과 그림 9에서 보여주는 시뮬레이션 결과를 보면 탈퇴자가 많을 경우 두 알고리즘에 비해 일정하게 유지되고 있으므로 큰 그룹에서 탈퇴자가 많은 경우에 유리하다. 두 번째 알고리즘은 번호표대로 트리를 새롭게 구성하므로 남아있는 멤버들간에 관계 변화가 적을수록 좋으며, 모든 다른 알고리즘보다 균형 트리에 가까운 트리를 만들 수 있기 때문에 좋다. 하지만 멤버가 중간에 나가는 경우에는 기존 트리를 구성할 경우 새롭게 지수계산을 해야 하기 때문에 나쁘고 그림 8과 그림9에서 보듯이 알고리즘 3과 비슷한 결과를 나타내주나 가입수가 탈퇴수보다 많은 경우 더 낮은 계산량으로 일정함을 알 수 있다. join = leave인 경우 트리를 재구성하지 않아서 좋지만 join>leave인 경우 (join-leave)수를 leave수만큼으로 서브 트리를 만들어 할당하도록 구성이 되어 있지만 다른 노드들에 영향을 미치지 않고 원래 계산을 해야 하는 노드에 연결되도록 구성이 되었기 때문에 계산량은 다른 알고리즘에 비해서 일정함을 알 수 있다. 세 가지 알고리즘을 분석해보면 기존의 개인멤버 연산 프로토콜과는 달리 연산을 수행하고자 하는 멤버들의 수보다는 기존의 멤버의 위치관계와 기존 멤버 수에 의존한다는 것을 알 수 있다.

본 논문에서는 수학적 분석을 통하여 성능을 분석하고 있다. 분석결과를 보다 정확하게 하기 위해서는 시뮬레이션을 했다. 동적인 서버를 사용하여 멤버들의 기여값으로 그룹키를 생성하는 contributory방식과는 다른 키분배 서버를 사용하는 OFT구조에 대해서도 일괄처리 알고리즘에 대한 연구가 필요하다.

참고문헌

[1] A. Perrig, D. Song, J.D. Tygar, "ELK, A New Protocol for Efficient Large-Group Key Distribution," Proc. of the IEEE Symp. on Security and Privacy, pp. 247-262, May 2001.

[2] Y.Kim, A. Perrig, and G. Tsudik, " Tree-based Group Diffie-Hellman Protocol,

" In Submission.

[3] R. Molva, A. Pannetrat, "Scalable Multicast Security with Dynamic Recipient Groups," ACM Trans. on Information and System Security, Vol. 3, No. 3. pp. 136-160, Aug. 2000.

[4] S. Rafaeli, L. Mathy and D. Hutchison. " EHBT: An Efficient Protocol for Group Key Management," Proc. of 3rd Int. Workshop on Networked Group Communication, LNCS 2233, pp. 159-171, Springer, 2001.

[5] S. Mittra, "Iolus: A Framework for Scalable Secure Multicasting," Proc. of the ACM SIGCOMM '97, pp. 277-288, Sept. 1997.

[6] L.R. Dondeti, S. Mukherijee, and A. Samal. "A Dual Encryption Protocol for Scalable Secure Multicasting," Proc. of the 4th Symp. on Computers and Communications, pp. 2-8, Jul. 1999.

[7] S.Rafaeli. "A Dencentralised Architecture for Group Key Management"

[8] M. Steiner, G. Tsudik, and M. Waidner, " Diffie-Hellman Key Distribution Extended to Group Communication," Proc. of the 3rd ACM Conf. on CCS, pp. 31-37, Mar. 1996.

[9] Y.Kim, A. Perrig, and G. Tsudik, " Communication-Efficient Group Key agreement," Proc. of the IFIP 16th Annual Working Conf. on Information Security, pp. 229-244, June 2001.