Ad-hoc 환경에서 신뢰적인 그룹 키 재전송 기법

홍석형° 김경민 이광겸 신용태
숭실대학교 일반대학원 컴퓨터학과
{stonehead°, kmkim, goodwin77, shin}@cherry.ssu.ac.kr

A Reliable Group Key Re-transmission Mechanism in Ad-hoc Environment

Sukhyung Hong° Kyungmin Kim Kwangkyum Lee Youngtae Sin
Dept. of Computing, Graduate School, Soongsil University

요 약

Ad-hoc 환경의 응용은 재난구조나 회의실 또는 강의실에서의 정보 교환과 같은 그룹 통에서 이용된다. Ad-hoc 환경은 무선 채널을 이용하므로 상대적인 낮은 대역폭과 높은 오류 발생률을 가지게 된다. 따라서 Ad-hoc 네트워크에서는 신뢰적인 전송이 요구된다. 이동 노드는 상대적으로 낮은 성능과 에너지의 제한으로 인해 유선 환경과 같은 신뢰적인 전송 기법을 Ad-hoc 환경에 적용하기에는 문제가 발생한다. Ad-hoc 환경의 무선 채널이 가지는 보안적인 취약성과 높은 에러율을 극복하는 신뢰적인 그룹 키 전송을 위한 재전송 기법을 제안한다. 신뢰적인 트리 형성하기 위해 n차 트리 구조를 이용한다. 손실 감지를 위한 ACK 메시지를 이용하고 손실 복구를 위한 재전송 기법에 대해 연구를 한다. 제안한 신뢰적인 그룹 키 전송을 위한 재전송 기법은 트리의 깊이의 차수가 루트 관리 노드, 서브 관리 노드와 로컬 멤버 노드로 구성되기 때문에 손실 감지와 손실 복구에 대한 연산의 오버헤드가 적다. 루트 관리 노드는 멤버 노드로부터 받은 개인키 정보를 이용하여 그룹 키를 생성하고 그룹 키 부분 정보를 서브 관리 노드에게 전송하고 서브 관리 노드에 대한 신뢰성을 책임진다. 서브 관리 노드는 루트 관리 노드로부터 받은 그룹 키 부분 정보를 로컬 멤버 노드에게 전송하고 로컬 멤버 노드에 대한 신뢰성을 책임진다. 루트 관리 노드와 서브 관리 노드를 관리 노드라 한다. 관리 노드가 신뢰적인 전송을 위해 관리하는 멤버 노드는 전체 그룹에 독립적으로 유지 가능하므로 확장성 및 효율성이 좋다. 관리 노드는 동적인 그룹에 따른 타이머를 설정함으로써 손실 감지에 대한 시간을 줄임으로써 효율적인 손실 감지 및 손실 복구를 한다. 임계값 설정으로 인한 중복 수신에 대한 오버헤드를 줄일 수 있다.

키워드 : Ad-hoc 네트워크, 신뢰성(Reliable), 손실 감지(loss detection), ACK, 손실 복구(loss recovery), 재전송

1. 서 론

Ad-hoc 네트워크는 임의의 노드들이 기반 통신시설(infrastructure)을 이용할 수 없는 경우에 임의의 통신을 수행하기 위해 다른 노드와 함께 통신을 수행하는 네트워크이므로 노드의 전송 범위 내의 노드들은 직접 통신이 가능하며, 전송 범위 밖의 노드들은 다른 중간 노드들의 도움을 받아 통신이 가능하다.

Ad-hoc 환경의 응용은 재난 구조나 회의실 또는 강의실에서의 정보 교환과 같은 그룹 통신에서 이용된다. 하지만 Ad-hoc 네트워크는 무선 채널을 이용하므로 상대적인 낮은 대역폭과 높은 오류 발생률과 보안상의 취약성을 가진다. 따라서 Ad-hoc 네트워크에서는 신뢰적인 전송과 안전한 통신이 요구된다.

신뢰적인 전송은 인터넷과 같은 비신뢰적인 네트워크 상에서 손실 감지(loss detection)와 손실 복구(loss recovery) 두 단계로 이루어진다.

유선환경과 같은 일반적인 패킷 전송의 경우에 손실 감지는 송신측 또는 수신측에서 이루어진다. 송신측에서는 수신자의 ACK를 받지 못한 채 타이머가 종료된 후에 손실 감지를 한다. 반대로 수신측에서의 손실 감지는 순서번호의 공백을 감지하고 NACK를 전송해 송신자가 패킷 손실을 감지하는 순간 재전송을 한다[1].

Ad-hoc 네트워크는 동적으로 그룹을 형성하고 이동 노드는 상대적으로 낮은 성능과 에너지의 제한으로 인해 유선환경과 같은 신뢰적인 전송 기법을 적용하기에는 문제가 발생한다.

따라서, Ad-hoc 네트워크에서 보안적인 측면과 손실 감지와 손실 복구를 고려한 그룹 키 전송을 위한 신뢰적인 전송 기법이 요구된다. 본 논문에서는 Ad-hoc 환경에서 무선 채널을 이용한 상대적인 낮은 대역폭과 오류 발생률을 극복하고 보안적인 측면을 고려한 신뢰적인 그룹 키 전송을 제안한다.

2장에서는 Ad-hoc 환경에서 그룹 키 전송을 위해 제안된 키 트리 구조인 OFT(One-way Function tree)[2]와 n차 키 트리[3] 형성에 대해 설명을 하고 n차 키 트리의 키 전송 방식에 대해 설명한다. 유선환경에서 제안된 트리 기반의 신뢰적인 전송 기법인 RMTP(Reliable Multicast Transport Protocol)와 TMTP(Tree-based Multicast Transport Protocol)에 대해 설명을 한 후에 3장에서 신뢰적인 그룹 키 전송을 위한 재전송 기법을 제안한다. 4장에서는 결론을 도출하고 향후연구를 제시한다.

2. 관련 연구

2.1 트리 기반의 그룹 키 전송

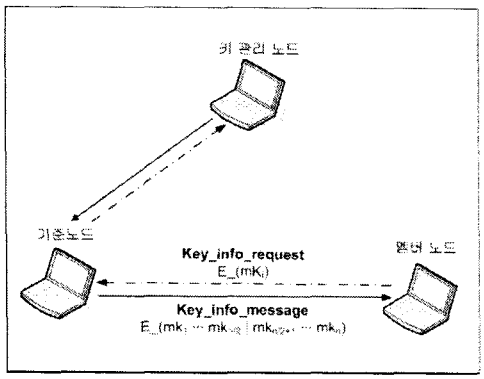
일반적인 이진 키 트리를 사용한 기법인 OFT[2]는 리프 노드들이 그룹 멤버로 대응된다. 이때의 루트 노드의 개인키는 그룹 통신을 위한 그룹 키로 사용된다. 그룹 키는 수집된 키 트리 정보를 사용하여 리프 노드로부터 상위 노드로 단방향 함수가 적용된 정보들을 통합하여 최종적으로 그룹 키를 생성한다. 이러한 경우 멤버 노드에서 그룹 키를 생성하려면 트리 깊이의 차수만큼의 계산이 필요하다.

OFT 키 트리는 경로상의 정보와 단방향 함수를 적용한 자식 노드들의 숨겨진 비밀 정보를 모두 사용하기 때문에 보안적인 측면에서는 견고하지만, 트리 깊이의 차수만큼 교환해야 하는 정보의 횟수가 늘어날 수 있으며 손실 감지와 손실 복구를 위한 재전송에 따른 오버헤드가 발생한다. 또한 그룹을 형성하는데 까지 소요되는 시간이 증가하는 단점을 가지고 있다.

OFT 키 트리 구조를 사용해서 그룹 키를 획득하는 기법은 그룹의 구성원이 늘어날수록 트리 깊이의 차수가 증가하지만, n차 키 트리 구조[3]는 구성원이 늘어나더라도 최단 경로를 유지할 수 있다.

n차 키 트리 구조는 그룹 키 전송에 대한 연산의 횟수를 줄이기 위해 최단 경로 키 트리를 구성하고, 안전한 키 정보의 분배를 위해 키 관리 노드와 기준 노드 구간, 기준 노드와 멤버 노드 구간으로 분산된 개인키를 사용하여 그룹 키를 생성하기 위한 부분 키 정보를 교환한다.

n차 키 트리 구조에서의 그룹 키 전송 기법은 [그림 1]에서 보여주는 것처럼 그룹에 가입하려는 노드가 그룹 키를 생성하기 위한 부분 키 정보를 키 관리 노드에서 기준 노드를 통해 받아온다.



[그림 1] 노드의 그룹 키 부분정보 획득 과정

그룹 키는 키 관리 노드가 멤버 노드로부터 받은 개인키 정보를 사용하여 키 트리를 구성하고, 그룹 키를 생성하기 위한 부분 키 정보를 전달한다.

n차 키 트리 구조는 그룹 키를 직접적으로 전달하지 않고 키 관리 노드의 그룹 키 부분 정보를 사용하여 멤버 노드에서 그룹 키를 생성하는 기법이기 때문에 키의 보안적인 측면에서 안전하다고 할 수 있다.

하지만, n차 키 트리 구조에서는 그룹 키 전송에 대한 손실 감지 및 손실 복구에 대해서는 고려하지 않았다. 따라서 n차 키 트리 구조에서는 그룹 키 부분 정보를 전송하는데 발생되는 손실에 대한 재전송이 요구된다.

2.2 트리 기반의 신뢰적인 멀티캐스트 프로토콜

유선 환경에서 제안된 트리 기반의 신뢰적인 전송 기법은 전체 노드의 그룹을 논리적인 트리 형태로 구성하여, 각 서브 트리에 해당하는 로컬 그룹 별로 손실 감지 및 손실 복구를 한다. 트리 기반의 신뢰적인 전송은 ACK에 대한 오버헤드와 재전송에 따른 중복 수신에 대한 문제를 줄일 수 있다[1].

한 노드가 맡아야 할 자식 노드의 수는 전체 수신자 수에 독립적인 상수로 유지 가능하므로, 확장성 측면에서 큰 장점을 가진다.

이러한 트리 구조의 장점으로 인해 Ad-hoc 환경의 그룹 기반 통신에서 동적으로 변하는 그룹 구성원에 대해 확장성 측면에서 유리하다.

트리 기반의 대표적은 프로토콜은 RMTP(Reliable Multicast Transport Protocol)과 TMTP(Tree-based Multicast Transport Protocol)이 있다.

RMTP[4]는 정적인 ACK 트리 구조를 이용하여 신뢰적인 멀티캐스트 서비스한다. 수신자들은 각자의 로컬 지역에 속하게 하고 지역마다 지정된 DR(Designated Receiver)로 하여금 지역 복구를 담당하도록 하여 확장성이 좋다. 수신자는 송신자에 ACK를 보내는 것이 아니라 주기적으로 ACK를 지역의 DR로 유니캐스트함으로써 ACK에 대한 오버헤드를 줄일 수 있으며, 주기적인 ACK는 손실 감지 즉시 재전송 요청하지 못하기 때문에 지연 시간이 증가한다.

재전송은 손실이 발생한 지역 내 수신자 수에 따라 유니캐스트 또는 멀티캐스트 되는데 멀티캐스트 시 대역폭을 효율적으로 사용하기 위해 손실을 겪은 수신자 수가 많은지 확인을 위한 대기가 필요하므로 지연 문제가 있다. 지연을 없애기 위해 즉시 유니캐스트할 수 있으나 이 경우 대역폭 증가라는 문제가 발생된다. 또한 Ad-hoc 환경은 동적으로 그룹이 변하므로 정적인 DR 선정에 대한 문제가 발생한다.

TMTP[5]는 ERS(Expended Ring Search) 기법을 사용하는 물리적 트리에 가까운 논리적 트리를 구성한다. 논리적 제어 트리상의 노드는 k개의 자식 노드의 신뢰성만을 책임을 지며 자식 노드는 부모 노드에게만 주기적으로 ACK를 유니캐스트한다. ACK를 보냄으로써 송신자 또는 부모 노드의 버퍼를 해제할 수 있게 하되 주기적

전송의 이유는 오버헤드 현상을 줄이기 위함이다. 송신자 또는 부모 노드는 그 자식으로부터 ACK를 받지 못한 채 타임아웃이 되면 제한된 범위의 멀티캐스트를 통해 재전송한다.

TMTP는 전체 수신자 그룹을 논리적인 트리 형태로 구성하여, 각 서브 트리에 해당하는 로컬 그룹별로 손실 감지 및 손실 복구를 담당하게 한다. 이로써 송신자의 처리 부담을 골고루 분산시킬 수 있음은 물론, 많은 수신자에 대해서도 높은 확장성을 갖는다.

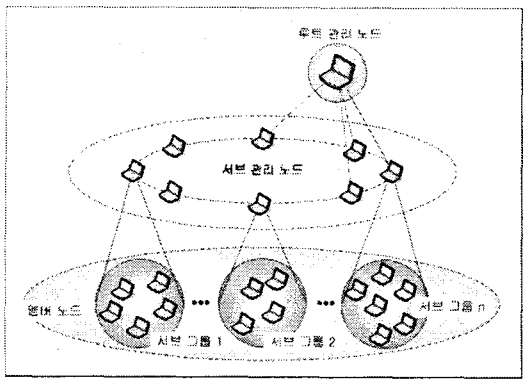
TMTP는 빠른 복구를 위해 NACK에 기반한 손실 복구를 병행한다. 자식 노드가 손실을 감지하면 NACK를 억제 기법을 쓰는 제한된 범위의 멀티캐스트로 보내고 부모는 TTL을 설정하여 제한된 범위의 멀티캐스트를 이용하여 지역 복구를 수행한다. 그러나 NACK 중복을 막기 위한 지연 시간과 TTL 메커니즘에 기반한 지역 복구는 오버헤드에 대한 문제가 발생한다. 또한 TTL의 무방향성과 멤버의 가입 순서에 따라 종속성으로 인해 ACK 트리를 하부 멀티캐스트 라우팅 트리와 유사하게 유지하는 것이 어렵다.

3. 신뢰적인 그룹 키 전송을 위한 재전송

제안하는 신뢰적인 전송을 위한 트리 구조는 기존의 n차 키 트리[3]에서 트리를 형성하기 위해 사용한 이웃 발견 메시지를 이용한다.

원 홉의 이웃 노드 개수를 비교하여 서브 관리 노드를 선출한다. 원 홉 단위의 서브 그룹들로 네트워크 전체 구성이 된다. 서브 그룹이 구성 된 다음에 서브 관리 노드들 비교하여 루트 관리 노드를 선출한다.

[그림 2]는 기존의 n차 트리 형성하는 메커니즘을 이용하여 신뢰적인 트리를 구성한 네트워크 구조를 보여준다.



[그림 2] 신뢰적 전송을 위한 네트워크 구조

트리 구조는 그룹 구성원이 늘어나더라도 트리 깊이의 차수는 3단계로 구성된다. 최단 경로 트리를 구성함으로써 손실 감지와 손실 복구에 대한 연산의 횟수가 줄어 재전송을 통한 그룹 형성하는데 소요되는 시간도 적다.

루트 관리 노드는 멤버 노드로부터 받은 개인키 정보를 이용하여 그룹 키를 생성하고 그룹 키 부분 정보를 멤버 노드에게 전송한다.

루트 관리 노드는 그룹 키 부분 정보를 서브 관리 노드에게 전송하고 서브 관리 노드는 이에 대한 응답으로 ACK를 전송한다. 루트 관리 노드는 타이머가 종료된 후에 서브 관리 노드로부터 받은 ACK로 손실 감지한다. ACK를 전송하지 않은 서브 관리노드에게 그룹 키 부분 정보를 재전송함으로써 신뢰적인 그룹을 형성한다.

서브 관리 노드는 루트 관리 노드로부터 받은 그룹 키 부분 정보를 서브 그룹의 멤버 노드에게 전송한다. 서브 그룹 멤버 노드는 이에 대한 응답으로 ACK를 서브 관리 노드에게 전송한다. 신뢰적인 트리의 깊이의 차수가 적기 때문에 손실 감지와 손실 복구에 대한 연산의 오버헤드가 적다.

3.2 ACK에 의한 손실 감지 및 손실 복구

루트 관리 노드와 서브 관리 노드를 관리 노드라 한다. 관리 노드는 멤버 노드에게 그룹 키 부분 정보를 전송하는 과정에서 타이머가 종료되면 ACK를 전송하지 않은 멤버 노드를 감지한다. 관리 노드는 손실 복구를 위해 그룹 키 부분 정보를 손실한 멤버 노드에게 재전송한다.

관리 노드는 그룹 키 부분 정보를 전송한 후에 타이머를 시작한다. 타이머가 동작하는 동안에 멤버 노드로부터 ACK를 수신 받는다.

관리 노드는 멤버 노드로부터 ACK를 수신하면 노드 리스트에 해당 멤버 노드의 ACK 플래그를 설정하고 수신한 ACK 개수를 증가시킨다.

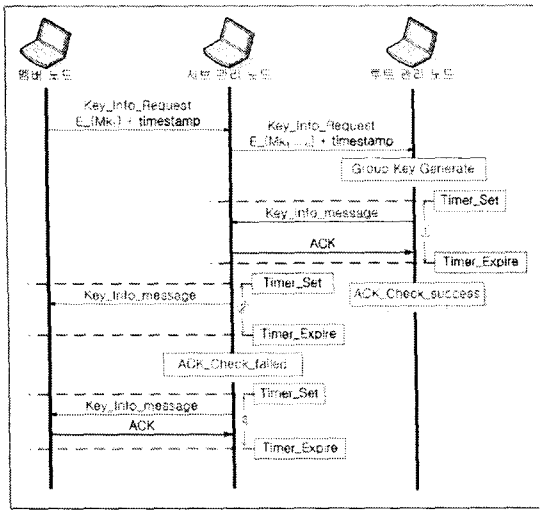
타이머가 종료되면 이웃 노드 개수와 ACK 개수를 비교하여 ACK 개수가 적다면 관리 노드는 손실을 감지한다. 관리 노드는 노드 리스트의 ACK 플래그를 검사하여 ACK 플래그가 설정되지 않은 멤버 노드에게 그룹 키 부분 정보를 재전송한다.

[그림 3]은 제안하는 신뢰적인 그룹 키 전송 과정의 순서를 설명한다.

트리가 형성된 후에 멤버 노드는 개인키 정보를 서브 관리 노드에게 전송한다. 서브 관리 노드는 멤버 노드로부터 받은 개인키 정보를 조합하여 루트 관리 노드에게 전송한다.

루트 관리 노드는 서브 관리 노드로부터 받은 개인키 정보를 이용하여 그룹 키를 생성할 수 있는 그룹 키 부분 정보를 생성한다.

루트 관리 노드는 그림 3의 ①에서 타이머를 설정하고 그룹 키 부분 정보를 서브 관리 노드에게 전송한다. 서브 관리 노드는 그룹 키 부분 정보를 수신하자마자 ACK로 응답을 한다. ①의 과정은 손실 없이 전송된 과정이다.



[그림 3] 신뢰적인 그룹 키 전송

서브 관리 노드는 루트 관리 노드로부터 그룹 키 부분 정보를 수신하고 ACK로 응답한 후에 이 정보를 멤버 노드에게 전송을 한다. ②는 서브 관리 노드가 그룹 키 부분 정보를 멤버 노드에게 전송을 하고 타이머가 종료될 때까지 ACK를 수신 받지 못한 상황이다. 타이머가 종료되면 서브 관리 노드는 노드 리스트의 ACK 플래그를 검사한다. 손실을 감지한 서브 관리 노드는 ③에서처럼 손실 복구를 위해 타이머를 재설정하고 그룹 키 부분 정보를 재전송한다.

그룹 키 부분 정보를 수신한 노드는 ACK를 상위 노드에게 전송 후 그룹 키 부분 정보를 이용하여 그룹 키를 생성한다.

신뢰적인 트리에서 로컬 그룹 별로 손실 감지를 위한 ACK는 관리 노드에게만 전송함으로써 손실 감지에 대한 오버헤드가 줄어든다. 관리 노드가 신뢰적인 전송을 위해 관리하는 멤버 노드는 전체 그룹에 독립적으로 유지가 가능하므로 동적으로 그룹을 형성하는 Ad-hoc 환경에서 확장성 및 효율성을 좋게 할 수 있다.

3.3 손실 감지를 위한 타이머 설정

Ad-hoc 네트워크는 무선 채널을 이용함으로써 상대적으로 낮은 대역폭을 가진다. 또한 멤버 노드들은 이동하기 때문에 거리 차에 따른 RTT가 틀리다.

관리 노드에서 손실 감지를 위한 타이머 설정을 위해 이러한 특성을 고려하지 않는다면 손실 감지 및 손실 복구를 위한 재전송에 대해서 상당히 오랜 시간이 소요될 수 있다.

따라서, Ad-hoc 환경의 특성을 고려한 적절한 타이머를 설정해야 한다. 관리 노드에서 적절한 타이머를 설정하기 위해서는 멤버 노드와의 RTT를 알아야 한다. 멤버 노드가 관리 노드로 개인키 정보를 전송할 때 타임스탬프를 포함하여 전송함으로써 관리 노드와 멤버 노드 사이의 RTT를 측정한다.

$$T_{avr} = \left(\sum_{i=1}^{n} RTT_i\right)/n \qquad \text{(수식 1)}$$

$$Timer = T_{avr} + (1-\alpha)T_{max} \qquad \text{(수식 2)}$$

[표 1] 변수의 정의

변수	정의
n	서브 로컬 멤버 노드의 개수
RTT_i	각 노드의 왕복 전송 시간
$Timer$	설정된 타이머
α, $(1-\alpha)$	관리자에 의한 가중치 확률
T_{avr}	로컬 멤버 노드의 RTT 평균
T_{max}	로컬 멤버 노드의 최대 RTT

관리 노드는 수집된 멤버 노드의 RTT와 이웃 노드의 개수 정보 n을 사용하여 (수식 1)과 같이 RTT 평균을 계산한다. 관리 노드는 타이머를 설정하기 위해 RTT 평균에 수집된 멤버 노드의 최대 RTT를 관리자에 의한 가중치 확률을 부여함으로써 (수식 2)와 같이 계산하여 손실 감지를 위한 타이머를 설정한다.

Ad-hoc 환경은 동적으로 변하기 때문에 관리 노드가 ACK를 수신하기 위한 타이머를 최소 시간인 T_{avr}만을 사용하면 ACK가 늦게 도착하는 노드에 대해 그룹 키 부분 정보를 재전송하는 확률이 증가한다. 따라서 동적인 환경을 반영하기 위해 타이머를 설정은 로컬 멤버 노드의 최대 RTT인 T_{max}에 관리자에 의한 가중치 확률을 부여함으로써 늦게 도착하는 ACK에 대한 재전송에 대한 확률을 줄일 수 있다.

3.4 손실 복구를 위한 임계값 설정에 따른 재전송

관리 노드는 타이머가 종료되면 ACK에 의해 손실 감지를 한다. 손실이 감지되면 손실 복구를 위해 재전송을 한다. 손실 복구를 위한 재전송에 따른 중복 수신에 대한 문제가 발생한다.

이러한 중복 수신에 대한 오버헤드를 줄이기 위해서는 관리 노드는 사전에 정의한 λ와 이웃 노드의 개수인 n에 의해 임계값을 결정한다.

임계값에 따른 손실이 발생한 노드의 수를 비교하여 유니캐스트로 전송할 것인지 멀티캐스트로 전송할 것인지 결정한다.

임계값보다 ACK를 전송한 노드가 많다면 손실이 발생한 노드가 전체 로컬 그룹 멤버보다 적기 때문에 멀티캐스트로 재전송한다면 중복 수신에 따른 오버헤드의 문제가 발생한다. 따라서 그룹 키 부분 정보를 수신하지 못한 노드에게 유니캐스트로 재전송한다.

유니캐스트 재전송에 따른 관리 노드는 재전송한 만큼의 에너지 소모가 발생하지만 멤버 노드에서의 중복 수신에 따른 오버헤드는 발생하지 않는다.

임계값보다 ACK를 전송한 노드가 적다면 손실이 발생한 노드가 전체 로컬 그룹 멤버보다 많기 때문에 유니캐스트로 전송하는 것은 관리 노드의 재전송에 따른 오버헤드가 심각하다. 따라서 이때에는 멤버 노드에서 중복 수신에 대한 오버헤드가 발생하지만 유니캐스트로 전송하는 관리 노드의 오버헤드보다 작다.

4. 결론 및 향후 연구

논문에서 제안한 신뢰적인 그룹 키 전송을 위한 재전송 기법은 트리의 깊이의 차수가 적기 때문에 손실 감지와 손실 복구에 대한 연산의 오버헤드가 적다.

관리 노드가 신뢰적인 전송을 위해 관리하는 멤버 노드는 전체 그룹에 독립적으로 유지 가능하므로 확장성 및 효율성이 좋다. 관리 노드는 동적인 그룹에 따른 타이머를 설정함으로써 손실 감지에 대한 시간을 줄임으로써 효율적인 손실 감지 및 손실 복구를 한다.

임계값 설정으로 인한 중복 수신에 대한 오버헤드를 줄일 수 있다.

논문에서 제안한 신뢰적인 그룹 키 재전송 기법은 손실을 최소화하기 위해 에너지 효율에 대해 고려를 하지 않았다. 따라서 에너지 효율을 고려한 신뢰적인 전송 기법에 대한 연구가 필요하다.

참고문헌

[1] 이동만, 윤원용, "Survey on Reliable Multicast over the Internet," 정보처리학회지, Vol.8.No.5, 2001

[2] McGrew, D. A. and Sherman, A. T., "Key establishment in large dynamic groups using one-way function trees," submitted to IEEE Transactions on Software Engineering 2003, pp. 444-458

[3] 이광겸, "Ad-hoc 환경에서 실시간 멀티캐스팅 서비스를 위한 키 관리 기법," 숭실대학교, 2006.6

[4] C.-G Lin and S. Paul, "RMTP:A Reliable Multicast Transport Protocol," IEEE INFOCOM 96, 1996.3

[5] R.Yavatkar, J.Criffioen, and M.Sudan, "A Reliable Dissemination Protocol for Interactive Collaborative Applications," ACM Multimedia '95, p. 333-344, 1995.11