동적 멀티미디어 그룹 통신을 위한 신뢰성 있는 다자간 전송 프로토콜(ReMP)의 성능 분석

손 지 연[†] · 원 유 재[†] · 오 수 형[††] · 임 경 식[†] · 황 승 구[†]

요 약

ReMP(Reliable Multipeer Protocol)는 멀티미디어 그룹 통신을 지원하기 위해 설계된 신뢰성 있는 다자간 전송 프로토콜이다. 본 논문에서는 ReMP가 제한된 버퍼 환경에서도 올바르게 동작함을 증명하였으며, 호스트에서의 처리량 관점에서 데이터 전송에 대한 성능 분석을 시도하였다. 또한 이를 토대로 ReMP를 다른 신뢰성 있는 멀티캐스트 프로토콜들과 성능 비교를 하였다. 분석 결과에 따르면, ReMP는 성능이나 메모리 요구량 관점에 비추어 볼 때, 다른 비교 대상 프로토콜들보다 광범위한 용도의 멀티미디어 그룹 통신에 적용되어질 수 있음을 알 수 있다.

A Performance Analysis of the Reliable Multipeer Protocol (ReMP) for Dynamic Multimedia Group Communications

Ji-Yeon Son[†] · Yoo-Jae Won[†] · Soo-Hyoung Oh[††] · Kyung-Shik Lim[†] · Seung-Ku Hwang[†]

ABSTRACT

Reliable Multipeer Protocol (ReMP) is a reliable multipeer data transfer protocol, which has been designed to support multimedia group communications. In this paper, we prove the correctness of ReMP and analyze the performance theoretically. In addition, ReMP is compared to other reliable multicast protocols focusing on the performance. Our analytical results show that ReMP can be more generally used for multimedia group applications, and are acceptable on the performance and the memory requirement.

1. 개 요

최근 네트워크를 통해 보다 고품질의 멀티미디어 그룹 통신 서비스들이 요구되면서, 신뢰성 있는 다자간 전송 프로토콜의 중요성이 크게 부각되고 있다. 이를 위해 기존에도 많은 다자간 전송 프로토콜[1, 10, 11]들이 제안되어졌는데, 이들을 데이타 전송의 관점에서 보면, 대체로 몇가지 부류로 나눌 수 있다. [2, 3]에서는 이들을 송신자 중심 프로토콜과 수신자 중심 프로토콜로 나누어 분류하였으며, [8]에서는 상기 두 부류에 링 기반 프로토콜과 트리 기반 프로토콜을 추가하였다. 이들 각 부류들에 대한 정의는 다음과 같이 일반화시켜 기술한다[8]. 송신자 중심 프로토콜은 멀티캐스트 전송에 있어 송신자 모든 수신자들로부터 응답을 받도록 하는 프로토콜을 말하며, 수신자 중심 프로토콜은 NACK(Negative Acknowledgement)을 기반으로 하는 프로토콜로서, ACK(Positive

† 정 회 원:한국전자통신연구원 멀티미디어 연구부
†† 정 회 원:해동정보시스템
논문접수:1997년 7월 29일, 심사완료:1997년 11월 17일

Acknowledgement) 패킷을 사용하지 않는 프로토콜로 정의한다. 또한 링 기반 프로토콜은 멤버들간에 논리적 링(logical ring)을 구성하여, 링을 따라 토큰을 전달하는 방식을 사용하는 프로토콜을 말한다. 이 프로토콜은 토큰 보유자로 하여금 송신자에게 ACK 패킷으로 응답하는 역할과, 다른 멤버들에게 데이타 재전송을 책임지는 역할을 담당하도록 한다. 반면, 트리 기반 프로토콜은 수신자들을 계층별로 그룹화하여 트리 형태를 구성하고, 각 중간 노드들로 하여금 바로 밑 단계의 자식(children) 노드들에 대해 재전송을 책임지도록 하는 프로토콜로 정의된다.

[2, 3]에서 저자들은 수신자 중심 프로토콜이 송신자 중심 프로토콜보다 전송률(throughput) 측면에서 훨씬 성능이 우수함을 보였다. 그러나, [8]에 의해 수신자 중심 프로토콜은 버퍼가 제한되어 있을 경우, 데드락(deadlock) 현상이 발생됨이 증명되었다. 또한, [8]에서는 트리 기반 프로토콜이 다른 부류들에 비해 전송률 측면에서 확장성이 가장 뛰어남을 보였다. 그러나 트리 기반 프로토콜은 트리에서의 중간 노드들에게 재전송 책임을 전가 시킴으로써 후에 언급하게 될 트리 상의 지연 문제를 유발시키며, 멤버들이 변경될 때마다 매번 ACK 트리를 재형성함으로써 부가적인 시간이 필요하다는 단점을 갖는다. 이는 멤버 변경이 자주 일어날 경우 프로토콜의 효율성이 크게 떨어질 수 있음을 의미한다. 현재 대부분의 응용 프로그램들이 동적인 멤버 관리를 필요로 한다는 점을 감안한다면, 트리 기반 프로토콜의 적용 범위는 매우 제한적이라고 할 수 있다.

본 논문에서는 ReMP(Reliable Multipeer Protocol) [7, 12, 13, 14]라고 하는 새로운 신뢰성 있는 다자간 전송 프로토콜을 소개하고자 한다. 이는 멀티미디어 그룹 통신 기능을 지원하기 위해 설계되어진 프로토콜로서 강력한 그룹 관리 기능을 가지고 있다. 본 논문에서는 데이타 전송의 성능에 초점을 맞추고 있으므로, 그룹 관리에 대한 자세한 사항들은 [7, 12]을 참조하기 바란다.

본 논문의 구성은 다음과 같다. 2장에서는 ReMP 프로토콜의 동작을 데이타 전송 측면에서 기술하며, 3장에서는 ReMP가 제한된 버퍼 환경에서도 올바르게 동작함을 증명한다. 이어 4장에서는 호스트에서의 처리량 관점에서 ReMP의 성능을 분석하며, 이를 토대로 5장에서는 ReMP와 다른 프로토콜들의 성능을 비교하여 본다. 마지막으로 6장인 결론에서는 5장에서의 분석 결과를 정리한 후, ReMP의 현재 상태 및 향후 계획을 기술하기로 한다.

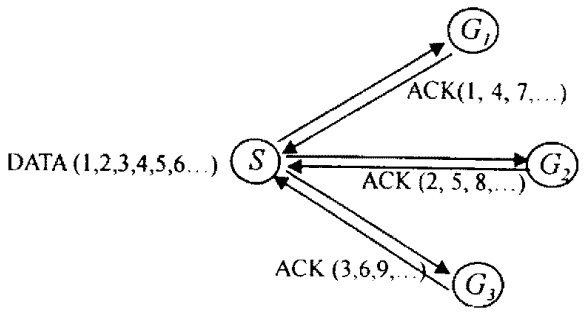
2. ReMP 프로토콜 기술

하나의 연결 내에 n명의 멤버들이 포함되어 있는 그룹을 $G=\{m_1, m_2,..., m_n\}$이라고 표기한다. ReMP에서 G에 속해 있는 각각의 멤버들은 멤버 집합 $M=\{m_0, m_1,..., m_{n-1}\}$에 일대일로 대응되어 지는데, i가 $0..n-1$의 범위 내에 있을 때 $m_i \in M$는 $m_{i+1} \in G$과 일치한다. M은 동적으로 변화되며, 이러한 변화를 시간의 흐름에 의해서 나타낼 때, $TS(M)$이라고 표기한다. 만일 $TS(M) < TS(M')$이면, M'는 M보다 나중에 형성된 멤버 집합이다.

ReMP에서는 그룹 G내에 멤버 집합 M을 생성하는 멤버 ($m_l \in G$)를 연결 관리자라고 하는데, 이때 연결 관리자는 연결 설정 과정에서 각각의 멤버들을 몇개의 서브그룹(sub-group) $G_i (i=1..\omega)$로 나누고, 이의 구분자인 i를 멤버들에게 할당하여 준다. 그룹 G는 멤버들간의 물리적 거리나 처리 능력 등 다양한 기준에 의해 서브그룹으로 나눌 수 있지만, 본 논문에서 이러한 기준은 사용자에 의해 주어진다고 가정한다. 이때 ω는 그룹 멤버수 n과 송신 윈도우의 크기의 절반인 $\frac{W}{2}$ 중에서 최소값으로 결정한다($\omega = min[n, W/2]$). 또한 $k=\left\lceil \frac{n}{\omega} \right\rceil$ 이라 할 때, 각 서브그룹의 크기는 k에 제한되어진다. 즉, $\|G_i\| \le k$이며, 이 조건을 서브그룹 크기 조건이라고 부른다.

서브그룹의 구분자인 i는 매 데이타 패킷마다 포함되어지며, 송신자에게 응답할 의무를 갖는 수신자들을 선택하는데 사용되어진다. ReMP에서는 이 수신자들을 *ACKer*라고 하는데, 같은 서브그룹에 속한 멤버들의 수에 따라 *ACKer*의 수가 결정된다. 따라서 상기 서브그룹 크기 조건에 의해, 하나의 패킷이 송신될 때마다 k개의 *ACKer*가 존재 한다. 이러한 방법을 통해서 ReMP는 송신측에서 요구되는 최소한의 송신 버퍼량을 일정 크기인 ω개 패킷으로 제한할 수 있다. 다시 말해, 송신자는 최대 ω개 만큼의 데이터 패킷을 송신하고 난 후에는 이전 패킷을 송신 버퍼에서 지울

수 있음을 의미한다. 이는, 수신자 수가 증가함에 따라 송신측에서의 메모리 요구량도 함께 증가되는 현상을 방지할 수 있다.



(그림 1) *ACKer*의 순환
(Fig. 1) Designation of *ACKer*

그림 1은 ReMP 내에서 *ACKer*가 순환되는 방식의 예를 나타낸 것이다. 먼저 ω가 3이고, 송신자 S가 DATA(1, 2, 3, 4, 5, 6...)의 데이터 스트림을 연속해서 전송한다고 한다고 가정한다. 이때, 서브 그룹 G_1에 속한 멤버들은 그림 1에서와 같이 1, 4, 7 데이터 패킷들에 대해 ACK 패킷으로 응답해야 한다. 또한 서브 그룹 G_2, G_3들도 각각 ACK(2, 5, 8...)과 ACK(3, 6, 9...)로 송신자에게 응답한다.

데이타를 송수신할 때, ReMP의 동작을 간략하게 기술하면 다음과 같다. 주어진 멤버 집합 M에 대해, 각각의 멤버 m_i는 연결관리자로부터 데이타를 송신할 수 있는 권한을 획득한 후에 데이타를 전송할 수 있다. 송신자들은 서브그룹 구분자를 포함한 데이타 패킷을 M의 모든 멤버들에게 멀티 캐스트하며, 이에 대한 재전송을 담당한다. 데이타 패킷을 올바르게 수신하면, 데이타 패킷 내의 서브그룹 구분자와 일치하는 k개의 *ACKer*들은 송신자에게 *ACK* 패킷으로 응답을 한다. 그 *ACK* 패킷 내에서 마지막으로 자신이 *ACKer*로 지정된 이후, 패킷 수신 상태에 관한 정보를 포함한다. 각 *ACKer*들에 대한 지정은 패킷 단위의 라운드 로빈 방식으로 이루어진다. 송신자는 일정 시간 동안 *ACKer*들로부터 k개의 *ACK*을 모두 수신받지 못하면, 다시 해당 패킷을 재전송한다. 수신자들의 경우 패킷 손실이 감지되면, 해당 패킷에 대한 NACK 패킷을 즉시 유니캐스트(unicast)한다. 앞에서 언급한 바와 같이, 손실된 데이타에 대한 재전송은 송신측에서 멀티캐스트에 의해 이루어지되, 재전송의 부담을 줄이기 위해서, 일정 시간동안 같은 데이타에 대해 NACK을 수신했을 경우에는 처음 요청에 대해서만 재전송을 하고, 나머지는 무시한다. 이는 송신자가 수신자들로부터 중복된 NACK을 계속 받을 경우, 재전송 데이타량이 급증함으로써 네트워크의 부하가 증가되는 현상을 피하기 위해서이다.

3. 프로토콜 동작의 올바름

본 논문에서는 버퍼가 제한되어 있는 환경에서 ReMP가 올바르게 동작한다는 것을 증명하기 위해서, 먼저 M내의 대다수 멤버들이 운용 가능하고, 연결관리자를 제외한 임의의 멤버측에 언제든지 고장이 발생될 수 있다는 전제를 한다. 연결관리자 측의 고장이나 연결관리자와 다른 멤버들간의 네트워크 링크 상에 문제가 생길 경우, M 내에서 새로운 연결관리자를 선정하는 방법도 생각할 수 있다. 그러나, 이에 대한 부담이 일반적으로 크다고 알려져 있다[6], 이러한 복구 가정 동안 손실되는 데이타량도 매우 클 것으로 판단되어 ReMP에서는 다른 멤버들에 의해 연결관리자의 고장이 감지될 경우, 바로 연결 해지 절차를 수행하도록 하고 있다. 구체적인 연결 해지 절차에 대한 설명은 본 논문에서 생략하기로 한다.

먼저, 본 장에서 사용하는 몇가지 용어에 대한 정의를 하고자 한다.

정의 1: M이 연결관리자 $m_i(m_i \in M)$를 포함하고 있고, 데이타를 순서적으로 전달하며, $TS(M) < TS(M')$ 조건을 만족하는 M'가 존재하지 않는 경우에는, M이 운용가능하다라고 정의한다.
정의 2: 두개의 멤버 집합인 M과 M'가 존재할 때, M'가 운용가능하고 $TS(M') > TS(M)$이면, $M \sim > M'$이라고 표기한다.

명제 1: 연결관리자 m_i가 G에서 운용 가능할 때, ReMP는 결국 멤버의 고장을 감지하고 복구할 수 있다.
증명: 본 명제는 멤버의 고장을 감지하고 복귀하는 알고리듬을 기술함으로써 증명할 수 있다.
(1) 고장 감지: 멤버 m_i 측에 고장이 발생하였다고 가정하면, *ACKer* 지정 방식에 의해서, m_i는 최대 ω개

의 데이타 패킷이 보내지기 전에 *ACKer*로 지정될 것이다. 이때 m_i에 대한 타이머 δ가 만료되고 나면, m_i의 고장을 감지하게 될 것이다.

(2)고장 복구:본 프로토콜에서는 복수의 멤버들이 m_i의 고장을 감지할 수 있다. 이를 감지한 멤버들은 G내의 다른 모든 멤버들에게 *site_failure* 패킷을 전송하고, 타이머 δ'를 가동시킨다. *site_failure* 패킷을 수신한 멤버들은 중복해서 발생하지 않도록 한다. 연결관리자가 하나 이상의 *site_failure* 패킷을 받고 나면, 새로운 멤버 집합인 M'를 생성한 후, 이를 모든 멤버들에게 통보하여 준다. 따라서, 결국 $((\|M'\| \geq \|M\|-1) \wedge (m_l \in M'))$ 조건을 만족하게 된다.

정리 1:만일 연결관리자 m_l가 운용 가능하고, 네트워크 분리가 발생되지 않는다면, M과 M'내 임의의 멤버들에 대해서 ReMP는 결국 $M \sim > M'$로 진행된다.

증명:연결관리자가 G에서 운용 가능하므로, 명제 1에 의해서, 연결관리자는 새로운 멤버 집합을 형성하여 멤버들에게 멀티캐스트 할 것이다. 연결관리자에 의해서 보내진 새로운 멤버 집합 M'는 순서를 보장하는 패킷에 의해 전송된다. 따라서 나머지 수신자들은 이를 올바르게 수신하게 될 것이며, 기존의 M을 M'로 수정할 것이다. 그러므로, ReMP는 결국 $M \sim > M'$로 진행된다.

증명:연결관리자가 G에서 운용 가능하므로, 명제 1에 의해서, 연결관리자는 새로운 멤버 집합을 형성하여 멤버들에게 멀티캐스트 할 것이다. 연결관리자에 의해서 보내진 새로운 멤버 집합 M'는 순서를 보장하는 패킷에 의해 전송된다. 따라서 나머지 수신자들은 이를 올바르게 수신하게 될 것이며, 기존의 M을 M'로 수정할 것이다. 그러므로, ReMP는 결국 $M \sim > M'$로 진행된다.

정리 2:만일 연결관리자가 운용 가능하다면, ReMP는 네트워크 분리가 발생한다고 하더라도, 결국 $M \sim > M'$로 진행된다.

증명:G가 만일 G_a와 G_b로 분리 되었을때, G_a는 연결관리자를 포함하고, G_b는 포함하지 않는다고 가정한다. $m_i \in G_a$인 일부 멤버들은 정리 1에 의해서 G_b에 속한 멤버들으 고장을 감지하게 된다. 이들은 복수의 고장 발생 메시지를 전송하며, 연결관리자는 이러한 메시지를 수신할 때마다 새로운 멤버 집합을 형성하게 되고, 결국 M'를 형성하게 될 것이다. 이를 연결관리자는 G_a에 속한 모든 멤버들에게 통보할 것이다. 반면 G_b내의 멤버, $m_j \in G_b$ 역시 G_a에 속한 멤버들의 고장을 감지하고, 이 사실을 멀티캐스트 한 후, 타이머 δ을 동작시킬 것이다. 그러나, G_b는 연결관리자를 포함하고 있지 않기 때문에, 각 m_j 노드에서는 결국 타이머 δ가 만료될 것이며, 상기에서 언급한대로 연결해지 절차를 수행하게 될 것이다. 따라서 어느 시점에서 두개의 멤버 집합이 동시에 정상적으로 운용되지 않게 될 것이다. 그러므로, M'는 결국 G_b 내에서 정상적으로 운영되게 될 것이다.

정리 3(Liveness):연결관리자가 운영 가능할 경우, ReMP는 데드락이 발생되지 않으며, 데이터의 전달도 무제한 지연되지 않는다.

증명:본 정리는 모순에 의해 증명하기로 한다. 데드락이 존재한다고 가정할 때, 다음 두가지 경우를 생각할 수 있다.

(1)먼저, ReMP가 정상적으로 수행될 때 데드락이 발생한 경우를 고려하기로 한다. 송신자가 저장할 수 있는 최대 패킷의 양을 B라고 하고, M내의 모든 수신자들에게 신뢰성 있게 데이타를 전달하기 위해 필요한 최소한의 송신버퍼량을 B'라고 가정한다면, 데드락 현상은 $B < B'$일 경우에 발생된다. 그러나, ω의 정의에 따라 $\omega \leq B$이며, 2장에서 언급한 서브그룹크기 조건에 의해서 $\omega = B'$이므로, $(B \geq \omega) \wedge (B' = \omega)$ 조건에 의해 항상 $B \geq B'$가 될 것이다. 따라서 $B < B'$ 조건은 결코 만족되지 않을 것이므로, ReMP는 데드락 현상이 발생되지 않는다.

(2)ReMP가 멤버의 고장이나 네트워크의 분리로 인해 데드락이 발생하였다면, 정리 1과 정리 2에 의해 결국 복구될 것이다. 그러므로, ReMP는 정상적으로 수행될 수 있으며, 어떠한 경우에는 데드락이 발생되지 않는다.

상기 증명에 의해 ReMP는 송신 버퍼가 제한되어 있는 환경에서도 올바르게 동작됨을 알 수 있다.

4. 성능 분석

본 논문에서는 ReMP의 성능 분석을 위해 [3]에서 사용한 모델을 이용하였다. 이 모델은 기존의 네트워크 대역폭의 관점에서 성능 분석을 시도했던 것과는 달리, 종단 시스템에서의 처리량(processing requirement)에 초점을 맞추었다. 이러한 접근 방식은 최근 호스트 측의 처리량 폭주가 데이타 손실의 큰 원인이 되고 있다는 점에서 타당하다고 보여진다.

성능 분석은 (1)단위 시간당 최대 패킷 전송량(thoughput)과 (2)한 패킷이 모든 수신자들에게 올바르게 전달되는데 소요되는 최대 지연 시간(delay)과 (3)송신 버퍼에서 패킷을 지울 수 있는 시간을 분석하는 과정을 포함한다.

최대 전송량은 각 송신자 측과 수신자 측에서 하나의 패킷을 처리하는데 소요되는 최소 시간을 계산하여 이의 역을 취함으로써 얻어질 수 있다. 또한 지연시간은 송신측과 수신측에서 하나의 패킷에 대한 처리 시간의 합에 순수 네트워크 통과 시간(propagation delay time)을 추가함으로써 구해질 수 있다.

4.1 가 정

본 논문에서는 성능 분석을 하는데 있어서 다음과 같은 가정을 한다.

(1)M내의 송신자는 연속적인 스트림의 형태로 데이타를 전송한다.

(2)수신측에서 발생되는 모든 패킷 손실은 서로 독립적으로 발생되며, 각각의 패킷 손실률 p는 수신자 측과 무관하다. 사실 대부분의 다자간 경로 설정 프로토콜들은 트리 형태를 기반으로 하고 있으므로, 특정 링크 상의 패킷 손실은 해당 링크를 공유하는 수신자들 전체에 영향을 미친다. 그러나, 본 논문에서는 프로토콜의 최대 전송량에 초점을 맞추고 있으므로, 이 가정이 이에 대한 최소 범위를 결정하는 역할을 할것으로 간주된다.

(3)ACK/NACK 패킷은 손실되지 않는다. 이 가정은 모델을 크게 단순화 시키고, 전체적인 성능 관점에서 프로토콜 동작을 바라볼 수 있도록 한다.

(4)송신자로부터 각 수신자에 이르는 순수 네트워크 통과 시간은 그 역방향을 통과하는데 걸리는 시간과 같다.

4.2 용 어

본 논문에서 사용되는 용어들은 나음과 같다.

R	수신자 수; $R = \|G\| - 1$.
X_f	사용자로부터 데이타를 받아 데이타 패킷을 생성하는데 걸리는 시간.
X_p	하나의 패킷을 전송하는데 걸리는 시간
X_a, X_n	하나의 ACK 또는 NACK 킷을 수신하여 처리하는데 걸리는 시간
X_t, Y_t	각 송수신측에서 타이머를 처리하는데 걸리는 시간, 여기에는 타이머를 동작시키거나 중단, 또는 실행시키기 위한 시간들이 포함된다.
Y_p	새로운 패킷을 수신하는데 걸리는 시간
Y_f	상위 계층에 데이타를 전달하는데 걸리는 시간
Y_a, Y_n	ACK 또는 NACK 패킷을 생성하여 전송하는데 걸리는 시간
p	수신측에서의 패킷 손실 확률
L_r^A L_r^N	패킷 마다 수신자 r에 의해서 보내진 ACK 또는 NACK의 수
M_r	패킷 한개에 대해 수신자 r이 성공적으로 받도록 하기 위한 전송 횟수
M	패킷 한개에 대해 모든 수신자들이 성공적으로 받도록 하기 위한 전송 횟수; $M = max_r\{M_r\}$
X, Y, A	프로토콜 내 송신측, 수신측, *ACKer*측에서의 단위 패킷당 처리 시간
Λ_x	ReMP의 전송량. 첨자인 x에는 송신자 s, 수신자 r, 또는 *ACKer*-노드 a일 수 있다. 시스템의 전체 전송량의 경우에는 첨자가 존재하지 않는다.
D	해당 패킷에 대해 모든 수신자들의 사용자에게 올바르게 전달하는데 걸리는 시간
T_p	초당 순수 네트워크 통과 시간. i.e., 단방향 링크를 통과하는데 걸리는 순수 지연 시간.

4.3 최대 전송량(Maximum Throughput) 분석

ReMP 프로토콜의 전송량을 구하기 위해서, 본 논문에서는 송신자측, 수신자측, *ACKer* 노드측의 순으로 각각 기대되는 비용을 계산한다.

송신자측에서의 처리 요구량을 X라고 표기할때, 여기에는 초기 데이타를 전송하는데 걸리는 시간, *ACKer*들로부터 k개 ACK을 수신하거나, 수신자들로부터 NACK을 수신하는데 걸리는 시간, 또한 재전송 시간들이 포함된다. 이는 다음과 같은 수식으로 표현

될 수 있다.

$$X = X_f + X_p(1) + kX_a + \sum_{i=1}^{L_N} X_n(i) + \sum_{i=2}^{M} X_p(m), \quad (1)$$

X_f는 상위 계층의 응용으로부터 데이타를 받기 위해 걸리는 시간이며, $X_p(m)$는 주어진 패킷에 대해 m-번째 전송을 성공적으로 수행하기 위해 걸리는 시간을 말하며, X_a는 해당 패킷에 대해 한 *ACKer*로부터 ACK을 받아 처리하기 위한 시간을 말한다. 또한 k는 *ACKer*들로부터 받은 ACK 패킷의 수이며, L^N은 수신한 NACK의 수, M은 모든 수신자들에게 올바르게 패킷을 전송하기 위해 요구되는 전송 횟수이다.

수식 (1)에 기대값을 취하면,

$$E[X] = E[X_f] + kE[X_a] + E[L^N]E[X_n] + E[M]E[X_p]. \quad (2)$$

L_r^N을 수신측 r에서 송신자에게 반환한 NACK의 수라고 가정할 때, L_r^N은 $M_r - 1$이고, $E[M_r]$은 $1/(1-p)$이므로, $E[L_r^N]$은 $p/(1-p)$값을 갖는다. 따라서, 모든 수신자들로부터 받은 평균 NACK의 수는 다음과 같다.

$$E[L^N] = Rp/(1-p) \quad (3)$$

수식 (3)을 수식 (2)에 대치하면, $E[X]$에 대해 다음과 같은 수식을 산출할 수 있다.

$$E[X] = E[X_f] + kE[X_a] + RpE[X_n]/(1-p) + E[M]E[X_p]. \quad (4)$$

Λ_s를 송신자측에서 수신자들에게 성공적으로 전송하기 위한 전송률이라고 한다면, $\Lambda_s = 1/E[X]$이다.

다음은 임의로 선택한 패킷에 대해 수신측에서의 평균 처리 요구량을 구하기로 한다. 수신측에서의 처리 요구량은 다음과 같이 구해질 수 있다.

$$Y = Y_f + P(M_r = 1)Y_p(1) + P(M_r > 1)\sum_{i=2}^{M} Y_p(i) + P(M_r > 1)\sum_{m=2}^{M_r} Y_n(m) + P(M_r > 2)\sum_{n=3}^{M_r} Y_t(n). \quad (5)$$

수식 (5)를 각 항목 별로 설명하면, 다음과 같다. 첫번째 항목은 상위 계층으로 패킷을 올바르게 전달하기 위해 소요되는 시간을 말하며, 두번째와 세번째 항목은 전송된 데이타 패킷을 올바르게 수신하여 처리하기 위해 요구되는 시간을 말한다. 또한 네번째는 패킷이 손실되었을 때, 송신자에게 NACK 패킷을 생성하여 보내는데 걸리는 시간을 말하며, 다섯번째는 타이머가 만료되었을 때의 처리 시간을 말한다. 마지막 두 항목은 초기 데이타 전송이 실패하였을 경우에만 해당된다.

상기에는 얻은 수식 (5)는 [2, 3]의 수신자 중심 프로토콜의 수신측 처리 요구량의 수식과 같다. 따라서 [2, 3]에서 구한 수신측에서 평균 처리요구량을 다음과 같이 다시 사용할 수 있다.

$$E[Y] = E[Y_f] + (1-p)E[M]E[Y_p] + \frac{p}{1-p}(E[Y_n] + pE[Y_t]). \quad (6)$$

수신측에서의 최대 패킷 처리율을 Λ_r라 하면, $\Lambda_r = 1/E[Y]$이다.

다음은, *ACKer*노드 측에서의 처리 요구량을 분석한다. 이는 수신측의 처리 요구량에 송신자에게 ACK 패킷을 생성하여 전송하는 시간이 더해진 값이다. 여기에 기대값을 구하면, 다음 식을 얻을 수 있다.

$$E[A] = E[Y] + E[Y_a]. \quad (7)$$

ACKer 측에서의 처리율을 Λ_a라고 하면, $\Lambda_a = 1/E[A]$이다.

ReMP의 전체 시스템의 최대 전송률은 송수신자, 또는 ACKer 노드 각각에서의 패킷 처리률의 최소값과 같다.

$$\Lambda = min\{\Lambda_s, \Lambda_a, \Lambda_r\}. \quad (8)$$

패킷 손실률 p가 상수이거나, $p \to 0$이라면, 수식 (4), (6), (7)에 의해 전체 시스템의 최대 전송률은 다음과 같이 제한되어진다.

$$1/\Lambda \in O\left(1 + \frac{Rp}{1-p}\right); p \text{ constant}, \quad (9)$$

$$1/\Lambda \in O(1) \quad ; p \to 0. \quad (10)$$

상기에서 보는 바와 같이 p가 상수일 때에는 수신자 수에 따라 처리량이 제한 받음을 알 수 있으며, $p \to 0$일 경우, ReMP의 최대 전송률은 O(1)이고, 수신자 수에 무관해짐을 알 수 있다.

4.4 전송 지연 시간 분석

본 논문에서의 전송 지연은 데이타 패킷이 송신측으로부터 모든 수신자측의 사용자에게 올바르게 전달되기 위해 소요되는 총 지연시간으로 정의한다[4]. 송신측에서 수신측까지의 지연 시간을 D라고 할때, 여기에는 송신측에서의 처리 요구량과 수신측에서의 처라 요구량, 그리고 데이타 패킷을 올바르게 전달하기 위해 순수 네트워크 통과 시간들이 포함된다. 이는 다음과 같이 표현될 수 있다.

$$D = X_p(1) + T_p + P(M_r = 1)Y_p(1)$$
$$+ P(M_r > 1)\sum_{i=2}^{M} Y_p(i) + (Y_a + T_p)/\omega + kX_a$$
$$+ P(M_r > 1)\sum_{m=2}^{M_r}(Y_n(m) + T_p)\sum_{i=1}^{L^N} X_n(i)$$
$$+ \sum_{j=2}^{M}(X_p(j) + T_p) + P(M_r > 2)\sum_{n=3}^{M_r} Y_t(n) + Y_f. \quad (11)$$

상기 수식에서 사용된 용어들의 대부분은 최대 전송률을 분석하면서 사용되어진 것들이다. 다만, T_p는 순수 지연 시간이며, $1/\omega$은 수신자가 *ACKer*로 지정될 확률을 의미한다.

수식 (11)에 기대값을 취하면, 다음과 같이 평균 전송 지연 시간을 구할 수 있다.

$$E[D] = E[X_p] + E[T_p] + (1-p)E[M]E[Y_p]$$
$$+ (1-p)(E[Y_a] + E[T_p])/\omega + kE[X_a]$$
$$+ \frac{p}{1-p}(E[Y_n] + E[T_p] + pE[Y_t]) + \frac{Rp}{1-p}E[X_n]$$
$$+ \frac{p \ln R}{1-p}(E[X_p] + E[T_p]) + E[Y_f]. \quad (12)$$

4.5 패킷 삭제 시간

패킷 삭제 시간은 하나의 패킷을 송신한 후, 송신 버퍼에서 해당 패킷을 삭제할 수 있는 시간이다. ReMP의 패킷 삭제 시간은 서브그룹 G_i의 개수인 ω에 제한된다. 즉, 송신자는 모든 G_i 그룹이 모두 한 차례씩 *ACKer*로 지정된 이후에는 해당 패킷을 송신 버퍼에서 지울 수 있음을 의미한다. 이는 송신자 측에서의 송신 버퍼량을 수신자의 수와 독립시킴으로써 확장성(scalability)을 증대 시키는 역할을 한다.

5. 결과 분석

본 장에서는 4장에서는 최대 전송률과 전송 지연 시간 분석을 위해 사용한 수식을 이용한다. 또한 1장에서 언급한 송신자 중심 프로토콜, 수신자 중심 프로토콜, 트리 기반의 프로토콜들과 ReMP의 장단점을 파악하고자 한다. ReMP 이외의 다른 프로토콜들에 대한 수식은 [2] [3] [8]에서 주어진 식을 이용한다. 이때 각 프로토콜마다 NACK 폭주를 막기 위한 알고리듬은 사용되지 않는다고 가정한다. 이는 본 논문에서의 가정 (3) (ACK 또는 NACK 패킷은 손실되지 않는다.)에 의해 상대적으로 다른 프로토콜보다 이득을 갖게 되기 때문이다. 또한 NACK 폭주 방지 알고리듬은 다른 프로토콜들보다 NACK 패킷의 손실률에 더욱 민감하기 때문에 본 논문에서 비교하는데 적합하지 않다.

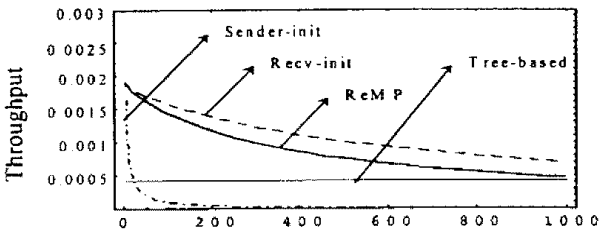
성능 비교를 위해서는 우선, ACK/NACK 패킷과 데이타 패킷을 송신하고, 수신하는데 걸리는 처리 시간에 대해 합당한 측정값이 필요하다. 이를 위해 본 논문에서는 기존의 연구들 가운데에서 펜티움 PC의 리눅스(Linux) 커널 환경에서 각각의 처리 시간들을 반복적인 실험을 통해 측정한 논문[9]의 결과를 이용하기로 한다. 다음 〈표 1〉은 [9]에서 측정한 결과치를 기반으로 작성된 것이다.

〈표 1〉 처리 시간들(microsecond 단위)
〈Table 1〉 Processing Times (in microseconds)

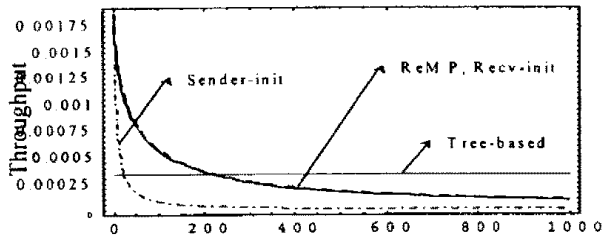
용어	평균 처리 시간	표준 편차
X_p	502	13.8
X_n, X_a	87	5.8
Y_p	487	12.6
Y_n, Y_a	86	5.1
X_t, Y_t	32	1.5
X_f, Y_f	22	1

성능 비교시에 사용되는 몇가지 변수들을 고려하여 보면, 먼저 ReMP에서 *ACKer*의 수인 k는 2장에서 언급한 바와 같이 $\left\lceil \frac{n}{\omega} \right\rceil$ 에 의해 결정되는데, 이때 $\omega = 64$로 한다. 또한 트리 기반의 프로토콜에서 사용되는 트리의 가지수(branching factor)는 20으로 한다.

(그림 2)의 (a)와 (b)는 서로 다른 패킷 손실률에 대해 수신자 수가 증가함에 따라 ReMP와 다른 프로토콜들의 전송률이 어떻게 변하는지를 보여 준다. 다음 그래프는 ReMP와 다른 프로토콜들의 전송량 수식들의 역에 의해 구해졌다.



(a) $p = 0.01$



(b) $p = 0.1$

(그림 2) 전송량의 비교

(Fig. 2) The throughput comparison

패킷 손실률이 낮은 환경에서는 (그림 2)(a)와 같이 수신자 중심 프로토콜의 전송량이 ReMP 보다 좋다. 그러나, (그림 2)(b)에 의해 패킷 손실률이 높아지면, ReMP 전송량과 수신자 중심 프로토콜의 전송량이 동일해짐을 볼 수 있다. 사실, ReMP 역시 NACK을 기반으로 하는 프로토콜이되, 제한된 버퍼 환경에서도 올바르게 동작하도록 설계되어진 프로토콜이다. 그러므로 ReMP의 전송량에는 기존의 수신자 중심 프로토콜의 전송량에 버퍼 문제를 해결하기 위해 소요되는 처리 시간이 추가된다. 따라서 추가된 시간만큼의 전송량 감소를 예측할 수 있으며, 이는 (그림 2)의 (a)에서 보이는 ReMP와 수신자 중심 프로토콜간의 전송량 차의 주원인임을 알 수 있다.

(그림 2)에서 보는 바와 같이, 패킷 손실률이 높아지고, 수신자 수 R이 증가할수록, 트리 기반 프로토콜은 일정한 전송량 값을 보이는데 반해, ReMP와 수신자 중심 프로토콜은 전송량이 감소됨을 보여준다. 반면, 패킷 손실률이 낮거나, 수신자 수가 그리 크지 않은 경우에는 ReMP와 수신자 중심 프로토콜이 다른 프로토콜들보다 전송량에 있어서 좋은 성능을 보이고 있다.

전송 지연의 관점에서의 성능을 비교하기 위해서는 ReMP 이외 다른 프로토콜들의 전송 지연에 대한 수식이 필요하다. 따라서 본 논문에서는 이들 수식들을 4.4절의 ReMP 전송 지연 수식을 유도하는 비슷한 방식으로 구하였다. 송신자 중심 프로토콜, 수신자 중심 프로토콜, 트리 기반 프로토콜의 전송 지연 시간을 각각 D^A, D^R, D^T라 할 때, 평균 전송 지연 시간들은 다음과 같다.

$$E[D^A] = E[M](E[X_p] + E[T_p]) + (E[M]-1)E[X_t] + R(1-p)E[M](E[X_a] + E[T_p]) + E[M](1-p)(E[Y_p] + E[Y_n] + E[T_p]) + E[Y_f], \quad (13)$$

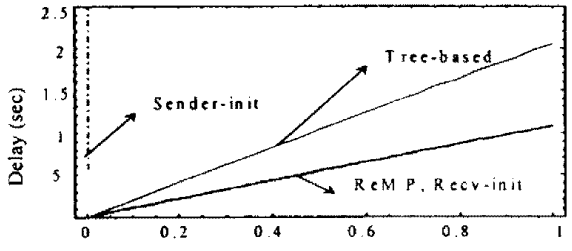
$$E[D^R] = E[M](E[X_p] + E[T_p]) + (1-p)E[M]E[Y_p] + \frac{p}{1-p}(E[Y_n] + E[T_p] + pE[Y_t]) + \frac{Rp}{1-p}E[X_n] + E[Y_f], \quad (14)$$

$$E[D^T] = \log_B RE[M](E[X_p] + E[T_p]) + \log_B R(E[M]-1)E[X_t] + B(1-p)E[M]E[X_h] + E[M](1-p)(E[Y_p] + E[Y_h] + E[T_p]) + E[Y_f]. \quad (15)$$

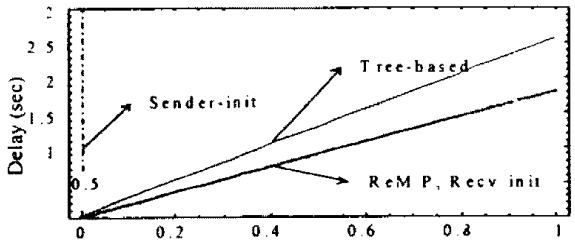
각 프로토콜의 상기 수식들은 대부분 [8]의 논문에서 전송량 성능 분석시에 사용되어졌던 것들이다. 이때, 수식 (13), (14)에서의 $E[M]$은 [2, 3]에 의해 $O(1 + \frac{p \ln R}{1-p})$인데 반해서, 트리 기반 프로토콜의 수식 (15)에서 사용된 $E[M]$은 [8]에 의해 $O(1 + \frac{p \ln B}{1-p})$에 제한됨을 주의해야 한다. 또한, 수식 (15)의 $\log_B R$은

트리의 최대 높이로서, [8]에서도 언급된 적이 있는 지연 문제(*latency problem*)와 연관된다. 예를 들어, 패킷이 송신측(트리의 최상위 노드라고 가정)에서 바로 손실되었을 경우에, 송신자는 한단계 아래의 자식(children)노드들에 대해서만 재전송 책임을 가지고 있으므로, 트리의 가장 밑단에 있는 노드(leaf node)들의 경우에는 트리 높이 만큼의 재전송 단계를 거쳐야 해당 패킷을 올바르게 받을 수 있다. 따라서 모든 수신자들이 패킷을 올바르게 전송받기 위해 소요되는 시간에는 트리의 최대 높이 만큼의 재전송 지연 시간이 포함되어져야 한다.

(그림 2)와 (그림 3)은 ReMP와 다른 프로토콜들의 전송 지연 시간 수식 (12), (13), (14), (15)을 기반으로 그려진 그림들이다.



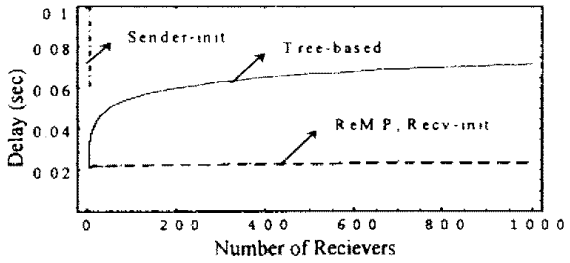
(a) $p = 0.01$, $R = 500$



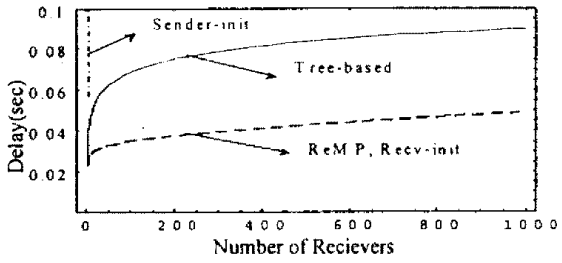
(b) $p = 0.1$, $R = 500$

(그림 3) 전송 지연 시간 비교
(Fig. 3) The delay comparison

(그림 3)은 서로 다른 패킷 손실률에 대해, 순수 네트워크 통과 시간 T_p가 ReMP와 다른 프로토콜들의 전송 지연에 얼마만큼의 영향을 보이는지를 나타낸 그림이다. 이 그림을 보면, 트리 기반 프로토콜이 ReMP나 수신자 중심 프로토콜보다 T_p의 변화에 더욱 민감하다는 것을 알 수 있다. 이는 앞에서도 언급한 바가 있는 지연 문제가 그 원인임을 추측할 수 있다. 또한 이 그림은 수신자 중심 프로토콜과 ReMP의 전송 지연의 변화가 T_p에 대해 동일하다는 사실도 보여주고 있다. 송신자 중심 프로토콜의 경우, 다른 프로토콜들과의 전송 시간 차가 너무 큰 관계로 이 그림에는 나타나지 않았지만, 매우 급격하게 변화함을 짐작할 수 있다.



(a) $p = 0.01$



(b) $p = 0.1$

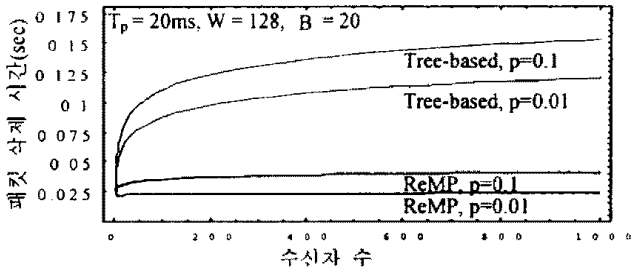
(그림 4) 전송 지연 시간 비교(수신자 수)
(Fig. 4) The delay comparison(delay versus number of receivers)

(그림 4)는 ReMP와 다른 프로토콜들을 수신자 수의 변화에 따른 전송 지연 시간의 관점에서 성능 비교를 한 것이다. 이때 순수 네트워크 통과시간은 20ms로 고정하였다. 그 결과는 기대하였던 바와 같이 ReMP와 수신자 중심 프로토콜이 트리 기반 프로토콜보다 전송 지연 측면에서 훨씬 성능이 우수함을 보여 주었다.

다음은, 패킷 삭제 시간의 관점에서 ReMP와 다른 프로토콜들을 비교하고자 한다. 먼저, 송신자 중심 프로토콜의 경우에는 다른 모든 수신자들로부터 ACK 패킷을 수신하여야만, 송신자 측의 송신 버퍼에서 해당 패킷을 삭제할 수 있다. 따라서, 송신자 측의 패킷 삭제 시간이 수신자 수가 증가함에 따라서 급격하게 증가할 것임을 쉽게 추측할 수 있다. 또한 수신자 중심 프로토콜의 경우에는 [8]의 논문에 의해서 패킷 삭

제 시간이 무한임이 증명되었다. 따라서 본 논문에서는 이에 대해 ReMP와 트리 기반 프로토콜만을 비교하기로 한다.

(그림 5)는 ReMP와 트리 기반 프로토콜의 패킷 삭제 시간을 비교한 것이다. 트리 기반의 프로토콜의 경우에는 패킷을 지우는 시점을 결정하기 위한 별도의 ACK(*aggregate ACK*)을 사용한다. 트리내에 가장 밑단계를 제외한 노드들, 즉 송신자 및 중간 노드들은 자신보다 바로 아래 단계의 모든 자식 노드들로부터 *aggregate ACK*을 받아야만, 해당 패킷을 버퍼에서 삭제할 수 있으며, 자신의 차상위 노드에게 다시 *aggregate ACK*을 전송할 수 있다. 따라서 트리 기반 프로토콜의 경우에는 패킷 삭제 시간을 위해 수식 (15)를 재사용할 수 있다. 반면, ReMP는 이를 위해 4.4절의 전송 지연 수식에 ω값을 적용한 결과를 이용할 수 있다.



(그림 5) 패킷삭제시간
(Fig. 5) Memory deallocation

(그림 5)는 ReMP와 비교하였을때, 트리 기반 프로토콜의 패킷 삭제 시간이 수신자 수 증가에 따라 지연 문제로 인해 큰 폭으로 증가함을 보여 준다. 따라서 제한된 버퍼 환경에서, 트리 기반 프로토콜의 전송량은 패킷 삭제 시간에 제한을 받게 될 것이다. 반면, ReMP의 경우에는 패킷 삭제 시간이 거의 일정함을 보여준다.

6. 결 론

본 논문에서는 새로운 신뢰성 있는 다자간 전송 프로토콜인 ReMP를 데이터 전송의 성능 관점에서 기술하고 분석하였다. ReMP는 수신자 측의 NACK 패킷을 기반으로 하되, 수신자들을 복수개의 서브그룹으로 나누어, 순서대로 *ACKer*를 지정함으로써, 제한된 버퍼 환경에서도 올바르게 동작하도록 설계되었다. 또한 5절의 분석 결과에서 본 바와 같이, 성능 관점에 있어서도 ReMP가 다른 프로토콜들에 비해 비교적 성능이 좋음을 알 수 있다.

[8]에서는 트리 기반 프로토콜이 전송량 측면에서 확장성이 가장 우수함을 보였으나, 본 논문을 통해서 트리 기반 프로토콜은 다음과 같은 제약이 있음이 밝혀졌다. (1)트리 구조로 인한 지연 문제가 순수 네트워크 통과 시간이 길어질수록, 전송 지연 시간에 크게 영향을 미친다는 것을 보여주었다. 그러므로, 트리 기반 프로토콜은 그 적용 범위가 전송 지연이 중요시되지 않는 응용 환경으로 제한된다. (2)수신자 수가 증가함에 따라 송신자 측에서의 패킷 삭제 시간이 급격하게 증가됨을 볼 수 있었다. 더군다나, 송신자 뿐만 아니라 트리 내의 모든 중간 노드들도 바로 밑단계의 자식 노드들로부터 응답을 받지 못하면, 메모리로부터 패킷을 삭제할 수 없다. 이 사실은 수신자 수가 증가할수록, 다른 프로토콜들에 비해 트리 기반 프로토콜의 멤버들 전체 메모리 요구량이 훨씬 커지게 될 것이라는 것을 추측할 수 있도록 한다. 또한 제한된 버퍼 환경에서 패킷 삭제 시간이 길어짐은 전송량 저하를 야기할 수 있다. (3)1장에서 언급한 바와 같이, 동적인 멤버 환경에서 트리 기반 프로토콜은 ACK 트리 설정 절차가 빈번하게 추가됨으로 해서 효율적으로 동작되기가 어렵다는 점이다. 따라서 트리 기반 프로토콜의 경우 동적인 멤버 환경에는 적합하지 못하다고 할 수 있다. (4)트리 내에 하나 이상의 중간 노드들에 고장이 발생할 경우를 대비하기 위해서 송신자뿐 아니라 중간 노드들까지도 모두 일정량의 버퍼를 부가적으로 유지해야 한다. 이상과 같은 제약 사항들에 비추어 볼 때, 트리 기반 프로토콜은 매우 제한된 환경에서만 효율적으로 동작됨을 알 수 있다.

반면, ReMP와 수신자 중심 프로토콜은 다른 프로토콜들에 비해 전송 지연 시간 측면에서 가장 좋은 성능을 보여주고 있다. 또한, 다른 프로토콜들과 달리, ReMP는 수신자 수가 증가 하여도, 송신 버퍼 요구량이 크게 변화되지 않음을 증명하였다. 따라서 ReMP는 상기의 트리 기반 프로토콜이 갖는 제약 사항 (1), (2)에 제한 받지 않는다. 또한, 제약 사항 (3),

(4)는 트리 구조로 인해 발생되는 근본적 문제이므로, ReMP의 경우에는 이들에게 제한되지 않는다. 그러므로, ReMP는 비록 확작성 면에서 트리 기반 프로토콜보다는 다소 떨어지지만, 다른 프로토콜들보다 다양한 용도로 사용될 수 있을 것으로 여겨진다.

현재 ReMP는 1차 버전 설계와 구현을 완료하고, 2차 버전을 시험 중에 있다. ReMP는 PC 상에서, WindowsNT 4.0의 UDP/IP 위에서 구현되었으며, 사용자 라이브러리 형태로 제공된다. ReMP의 향후 계획으로는 본 부서에서 개발하고 있는 ITU-T T.120 프로토콜 스택에 접목시키는 계획을 가지고 있다. 이는 다자간 통신에서 멀티캐스트 프로토콜이 갖는 장점을 T.120에 제공해줄 수 있다는 점과, T.125(MCS)를 통해 ReMP와 T.123에서 규정하고 있는 전송 계층 프로토콜과의 상호 운용이 가능하다는 점에서 의미가 있다고 하겠다. 또한 확장성 문제에 대한 보완도 고려 중에 있다.

참 고 문 헌

[1] Brian Whetten, Simon Kaplan, and Todd Montgomery, "A High Performance Totally Ordered Multicast Protocol," *In Theory and Practce in Distributed Systems*, number 938 in LCNS, Spring Verlag, 1994.

[2] S. Pingali, "*Protocol and Real Time Scheduling issues for Multimedia Applications*," *PhD thesis*, University of Messachusetts Amherst, September 1994.

[3] S. Pingali, D.Towsley, and J.F. Kurose, "A comparison of sender-initiated and receiver-initiated reliable multicast protocols," *Performance Evaluation Review*, Vol.22, pp.221-230, May 1994.

[4] M.Peyravian, "An improved selective repeat protocol and its performance in high-speed environments," *Computer Networks and ISDN Systems*, Vol.26, pp.1595-1605, 1994.

[5] Weijia Jia, Edgar Nett, "Verification of RMP:An efficient Relibale Multicast Protocol," *Proceedings of second International Symposium on Parallel Architecutres, Algorithms, and Networks(I-SPAN' 96)*. pp. 388-393, 1996.

[6] Shyh-Wei Luan, Virgil D. Gligor, "A Fault-Tolerant Protocol for Atomic Broadcast," *IEEE Transactions on Parallel and Distributed Systems*, Vol.1, No.3, pp.271-285, July 1990.

[7] 손지연, 원유재, 오수형, 박준석, "신뢰성 있는 다자간 전송 프로토콜에서의 그룹관리," 한국통신학회 *COMSW'97* pp.464-467, 속초, 1997.

[8] Brain Neil Levine, JJ Garcia-Luna-Aceves, "A Protocols," *MS thesis*, University of California, June 1996.

[9] Sneha K. Kasera, Jim Kurose, and Don Towsley, "Scalable Reliable Multicast Using Multiple Multicast Groups," *CMPSIC Technical Report TR*, Oct 1996.

[10] S. Armstrong, A.Freier, and K.Marzullo, *RFC1301*:Multicast Transport Protocol, 1992.

[11] Bormann, C., Ou, J., Gehrcke, H-C, Kerschat, T., and Seifert, N., "MTP-2:Towards Archieving the S.E.R.O. Properties for Multicast Transport," *ICCCN 94*, San Francisco, Sep. 1994.

[12] 원유재, 손지연, 오수형, 박준석, 임경식, 박치항, "멀티미디어 그룹통신을 위한 신뢰성 있는 멀티미디어 프로토콜의 설계 및 구현," 한국정보과학회 논문지(C) 4권 1호 게재예정, 1998년 2월.

[13] JiYeon Son, YooJae Won, JunSeok Park, and KyungShik Lim "ReMP:Reliable Multipeer Protocol for Multimedia Group Communications," the 16th IASTED International Conference on Applied Informatics, to be published, Feb. 23-25, 1998.

[14] 손지연, 오수형, 원유재, 박준석, 임경식, "신뢰성 있는 다자간 전송 프로토콜의 구현," 한국정보처리학회 제10회 산·학·연 멀티미디어 산업기술 학술대회, pp.293-297, 11월 1997.

손 지 연

1991년 숙명여자대학교 전산학과 졸업(이학사)

1991년~현재 한국전자통신연구원 연구원

관심분야: 고속/멀티미디어 그룹통신, Protocol engineering, Parallel and distributed systems

원 유 재

1985년 충남대학교 계산통계학과 졸업(이학사)

1987년 충남대학교 대학원 계산통계학과 졸업(전산학 석사) 충남대학교 대학원 컴퓨터과학과 재학

1987년~현재 한국전자통신연구원 선임연구원

관심분야: protocol engineering, mobile computing., multimedia communication, parallel and distributed systems

오 수 형

1988년 서울대학교 천문학과 학사

1991년 한국과학기술원 전산학과 석사

1991년~1997년 한국전자통신연구원 연구원

1998년~현재 해동정보통신 과장

관심분야: 컴퓨터통신, 컴파일러, 컴퓨터 언어

임 경 식

1982년 경북대학교 전자공학과 전산 전공(공학사)

1985년 한국과학기술원 전산학과 졸업(전산학 석사)

1994년 University of Florida 전산학과 졸업(전산학 박사)

현재 한국전자통신연구원 컴퓨터통신연구실장/책임연구원

관심분야: mobile computing, wireless networks, high-spped communications networks, and parallel and distributed systems

황 승 구

1979년 서울대학교 전기공학과 졸업(학사)

1981년 서울대학교 전기공학과 졸업(석사)

1986년 University of Florida 전기공학과(박사)

1982년 7월~현재 한국전자통신 연구원 멀티미디어연구부 부장

1994년~1995년 미국 SRI International(International Fellow)

관심분야: 멀티미디어 시스템, HCI, 로보틱스