이중 비실시간 연결 구조 상에서의 실시간 RPC 스케쥴링 기법

(An RPC Scheduling Scheme over Dual Non-Real-Time Connection Structure)

이 정 훈 † 박 재 우 †† 천 원 기 ††
(Junghoon Lee) (Jaewoo Park)(Wonkee Chun)

요 약 본 논문은 이중의 비실시간 네트워크 상에서의 실시간 원격 프로시쥬어 호출(RPC)을 위한 메시지 스케쥴링 기법을 제안하고 평가한다. 제안된 기법은 시간제약 조건을 고려한 메시지 스케쥴에 의해 RPC 트랜잭션의 종료시한 만족도를 향상시키기 위하여 각 네트워크에 메시지 분배 비율을 달리함으로써 부하를 차별화하고 종료시한이 촉박한 메시지는 낮은 부하의 네트워크를 통해 전송한다. 생성간격 시간, 서버 실행시간, 여유시간 등과 같은 네트워크 인자를 기반으로 SMPL에 의해 수행된 모의실험 결과는 적절한 분할 기준치를 설정했을 때 제안된 기법이 실시간 RPC의 성능을 향상시킬 수 있으며 분할 기준치는 통계적 모델에 의해 효율적으로 찾아질 수 있음을 보인다.

Abstract This paper proposes and analyzes the performance of a message scheduling scheme for real-time RPC(Remote Procedure Call) over dual non-real-time networks. The proposed scheme partitions network load according to the slackness of RPC transactions. By transmitting more urgent messages via the lower load network, we can expect the enhancement of deadline meet ratio of RPC transactions. We measure the performance of our scheme by simulation using SMPL on the various RPC parameters such as interarrival time, service time and maximum slack ratio. The result indicates that our scheme improves the real-time performance. Furthermore, a particular statistical model is selected to determine the distribution ratio efficiently.

1. 서 론

실시간 시스템(real-time system)은 그 정확성이 결과의 정확함과 아울러 결과가 산출되는 시간에 의해 결정되는 시스템으로 정의되며 시간제약 조건(time constraint)에 대한 만족도 요구 수준에 따라 경성 실시간 시스템(hard real-time system)과 연성 실시간 시스템(soft real-time system)으로 구분되는데 시스템의 규모가 확장되고 다양한 실시간 응용이 출현함에 따라 분산 시스템(distributed system) 구조를 따르고 있다[1]. 자동제어 시스템, 우주선 항해 시스템 등과 같은 경성 실시간 시스템의 작업들은 반드시 종료시한(deadline) 이내에 수행이 완료되어야 하며 그렇지 못한 경우는 시스템에 치명적인 영향을 줄 수 있다. 따라서 경성 실시간 시스템은 고비용 특수 하드웨어 요소에 기반한 복잡한 시스템 설계와 아울러 주어진 태스크 집합에 대한 오프라인 분석 및 스케쥴을 수행하여 작업들의 시간제약 조건을 만족시키도록 한다. 반면 실시간 트랜잭션 시스템[2] 혹은 멀티미디어 표현 시스템 등과 같은 연성 실시간 시스템은 일부 작업의 종료시한 내 수행 실패를 허용할 수 있으며 이미 주어진 자원들을 효율적으로 이용하여 가능한 한 많은 작업들이 종료시한 내에 수행할 수 있도록 스케쥴한다[3].

TINA(Telecommunication Information Network Architecture)와 같이 거대한 규모를 갖는 정보망이 실

· 본 연구는 과학기술부의 원자력 연구 프로그램의 지원에 의해 수행되었음
† 정 회 원 : 제주대학교 전산통계학과 교수
jhlee@venus1.cheju.ac.kr
†† 종신회원 : 제주대학교 에너지공학과 교수
jwpark@cheju.cheju.ac.kr
wgchun@cheju.cheju.ac.kr
논문접수 : 2000년 5월 9일
심사완료 : 2000년 11월 7일

시간 서비스를 필요로 하게 되고 정보망을 이용한 다양한 실시간 응용이 등장하게 되었으며 이러한 기반구조는 응용의 시간제약 요구나 서비스의 질(Quality of Service) 요구에 따라 응용의 수행에 필요한 자원을 할당할 수 있다[4]. 자원의 할당에 의해 일정 수준의 종료시한 만족도를 보장하게 되는데 보장 과정은 네트워크나 운영체제 등 하부 플랫폼의 특성에 따라 최선 노력(best-effort) 수준에서부터 결정적인(deterministic) 수준까지 제공될 수 있다. 시스템이 제공할 수 있는 보장 수준이 하부 플랫폼의 기능에 따라 결정이 되기 때문에 경성 실시간 응용은 경성 실시간 보장을 제공할 수 있는 플랫폼에서만 설치되어 수행될 수 있는 반면 연성 실시간 응용은 충분한 종료시한 만족도(deadline meet ratio)를 지원할 수 있는 비실시간 요소들로 구성된 플랫폼 상에서 수행이 가능하다. 이러한 연성 실시간 응용을 지원하기 위해서 통신망은 운영체제의 프로세스 스케쥴링 정책이나 하부 네트워크의 프로토콜 등의 특성에 기반하여 효율적으로 작업을 스케쥴하여야 한다.

분산 실시간 시스템에서 프로세스들은 서로 다른 노드에 위치하여 시스템이 제공하는 특정한 상호작용에 의해 서로 협력하여 주어진 작업을 수행하게 되는데 이러한 상호작용은 메시지의 교환을 수반한다[5]. 작업이 종료시한 이내에 완수되기 위해서는 이들이 교환하는 메시지 또한 부여된 종료시한 이내에 전송이 완료되어야 한다는 시간제약 조건을 갖게 된다[6]. 각 프로세스는 메시지 교환에 있어서 원격 프로시저 호출(RPC: Remote Procedure Call)이나 분산 객체 연산(Distributed Object Operation) 등 다양한 프리미티브를 이용할 수 있는데 RPC는 분산 시스템에서 서비스 요청과 수행에 있어서 가장 기본적인 통신 프리미티브로서 대부분의 RPC 구현은 지역 환경에서의 프로시저 호출과 동일한 시맨틱을 제공하도록 설계되었다[7]. RPC와 같은 통신 프리미티브들의 시간제약 조건을 만족시키기 위해서는 우선적으로 시간제약 조건을 표현할 수 있어야 하며 시스템은 이에 의해 프리미티브들이 생성하는 메시지들을 응용의 요구에 맞추어 스케쥴하여야 한다.

RR(Request-Reply) 시맨틱을 갖는 전형적인 RPC 트랜잭션은 클라이언트와 서버 사이의 호출 및 결과 전달 등 메시지의 교환을 필요로 하며 클라이언트로부터의 프로시저 호출은 원격 서버에서 해당 연산을 수행시킨다[8]. 다른 실시간 서비스와 같이 실시간 RPC의 정확성은 결과를 산출해내는 시간뿐만 아니라 계산 결과의 정확성에 의존하여 호출·프로시저 수행·결과 전달 등 일련의 과정이 종료시한 내에 수행되어야만 결과가 유용하게 된다[9]. RPC의 시간제약 조건을 만족시키기 위해서는 RPC에서 파생된 메시지들과 서버에서의 수행이 시간제약 조건을 고려하여 스케쥴되어야 하는데 서버에서의 수행은 서버 측의 운영체제에서 제공하는 스케쥴링 기법을 따른다. 반면 메시지의 스케쥴링에 있어서는 클라이언트와 서버가 서로 다른 노드에 위치하기 때문에 야기되는 임의의 네트워크 지연시간을 우선적으로 고려하여야 하며 이는 하부 네트워크 구조, 즉 프로토콜이나 매체 접근 제어 기능과 매우 밀접한 관계가 있다. 연성 실시간 시스템을 구축함에 있어서 실시간 네트워크 등과 같은 특수 하드웨어를 사용하는 것이 항상 가능하지는 않을 뿐 아니라 이와 같은 실시간 하드웨어 요소는 매우 고가이며 손쉽게 이용될 수 없다. 더욱이 이미 구축되어 있는 정보망 위에 실시간 응용을 탑재하는 경우에는 하부의 플랫폼이 실시간 하드웨어로 구성되어 있지 않을 가능성이 높다. 예를 들어 이더넷이나 토큰 링과 같은 기존의 표준 네트워크와 이에 대응되는 드라이버들은 메시지 전송의 예측가능성(predictability)를 제공하지 못하고 메시지의 전송 순서를 결정하는데 있어서 시간제약 조건을 고려하지 않은 채 메시지를 단순하게 FCFS(First Come First Service) 순서로 전송하는 경향이 있다.

본 논문은 연성 실시간 시스템을 구축함에 있어서 이중화된 비실시간 네트워크를 이용하여 RPC 트랜잭션의 종료시한 만족도를 개선하는 효율적인 스케쥴링 기법을 제안하고 그 성능을 평가한다. 고려될 수 있는 이중화된 비실시간 네트워크 구조는 다음과 같다. 첫째, 모든 노드가 물리적으로 이중 네트워크에 연결되어 각 노드는 하나의 네트워크를 선택하여 메시지를 전송한다. 둘째, 광역 정보망 상에서 RPC를 교환하는 노드들에 대해 두 개의 그룹 연결을 설정하고 Controlled Load Service를 이용하여 각 연결에 개입된 네트워크 요소들의 자원을 할당함으로써 각 연결에게 일정량의 대역폭을 할당한다[10]. 이때 할당된 대역폭은 연결에 참여한 노드들이 공유한다. 마지막으로 하부 통신 구조가 ATM (Asynchronous Transfer Mode) 네트워크라면 ATM LAN Emulation 기능을 이용하여 RPC를 교환하는 노드들간에 이중 다자간(multi-party) ATM 연결을 설정하고 역시 일정한 대역폭을 할당한다[11]. 이와 같은 이중화된 연결 구조는 특수 하드웨어를 사용하지 않고 연성 실시간 응용을 위한 하부 통신구조로 고려될 수 있다.

이러한 비실시간 이중 연결 구조에서 RPC 메시지들

의 종료시한 만족도를 개선하기 위해서는 메시지의 종료시한을 고려한 스케줄링이 필요하여 종료시한이 촉박한 RPC 메시지는 종료시한에 여유가 있는 메시지 보다 신속히 전송이 완료되어야 한다. 그러나 하부 네트워크가 단순히 FCFS 순으로 전송한다면 이와 같은 우선순위를 부여할 수 없다. 네트워크의 부하가 낮을수록 메시지의 전송 시간은 단축되므로 이중화된 네트워크 구조 상에서 두 네트워크의 부하를 다르게 유지하고 낮은 부하의 네트워크를 통해 촉박한 시간제약 조건을 갖는 RPC 메시지를 전송한다면 종료시한에 따른 우선순위를 부여할 수 있게 되어 종료시한 만족도를 개선할 수 있다. 본 논문의 구성은 다음과 같다. 1장에서 문제를 제기한 후 2장에서는 기존의 실시간 RPC에 관련된 연구에 대해 소개한다. 이어 3장에서는 네트워크 모델을 기술하고 본 논문에서 제안하는 스케줄링 기법에 대해 상세하게 설명한다. 4장에서는 모의실험 결과를 보인 후 마지막으로 5장에서 결론을 도출한다.

2. 관련 연구

실시간 RPC에 관련된 연구로서 M. Dao 등은 비신뢰적인, 즉 메시지의 전송 시간이 예측가능하지 않은 네트워크 상에서 실시간 RPC 성능을 개선하기 위해 서버로 하여금 요청의 수행 전에 현재 대기중인 작업의 스케줄을 검사하여 해당 RPC의 종료시한 만족 가능성이 있는지 검사하고 가능성이 있을 때에만 요청을 수행하도록 함으로써 서버의 불필요한 수행시간을 감소시키는 기법을 제안하였다[9]. 종료시한 만족 가능성이 낮다고 판단되면 클라이언트에게 조기에 이를 통지하도록 하는데 이러한 사전 승인 과정이 실질적으로 RPC 종료시한 만족도를 증가시키지는 않지만 클라이언트로 하여금 시간 오류를 회복하기 위한 시간을 더 많이 갖게 한다. 이 방식에서 네트워크에서의 지연시간이 예측가능하지 않으므로 서버에서의 불필요한 연산이 완전히 배제되지는 않으며 네트워크 스케줄과 서버 스케줄과의 연계성이 직접적으로 고려되지는 않았다.

J. Lee는 실시간 RPC를 지원하기 위해 관련 메시지들을 효율적으로 스케줄하는 통신 시스템을 설계했는데, 종료시한과 유형, 즉 요청인지 응답인지에 따라 RPC 관련 메시지를 동적으로 스케줄한다[12]. 응답 메시지 전송에서의 종료시한 초과는 이전까지의 수행 과정을 무의미하게 하므로 응답 메시지에 우선순위를 주기 위하여 요청 메시지는 클라이언트가 여분의 대역폭이 있을 경우에만 전송하도록 하였다. 더욱이 서버는 응답 메시지를 보낼 수 있는 네트워크 시간이 확보 가능한 경우에만 수신된 요청을 수행하여 서버 스케줄과 통신 스케줄을 결합함으로써 서버 실행시간의 낭비를 완전히 제거하는 것이 가능하다. 그러나, 제안된 기법은 TDMA 구현 이더넷(Time Division Multiple Access-implemented Ethernet)과 같이 메시지 전송을 위한 네트워크 시간이 동적으로 할당 가능한 특수한 네트워크에만 적용이 가능하며 표준적인 비실시간 네트워크 환경에는 적용이 불가능하다.

B. Kao는 이중의 비실시간 네트워크상의 메시지 여유시간(slack)에 따라 부하를 분배하는 메시지 스케줄링 정책을 제안하였다[13]. 이 방식은 연성 실시간 메시지의 종료시한 만족도의 개선에 초점을 두었으나 요청-응답 RPC 시맨틱을 고려하지 않을 뿐 아니라 각 네트워크는 비실시간 네트워크에서 적용이 불가능한 EDF (Earliest Deadline First) 정책에 따라 메시지를 스케줄하도록 가정하였다.

실시간 CORBA(Common Object Request Broker Architecture)는 동적 CORBA 환경에서 분산 객체들간의 상호 작용에 있어서 종단간(end-to-end) 시간제약 조건을 지원하도록 설계되었다[14]. 동적 CORBA 시스템은 CORBA 시스템에서 전역적 우선순위 할당을 기반으로 한 시간제약 조건을 만족시키도록 하는데 최선 노력(best-effort) 방식을 따르고 있으며 역시 하부 네트워크의 스케줄링에 의한 성능개선 보다는 네트워크를 통한 메시지의 전송시간이 바운드되어 있다는 가정하여 이 시간을 기반으로 미들웨어(middleware) 상에서 실시간 응용을 지원하는데 중점을 두고 있다.

3. 스케줄링 기법

3.1 네트워크 모델

전형적인 실시간 RPC 트랜잭션은 클라이언트로부터 서버로 호출의 전달, 서버에서의 수행, 서버로부터 클라이언트로 결과의 전달 등 순차적인 과정으로 구성된다. 클라이언트의 호출 과정은 시간제약 조건, 목적지 서버 주소 등과 아울러 수행을 원하는 프로시저에 대한 정보, 이에 관련된 입력 인자 등을 하부 네트워크를 통해 서버로 전송한다. 요청을 받으면, 서버는 메시지에 명시된 입력 인자를 기반으로 요청된 프로시저를 수행하는데 동시에 여러 개의 요청이 도착한 경우 요청들은 서버 큐에 삽입되어 추후에 수행할 수 있도록 한다. 서버 큐의 대기 정책은 서버가 수행되는 운영체제 의해 결정되며 이에 가능한 정책으로는 FCFS, EDF(Earliest Deadline First) 등이 있다. 요청에 대한 수행이 완료되면 서버는 서비스를 요구한 클라이언트에게 계산 결과

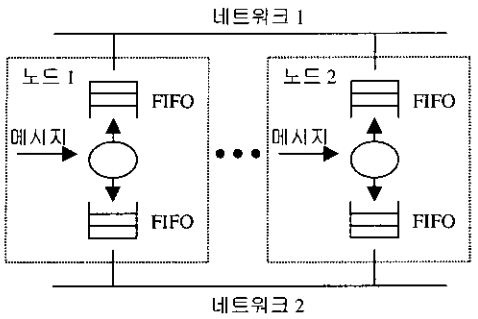

그림 1 네트워크 구조

를 전송하는데 이러한 RPC 트랜잭션 과정이 모두 종료 시한 내에 완료되어야 RPC의 시간제약 조건을 만족시킬 수 있다.

그림 1은 본 논문에서 대상으로 하고 있는 네트워크 모델을 보이고 있는데 각 노드는 두 개의 네트워크에 연결되거나 두 개의 그룹 연결에 소속되어 있다. 네트워크는 실시간 네트워크가 아닌 일반적인 표준 네트워크로서 이더네트와 같이 FCFS 순에 따라 메시지를 전송한다. 또 ATM 연결인 경우 연결에 개입된 ATM 교환기들이 단순히 연결 내에서 FCFS 방식으로 셀을 교환한다. 한 노드에서 메시지가 생성되면 두 네트워크 큐 중에 한 곳에 삽입되며 삽입된 이후에는 메시지를 현재 속한 큐에서 다른 큐로 이동할 수 없다. 큐의 선두에 위치하게 되면 메시지는 네트워크의 MAC(Medium Access Control) 계층 프로토콜에 따라 전송되는데 예로는 이더넷 상의 CSMA(Carrier Sense Multiple Access)/CD(Collision Detect)가 있다. 일반적으로 전송 지연시간은 기하학적인 분포를 따르는데 P를 네트워크 부하라 할 때 지연시간이 k일 확률은 식 (1)과 같이 표현되어 평균적인 전송시간은 네트워크 부하에 의해 결정된다[6].

$$\Pr(k) = P \cdot (1-P)^{k-1} \quad (1)$$

메시지 전송에 있어서 지연시간은 네트워크 부하가 주어질 때 평균 시간에 대한 분석은 가능하지만 각 전송시간에 대한 정확한 사전 예측은 불가능할 뿐 아니라 경성 실시간 시스템에서 보장을 위해 사용하는 최악의 지연시간도 계산할 수 없다. 따라서 RPC 트랜잭션의 종료시한 만족도를 향상시키도록 메시지를 스케줄하는 과정에 있어서 각 메시지들을 네트워크에 분배하는 정책이 주관심사가 되며 이 분배 정책이 실시간 성능에 가장 큰 영향을 준다.

3.2 스케줄 기법

앞에서 언급된 바와 같이 하나의 RPC 트랜잭션은 하나 이상의 노드가 개입된 일련의 수행 과정을 포함하므로 트랜잭션 수행 중에 종료시한을 초과할 수 있는데 이런 경우 트랜잭션의 남은 과정을 수행하는 것은 바람직하지 못하다. 이미 종료시한을 초과한 트랜잭션의 수행은 공유된 네트워크나 서버의 CPU 시간을 소비하기 때문에 종료시한을 만족시킬 가능성이 있는 다른 트랜잭션의 수행시간을 연장하여 전체적인 종료시한 만족도를 감소시킨다. 따라서 이러한 트랜잭션의 수행을 기각하여야 하는데 기각 과정은 네트워크 큐나 서버의 큐에서 일어날 수 있다. 서버의 큐에서 요청이 디스패치될 때 서버는 요청의 종료시한과 수행시간을 비교하여 초과 여부를 판단할 수 있으며 네트워크 큐에서는 MAC 프로토콜을 수행하기 전에 이미 RPC의 종료시한을 초과했는지 판단한다[15]. 일반적인 네트워크 인터페이스에서 비록 MAC 프로시저가 수행되기 이전에 메시지 전송을 기각할 수는 있으나 MAC 프로시저가 시작하면 도중 기각은 불가능하다. 따라서 매체 접근 과정이 길어져서 종료시한이 초과되는 경우 네트워크 시간의 낭비가 불가피하게 발생한다.

각 노드에 있어서 요청 혹은 결과 메시지가 생성되었을 때, 두 네트워크 중 하나에 의해 분배되어 전송되어야 하는데 분배하는 기법으로서 균등 분할(even partition) 방식과 여유시간 분할(slack partition or chop partition) 방식을 고려할 수 있다. 균등 분할 방식은 메시지를 두 네트워크에 똑같은 확률로 할당하여 충분한 네트워크 부하에서 메시지의 전송시간을 반으로 감소시킨다. 반면 여유시간 분할 방식은 각 RPC 여유시간에 따라 분배하는데 여유시간은 일반적으로 작업의 종료시한과 실행시간 사이의 차이로서 정의되며 작업의 시간제약 조건을 만족시키기 위해서는 늦어도 여유 시간 이내에 작업의 수행이 시작되어야 한다. 균등 분할 기법은 평균적으로 각 네트워크 큐의 평균 길이와 전송시간을 동일하게 하여 두 네트워크의 RPC 부하를 같게 하므로 균등 분할 방식은 모든 RPC 트랜잭션의 여유시간이 같을 때에 최선책이 된다. 그러나 각 RPC 메시지는 다른 여유시간을 갖기 때문에 촉박한 종료시한을 갖는 RPC가 생성한 메시지는 신속히 전송되어야 하는 반면 여유시간이 충분한 RPC에서 생성된 메시지는 전송시간이 여유시간 범위 내에서 연장되어도 종료시한을 만족시킬 수 있다.

여유시간 분할 방식은 각 RPC 메시지를 트랜잭션의 여유시간에 따라 네트워크에 분배하는데 RPC의 여유시간을 계산하는데 필요한 수행시간은 서버에서의 수행시

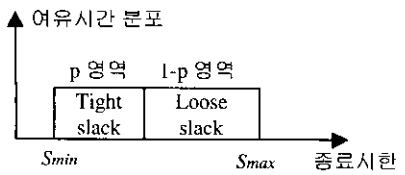

그림 2 여유시간 분할

간과 네트워크 수행 시간의 합으로 구성된다. 서버에서의 수행시간은 사전에 예측이 가능하거나 최소한 최악의 수행시간에 대해 계산할 수 있는데 반해 네트워크에서의 수행시간은 MAC 과정 때문에 정확한 예측이 불가능하므로 메시지의 길이에 따른 순수한 전송시간으로 설정한다. 결국 같은 RPC에 속한 요청과 응답 메시지는 같은 네트워크를 통해 전송된다. 여유시간 분할 방식은 RPC 트랜잭션들의 여유시간 분포가 사전에 알려져 있다는 가정 하에 한 네트워크의 부하를 낮게 유지하여 여유시간이 촉박한 메시지들을 전송하고 또다른 네트워크에는 종료시한 만족에 있어서 비교적 여유가 있는 메시지를 전송한다. 이를 위해 그림 2에서 보는 바와 같이 분할 기준치(chop value) p를 기반으로 네트워크를 선택하는데 여유 시간이 B. Gao 등의 연구에서처럼 S_{min}에서 S_{max}까지 균일하게 분포한다고 가정한다면 여유시간 분포상 0부터 p 범위에 해당하는 메시지들은 네트워크 1을 통해 전송하고 기타 $(1-p)$에 해당하는 메시지들은 네트워크 2를 통해 전송하도록 분배한다[13]. 이때 p는 0에서부터 0.5 사이의 값을 가질 수 있으며 만약 p 값이 0이라면 오직 하나의 네트워크를 통해 전송하는 경우와 동일하고 p가 0.5라면 균등 분할 방식과 같게 된다. 이외의 값에서는 두 네트워크의 부하가 다르게 유지되며 이 분할 기준치는 RPC 부하와 같은 네트워크 인자에 따라 설정된다.

3.3 분할 기준치의 결정

여유시간 분할 방식은 분할 기준치의 값에 의해 그 성능이 좌우된다. 만약 시스템의 인자들이 변하지 않는 정적인 특성을 갖고 있다면 사전에 다양한 모의실험이나 환경 실험을 수행하여 인자들과 최적의 분할 기준치에 대한 관계를 통계적인 모델에 의해 구한 후 시스템 운영 중에는 이 모델에 의해 분할 기준치를 설정할 수 있다. 반면 시스템의 인자들이 동적으로 변화한다면 분할 기준치를 이에 따라 같이 변경하여야 하는데 이러한 동적 적응을 위해서는 네트워크 상에 조정자(coordinator) 노드가 필요하다. 조정자 노드는 네트워크 상의 다른 노드들로부터 주기적으로 RPC 부하와 같은 인자들과 현재의 종료시한 만족도와 같은 상태 정보들을 수집하여 새로운 분할 기준치를 설정하고 이를 방송에 의해 다른 노드들에게 알려 분할 기준치를 갱신하도록 한다.

4. 모의실험 결과와 분할 기준치 추정

본 절에서는 SMPL을 사용한 모의실험을 통하여 제안된 RPC 스케쥴링 기법의 성능을 측정한 결과를 제시하고 분석한다[16]. 모의실험에 있어서 주요한 가정은 다음과 같다. 시스템에서 모든 RPC 트랜잭션들은 동일한 중요도를 가지며 종료시한 만족도가 가장 주요한 시스템 성능 요소라고 가정한다. 만약 각각의 RPC가 서로 다른 중요도를 가진다면 성능 목표는 각 RPC 트랜잭션의 종료시한 만족에 의해 얻어지는 결과의 가중치 값을 극대화시키도록 조정되어야 한다. 또한 각 RPC들은 다른 RPC 트랜잭션과 독립적으로 수행되며 또한 서버는 EDF 정책에 따라 수신된 요청을 스케쥴하고 수행한다고 가정한다. 모의실험은 이중의 10 Mbps 이더넷 모델에 기반을 두고 있으며 모의실험의 단순화를 위하여 모든 메시지는 최대 이더넷 프레임 길이를 초과하지 않도록 함으로써 추가적인 분할(fragmentation)이나 재조립(reassembly) 과정이 없이 하나의 MAC 프레임을 통하여 전송될 수 있도록 하였다. 요청 메시지의 평균 길이는 500 바이트, 반면 응답 메시지의 평균 길이는 1000 바이트로 설정하여 지수 분포를 따르도록 하였다.

표 1 인자들의 기본값

RPC 생성간격 시간	0.0008
분할 기준치	0.45
최소 여유시간 비율	2
최대 여유시간 비율	32
서비스 시간	0.0001

제안된 메시지 스케쥴링 기법에 있어서 RPC의 종료시한 만족 비율에 영향을 미치는 네트워크 인자로서 RPC의 생성간격 시간(interarrival time), 서비스 시간, 분할 기준치, 여유시간 비율 등이 있으며 본 모의실험은 이와 같은 각 인자에 따른 종료시한 만족도를 측정한다. 모의실험은 우선적으로 각 인자들이 종료시한 만족도에 주는 영향을 측정하기 위해 표1에서 보이는 고정된 기본 값을 기반으로 각 실험마다 하나의 인자를 변화시켰다. 표 1에서, RPC 생성간격 시간과 서비스 시간은 지수 분포를 따르며 시간 단위는 1초를 기준으로 하였다.

또 기본 값에 의해 여유 시간은 균일하게 평균 서비스 시간의 2 배에서 32 배까지 분포한다. 네트워크 상에는 RPC 트래픽만이 존재한다고 가정하였으나 다른 트래픽이 존재하는 경우에도 트래픽이 네트워크에 진입하는 비율을 조정함으로써 각 네트워크의 부하를 다르게 유지시킬 수 있다.

첫 번째 실험에서는 RPC 부하에 해당하는 RPC 생성 간격 시간을 변화시키며 종료시한 만족도를 측정하였는데 그림 3에서 보여지는 바와 같이 간격시간이 증가함에 따라, 즉 부하가 적정하게 부여될 때 여유시간 분할 기법의 종료시한 만족도가 균등 분할 방식에 비해 최대 7%까지 증가한다. 그러나 RPC 부하가 증가하면 균등 분할 방식이 평균 네트워크 큐의 길이를 줄이므로 더 나은 성능을 보여준다. 이와 같이 부하가 높은 경우 0.45가 아닌 다른 분할 기준치 값을 사용한다면 여유시간 분할 방식이 그 격차를 줄이거나 더 좋은 성능을 보일 수 있다. 또 표 1과 다른 기본값을 사용하는 경우도 그림 3과 유사한 성능곡선 패턴을 보인다.

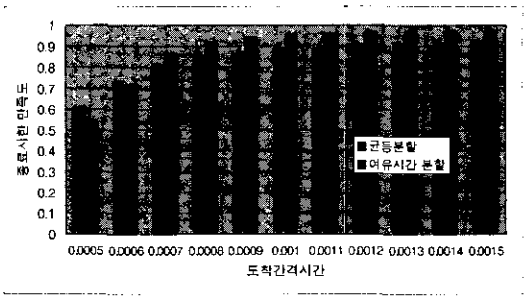

그림 3 RPC 도착간격 시간 대 종료시한 만족도

그림 4는 최대 여유시간에 따른 종료시한 만족도를 측정하기 위해 최대 여유시간을 서비스 시간의 10 배부터 100 배까지 변화시켰다. 균등 분할과 여유시간 분할 기법 모두 최대 여유시간이 증가함에 따라 더 나은 성능을 보이고 있는데 실험에서와 같이 분할 기준치의 값이 적절하게 선택된다면 여유시간 분할 방식은 주어진 인자의 모든 범위에 걸쳐 균등 분할 방식보다 좋은 성능을 보일 뿐 아니라 최대 여유시간이 증가함에 따라 성능향상도도 함께 증가한다. 그림 5는 주어진 네트워크 인자 집합에 대해 분할 기준치의 영향을 보여주는데 실험에서 주어진 네트워크 인자에 있어서는 0.45 부근에서 성능 격차를 최대화하고 있으며 성능 격차를 최대화하는 분할 기준치 값은 RPC 부하의 증가에 따라 0.5에 근접하는 경향을 보인다. 그림 6은 서버 실행 시간에 따

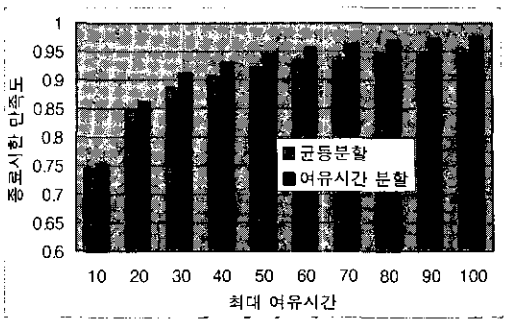

그림 4 최대 여유시간에 따른 종료시한 만족도

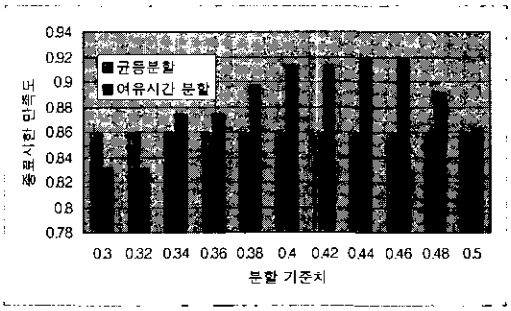

그림 5 분할 기준치에 따른 종료시한 만족도

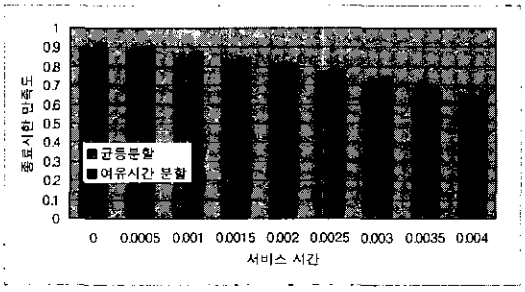

그림 6 서비스 시간 대 종료시한 만족도

른 종료시한 만족도를 나타내고 있는데 이 실험에서는 서버의 수행 시간을 0부터 0.004까지 변화시켰다. 서비스 시간이 증가함에 따라 메시지 전송을 위한 시간이 적게 남아있게 되며 이와 같은 상황은 RPC 부하가 높은 상황과 동일하다. 종료시한 만족도가 응용이 요구하는 수준 이하로 떨어진다면 네트워크는 더욱 높은 대역폭을 갖는 네트워크로 교체되거나 더 많은 대역폭을 할당받아야 한다.

제안된 스케쥴링 기법에 있어서 주어진 네트워크 인자에 대해 최적의 분할 기준치를 설정하는 것이 시스템의 실시간 성능에 가장 중요한 과정인데 이는 실험에

의한 통계 분석에 의해 결정될 수 있다. 시스템에서 평균 서비스 시간과 최대 여유시간은 정적인 인자인데 반해 도착간격 시간은 동적으로 변화하는 특성을 갖는다. 조정자 노드는 각 노드들로부터 현재 RPC 트랜잭션의 도착간격 시간에 대한 정보를 수집하여 이를 기반으로 분할 기준치를 결정하고 이를 다시 각 노드들에게 알린다. 식 (2)는 최대 여유시간 비율 50, 평균 서비스 시간 0.0005에서 생성간격 시간 λ에 따른 분할 기준치를 SAS를 이용하여 통계적으로 분석한 결과이다[17]. 분할 기준치를 정확하게 설정하기 위해서는 가능한 한 변수들의 개수를 줄이는 것이 필요한데 실시간 RPC의 특성상 서버측에서 수행시간을 예측할 수 있으면 서비스 시간과 메시지 크기는 사전에 알려져 있다고 볼 수 있으며 여유시간 비율은 각 호출의 특성에 따라 결정된다고 볼 수 있다. 따라서 동적인 인자는 생성간격시간으로 단순화되며 이 생성간격 시간의 모형을 결정하기 위해 일반적으로 알려진 여러 모형을 선택하고 이들의 모형 부합성은 Least Square 와 Least Median을 사용해 평가한 결과 e^x 모형이 가장 잘 부합하였다. 따라서 $p = a \cdot e^{b\lambda} + c$로 설정하고 Least Square 방식으로 a, b, c 상수의 값을 결정하였다.

$$p = 579.70318 \cdot e^{-\frac{\lambda}{2}} - 579.038061 \quad (2)$$

이와 아울러 그림 7은 실험에 의해 찾아진 최적의 분할 기준치와 식 (2)에 의해 추정된 값으로 분할 기준치를 설정하였을 때 생성간격 시간에 따른 종료시한 만족도를 보이고 있는데 성능의 차이는 최대 0.3 % 이내이다. 또 그림 5에서 보이고 있는 것처럼 만약 분할 기준치를 최적의 값과 동일하지는 않더라도 근사하게 설정했을 때 이와 유사한 종료시한 만족도를 보일 수 있으므로 통계적 분석에 의한 방식은 효율적으로 분할 기준치를 설정할 수 있다.

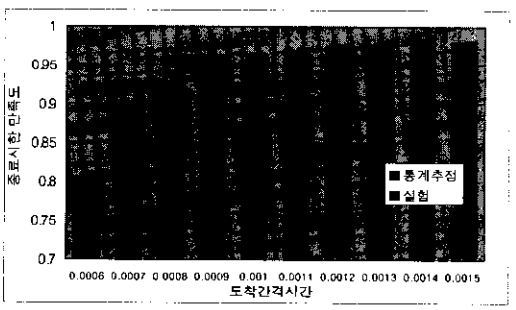

그림 7 분할 기준치의 추정과 실험

5. 결 론

본 논문은 이중 비실시간 연결 구조 상에서의 실시간 RPC 스케쥴링 기법을 제안하고 그 성능을 평가하였다. 제안된 기법은 보다 많은 실시간 RPC의 종료시한 만족을 목표로 하고 있으며 비실시간 네트워크가 메시지들의 시간제약 조건을 고려하지 않고 메시지들을 스케쥴하는 단점을 극복하기 위하여 두 네트워크의 부하를 다르게 유지하여 평균적인 전송시간을 다르게 한다. 이에 따라 연결 구조 상에서 두 개의 우선순위 수준이 존재하게 되며 결과적으로 RPC 트랜잭션의 여유시간에 따라 RPC가 파생시킨 메시지들이 우선순위를 부여받는다. 즉, 촉박한 종료시한을 갖는 메시지들은 부하가 낮은 네트워크를 이용해 전송이 되며 비교적 여유시간이 많은 메시지들은 부하가 높은 네트워크를 이용해 전송이 된다. 이때 네트워크 부하의 분할 기준치가 종료시한 만족도에 가장 큰 영향을 주게 되는데 이 분할 기준치는 시스템 내에서 RPC 생성간격 시간, 서버들의 평균 요청처리 시간 및 여유시간 비율 등 다양한 인자들에 의해 결정이 된다. 따라서 통계적인 분석에 의해 분할 기준치를 결정할 수 있도록 모델을 제시하였으며 이에 의해 최적에 가까운 분할 기준치 값에 의해 실시간 RPC의 성능 향상을 기할 수 있다. 최종적으로 본 논문에서 제시한 스케쥴링 기법은 이중화된 비실시간 구조를 이용해 저렴하게 연성 실시간 시스템을 구축할 수 있도록 함은 물론 광역 정보망으로 하여금 이중 연결 구조를 이용하여 효율적으로 연성 실시간 응용을 지원할 수 있도록 한다. 또 본 논문에서는 프로토콜 스택으로 RPC 계층을 전제하였으나 프로토콜 계층의 구조는 점차 경량화하고 있는 추세이다. 그러나 많은 하부의 네트워크는 아직도 비실시간 매체 접근 프로토콜을 사용하거나 실시간 전송을 보장하지는 못한다. 본 논문에서 제시된 방식은 이러한 비실시간 구조 상에서 프로토콜 스택과 관련없이 요청-응답 형태의 통신 특성을 갖는 응용들에 적용될 수 있다.

참 고 문 헌

[1] K. Arvind, Krithi Ramamritham and John A. Stankovic, "A local area network architecture for communication in distributed real-time systems," *Journal of Real-Time Systems*, Vol. 3, pp.115-147, May 1991.

[2] J. Kim, H. Shin, "Priority-driven concurrency control based on data conflict state in distributed

real-time databases," *Microprocessor and Microprogramming*, pp.491-498, 1993.
[3] K. Ramamritham, J. Stankovic, "Dynamic task scheduling in hard real-time systems, *IEEE Software*, pp.65-75, July 1984.
[4] W. Barr, T. Boyd, Y. Inoue, "TINA initiative," *IEEE Communication Magazine*, pp.70-76, March 1993.
[5] G. Coulouris, J. Dollimore, T. Kindberg, *Distributed Systems; Concepts and Design*, 2nd Ed., Addison-Wesley, pp.128-130, 1994.
[6] I. Lee, S. Davidson, "A performance analysis of timed synchronous communication primitives," *IEEE Trans. on Computers*, pp.1117-1131, September 1990.
[7] A. Birrel, B. Nelson, "Implementing remote procedure calls," *ACM Trans. on Computer Systems*, pp.39-59, 1984.
[8] A. Sinha, "Client-server computing," *Communication of ACM*, pp.77-98, 1992.
[9] Marina Dao and Kwei-Jay Lin. "Remote procedure call protocols for real-time systems," *Proceedings of IEEE Euromicro Workshop*, pp.216-223, 1991.
[10] J. Wroclawski, *Specification of the Controlled-Load Element Service*, RFC 2211, September 1997.
[11] P. Newman, "ATM local area networks," *IEEE Communication Magazine*, pp.86-98, March 1994.
[12] J. Lee, "Design of a communication system capable of supporting real-time RPC," *4-th Real-Time Application Workshop*, pp.99-102, 1996.
[13] B. Gao, H. Garcia-Molina, "Scheduling soft real-time jobs over dual non-real-time servers," *IEEE Trans. Parallel and Distributed Systems*, pp.56-68, Jan. 1996.
[14] D. Schmidt, D. Levine, S.Mungee, "The design and performance of real-time object request brokers," *Computer Communications*, pp.294-324, 1998.
[15] J. Lee, S. Park, "Design of a DAVIC residential network based on Ethernet," *Proc. RTCSA*, pp.223-228, Oct. 1997.
[16] M. H. MacDougall, *Simulating Computer Systems: Techniques and Tools*, MIT Press, 1987.
[17] 성내경, *SAS/STAT-회귀분석*, 자유 아카데미, 1991년
[18] S. Weisberg, *Applied Linear Regression*, 2nd Ed., John Wiley & Sons, 1985.

이 정 훈
1988년 서울대학교 컴퓨터공학과 공학사. 1990년 서울대학교 컴퓨터공학과 석사. 1996년 서울대학교 컴퓨터공학과 박사. 1990 ~ 1991, 1996 대우통신 근무. 1997년 ~ 현재 제주대학교 자연과학대학 전산통계학과(조교수).

박 재 우
1979년 서울대학교 원자력공학과 공학사. 1983년 펜실베니아 주립대 원자력 공학과 석사. 1995년 미주리대학교 원자력공학과 박사. 1979년 ~ 1981년 한국전력기술주식회사 근무. 1988년 ~ 현재 제주대학교 공과대학 에너지공학과(교수)

천 원 기
1981년 한양대학교 기계공학과 공학사. 1984년 유타대학교 기계공학과 석사. 1986년 유타대학교 기계공학과 박사. 1986년 ~1994년 한국에너지 기술연구소 선임연구원. 1994 ~ 현재 제주대학교 공과대학 에너지공학과(부교수).