FDDI 기반 실시간 데이타 수집 네트워크에서의 최선노력 오류제어 기법

(A Best-Effort Error Control Scheme on FDDI-based Real-Time Data Collection Networks)

이 정 훈 † 김 호 찬 ††
(Junghoon Lee) (Hochan Kim)

요 약 본 논문은 FDDI에 기반한 경성 실시간 네트워크에서 메시지의 종료시한을 고려하여 주기내 전송을 지원하는 오류제어 기법을 제안하고 그 성능을 분석 및 평가한다. 오류제어 과정에서 필수적인 재전송 요구와 오류발생 메시지의 재전송 과정이 일반적인 실시간 메시지의 전송에 영향을 주지 않도록 하기 위하여 재전송 요구는 FDDI 매체 접근 제어 프로토콜이 지원하는 비동기 대역폭을 이용하여 수행되며 재전송 과정은 대역폭 할당 알고리즘이 불가피하게 생성한 과할당된 대역폭을 이용한다. 또 재전송 요청 시점을 결정하기 위해 수신자는 자신이 수신한 토큰의 수를 계수한다. 분석 결과와 SMPL을 이용한 실험 결과는 제안된 방식이 실시간 통신을 위한 오류제어 기능으로서 네트워크 오류를 극복하여 메시지의 종료시한 만족도를 증가시킬 수 있으며 이중화된 네트워크에 비견할 만한 성능을 보임으로써 저비용으로 실시간 네트워크를 구축할 수 있음을 보인다.

Abstract This paper proposes and analyzes an error control scheme which tries to recover the transmission error within the deadline of a message on FDDI networks. The error control procedure does not interfere other normal message transmissions by delivering retransmission request via asynchronous traffic as well as by delivering retransmitted message via overallocated bandwidth which is inevitably produced by the bandwidth allocation scheme for hard real-time guarantee. The receiver counts the number of tokens which it meets, determines the completion of message transmission, and finally sends error report. The analysis results along with simulation performed via SMPL show that the proposed scheme is able to enhance the deadline meet ratio of messages by overcoming the network errors. Using the proposed error control scheme, the hard real-time network can be built at cost lower than, but performance comparable to the dual link network.

1. 서 론

실시간 시스템(real-time system)은 그 정확성이 결과의 정확함과 아울러 결과가 산출되는 시간에 의해 결정되는 시스템으로 정의되는데 실시간 시스템의 규모가 확장되고 다양한 실시간 응용이 출현함에 따라 분산 시스템(distributed system) 구조를 따르고 있다[1]. 이러한 분산 실시간 시스템에서 프로세스들은 메시지를 통해 서로 협력하여 작업을 수행하게 되며 작업이 종료시한(deadline) 이내에 완수되기 위해서는 이들이 교환하는 메시지 또한 부여된 종료시한 이내에 전송이 완료되어야 한다는 시간 제약조건(time constraint)을 갖게 된다. 메시지의 시간 제약조건을 만족시키기 위해서는 메시지들의 전송이 정확하게 스케쥴되어야 하는데 이는 네트워크 프로토콜 계층구조상 데이타 링크 계층의 기능에 속한다[2]. LAN(Local Area Network)과 같이 공유된 네트워크를 통해 모든 노드들이 연결되어 있는 다중접근 네트워크(multiple access network)에서 데이타 링크 계층의 기능은 분산된 노드들로부터의 네트워크 접근을 중재하여 주어진 싯점에 어느 노드가 네트워크를 사용할지를 결정하는 것이다[3]. 실시간 MAC

· 본 연구는 과학기술부의 원자력 연구 프로그램의 지원에 의해 수행되었음.
† 정 회 원 : 제주대학교 전산통계학과 교수
jhlee@venus1.cheju.ac.kr
†† 비 회 원 : 제주대학교 전기공학과 교수
hckim@cheju.cheju.ac.kr
논문접수 : 2000년 4월 14일
심사완료 : 2001년 5월 25일

(medium access control) 프로토콜의 한 예인 FDDI (Fiber Distributed Data Interface)는 링 형태의 토폴로지를 따라 토큰을 회전시키고 토큰을 소유하고 있는 노드만이 메시지를 전송하도록 한다[4]. FDDI 네트워크에서 실시간 메시지 스트림의 전송을 스케쥴하기 위해서는 오프라인 시에 주어진 메시지 스트림의 특성을 분석하여 각 노드 혹은 스트림이 토큰을 소유하였을 때 네트워크에 접근할 수 있는 시간, 즉 용량 벡터(capacity vector)를 결정하며 이에 따라 네트워크 운영시에 각 노드는 결정된 시간만큼 메시지를 전송한다[5]. 이 과정에서 메시지 스트림 분석과 대역폭 할당 과정은 새로운 연결이 설정될 때에도 수행될 수 있다.

FDDI와 같은 실시간 통신 프로토콜들이 메시지들의 시간 제약조건을 만족시킬 수는 있지만 메시지 전송 중의 오류는 불가피하게 발생하며 전송 오류는 메시지의 손실을 의미한다. 지역망 구조에서 전송 오류는 기본적인 매체 오류율, 외부 자극에 의한 일시적인 교란, 혹은 수신자의 버퍼용량 초과 등의 요인에 의해 발생하며 통신 기술의 발달로 인해 매체 오류율이 감소하고 있기는 하지만 완전히 제거되지는 못하고 있다[2]. 또 멀티미디어 응용의 도입은 메시지의 크기를 증가시켜 매체 오류율의 영향을 증폭시키고 있다. 프로토콜 계층 구조상 논리적 링크 제어(LLC: Logical Link Control) 계층이 오류를 탐지하고 보정하는 기능을 수행하는데 go-back-N, stop-and-wait, selective repeat 등 기존의 전형적인 오류제어 기능은 그 과정이 메시지 전송 시간을 연장하게 되어 메시지들이 시간 제약조건을 갖는 실시간 통신에는 적합하지 못하다[6]. 예를 들어 선택적 재전송(selective retransmission) 방식은 일반적인 데이타 통신에서 널리 사용되는 오류제어 기법으로서 메시지의 전송이 끝난 후 확인(acknowledgment) 혹은 재전송 요청(retransmission request) 메시지에 의해 수신자가 송신자에게 수신의 확인이나 재전송 요구를 하며 송신자는 이에 의해 메시지를 재전송한다. 이 기법은 전송중 오류가 발생한 메시지만 선택적으로 재전송하므로 대역폭을 효율적으로 이용할 수는 있지만 확인 혹은 재전송 요청 메시지의 전송과 이에 따르는 메시지의 재전송 과정이 메시지 전송 시간을 연장하여 메시지의 종료시한을 초과하게 할 수 있을 뿐 아니라 재전송되는 메시지가 일반적인, 즉 재전송되지 않는 실시간 메시지 전송을 지연할 수 있다. 또한 재전송 요청 메시지가 일반 메시지들과 함께 스케쥴되어야 하며 전송된 메시지들을 위해 버퍼가 할당되어야 하므로 실시간 시스템에서는 적용하기 어렵다. 결국 실시간 통신에서의 오류제어는 메시지들의 시간 제약조건을 고려하여 종료시한 이내 제어 가능한 오류의 수를 증가시켜야 한다.

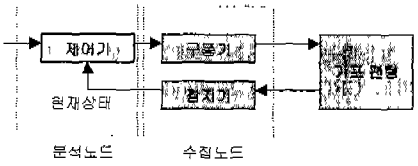
FDDI에 기반한 실시간 통신에서는 대역폭 할당 기법이 최악의 경우를 고려하여 실시간 메시지를 위한 용량 벡터를 결정하기 때문에 각 스트림은 여분의 대역폭을 갖게 되며, 실시간 스트림에 할당하고 남은 대역폭은 비실시간 메시지들이 이용하게 된다. 특정 실시간 메시지 스트림에 있어서 과할당된 대역폭을 이용한 재전송은 종료시한 이내에 수행이 가능할 뿐 아니라 자신에게 배타적으로 할당된 대역폭을 사용하므로 다른 메시지의 전송에 전혀 영향을 주지 않는다. 또 수신자가 송신자에게 전송하는 재전송의 요청은 대역폭을 사전에 확보하기 어려우므로 비실시간 대역폭을 통해 전송하여 수행한다면 다른 실시간 스트림의 전송에 전혀 영향을 주지 않는다[7]. 이와 같이 확장된 MAC 기능은 비록 메시지 전송 중에 발생하는 모든 오류에 대해 제어를 할 수 있는 것은 아니지만 종료시한을 고려하여 오류제어를 수행하고 상위 계층에게 보다 나은 메시지 전송율을 제공할 수 있다.

본 논문은 FDDI에 기반한 경성 실시간 통신을 위한 오류 제어 기법을 제안하고 평가함을 목적으로 하며 다음과 같이 구성된다. 2장에서는 연구배경과 본 논문에서 대상으로 하고 있는 응용에 대해 소개하며 3장에서는 관련연구로서 기존의 실시간 통신을 위한 오류제어 기법과 실시간 통신에 적용가능한 오류제어 기법을 소개한다. 이를 바탕으로 4장에서는 본 논문에서 제안하는 실시간 오류제어 기법의 기능과 동작을 자세히 기술하고 5장에서는 제안된 기법을 분석함과 아울러 모의실험 결과를 보이고 마지막으로 6장에서는 본 논문을 요약하고 결론과 추후 연구과제를 도출한다.

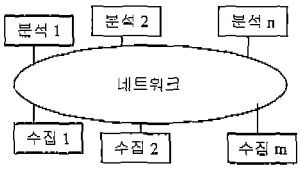
2. 연구배경

ET(Electric Tomography)와 같은 실험 환경은 다른 실시간 시스템에서의 제어 루프와 마찬가지로 그림 1(a)와 같이 데이타의 수집 노드와 분석 노드 사이의 관계가 폐쇄된 루프(closed loop)의 형태를 가지며 수집된 데이타가 분석 노드로 전송되어 실시간으로 분석되어 다음의 동작을 지시하여야 한다[8]. 그러나 다수의 수집 노드와 분석 노드가 존재할 때 각각 마다 폐쇄된 루프를 형성한다면 실험 환경을 구축하는데 있어서 그 비용이 증가하게 되며 각 루프 마다의 상호작용 인터페이스에 일관성을 유지할 수 없다. 따라서 이러한 구조를 그림 1(b)와 같이 하나의 개방된 루프 형태로 변경하여 범용 네트워크로 연결한다면 실험 환경의 구축 비용이

절감되고 새로운 수집 노드와 분석 노드를 설치하는데 효율성을 기할 수 있다. 이 과정에서 수집 노드 상에 데이타는 주기적으로 발생하며 이들은 일정 시간 이내에 분석 노드에 전달되어 분석이 수행되어야 하는 실시간 제약조건을 갖는다. 따라서 FDDI와 같이 실시간 통신을 지원할 수 있는 네트워크를 하부 통신구조로서 사용하여야 하며 각 노드 마다 메시지 스트림의 주기와 메시지 양에 기반하여 일정량의 대역폭을 할당받아야 한다. 한 네트워크에 여러 노드가 연결되므로 다양한 트래픽이 존재할 수 있으나 각 노드는 할당된 대역폭을 배타적으로 사용할 수 있기 때문에 비실시간 메시지의 간섭을 배제하여 실시간 메시지의 전송을 보장하게 된다.



(a) 폐쇄된 제어 루프



(b) 개방된 제어 루프

그림 1 데이타 수집 구조

각 샘플링 주기마다 수집된 데이타는 그 크기가 일정하며 하나의 메시지 혹은 데이타 블록을 형성하는데 이 메시지가 분석 노드에게 전달되는 과정에 있어서 그림 2와 같이 다수의 FDDI 프레임으로 분할되어 전송된다. 한 메시지에 속한 모든 프레임들은 한 주기 이내에 전송이 완료되어야 하며 메시지 전송과정에 있어서 한 비트의 오류는 전체 메시지의 손실로 파급되기 때문에[9] 데이타의 전송 오류는 수집된 데이타의 손실을 의미할 뿐 아니라 데이타 분석에 있어서의 정확도를 감소시키는 원인이 된다. 비록 네트워크 성능의 개선이 전송 오류율을 충분히 감소시킨다 하더라도 한 번 수집되어 전송되는 데이타의 양이 증가한다면, 즉 메시지의 크기가 증가한다면 일부 비트의 오류가 전체 메시지의 오류로 파급되므로 데이타의 손실되는 양도 따라서 증가한다.

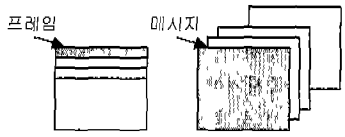


그림 2 수집 데이타 전송을 위한 메시지와 프레임

3. 관련연구

3.1 오류제어 기법

실시간 데이타 링크 계층에서는 두 종류의 오류제어 기법이 널리 사용되는데 이는 시간적 여분(temporal redundancy)과 공간적 여분(spatial redundancy)을 이용하는 방식이다[2]. 시간적 여분을 이용하는 방식은 메시지의 종료시한이 비교적 긴 분산 응용에 적합한데 이 방식은 물리적인 네트워크의 중복없이 동일한 네트워크 상에서 메시지를 중복 전송하여 오류율을 줄이거나 재전송 요청 혹은 타임아웃에 뒤따르는 재전송에 의해 오류를 제어한다. 재전송 요청이나 타임아웃을 이용한 방식은 재전송에 관련된 메시지들이 하부 네트워크 프로토콜에 맞추어 스케줄되어야 한다는 문제점을 갖고 있으며 중복 전송에 의한 방식은 대역폭의 낭비를 크게 하는 문제점을 갖고 있다. 시간적 여분 방식의 예로서 MARS 시스템을 들 수 있는데, 이 시스템에서는 그림 3에서 보는 바와 같이 오류 발생 여부에 관계없이 모든 메시지들이 이중으로 중복되어 전송된다[10].

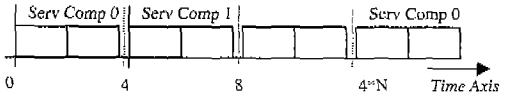


그림 3 MARS의 중복전송 방식

반면 공간적 여분을 사용한 방식에서는 중복된 메시지들이 서로 다른 채널이나 네트워크를 통해 전송되며, 메시지의 종료시한이 촉박하여 재전송이나 중복전송을 허용할 수 없을 경우에 적합하다[11]. 이 방식은 항상 메시지들이 다중으로 중복되어 전송되므로 메시지들에 의해 요구된 대역폭의 중복 배수 만큼의 대역폭을 필요로 하게 되어 상당한 대역폭의 낭비를 초래할 뿐 아니라 여러 경로로 다른 시간대에 수신되는 메시지들을 처리하여야 하므로 수신자의 기능이 복잡해진다.

이상의 기법들 이외에 오류 보정 코드에 의한 전향 오류제어 방식이 효율적으로 보이지만 정보의 낭비가 크게 되며 처리시간이 길어지게 된다. 그 예로 Hamming 코

드 방식에서는 메시지의 크기를 m 비트라 할 때 한 비트의 오류를 보정하기 위한 오버헤드 k는 $2^k \geq m+k+1$의 관계식을 만족시켜야 되는데 메시지의 크기가 커질수록 대역폭 낭비 요소인 k의 길이가 증가한다[12]. 이 경우에도 2 비트의 동시 오류는 탐지만 가능하며 보정은 할 수 없다. 오류 보정 코드는 오류가 발생하지 않는 경우에도 고정적으로 부가적인 정보가 추가되므로 대역폭의 낭비를 초래하며 송신자와 수신자 측에서 오류를 보정하기 위해 많은 데이타 연산을 수행하여야 한다. 이와 아울러 오류보정 코드는 메시지의 오류율을 감소시킬 수 있으며 오류제어 기능과 결합한다면 오류처리율을 개선할 수 있다.

또 임의의 하부 네트워크를 기반으로 트랜스포트 계층에서 오류를 제어하는 방식의 예로서 사전 재전송 방식은 오류의 발생 여부를 최종적으로 확인하기 위해 기다리지 않고 오류의 가능성이 있다고 판단될 때 미리 재전송을 수행하도록 한다[13]. 이는 오류 복구가 종료시한 이전에 이루어질 수 있는 시간적 여유를 갖도록 함을 목적으로 하고 있다. 그러나 이 방식은 오류의 가능성을 판단하기 위해 사전 재전송 시점을 결정하여야 한다는 추가적인 요구사항을 갖고 있으며 재전송이 다른 실시간 스트림의 전송을 지연할 수 있다는 문제점을 갖고 있다. 또 MAC 계층 보다는 트랜스포트 계층에서의 오류 제어를 대상으로 하고 있기 때문에 다양한 하부 MAC 구조에 적용될 수는 있으나 실시간 MAC의 기능을 이용하지 못한다. 기타 트랜스포트 계층의 오류제어 기법들이 실시간 통신에 일부 적용이 가능하지만 시간제약 조건을 직접적으로 고려하고 있지는 않다[14, 15].

3.2 FDDI 상에서의 실시간 통신

실시간 메시지들은 주기적으로 발생되며 이들은 주기 내에 전송이 완료되어야 한다. FDDI 네트워크에서 메시지들의 시간 제약조건을 만족시키기 위해서는 TTRT (Target Token Rotation Time)나 용량 벡터와 같은 네트워크 인자들이 정확하게 설정되어야 한다. TTRT는 토큰이 회전하는 기대치로서 각 노드들은 자신의 메시지 특성에 따라 TTRT 중 일부를 할당받는데 이 할당받는 양이 용량벡터이다. 각 노드는 토큰을 받으면 최대 용량 벡터만큼 실시간 메시지를 전송할 수 있으며 TTRT 보다 토큰이 일찍 도착했을 때에만 비실시간 메시지를 전송할 수 있다. 실시간 시스템에 있어서 모든 실시간 메시지 스트림에 대한 정보는 오프라인 시에 알려지며 메시지 스트림 S_i는 주기 P_i와 전송 시간 C_i로 특성화되므로 통신 스케쥴러는 주어진 $\{S_i\}$ 집합에 대해 실시간 제약조건을 만족시키도록 TTRT와 용량 벡터 $\{H_i\}$를 결정하여야 한다. 한 스트림이 토큰을 맞이하는 횟수는 각 주기마다 다르게 되며 현재까지 제안된 대역폭 할당 기법들은 모두 경성 실시간 제약조건을 만족시키기 위하여 최소의 전송시간을 갖는 주기를 기반으로 용량 벡터를 할당하기 때문에 각 노드는 충분한 대역폭을 할당받게 된다[5].

4. FDDI 상에서의 오류제어 기법

4.1 오류의 탐지 및 보고

오류제어를 위해서는 수신자 노드는 수신된 메시지의 오류를 탐지하고 이를 송신자에게 보고하여야 한다. FDDI 네트워크에서 전송 오류의 탐지는 프레임에 포함된 FCS(Frame Check Sequence)와 FS(Frame Status) 필드를 이용하는데 FCS는 프레임의 오류를 탐지하기 위한 32비트 CRC(Cyclic Redundancy Check)를 포함하고 있으며 FS는 프레임 전송중에 발생한 오류를 나타내기 위한 ED(Error Detection) 심볼을 포함하고 있다[4]. FDDI 프레임이 링 토폴로지를 따라 전송될 때 각 노드는 수신된 프레임을 검사하고 오류가 탐지되면 ED 필드를 세팅하는데 모든 프레임은 자신을 전송한 노드로 환원되어 삭제되기 때문에 전송 노드는 프레임에 오류가 발생했는지 판단할 수 있다. FCS를 이용한 오류의 탐지와 보고는 메시지의 손상에 기인한 오류를 탐지하고 보고할 수 있기는 하지만 수신 버퍼 초과에 기인한 기각과 같이 수신자 내부 원인에 기인한 오류는 보고할 수 없으며 링 구조상 수신 노드 이후에서 발생한 오류에 대해 송신자와 수신자 사이의 불일치가 발생할 가능성도 내포하고 있다.

이를 극복하기 위해서는 추가적인 재전송 요청 메시지를 통해 송신자에게 오류 발생을 보고하여야 하며 이 재전송 요청 메시지는 실시간 메시지의 전송에 지연을 주지 않도록 비동기 대역폭을 이용하여 전송하여야 하는데 재전송 요청은 SNR 방식과 같이 재전송 리스트를 주기적으로 전송하는 방안이 바람직하다[15]. 즉, 수신자는 각 주기마다 재전송 리스트를 초기화하고 프레임 수신시에 오류가 발생하면 오류 프레임을 리스트에 추가하여 재전송 리스트를 주기적으로 송신자에게 보고한다. 재전송 시점의 결정은 FDDI 네트워크에서는 효율적으로 계산될 수 있는데 연결 설정 시에 각 스트림 마다 한 메시지가 몇 개의 프레임으로 나뉘어 전송되는지 계산될 수 있으므로 수신자는 메시지의 송신 시작 시점부터 자신이 수신하는 토큰의 개수를 세고 있으면 한 메시지에 속한 프레임의 전송이 모두 끝났는지 알 수 게

된다. 그림 4에서는 그 예를 보이고 있는데 한 메시지가 u 개의 프레임으로 나뉘어 전송된다면 수신자는 자신이 수신한 토큰의 수에 의해 송신자의 전송이 완료되었는지 판단할 수 있다. 처음으로 수신된 프레임이 2라고 한다면 카운터를 2로 설정하고(1번 프레임은 재전송 리스트에 추가) 이후부터 토큰의 수신 회수를 측정하여 최종적으로 u가 된다면 재전송 리스트를 전송한다. 이 방식에 의해 임의의 프레임에 손상이 발생하더라도 토큰만 정확하게 수신이 된다면 수신자는 쉽게 재전송 시점을 결정할 수 있다. 물론 한 메시지에 속한 모든 프레임이 손상되면 재전송 요구를 할 수 없는데 이 경우는 어차피 종료시한 이내에 오류 복구를 할 가능성이 거의 없다. 재전송되는 프레임의 경우 임의의 순서번호를 갖기 때문에 최초 수신 프레임 번호 인식에 오류를 발생시킬 수 있으므로 일반적인 전송과 재전송되는 프레임을 구분하기 위해 MAC 프레임에 포함된 예약 비트를 사용한다.

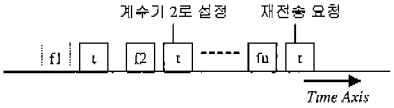


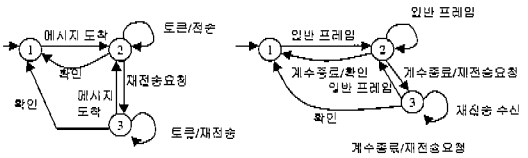
그림 4 프레임 오류의 탐지 및 보고

4.3 재전송 기법

수신자 노드는 오류 때문에 순서에 어긋나게 수신된 노드를 재조립할 수 있어야 하며 이를 위해 C_i 크기의 버퍼가 필요하다. 송신자 역시 추후의 재전송을 위해 C_i 크기의 버퍼를 할당받는데 재전송 가능성은 주기 이내에서만 발생 가능하며 일반적으로 송신 버퍼는 새로운 메시지가 발생할 때마다 자신의 버퍼에 덮어쓰게 된다. 또 다른 오류 제어 기법과 마찬가지로 각 프레임 마다 순서번호를 추가하며 수신자는 재전송에 의한 프레임 수신 순서의 전복을 복구할 수 있다고 가정한다.

그림 5는 송신자와 수신자의 오류 제어를 위한 상태도를 보여주고 있다. 그림 5(a)는 송신자 측의 상태도를 보여주고 있는데 역시 상태 1은 초기 상태로서 노드 내부에서 전송할 메시지가 생성되면 상태 2로 천이하며 상태 2에서는 토큰을 받을 때마다 한 프레임씩 전송한다. 상태 2에서 수신자로부터 재전송 요청과 확인을 받을 수 있으며 이들 메시지가 손실된 경우에는 다른 메시지가 도착한다. 확인을 받으면 상태 1로 천이하여 다음 메시지의 도착을 기다리게 되며 재전송 요청을 받으면 상태 3으로 천이하여 토큰을 맞이할 때마다 재전송 리스트에 포함된 프레임을 하나씩 전송한다. 상태 3에서 재전송 중 새로운 메시지가 도착하면 재전송 과정을 기각하고 상태 2로 천이하여 일반 프레임을 전송한다.

그림 5(b)에서 보는 바와 같이 수신자는 상태 1에서 일반 프레임을 처음으로 수신하면 상태 2로 천이하며 상태 2에서 토큰을 맞이할 때 수신 계수기를 최초로 수신된 프레임의 번호로 설정함과 아울러 재전송 리스트를 초기화한다. 상태 2에서는 계속해서 프레임을 수신하면서 토큰을 맞이할 때마다 계수기를 증가시키는데 수신되지 못한 프레임의 번호를 재전송 리스트에 기록한다. 계수기의 값이 목적치가 되면 메시지의 오류 발생 여부에 따라 확인 혹은 재전송 리스트를 송신자에게 보내고 각각 상태 1 혹은 3으로 천이한다. 상태 3은 재전송되는 프레임들을 수신하는 상태로서 이 상태에서 일반 프레임이 수신되면 이전 메시지의 프레임들에 대한 수신을 포기하고 상태 2로 천이한다. 물론 계수기와 재전송 리스트를 수신된 메시지에 따라 초기화 한다. 상태 3에서 재전송에 의해 오류가 복구되었을 때는 상태 1로 천이하여 메시지의 수신을 완료한다.



(a) 송신자 노드 (b) 수신자 노드

그림 5 송수신자 노드의 상태도

오류의 보고는 수신자가 송신자에게 재전송 리스트를 전송함으로써 수행되는데 만약 수신자가 토큰을 받았을 때 충분한 비동기 대역폭이 없다면 보고를 할 수 없으며 오류 제어가 불가능하다. 송신자는 한 메시지의 전송이 모두 완료된 후 재전송 요청을 받으면 주기내에서 이후 토큰을 맞이할 때만 재전송을 수행하고 또 다시 새로운 메시지가 도착하면 이전 재전송 요청을 무시한다.

5. 분석 및 모의 실험 결과

본 절에서는 오류제어 기능이 없는 단일 네트워크와 제안된 오류제어 기능을 갖는 네트워크의 메시지 오류율을 분석 및 모의실험에 의해 측정한다. 제안된 방식의 성능과 비교하기 위한 또 다른 기준으로 MARS와 같이 프레임을 중복하여 전송하는 방식을 선택한다. 이 방식에서는 대역폭의 낭비가 심하지만 각 프레임의 중복이 모두 손실되었을 때에만 프레임의 손실로 처리되므로 메시지 오류율이 상대적으로 낮다.

5.1 분석

3.2에서 소개된 바와 같이 주어진 스트림 집합 $\{S_i\}$의 원소 Si는 트래픽 특성으로 주기 Pi와 전송시간 Ci를 갖는다. 대역폭 할당 기법은 이를 기반으로 TTRT와 용량벡터 $\{Hi\}$, 즉 노드의 최대 토큰 보유 시간을 결정한다. 결국 $\{Si\}$에 대해 메시지의 오류율, 혹은 전송오류에 의한 실기율(deadline miss ratio)은 식 (1)과 같이 정의되는데 F_i와 E_i^X는 각각 메시지 스트림 S_i의 발생빈도와 오류 발생율이다.

$$E^X = F_i \cdot E_i^X, \quad X\text{는 오류제어 방식} \tag{1}$$

발생 빈도 F_i는 T를 충분히 큰 임의의 시간 구간이라 정의할 때 식 (2)와 같이 계산된다.

$$F_i = \frac{\frac{T}{P_i}}{\sum_{i=1}^{n}\frac{T}{P_i}} = \frac{\frac{1}{P_i}}{\sum_{i=1}^{n}\frac{1}{P_i}} = \frac{1}{P_i \cdot \pi}, \text{ 단 } \pi = \sum_{i=1}^{n}\frac{1}{P_i} \tag{2}$$

ρ를 비트 오류율이라고 할 때 오류제어를 수행하지 않는 단일 네트워크 상에서 스트림 S_i에 오류가 발생할 확률 E_i^s는 잘 알려진 바대로 크기 C_i인 메시지에 오류가 발생할 확률로 계산되므로 식(3)과 같이 계산된다[6].

$$E_i^s = 1-(1-\rho)^{C_i} \cong \rho \cdot C_i \tag{3}$$

이 과정에서 프레임 분할은 오류율에 영향을 주지 않으며 결국 오류제어를 수행하지 않는 단일 네트워크에서의 전체적인 메시지 오류율 E^s 는 식(4)와 같이 계산된다.

$$\begin{aligned} E^s &= \sum_{i=1}^{n} F_i \cdot E_i^s = \sum_{i=1}^{n}\frac{1}{\pi P_i} \cdot E_i^s \\ &= \frac{1}{\pi} \cdot \sum_{i=1}^{n}\frac{E_i^s}{P_i} = \frac{1}{\pi} \cdot \sum_{i=1}^{n}\frac{\rho \cdot C_i}{P_i} \\ &= \frac{1}{\pi} \cdot \rho \cdot \sum_{i=1}^{n}\frac{C_i}{P_i} = \frac{1}{\pi} \cdot \rho \cdot U, \text{ 단 } U = \sum_{i}^{n}\frac{C_i}{P_i} \end{aligned} \tag{4}$$

제안된 기법의 오류율을 구하기 위해 복구 가능한 오류율을 구하도록 하며 이는 한 프레임의 오류 발생율과 한 주기 내에서 사용가능한 여분의 대역폭에 의한 함수로서 계산된다. 한 프레임의 오류 발생율 e_i는 식 (3)에 C_i 대신 H_i를 대입함으로써 $\rho \cdot H_i$ 와 같이 계산되며 한 주기내 여분의 대역폭 R_i는 전체 실시간 트래픽, 비실시간 트래픽 및 오류 처리를 위한 트래픽의 확률 함수에 의해 결정된다. 즉 네트워크 전체적으로 실시간 트래픽이 많다면 여분의 대역폭이 감소하게 되며 또 각 주기가 $TTRT$로 약분되지 않는 경우 각 주기마다 맞이하는 여분의 토큰을 맞이하는 횟수가 다를 뿐 아니라 분석적 모델로 표현될 수 없다[3]. 이 과정에서 R_i의 확률 밀도 함수를 구할 수는 없지만 평균값을 구할 수는 있으므로 평균값을 사용하여 근사치를 구하도록 한다. 각 스트림이 주기 내에서 맞이하는 평균 여분 대역폭 R_i는 전체 대역폭에서 실시간 및 비실시간 트래픽이 차지하는 비율을 제거함으로써 식 (5)와 같이 계산된다.

$$R_i = (1-U) \cdot P_i - \delta \tag{5}$$

식(5)에서 δ는 기타 트래픽이 차지하는 비율로서 P_i 내에서 평균 비동기 트래픽과 오류제어 트래픽이 차지하는 비율, 토큰 회전 오버헤드 및 오류를 보고하는데 있어서의 지연시간을 포함한다. 결국 한 주기 내에서 각 스트림은 평균적으로 재전송 과정 중에 발생한 프레임 오류를 포함하여 $\lfloor R_i / H_i \rfloor$개 까지의 프레임 오류를 제어할 수 있으며 스트림 S_i에 있어서의 오류율 E_i^P을 계산하기 위해 u를 한 메시지 전송에 필요한 프레임 수, v를 제어가능한 오류발생 프레임 수라 하면 이들은 각각 $\lfloor C_i / H_i \rfloor$와 $\lfloor R_i / H_i \rfloor$로 계산된다. 복구가능한 오류는 주기 내에서 k 개의 프레임 오류가 발생했을 때 이를 오류제어 기능에 의해 처리하는 확률로서 k는 0부터 u까지의 값을 가지되 v보다는 작거나 같아야 한다. 결국 제안된 방식에 있어서 S_i의 오류율 E_i^p는 식(6)과 같이 계산되며 이에 따라 메시지 오류율 E^P는 식(1)에 E_i^X 대신 E_i^P 를 대입함으로써 구해진다.

$$E_i^P = 1 - \sum_{k=0}^{v} {}_uC_k \, e_i^k \cdot (1-e_i)^{u-k} \cdot (1-e_i)^k \tag{6}$$

이중 네트워크를 통한 중복 전송 방식은 모든 프레임이 이중으로 중복되어 전송되므로 두 프레임이 모두 손실되었을 때에만 프레임의 손실이 발생한다. E_i^D 는 식 (7)과 같이 계산되며 역시 메시지 오류율 E^D는 식(1)에 E_i^X 대신 E_i^D 를 대입함으로써 구해진다.

$$E_i^D = 1-(1-(\rho \cdot H_i)^2)^u \tag{7}$$

5.2 모의 실험

제안된 기법과 오류제어가 없는 네트워크, 이중링크에 의한 중복 전송 등의 기법에 대해 매체 오류율, 사용율 및 비동기 트래픽 등의 네트워크 인자에 따라 종료시한 만족도를 SMPL을 이용한 모의실험에 의해 측정하였다[16]. 제안된 기법에 대해서는 5.1에서 제시된 분석에 의한 결과도 함께 도시한다. 매체 오류율에 따른 종료시한 만족도 측정을 위해 70~79%까지의 사용율을 갖는 스트림의 집합을 40개 생성하고 스트림의 특성에 따라 대역폭을 할당한 후 매체 오류율을 10^{-5}에서 10^{-9}지 변화시키면서 종료시한 만족도를 측정하였다. 본 실험에서 대역폭 할당은 기존의 연구결과를 이용하여 식 (8)과 같이 결정하였다[3]. 식 (8)에서 P_{min}은 주기의 최소값이며 γ는 토큰 회전에 따른 낭비시간이다.

$$TTRT = \frac{P_{min}}{\left\lfloor \frac{-3+\sqrt{9+\frac{8P_{min}}{2}}}{2} \right\rfloor}, \qquad H_i = \frac{C_i \cdot P_i}{U} \cdot (TTRT-\gamma) \tag{8}$$

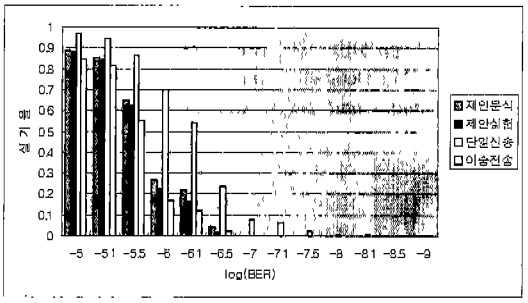


그림 6 매체 오류율 대 실기율

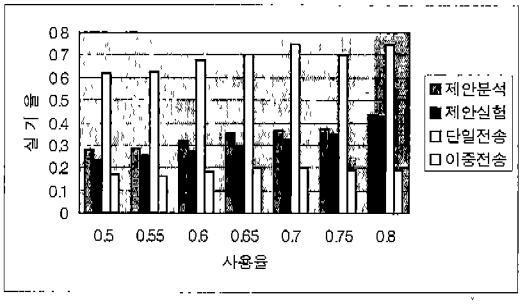


그림 7 사용율 대 실기율

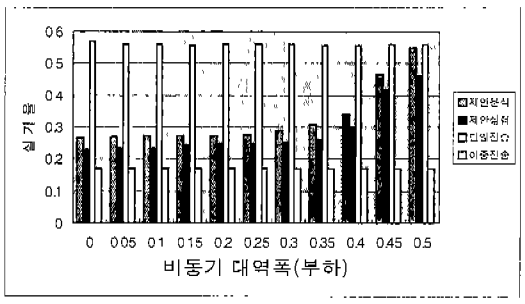


그림 8 비동기 대역폭 대 실기율

그림 6은 매체 오류율과 메시지들의 실기율을 보여주고 있으며 제안된 방식이 단일 전송 방식에 비해 실기율을 현격히 감소시킬 수 있을 뿐 아니라 이중 전송 방식에 유사한 종료시한 만족도를 보이고 있다. 즉 이중화와 같은 고비용 요소나 중복 전송과 같은 대역폭의 낭비없이 메시지 스트림의 종료시한 만족도를 개선할 수 있음을 나타낸다. 단일 전송과 이중 전송 방식은 모의실험과 분석 결과가 3% 이내의 오차를 보이고 있는데 반해 제안된 방식은 분석 결과의 실기율이 모의실험 결과보다 최대 8%까지 낮은데 이는 R_i를 계산하는 과정에서 확률 밀도 함수가 아닌 평균값을 사용했기 때문으로 추정된다. 또 비동기 트래픽이 없다고 가정하였으므로 모든 노드에 있어서 대부분의 주기에서 토큰은 TTRT보다 일찍 도착하게 되어 오류의 보고가 한 토큰 회전 시간 이내에 수행이 완료된다.

그림 7은 사용율에 따른 실기율을 보여주고 있는데 매체 오류율은 10^{-6}으로 고정하고 스트림 집합의 사용율을 50 %에서 80 %까지 변화시켰으며 역시 비동기 트래픽은 발생하지 않는다고 가정하였다. 사용율이 높아질수록 제안된 기법의 실기율이 증가하는 이유는 복구할 수 있는 오류의 수에 직접적으로 영향을 주는 R_i가 감소하기 때문이다. 사용율이 낮아질수록 모든 프레임 마다 이중으로 중복되어 전송할 확률이 높아지므로 중복 전송하는 방식과 성능이 유사하게 된다. 그림 8은 비동기 대역폭의 영향을 측정하기 위해 사용율 70%인 스트림 집합을 40개 생성하고 매체 오류율을 10^{-6}으로 설정한 후 여분의 대역폭에 대한 비동기 트래픽의 발생율을 0%부터 50%까지 발생시켰다. 비동기 트래픽이 증가할수록 오류 보고를 못할 가능성이 증가할 뿐 아니라 여분의 대역폭도 따라서 감소하므로 제안된 기법의 실기율이 증가한다.

6. 결 론

기존의 연구에 의하면 최근까지 제안된 오류제어 기법들이 데이타의 종료시한을 직접적으로 고려하지 않았기 때문에 슬라이딩 윈도우와 같은 고전적인 오류제어 기법에 비해 약간의 개선을 보여줄 뿐 시간 오류에 대한 근본적인 해결을 하고 있지는 않다고 할 수 있다. 결국 MAC 계층에서 메시지의 오류율을 감소시켜 상위 계층에서의 오류율을 감소시키는 방안이 바람직한데 본 논문에서 제안된 방식은 하부 네트워크의 MAC에 기반하여 이의 특성을 이용함으로써 기존 오류제어 방식의 비실시간성을 극복할 수 있다. 제안된 방식은 토큰의 수신 회수를 측정하여 오류 보고 시점을 결정한 후 비동기 대역폭을 통해 전송하고 재전송은 과할당된 대역폭을 통해 수행함으로써 다른 실시간 트래픽의 전송에 전혀 영향을 주지 않는다. 이중 전송에 의한 오류율 감소보다 실기율을 더욱 감소시키기 위해서는 오류제어 라운드를 여러 번 수행할 수 있도록 대역폭을 할당할 때 추가적으로 미리 확보하여 R_i를 늘려야 한다. 또 본 논문에서는 메시지의 종료시한이 주기와 같다고 가정하였으나 종료시한이 주기보다 큰 경우는 여러 주기간에 오류제어가 수행될 수 있도록 보완되어야 한다.

참 고 문 헌

[1] K. Arvind, K. Ramamritham, J. 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. Kurose, "Real-time communication on packet-switched networks," *Proc. of IEEE*, Vol. 82, No. 1, pp.122-129. Jan. 1994.

[3] N. Malcolm and W. Zhao, "Hard real-time communication in multiple-access networks," *Journal of Real-Time Systems*, pp.37-77, 1995.

[4] S. Mirchandani, R. Khanna, *FDDI Technology and Applications*, Johm Wiley and Sons, Inc., 1993.

[5] B. Chen, G. Agrawal, W. Zhao, "Optimal synchronous capacity allocation for hard real-time communications with the timed token protocol," *Proc. Real-Time Systems Symposium*, pp.198-207, 1992.

[6] M. Schwartz, *Telecommunication Networks: Protocols, Modeling and Analysis*, Addison Wesley Publishing Company, 1987.

[7] J. Lee, C. Kim, "Design of an error control scheme for hard real-time communication on FDDI networks," *IEEE TENCON*, pp.828-831, Sep. 1999.

[8] 기초전력공학 공동연구소, 전기 임피던스 단층촬영기법에 의한 이상유동장 기포분포 측정 알고리즘 개선, 1999

[9] A. Romanow, S. Floyd, "Dynamics of TCP traffic over ATM networks," *IEEE Journal on Selective Areas in Communications*, Vol. 13, No. 5, pp. 633-641, May. 1995.

[10] H. Kopetz, A. Damm, J. Reisinger, W. Schwabl, "The real-time operating systems of MARS," *Proc. Operating Systems Review*, pp.141-157, 1988.

[11] H. Garcia-Molina, B. Gao, D. Barbara, "Aggressive transmissions over redundant paths," *Proc. Distributed Computing Systems*, pp.198-207, 1991.

[12] Z. Kohavi, *Switching And Finite Automata Theory*, McGraw-Hill, 1978.

[13] 정충일, 권도한, 박창윤, "멀티미디어 통신에서 적시 재전송에 기반한 선택적인 오류 제어 방법", 정보과학회논문지(A), 26권 10호, pp.1225-1236, 1999년 10월.

[14] F. Gong, G. Parulkar, "An application-oriented error control scheme for high-speed networks," *IEEE Trans. Networking*, Vol. 4, No. 5, pp.669-683, October 1996.

[15] A. N. Natravali, W. D. Roome, K. Sabnani, "Design and implementation of a high-speed transport protocol," *IEEE Trans. Communication*, Vol. 38, No. 11, pp.2010-2024, Nov. 1990.

[16] M. H. MacDougall, *Simulating Computer Systems: Techniques and Tools*, MIT Press, 1987.

이 정 훈

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

김 호 찬

1987년 서울대학교 제어계측공학과 공학사. 1989년 서울대학교 제어계측공학과 석사. 1994년 서울대학교 제어계측공학과 박사. 1995년 ~ 현재 제주대학교 공과대학 전기공학과(조교수).