전동차용 통신 네트워크 프로토콜 구현 및 성능평가에 관한 연구

論 文
48A-12-14

A Study on the Implementation and the Performance Evaluation of the Train Communication Network

李 尙 哲* · 朴 宰 賢** · 張 來 赫***
(Sang-Chul Lee · Jae-Hyun Park · Nae-Hyuck Chang)

Abstract - This paper evaluates the real-time performance of the Train Communication Network (TCN) that consists of WTB and MVB. A run-time scheduling algorithm for the hard-real time communication was proposed and its performance was evaluated. Also, a new addressing method and the adaptive tree algorithm were suggested to enhance performance. The overall performance was evaluated by computer simulation using Arena.

Key Words : Train Communication Network, real-time communication

1. 서 론

지하철 및 고속전철과 같은 전동차 시스템은 고도의 기술이 집적화 된 제어기술과 함께 온라인 감시, 자기진단 및 승객 정보서비스 등의 기능들을 요구하고 있다. 이러한 다양한 기능들의 수행을 위하여 분산화와 모듈화가 이루어지고 있어 차량 네트웍 시스템이 가지는 역할이 중요해지고 있다. 차량 네트워크 시스템의 표준화는 분산된 제어기기, 예를 들어 추진제어기, 제동제어기, 진단기기, 운전자 입력기기 사이에 빠르고 정확한 데이터 교환을 위한 유연성(Flexibility)을 제공하는데 기본적인 목적을 두고 있다. 그러나 전동차용 네트워크 프로토콜 설계에는 다음과 같은 어려운 점이 있다. 첫째, 차량내의 하나의 데이터 버스를 크게는 지능형 스테이션부터 작게는 간단한 I/O 기기들이 공유해야 한다. 둘째, 데이터 서비스에 있어서는 제어신호와 같이 전송시간이 한계치를 넘지 않도록 시간제약성(Time Constraints)을 요구하는 데이터가 있는가 하면, 시간제약성은 작지만 상당한 크기의 데이터 전송을 필요로 하기도 한다. 셋째, 전체 차량시스템 차원에서 보면 차량간의 연결과 차량 구성의 변화에 대한 지능성이 필요하게 된다. 국제 표준화 기구 중 하나인 International Electrotechnical Commission(IEC)에서는 차량간 혹은 차량내 전자 장치들의 상호운용성을 목적으로 한 전동차용 통신 네트워크 프로토콜(Train Communication Network Protocol)을 IEC 61375-1로 국제 표준(International Standard)화 하였다[1-3]. 전동차용 통신 네트워크는 개방형 통신 프로토콜로서 차량내 기기뿐만 아니라 차량의 교환 및 연결에 대한 유연한 상호운용성을 목적으로 하며, ABB, AEG, Siemens 등과 같은 유럽의 여러 철도관련 업체에 의해 지원, 개발되고 있다.

전동차와 같이 고도의 안전성과 실시간성을 요구하는 분산 제어 시스템에서 네트워크의 안전성과 실시간성은 전체 제어시스템의 성능과 직결되며, 네트워크 프로토콜의 성능분석은 네트워크 시스템 구축뿐만 아니라 응용 프로세스 개발에 필수적이다. 이에관한 연구는 실시간 필드버스를 중심으로 상당부분 이루어져 있으나, 전동차용 통신 네트워크의 오류는 인명을 해치는 사고로 직접 연결된다는 점과 운행 중 차량의 연결 및 구성변화 등과 같은 전동차고유의 특징으로 볼 때 필드버스와 같은 기존의 유사한 연구가 직접적으로 적용되기 힘들다[4-6].

본 논문은 전동차용 통신 네트워크 프로토콜에 대한 고찰과, 성능에 가장 큰 영향을 주는 데이터 링크 계층을 집중적으로 분석한다. 이와 함께 실시간 네트워크 환경 구축을 위한 실시간 데이터의 스케쥴링 기법을 제시하고 처리용량과 지연시간을 계산한다. 또한 매체 접근 제어(MAC) 방식으로 사용되고 있는 트리 알고리즘상의 어드레싱 기법과 적응기법을 제시하고 성능분석을 위한 모델링과 시뮬레이션을 수행한다. 실제적인 유용성과 적용성을 판단하기 위하여 전동차용 통신 네트워크의 일부인 차량내 통신버스를 구현한다. 이러한 분석과 방법제시로 본 논문은 실제 네트워크의 적용시 테스트 데이터를 제공한다.

본 논문의 구성은 2장에서는 전동차용 통신 네트워크 프로토콜을 개괄적으로 소개하고, 3장에서는 성능과 관련된 특성을 분석하고 적용할 수 있는 알고리즘을 제안, 이들에 대한 성능평가를 수행한다. 4장에서는 유용성과 성능을 평가하기 위하여 개발된 차량내 통신버스 컨트롤러의 구현과 관련된 내용과 실험 결과를 기술한다.

* 非 會 員 : (주)로커스 硏究員
** 正 會 員 : 仁荷大 自動化工學科 副敎授 · 工博
*** 非 會 員 : 서울大 컴퓨터工學科 專任講師 · 工博
接受日字 : 1999年 2月 25日
最終完了 : 1999年 11月 2日

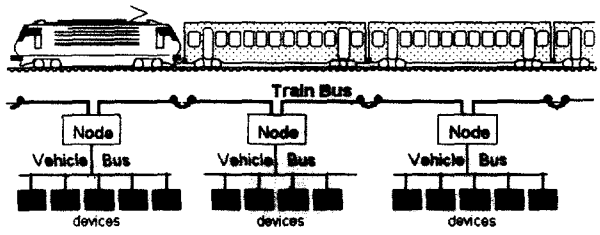


그림 1 전동차용 통신 네트워크의 계층적 구조
Fig. 1 Hierarchical structure of TCN

2. 전동차용 통신 네트워크의 구조 및 서비스

전동차용 통신 네트워크는 그림 1과 같이 서로 다른 차량에 있는 노드를 연결하는 차량간 버스(Wired Train Bus)와 차량 내에 있는 기기를 연결하는 차량내 버스(Multifunction Vehicle Bus)로 구성되어 있다. 이렇게 계층구조를 가지는 이유는 각 버스가 가지는 기능 및 특성과 깊은 연관이 있는데 차량간 버스는 전송 속도가 1Mbps로 기존의 차량과의 호환성을 위하여 길이 860m에 22개의 차량과 32개까지의 노드를 지원하며 차량의 구성 변화가 자주 일어나는 점을 고려하여 설계되었다. 따라서 차량의 구성 변화에 따른 자동 인증(Inauguration)을 위하여 하드웨어적인 주소화가 지원되며, 노드의 고장 시 모든 응용프로세스에게 새로운 환경을 구축하도록 알려주는 기능 등 초기화 기능과 안정성에 중점을 두었다. 반면 차량내 버스는 전송 속도가 1.5Mbps로 일반적으로 구성 환경이 자주 바뀌지 않는 특성과 보다 높은 효율의 서비스를 위하여 설계되었다. 이와 같이 두 버스는 각각의 역할은 다르지만 OSI(Open System Interconnection; 개방형 시스템간 상호 접속) 참조 모델 7계층 중에서 물리계층과 데이터링크계층을 제외한 나머지 계층들은 같은 프로토콜 스택을 사용함으로써 이식성과 호환성이 높고 유지 보수가 용이하도록 설계 되었다.(그림 2 참조)

OSI 7계층이란 서로 다른 네트워크 구조를 가진 이기종 컴퓨터간의 통신을 위해 ISO (International Standardization Organization) 에서 규격화한 네트워크 구조로, 최하위의 실제적인 매체를 담당하는 물리계층, 물리계층에 데이터를 실어 넣거나 물리계층의 신호를 받아 인식 할 수 있는 데이터로 만들어주는 데이터링크계층, 통신 경로를 확보해 주는 네트워크계층, 안정적인 전송을 확보하기 위한 트랜스포트계층, 이 위에 세션계층과 프리젠테이션계층, 마지막으로 최상위 어플리케이션계층으로 구성된다. OSI 7계층 중에서 트랜스포트계층에서 컴퓨터간의 안정적인 연결을 책임지므로, 세션계층부터는 소프트웨어적인 고려만을 하면 되는데 따라서 실제 네트워크 구현 시 세션계층, 프리젠테이션계층, 어플리케이션계층은 뚜렷한 구분이 없이 구현되기도 한다.

전동차용 통신 네트워크는 차량간 버스와 차량 내 버스가 서로 다른 구조와 서로 다른 버스를 사용하며 연결되고, 세 가지 종류의 데이터, 즉 프로세스 데이터(Process data), 메시지 데이터(Message data), 관리용 데이터(Supervisory data)의 형태로 서비스를 제공한다.

프로세스 데이터란 차량상태, 예를 들어 차량의 속도, 모터전류, 공기압력 등과 같은 전동차의 상태를 감시하는 데이터와 주기적 제어명령 등과 같이 길이는 비교적 짧지만 시간제약성이 있는 데이터를 말한다. 전동차용 통신 네트워크는 이러한 프로세스 데이터에 대해 차량내 통신 버스 안에서는 50ms, 차량간 통신 버스를 이용 시에는 100ms내의 전송지연을 보장한다. 따라서 통신에 필요한 최소한도의 기능을 수행하는 단순한 구조를 가지면서 빠른 응답시간을 가지기 위하여 OSI 7계층 중 1, 2, 7 계층만을 사용한다. 데이터의 전송은 기본 주기를 프로세스 데이터와 메시지 데이터를 보내기 위한 시간대를 일정 비율로 나누어 이루어지는데, 프로세스 데이터는 할당된 시간대에 마스터의 전송요구에 따라 브로드캐스팅(Broadcasting)된다. 따라서 전송매체를 접근 할 때 충돌을 피할 수 있고 실시간성을 보장받을 수 있다. 또한 해당하는 데이터를 필요로 하는 디바이스들은 한 번의 전송만으로 각 디바이스들이 같은 데이터를 전송 받을 수 있는 장점이 있다.

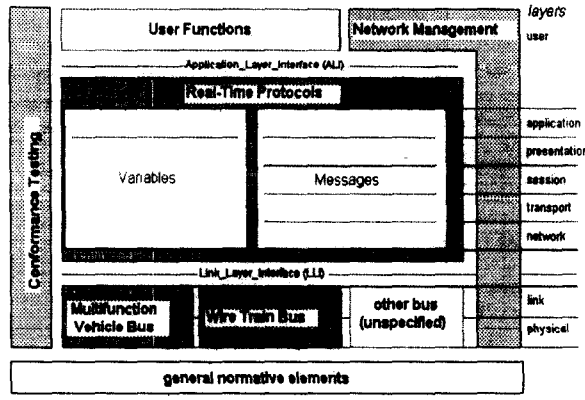


그림 2 프로토콜 스택 구조
Fig. 2 Protocol stack structure

이에 반하여 시간제약성이 상대적으로 덜하며, 이벤트 발생적이고, 길이가 비교적 긴 데이터, 예를 들어 승객정보, 진단정보 등을 메시지 데이터라 하며 안정적이고 신뢰성 있는 통신을 위하여 7계층 모두를 사용한다. 메시지 데이터는 기본주기 내에 산발적으로 발생하는 메시지를 위해 예약된 시간대에 보내지는데, 마스터는 메시지 데이터를 가진 디바이스를 먼저 탐색하게 되고, 하나 이상의 디바이스들이 메시지를 가지고 있을 경우 프로세스 데이터와는 달리 충돌이 일어난다. 응용프로세스는 클라이언트/서버와 같이 호출자(Caller)/응답자(Replier)로 메시지를 교환한다. 메시지는 여러 패킷으로 나누어지며 각각의 패킷에는 보내는 자신의 어드레스와 받는 목적지의 어드레스를 가지고 있으며 차량간 버스의 노드에서는 네트워크 계층에서 이를 라우팅하게 된다. 또한 신뢰성을 높이기 위한 흐름제어 및 에러복구가 트랜스포트 계층에서 지원된다. 이를 위하여 각 디바이스들은 고유의 어드레스를 가지며 전송하기 위한 큐와 전송을 받기 위한 큐를 사용하고 있다. 이와는 다르게 순전히 네트워크의 유지보수를 위한 데이터, 예를 들어 디바이스 상태, 마스터권 이양 등을 관리용 데이터라고 하며 네트워크 관리용 응용프로세스와의 인터페이스를 지원한다. 이는 각 데이터의 특성에 따라 프로세스 데이터 서비스 혹은 메시지 데이터 서비스와 같은 방식으로 사용되기 때문에 네트워크 성능이나 전송특성과 밀접한 관계가 없다.

3. 성능분석 및 평가

본 장에서는 TCN 상에서의 프로세스 데이터에 대한 확정론적인 분석 및 메시지 데이터에 대한 확률론적인 컴퓨터 모의 실험을 통해 TCN의 성능을 예측하고, 각각 적합한 구현 방법을 제시한다.

3.1 프로세스 데이터 서비스

3.1.1 스케쥴링

전동차 차량 제어를 위한 응용 프로세스들은 일반적으로 시간제약성을 가지며, 이러한 시간적인 제약을 준수하지 못하였을 경우 치명적인 오류를 범하게 된다. 따라서 이러한 시스템에서의 실시간 네트워크 프로토콜은 일반적으로 필드버스와 같이 크게 응용계층, 데이터링크계층, 물리계층으로 구성되어 있다. 응용계층에서의 데이터 전송은 이벤트 발생에 의한 전송(Event triggered)과 시간 발생에 의한 전송(Time triggered)이 있다. 전동차용 통신 네트워크의 실시간 통신 서비스인 프로세스 데이터 서비스는 데이터링크계층이 시분할 다중방식을 사용하기 때문에 주기적인 특성을 가지는 차량의 추진이나 제동과 관계된 응용프로세스에게 시간 발생에 의한 전송에 적합하게 되어 있다. 이러한 주기적인 데이터는 일반적으로 피드백 제어 루프에 사용되며 전송오류, 지연 및 지터(jitter)는 제어시스템의 성능 저하나 불안정의 요인이 된다[7].

마스터와 슬레이브로 구성되어 있는 비대칭형 프로토콜에 있어서 이러한 통신상의 시간적 요구는 매체 접근 권한을 부여하는 버스 관리자에 의한 스케쥴링 성능에 의존하게 된다. 전동차용 통신 프로토콜의 주기적 전송구간에서 전송되는 프레임은 포트 단위로 전송하는데 각 데이터는 해당하는 논리적 포트에 할당되며 각 포트는 고유의 주기, 즉 개별 주기를 가지게 된다. 따라서 전송지연의 데드라인은 개별주기가 되며 전송지연의 불규칙이 지터가 된다. 모든 개별 주기들을 바탕으로 버스 마스터는 폴링 사이클인 거시사이클을 가지는 스캔리스트를 생성해야한다. 위와 같이 개별주기를 가지는 데이터들을 실행 전에 스케쥴하는 방법으로는 주로 최소공배수에 의한 방법과 오버샘플링을 고려한 방법이 있는데, 전자는 메모리를 많이 차지하며 후자는 지터가 생기는 단점이 있다[8][9].

프로세스 데이터 서비스에선 개별주기가 기본주기와 2의 제곱승의 곱으로 정해진다. 따라서 거시사이클은 가장 긴 개별주기가 되며, 기본주기는 가장 짧은 개별주기와 일치하게 된다. 개별주기의 결정과정에서의 위와 같은 제약성은 응용프로세스의 샘플링을 결정하는데 많은 제약을 두는 것처럼 보인다. 그러나 모든 개별주기가 갖는 배수 혹은 약수의 관계는 비율 단조형 스케쥴링(rate monotonic scheduling) 방법을 사용하면 예상되는 모든 지터, 각 개별 주기와 다르게 스케쥴링 되어 나타나는 지터와 주기적 전송구간에서 차지하는 위치에 따른 지터를 방지할 수 있다[11]. 또한 스케쥴링의 결과가 일정한 분포를 가지게 하기 위하여 주기에 따른 오프셋을 두어 배치할 수도 있다. 위와 같이 개별주기를 정하는 알고리즘에 의하여 가장 짧은 개별주기와 기본주기(1,2,4 혹은 8ms) 중 짧은 시간을 기본주기로 하여 표 1과

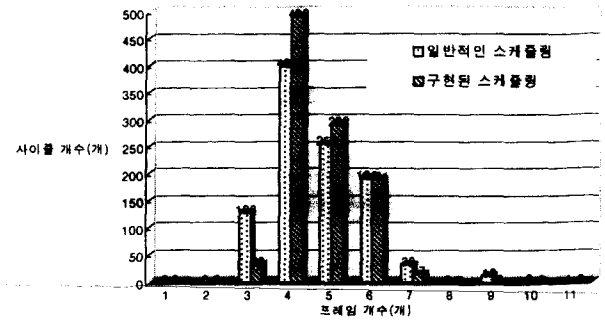


그림 3 스케쥴링 결과

Fig. 3 Scheduling result for test data

표 1 스케쥴링 표본 데이터

Table 1 Test data for scheduling

데이터주기 (ms)	1	2	4	8	16	32	64	128	256	512	1024
데이터 수	1	2	4	6	7	10	7	6	4	2	1

같은 테스트 데이터를 대상으로 그림 3과 같은 결과를 얻었을 수 있다. 사용한 테스트 데이터는 한국형 고속 전철에서 실제 사용되는 데이터를 기반으로 생성되었다. 기본주기 하나 하나를 사이클이라 부르면 그림 3 에서와 같이 분포를 고려하지 않은 경우에는 사이클에 최대 11개의 포트가 스케쥴링 되었지만, 이를 고려해 구현된 스케쥴링은 주로 4~6개로 고르게 분포하고 10개 이상의 포트가 할당된 사이클은 존재하지 않는다.

앞에서도 언급한 바와 같이 개별주기들이 가지는 제약, 즉 약수와 배수 관계는 개별주기를 거시주기에 할당하는 스케쥴링을 용이하게 해 주며 지터의 발생을 막는다. 이러한 주기, 폴링해야 할 포트, 연산자등은 테이블에 저장되고 이를 바탕으로 기본주기마다 연산자를 증가 혹은 감소시키며 각 포트의 개별주기 혹은 0일 경우 폴링하는 방법이 사용된다. 특히 일정한 분포를 가지기 위하여 각 연산자의 초기값을 다르게 사용할 수도 있다. 이러한 방법은 위에서 설명한 실행전 스케쥴링과 같은 결과를 낼 수도 있다. 또한 표 1에서 실행전 스케쥴링으로 인한 폴링리스트의 크기는, 모든 포트가 개별주기에 맞추어 메모리에 할당되므로 거시주기를 개별주기로 나눈 값의 합이 되므로, 포트를 폴링하기 위한 프레임의 크기인 2바이트를 곱하면, 2*(1024 + 2*512 + 4*256 + 6*128 + 7*64 + 10*32 + 7*16 + 6*8 + 4*4 + 2*2+1) = 9578바이트가 되지만 실행시간 스케쥴링을 위해선 단지 프레임, 주기, 연산자만을 필요로 하기 때문에 이의 크기를 곱하면 6*(1 + 2 + 4 + 6 + 7 + 10 + 7 + 6 + 4 + 2 + 1) = 300바이트의 메모리만을 사용한다. 따라서 개별주기의 최소공배수의 크기가 큰 경우 실행전 스케쥴링은 폴링리스트의 크기가 매우 커지므로 큰 메모리가 필요한 반면 실행시간 스케쥴링은 상대적으로 작은 메모리를 사용하게 된다. 이밖에 실행 중 새롭게 스케쥴링이 되어야 할 경우 실행시간 스케쥴링은 새로운 리스트를 만들지 않고 단지 테이블만 갱신을 요하므로 유연성이 높은 장점이 있다.

3.1.2 전송효율 및 전송용량

전동차용 통신 네트워크와 같이 마스터/슬레이브로 구성되어 마스터의 전송 요구에 의해서만 통신이 이루어지는 비대칭형 프로토콜은 특정 마스터가 존재하지 않는 TCP/IP와 같은 대칭형 프로토콜과는 달리 버스관리자가 보내는 마스터 프레임과 이에 응답하는 슬레이브가 보내는 슬레이브 프레임을 사용하게 되는데, 이 둘을 합쳐 텔레그램(Telegram)이라고 한다. 버스 마스터가 연속적인 폴링 한다고 가정할 경우 하나의 텔레그램 전송에 걸리는 시간은 연속적인 텔레그램의 시간 간격과 같으므로 마스터 프레임 전송시간(Master frame duration), 슬레이브 프레임 전송시간(Slave frame duration), 전송지연시간(Propagation delay), 실행시간(Procedural delay)을 더한 값이 된다. 차량간 버스는 차량내 버스에 비해 전송속도가 낮으며 프레임 크기도 크기 때문에 텔레그램 전송속도가 상대적으로 낮다. 64, 256비트 크기를 가지는 포트의 차량내 버스에서 텔레그램 전송시간은 82μs, 226μs이며 256, 1024비트 크기를 가지는 포트의 차량간 버스에서 텔레그램 전송시간은 320μs, 1180μs가 된다. 슬레이브 프레임 전송시간을 제외한 나머지 항들은 고정적이거나 상대적으로 매우 작은 값을 가진다. 총 전송시간(Total transmission time)에 대한 유용한 데이터 전송시간(Transmission time of the useful information)으로 정의되는 전송효율(Transmission Efficiency)은 바로 이 슬레이브 프레임의 길이와 데이터 집적도의 영향을 받는다.

표 2는 데이터 크기에 대한 차량내 버스의 텔레그램 전송시간과 전송효율을 같은 비대칭형 프로토콜을 사용하는 필드버스의 하나인 FIP와 비교하였다[13]. 데이터 크기가 작을 경우 차량내 버스와 FIP 모두 비교적 낮은 효율을 보이고 있지만, 이 때 유용한 데이터는 브로드캐스팅 되므로 여러 번의 전송을 필요로 하는 다른 네트워크와는 그 성격이 다르다. 여기서 계산된 전송효율은 응용계층의 데이터 영역을 고려한 것인데 실제 프로세스 데이터는 일반적으로 수 비트의 크기를 가지므로 프로세스 데이터가 저장되는 논리적 포트의 데이터구조 및 데이터 집적도를 높이는 것이 실제로 중요하다.

전송효율과 더불어 전송량은 네트워크 성능의 척도가 되며 이는 전송속도, 프레임구조, 전송 방법 등과 연관이 있다. 프로세스 데이터 서비스에서는 전송량 분석은 실행전 스케줄링을 위하여 반드시 선행되어야 하며, 특히 서로 다른 두 가지 이상의 버스상에서의 스케줄링에 유용한 자료가 된다. 전송량은 모든 주기적 전송구간에서 일정량의 주기적 데이터의 전송이 일어난다는 가정하에 식(1)과 같이 모든 데이터의 개별주기에 대한 텔레그램 전송시간의 합이 주기적 전송구간보다 작은 경우에 스케줄링이 가능하고 이 경우가 최대 주기적 전송량이 된다.

$$\sum_i \frac{T_{mm}(i)}{T_{ip}(i)} \le 1-\left(\frac{T_{spo}}{T_{bp}}\right) \qquad (1)$$

위 식에서 $T_{mm}, T_{ip}, T_{spo}, T_{bp}$는 각각 개별 데이터 전송시간, 개별주기, 기본주기 내의 산발적인 데이터를 위한 시간, 기본주기이다.

표 3과 표 4는 각각 64비트 크기를 가지는 차량내 통신버스와 256, 1024비트 크기를 가지는 차량간 통신버스에서의 개별주기와 산발적 전송구간에 따른 스케줄링 가능치를 계산한 결과이다. 스케줄링 가능치는 주어진 조건하에 전송가능한 포트의 수 이다. 표준에 따르면 차량내 버스에선 일반적으로 256개의 논리적 포트를(실제 여러 프로세스 데이터들은 하나의 포트에 할당된다) 가질 것을 권고하는데, 이를 위해서는 평균적으로 개별주기가 32ms 정도를 가지는 것이 가장 적당하다는 결론이 나온다. 또한 차량간 전송버스에서는 각 노드가 논리적인 포트를 가지지 않고 물리적인 포트만을 가지고 있는데 기본주기는 25ms로 정해져 있다. 만약 노드의 개별주기가 25ms이고 포트의 크기가 1024bit라

표 2 차량내 통신버스의 전송효율
Table 2 Transmission efficiency of MVB

Data Size	차량내 버스(1.5Mbps)			FIP (1Mbps)
	Telegram Duration (μs)	Transmission time of information (μs)	Efficiency (%)	Efficiency (%)
2	50	10.67	21.34	9.75
4	61	21.33	34.97	17.77
16	130	85.33	65.64	46.37
32	226	170.67	75.52	63.36

표 3 차량내 통신버스의 스케줄링 가능치
Table 3 Schedulability in MVB

포트개수 (개)		산발적 전송구간 / 기본주기				
		0.3	0.35	0.4	0.45	0.5
개별주기 (ms)	1	9	8	7	7	6
	2	17	16	15	13	12
	4	34	32	29	27	24
	8	68	63	59	54	49
	16	137	127	117	107	98
	32	273	254	234	215	195
	64	546	507	468	429	390
	128	1093	1015	937	859	780
	256	2185	2029	1873	1717	1561
	512	4371	4059	3746	3434	3122
	1024	8741	8117	7493	6868	6244
	2048	17483	16234	14985	13737	12488
	4096	34966	32468	29971	27473	24976

표 4 차량간 통신버스의 스케줄링 가능치
Table 4 Schedulability in WTB

포트개수 (개)		산발적 전송구간 / 기본주기				
		0.3	0.35	0.4	0.45	0.5
포트 크기 (bit)	256	54	50	46	42	39
	1024	14	13	12	11	10

면 10~14개의 노드를 위한 전송만 가능하게 된다. 차량내 프로세스 데이터에서 평균 개별주기가 32ms인 경우 약 6.25%만이 차량간 통신버스를 통할 수 있게 된다. 이러한 이유로 차량내 버스 보다 상당히 낮은 전송량을 가지는 차량간 통신버스의 부하가 예상된다. 또한 차량간 통신버스를 경유하는 경우 차량내 통신버스간 최대통신지연시간(Worst case transmission time)은 각 버스에서의 개별주기의 합과 일치하게 된다.

3.2 메시지 데이터 서비스

3.2.1 트리 알고리즘

주기적 전송구간 사이의 산발적 전송구간에서는 매체 엑세스 제어 계층의 프로토콜에 따라 그 정성적인 성능이 결정된다. 트리 알고리즘은 각 스테이션의 주소에 의한 이진 탐색 트리(Binary Search Tree)를 이용한 것으로, 차량내 버스의 산발적 전송구간에서 발생할 수 있는 충돌(Collision)을 해결하기 위한 매체 액세스 제어 프로토콜로 사용되고 있다. 그림 4는 8개의 디바이스를 대상으로 한 트리 알고리즘을 보여준다. 버스 마스터는 처음 모든 디바이스들을 대상으로 폴을 하고 충돌이 있을 시 반으로 나눈 그룹으로 다시 폴을 한다. 예를 들어 000과 010 이라는 디바이스 어드레스를 가지는 디바이스들이 동시에 메시지를 가지고 있었다면, 충돌은 그림 4에서 최초 한번과 xx0으로 폴할 경우 2번이 발생한다. 이와 같은 알고리즘의 단점은 충돌을 해소하기 위한 제어 프레임의 개수가 많음으로써 대칭형 프로토콜의 매체 액세스 제어 보다 낮은 효율을 가지고 있다는 점이다. 반면 하나의 세그먼트에 최대 32개의 디바이스를 지원하는 차량내 버스에서는 트리 레벨(Level)이 최대 5이기 때문에 5회 이상의 연속적인 충돌은 발생하지 않는다는 장점이 있다. 보다 나은 성능을 위하여 탐색 경로 설정, 어드레싱, 적응성 탐색 등을 고려할 수 있다. 탐색 경로의 개선의 예는 그림 4에서와 같이 무응답일 경우 하나의 레벨을 낮추어 찾는데, 만약 어드레스의 최상위 비트만 다른 111과 011 디바이스에서 이벤트가 발생하였다면, 9개의 텔레그램을 7개의 텔레그램으로 줄이는 성능향상이 있게 된다.

또한 레벨 n의 트리에서 만약 2개의 디바이스에서 산발적인 데이터 전송 이벤트 발생 시 필요한 텔레그램의 최소값은 레벨 0에서 분기되는 경우이고 최대값은 레벨 n에서 분기되는 경우이다. 따라서 필요한 텔레그램 수의 최대값을 낮추기 위해서는 가능한 밸런스형 트리 구조를 가져야 하는데 이것은 그룹어드레스의 마스크 형식의 영향을 받는다. 즉 각 레벨에 따른 최하위 비트로 각 분기점에서 나누어지므로 연속적인 어드레싱, 예를들어 0,1,2,3.... , 방법이 밸런스형 트리를 구축하는 방법이라고 할 수 있다. 특히 자주 이벤트가 발생하는 두 디바이스에 대해서 최하위 비트만 다른 어드레싱은 충돌을 최소화 시켜준다.

이와 같은 어드레싱은 정적인 기법인데 반해 적응 기법은 실행 중에 네트워크 트래픽에 의거한 동적인 방법이다. 트리 구조를 감안하면 충돌해소주기에서 모든 노드의 전송 이벤트가 발생하였을 경우 노드를 찾는 시간이 약 절반을 차지하게 된다. 만약 n개의 노드에서 산발적인 데이터 전송 이벤트의 발생이 있다고 가정하면, 적응기법은 레벨 0에서가

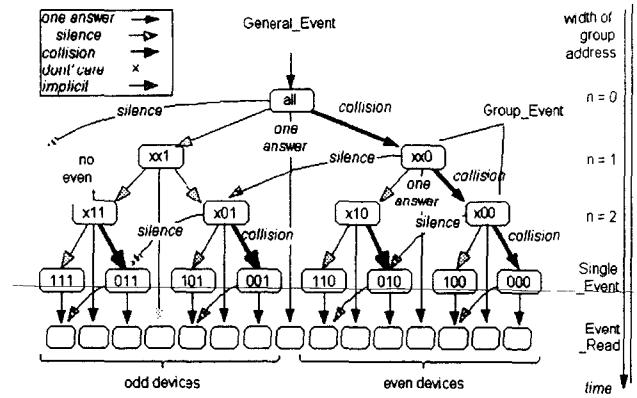


그림 4 이벤트 중재 트리

Fig. 4 Event arbitration tree

아니라 트래픽에 맞는 레벨에서 이벤트 제안이 시작되므로 레벨 0에서의 시작으로 인한 필연적인 충돌을 피할 수 있게 된다. 레벨 i에서 가질 수 있는 노드는 2의 i제곱승이 되므로 $i = \log_2 n$ 과 같은 레벨이 가장 적합하게 된다. 일반적인 트래픽 예측방법은 전 충돌 해소 주기동안 전송한 노드 수를 택하는 것이다. 그러나 이러한 방법은 모니터링을 통한 트래픽을 예측해야 하는 전제조건과 이벤트 발생 수가 작은 경우 무효한 단점이 있다. 이러한 방법은 실제 대칭형 프로토콜에서는 모든 노드의 동기화 등으로 인한 문제로 적용이 어려우나 비대칭형은 차량내 버스에서는 마스터에서만의 구현으로 가능하게 된다.

3.2.2 모델링 및 시뮬레이션

데이터링크 계층 하에서의 동작 특성이 확정론적(Deterministic)인 프로세스 데이터 서비스는 앞에서와 같은 분석과 계산에 의한 성능분석이 가능하지만, 확률론적(Stochastic)인 메시지 서비스의 동작 특성은 앞에서와 같은 방법으로는 성능분석이 불가능하다. 따라서 본 논문은 전동차용 통신 네트워크의 비실시간 통신 성능분석을 위하여 상용 시뮬레이션 툴인 Arena를 사용하여 동적 이산 확률론적인 큐 모델(Dynamic Discrete Stochastic Queuing Model)을 개발하였으며 시뮬레이션을 수행하였다.

메시지 데이터의 전송지연 시간은 클라이언트 응용프로세스로부터의 전송요구 시점에서 서버 응용프로세스까지의 도착하는 시점까지의 경과시간을 의미한다. 따라서 전송지연시간은 패킷화, 흐름제어, 라우팅 등에 의한 데이터처리 시간과 전송큐에서의 대기시간 그리고 물리적 전송시간으로 나눌 수 있다. 개발된 모델은 차량내 통신버스와 차량간 통신버스의 데이터링크 계층의 하위 부분 위주로 모델링 하였다. 그러나 이러한 방법은 전송큐의 대기시간과 물리적 전송시간을 고려한 것이며 간단 명료한 모델링을 위하여 다음과 같은 가정을 하였다.

- 모든 전송 실패는 없다.
- 전송 측과 받는 측의 데이터 처리 시간은 고려하지 않는다.
- 전송되는 프레임에는 제어프레임이 없다.
- 항상 트랜스포트 레벨에서의 연결은 유지되며 일정한

윈도우크기를 갖는다.

- 모든 메시지데이터의 크기는 일정하므로 동일한 패킷을 발생시킨다.
- 전송되는 프레임의 크기는 같으며 물리적 전송시간은 동일하다.
- 라우팅 시간은 일정하다.

개발된 모델은 각 버스의 매체 제어 엑세스 프로토콜을 모델링한 논리적 제어부(Logical Control Part)와 실제 데이터를 생성하는 물리적 스테이션부(Physical Station Part)로 구성되어 있다[10]. 각각의 부분은 Arena[12]에서 제공하는 기본적 모듈을 조합하여 구성된 템플릿들로 이루어져 있다. 이러한 템플릿을 기본 단위로 하여 모델이 개발되었으므로 모델을 변경, 확장하기 쉽다. 그림 5와 그림 6과 같이 기본 구조는 논리적 제어 기본 템플릿 아래에 물리적 스테이션 기본 템플릿이 형성되어 있는 것인데, 차량내 통신 버스는 트리 구조로 버스 논리적 제어 기본 템플릿이 놓여 있고, 차량간 통신 버스는 라운드 로빈(round robin)방식을 위해 체인 형태로 연결되어 있다. 각 그림에서 물리적 스테이션 기본 템플릿과 점선으로 연결된 논리적 스테이션 기본 템플릿이 네트워크상의 하나의 스테이션이 된다. 그림 5와 그림 6의 기본 템플릿을 요약하면 다음과 같다.

- 차량내 버스 논리적 제어 기본 템플릿(MLCBT; MVB Logical Control Basic Template): 트리 알고리즘에서 트리 하나의 노드를 의미하는 템플릿. 따라서 8개의 스테이션을 제어하기 위해 이러한 템플릿이 15개가 필요하다.
- 차량내 버스 물리적 스테이션 기본 템플릿(MPSBT; MVB Physical Station Basic Template): 메시지를 생성하고 전송큐에 저장하고 있는 차량내 통신버스의 한 스테이션 기능을 하는 템플릿. 8개의 차량내 스테이션이 존재하므로 이러한 템플릿이 8개가 존재하며 다른 스테이션과는 독립적으로 작동된다.
- 차량간 통신버스 논리적 제어 기본 템플릿(WLCBT; WTB Logical Control Basic Template): 차량간 통신버스의 매체 엑세스 제어 계층의 기능을 수행하는 템플릿. 여기서 첫 번째 시간 지연은 엔터티의 무한한 순환을 막기 위하여 차량간 통신버스의 텔레그램 전송시간의 작은 부분을 할당한다. 8개의 노드를 대상으로 하였으므로 이러한 템플릿이 8개가 사용되었다.
- 차량간 버스 물리적 스테이션 기본 템플릿(WTB Physical Station Basic Template): 메시지를 생성하고 전송큐에 저장하고 있는 차량간 통신버스의 한 노드의 기능을 하는 템플릿. 두 버스를 연결시키는 모듈, 애니메이션 및 카운터 및 기록 모듈들을 이용하여 8개의 노드를 가지는 차량간 통신버스와 8개의 스테이션을 가지는 차량내 통신버스 시스템을 모델링 하였다.

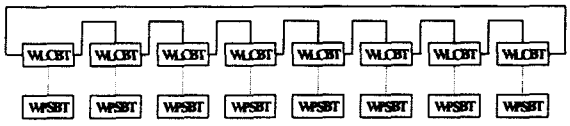


그림 5 간략화된 WTB 모델링

Fig. 5 Simplified WTB simulation model

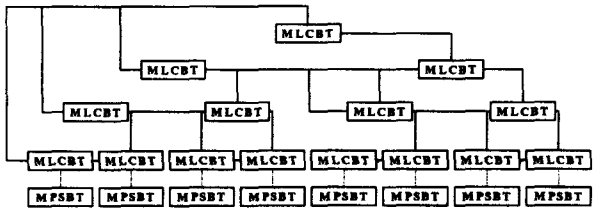


그림 6 간략화된 차량간 통신 버스 모델

Fig. 6 Simplified MVB simulation model

표 5 시뮬레이션 파라미터

Table 5 Parameters for computer simulation

파라미터	차량내 통신 버스	차량간 통신 버스
메시지 발생 간격	평균값 10, 20, 30, 40, 50ms의 확률 지수분포	
메시지 길이(bit)	256	256
패킷 길이(bit)	64	256
윈도우 크기	4	1
스테이션 수	8	8
기본 주기(ms)	4	25
산발적 전송구간의 비율(%)	40	40

시뮬레이션에서 적용된 각 버스에서의 텔레그램 전송시간, 기본주기, 스테이션 수, 메시지 크기는 변동이 없으며 표준에서 정의한 사양 혹은 계산에 의하여 주어졌다. 반면 각 스테이션의 메시지 발생빈도, 산발적 전송구간, 시뮬레이션 시간은 변수로 사용되었다. 각각의 시뮬레이션 결과는 변수와 모델의 변화에 따른 것이며, 본 시뮬레이션 수행에서 사용된 파라미터를 표 5와 같이 나타내고 있다.

3.2.3 전송지연

그림 7은 각 스테이션은 동일한 분포의 패킷 발생에서 트리 알고리즘이 가지는 이진 트리 형식으로 인하여 각 스테이션의 어드레싱 방법이 미치는 영향을 나타낸 것이다. 트리를 검색할 때 어드레스의 최하위 비트값이 0인 경우를 먼저 검색하기 때문에 홀수의 어드레스를 가지는 디바이스가 짝수의 어드레스를 가지는 디바이스보다 지연시간이 길게 나타난다. 그러나 디바이스의 주소에 따라 패킷의 큐에서의 대기시간은 수십μs의 차이를 보이긴 하지만 패킷의 발생 빈도에 더 민감한 것으로 나타난다.

그림 8은 항상 전체 이벤트 제안으로 시작하는 트리 알고리즘과 네트워크 트래픽을 고려한 적응기법을 적용한 경우 4개의 패킷으로 나누어지는 메시지의 전송지연 시간을 나타낸 것이다. 트래픽에 따른 초기 시작 노드를 앞 절에서 구한 값으로 정하였을 경우 전체적으로 약 1ms 정도의 전송지연 효과를 나타낸다. 그러나 본 시뮬레이션에서는 현재의 네트워크의 트래픽을 고려한 것으로 실제는 결과 보다 낮은 효과를 나타낼 것으로 예상된다. 따라서 과거의 트래픽을 고려하여 현재의 트래픽을 예측하는 성능에 따라 이 적응기법의 효과가 달라지게 된다. 또한 여러 패킷을 가지는 메시지의 경우 각 스테이션이 가지는 어드레스로 인한 효과는 무시할 정도로 나타나고 있다.

그림 9는 차량내 통신버스에서 메시지의 통신지연을 분석한 것이다. 메시지 발생시간 간격이 일정시간보다 짧아지면 공통적으로 통신지연이 급격히 늘고 있음을 알 수 있다. 위의 실험 조건에서, 차량내 통신버스에서 메시지 발생간격이 약 15ms보다 적은 경우에는 평균 지연 시간 보다 작게 되어 전송되는 데이터의 양보다 데이터의 생성이 많게되므로 전송큐의 범람이 예상된다. 만약 실제 네트워크 구성 시 이러한 전송큐의 범람이 예상된다면 차량내 통신버스를 분리하는 등의 노력이 필요하다. 또한 약 30ms를 넘는 경우에는 지연시간 감축 폭도 작아져서 일반적으로 5ms안에 전송된다는 것을 알 수 있다.

차량간 통신 버스에서의 전송지연은 차량내 통신버스보다 큰 것으로 나타나고 있는데 실제 이보다 더 큰 문제는 부착된 노드의 수가 가변적이라는 것이다. 이러한 요소가 네트워크 환경설정 및 성능예측에 어려움을 더하고 있는데 그림 10은 차량간 통신 버스에서 노드의 변화에 따른 전송시간을 나타낸 것이다. 기본 주기 25ms의 60%인 10ms를 산발적 전송구간으로 사용할 때 약 31개의 256비트 패킷들을 전송할 수 있다. 따라서 평균 25ms의 패킷의 발생인 경우 노드의 개수가 32개일지라도 큰 전송지연의 차이를 보이지 않고 있다. 반면 평균 10ms의 패킷 발생 빈도를 갖는 노드들인 경우 패킷의 전송지연이 노드 개수에 민감한 반응을 보이고 있다.

그림 11은 하나의 차량내 통신버스 스테이션에서 차량간 통신버스를 통하여 다른 차량내 통신버스 스테이션으로 메시지 전송 시 걸리는 시간 지연이다. 이러한 경우 전송지연의 상당부분을 차량간 통신버스에서 발생한다. 이러한 이유는 차량간 통신버스의 텔레그램 전송시간이 큰 이유에서 있기도 하나 대부분 기본주기가 길어서 나타나는 현상이다. 두 버스에서 각 스테이션의 메시지 발생시간 간격이 20ms 이하인 경우 대부분의 전송지연시간은 거의 일정하게 20~40ms범위를 나타내고 있다.

4. 차량내 통신 버스 구현 및 실험

현재 한국형 고속전철의 시제차에 탑재용으로 개발되고 시험중인 차량내 통신 버스 제어기인 IMC(Intelligent MVB Controller)는 모토로라의 고성능 통신 컨트롤러인 MC68360을 기반으로 하며 산업 표준형인 VMEbus를 통하여 주제어기와 연결된다. IMC의 CPU인 MC68360은 25MHz의 클럭 스피드로 동작하고 프로그램의 수행을 위해 보드상의 2Mbyte의 스태틱 메모리(Static Memory)와 512Kbyte의 플래쉬 메모리(Flash memory)를 이용한다. 차량내 통신 버스는 전용 컨트롤러인 MVBC가 담당하고 데이터의 저장, 컨트롤 비트 등의 저장 등을 위해 256Kbyte로 구성된 전용의 트래픽 메모리(Traffic Memory)를 이용한다. 이렇게 충분한 트래픽 메모리의 확보로 IEC에 정의한 클래스 4의 차량내 통신 버스 제어기의 기능을 무리 없이 수행할 수 있다. 탑재된 MVB 펌웨어(firmware)의 안정성과 정확한 결과가 요구되는 정확한 시간까지 제공되어야 하는 실시간 응용 시스템을 개발하기 위하여 실시간 운영체제인 VRTX/sa를 탑재하였다.

프로토콜 스택은, 그림 12와 같이 디바이스의 기능 즉, 마

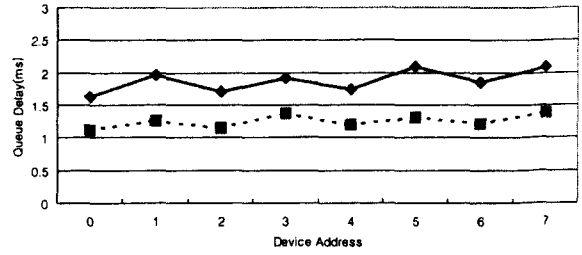


그림 7 어드레스에 따른 패킷의 큐에서의 지연시간
Fig. 7 Queueing delay of packet according to address

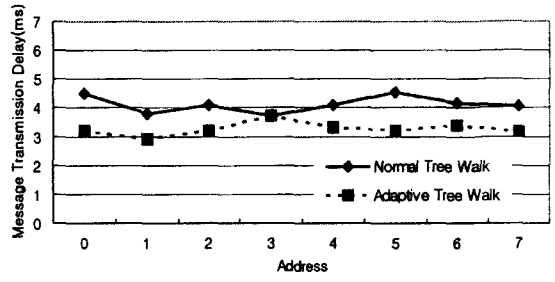


그림 8 적응기법에 의한 성능향상
Fig. 8 Performance using adaptive tree algorithm

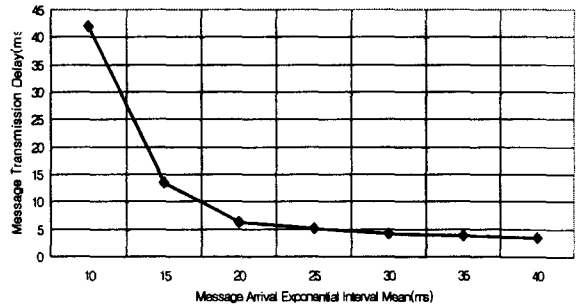


그림 9 메시지 발생에 따른 차량내 버스의 전송지연
Fig. 9 Message transmission delay on MVB

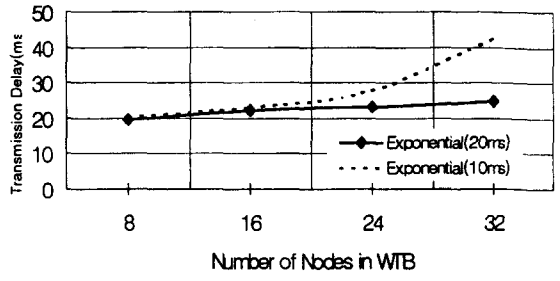


그림 10 차량간 통신버스에서의 메시지 전송지연
Fig. 10 Message transmission delay on WTB

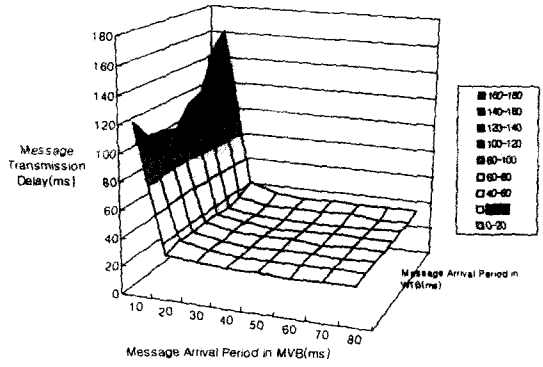


그림 11 차량내 통신버스 사이에서 메시지 평균 지연시간
Fig. 11 Average message transmission delay

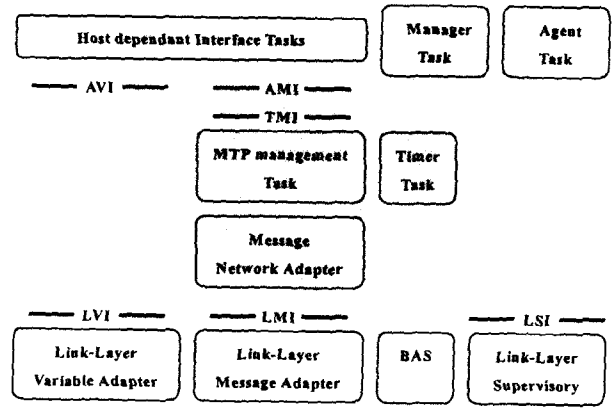


그림 12 구현된 소프트웨어 프로토콜 스택
Fig. 12 Implemented software protocol stack

스터 혹은 슬레이브에 따라 7에서 10개의 태스크들로 구성되어 있다. 각 태스크의 기능은 다음과 같다.

- Link-Layer Variable Adapter task (LVA): 프로세스 데이터 서비스를 위하여 최대 4096개의 논리적 포트와 관련된 프리미티브(primitive)들을 수행하는 태스크로서 각 데이터에 대한 포트 할당, 집적화, 리프레쉬 타임 계산을 담당한다.
- Link-Layer Message Adapter task (LMA): 메시지 데이터 서비스를 위하여 전송 큐(Queue)와 관련된 프리미티브들을 수행하는 태스크로서 링버퍼(Ring-buffer) 구조를 가진 전송을 하기 위한 큐와 전송을 받기 위한 큐를 관리한다.
- Link-Layer Supervisory Adapter task (LSA): 관리용 데이터 서비스를 위하여 물리적 포트와 관련된 프리미티브들을 수행하는 태스크로서 각 디바이스의 상태와 스케쥴링을 위한 정보를 교환하는 역할을 담당한다.
- Bus Administrator Software task: 산발적 전송구간에서 이벤트 중재를 위해 마스터 디바이스에서 실행되는 태스크로서 트리 알고리즘을 이용한다.
- Message Network Adapter task (NMA): 인바운드(Inbound), 아웃바운드(Outbound) 패킷의 전송 및 패킷라우팅을 담당한다.
- Message Transport Protocol State Management task (MTP): 트랜스포트 흐름제어를 위한 상태 머신 관리 태스크로서 사용자, 네트워크, 혹은 타이머로부터의 이벤트에 대한 처리와 상태 천이를 일으킨다. 기본적으로 연결을 관리하고 윈도우 슬라이딩 프로토콜에 의한 패킷전송, 재전송을 담당한다. 지금까지는 단지 한 순간에 하나의 연결만 지원하고 있다.
- Timer task : 시간경과에 따른 이벤트(Time-out)를 발생시키는 태스크로 여기서는 주어진 시간에 하나의 타이머만 작동한다.
- Manager task: MVB관리 및 환경설정을 위한 태스크로서 각 디바이스 상태를 점검하며 에이전트와의 통신과 더불어 앞장에서 소개한 방법으로 스케쥴링을 담당한다.
- Agent task: 디바이스 관리 및 환경설정을 위한 태스크로서 자신의 디바이스 상태 점검과 스케쥴링을 위한 포트정보를 매니져에게 알려준다.
- Host dependant interface tasks: 호스트 인터페이스를 위한 태스크들로서 VME버스를 인터페이스 하기 위한 듀얼포트램을 관리하는 데몬(Daemon)을 비롯하여, 이로부터 생성되는 서브 태스크들과 시리얼 인터페이스를 위한 데몬이 있다.

차량내 통신버스를 통하여 하나의 제공자 세션에서 다른 소비자 세션으로의 100바이트의 메시지 전송 시 윈도우 크기 4의 슬라이딩 윈도우 동작을 시험해 본 결과 패킷의 충돌이 없는 상황에서 5개의 패킷을 전송하는데 약 100ms 정도 소요 되었다. 구현된 IMC360상에서의 가장 큰 전송지연 요소는 충돌해소주기 동안 각 디바이스의 늦은 반응이었으며 시뮬레이션 상에서 네트워크 부하가 작은 경우 5ms이하가 되었는데, 충돌이 없는 경우 연결을 위한 6개, 데이터 전송을 위한 15개, 응답을 위한 6개, 즉 약 27개의 텔레그램이 필요하게 된다. 이는 약 1.8ms의 산발적 전송구간을 필요로 하며 데이터 처리 시간을 감안할 때 시뮬레이션의 결과와 일치하게 된다.

5. 결 론

전동차용 통신 네트워크 프로토콜은 전동차 차량 시스템에 있어서 기간망과 같은 역할을 담당하며 차세대 표준 인터페이스로서 차량 탑재용 전자기기 설계에 큰 영향을 미칠 것으로 예상된다. 본 논문은 이러한 네트워크의 환경구축 및 응용프로세스 개발을 위하여 프로토콜의 구현 및 시뮬레이션에 의한 실시간성과 성능을 분석하였다. 분석 결과에 따르면 논문에서 제안된 스케쥴링 방법을 사용한 차량내 버스통신은 수ms 단위의 실시간 통신을 지원하며 실시간 분산 시스템에 적합한 특성을 가지고 있다. 그러나 차량간 통신은 상대적으로 많은 제약과 낮은 성능을 가지고 있는 것으로 판단된다. 또한 시간적 제약성보다는 신뢰성 있는 메시지의 전송지연시간을 단축하기 위하여 어드레싱기법, 적응기법을 제시하고 이에 대한 효과를 알아보기 위하여 시뮬레

그림 13 개발된 차량내 통신버스 제어기
Fig. 13 Intelligent MVB Controller

이션 모델을 개발하였다. 구현된 컨트롤러 상에서의 실험은 아직 미비한 점이 있으나 그 유용성을 확인할 수 있었다. 본 논문에서는 주로 네트워크 성능에서 주된 영향을 주는 데이터 링크 이하 계층을 위주로 연구하으나 이밖에 응용프로세스와의 동기화와 고장 포용성, 라우팅, 다중캐스팅과 다중연결, 응용계층의 개발과 네트워크 관리기법에 관한 연구가 절실히 요구되고 있다.

감사의 글

본 연구는 1997년도 인하대학교 교내연구비 지원의 지원에 의하여 이루어진 연구입니다.

참 고 문 헌

[1] IEC 1375 Standard, Train Communication Network: Part(1)General Architecture (2)Real-Time Protocol (3)Multifunction Vehicle Bus (4)Wire Train Bus (5)Train Network Management (6)Train Communication Conformance Testing, 1996.

[2] P.Etter and H.Kirrman, "Use of standardized integrated communication systems on vehicles and trains", Proceedings of the International Conference on Speedup Technology for Railway and Maglev Vehicles, Vol.1, 1993.

[3] H.Kirrman and P.A.Zuber, IEC/IEEE Train Communication Network, 1996.

[4] G.Cau and Gianfranco, "From the research to the applications of a new concept of data communication infrastructure on board of rolling stocks", World Congress on Railway Research 97, 1997.

[5] G.Fadin and F.Cabaliere, "IEC TC9 WG22 Train Communication Network Dependability and Safty concepts", World Confress on Railway Research 97, 1997.

[6] F.Vasques and G.Juanole, "Pre-Run-Time Schedulability Analysis in Fieldbus Networks", IEEE, 1994.

[7] S.H.Hong, "Scheduling Algorithm of Data Sampling Times in the Integrated Communication and Control Systems", IEEE Trans. Control System Technology Vol.3, 1995.

[8] Y.S.Kim and W.H.Kwon, "Pre-runtime Scheduling Methods of Data Link Layer in the Fieldbus environment", Workshop on Distributed Computer Control Systems, 1997.

[9] S.Cavalieri, A.D.Stefano and O.Mirabella, "Pre-run-time Scheduling to Reduce Schedule Length in the Fieldbus Environment", IEEE Trans. on Software Engineering, 1995.

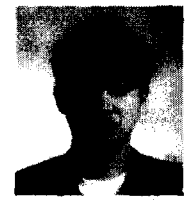
[10] J.H.Park and S.C.Lee, "Performance Evaluation of the Train Communication Network", Workshop on Distributed Computer Control Systems, 1998.

[11] C.M.Krishna and K.G.Shin, Real-time Systems, Mcgraw-Hill, 1997.

[12] W.D.Kelton, R.P.Sadowski and D.A.Sadowski, Simulation with Arena, Mcgraw-Hill, 1998.

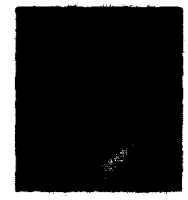
[13] EN50170 standard, The WorldFIP protocol, 1996.

저 자 소 개



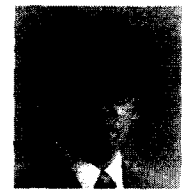
이 상 철 (李 尙 哲)

1973년 12월 5일 생. 1997년 인하대 자동화공학과 졸업, 1999년 동 대학원 석사. 1999년~현재 (주)로커스 정보통신 연구소 연구원.
Tel : 3149-9281, Fax : 3149-9148,
E-mail: sclee@locus.co.kr



박 재 현 (朴 宰 賢)

1963년 10월 8일 생. 1986년 서울대 제어계측공학과 졸업, 1988년 동 대학원 석사, 1994년 동 대학원(공박). 1995-현재 인하대 부교수. 관심 분야는 실시간 시스템, 컴퓨터 네트웍, 내장형 시스템, 이산현상 시스템.
Tel : (032) 860-7713, Fax : (032) 863-4386
E-mail: jhyun@inha.ac.kr



장 래 혁 (張 來 赫)

1965년 3월 19생. 1989년 서울대 제어계측공학과 졸업. 1992년 동 대학원(석사). 1996년 동 대학원(공박). 1997-현재 서울대 컴퓨터공학과 전임강사. 관심분야는 디지털 시스템 설계, 내장형 실시간 시스템, 이산현상시스템.
Tel : 880-1834, Fax : 886-7589
E-mail: naehyuck@snu.ac.kr