工學碩士 學位論文

분산된 ECU간의 Fault-Tolerance한 통신 방식에 관한 연구

指導敎授 邊 基 植

이 論文을 工學碩士 學位論文으로 提出함

2005年 2月

釜 慶 大 學 校 大 學 院

制 御 計 測 工 學 科

金 成 勳

金成勳의 工學碩士 學位論文을 認准함

2004年 12月 23日

主 審 工學博士 黃 龍 淵

委 員 工學博士 金 萬 高

委 員 工學博士 邊 基 植

목 차

Study for the Fault-Tolerant communication method between the Distributed ECUs

Sung Hoon Kim

Department of Control and Instrumentation Engineering

Graduate School of

Pukyong National University

Abstract

Communication between the distributed ECUs(Electronic Control Units) of the safety-critical application have to meet fault-tolerant communication protocol system. Because commercial in-vehicle communication protocol methods(e.g., CAN, LIN, etc) by this time can not implement fault-tolerant communication systems, this research suggests the TTP/C(Time-Triggered Protocol Class C) communication system. This paper is composed of the concepts and the principles of the TTP/C, and followed with hardware design for the TTP/C communication protocol module and its operation program. It is showed that the important feature of TTP/C is its ability to guarantee that no single failure of a

node can disturb the communication of the remaining nodes in the cluster. Futhermore, it is proved that two serial communication redundancy channels can be utilized for fault-tolerant to ensure that critical messages arrive even in presence of a broken channel.

제1장 서 론

현대 산업의 발전에 따라 제품의 안전성(Safety), 편의성(Convenience), 신뢰성(Reliability)등에 대한 요구가 증가되고 있다. 그 결과로 산업현장에서 사용되는 많은 기계 시스템들을 효율적으로 제어하기위해서, 보다 많은 전자제어시스템(ECU: Electronic Control Unit)의 적용 비율이 점차적으로 증가되고 있다[1],[2]. 예를 들어 자동차의 경우 ECU를 사용하게 되면 운전자의 운전습관을 습득하여 RPM(Revolution Per Minute)에 맞는 적정 연료를 계산해서 분배하고, 연료 차단을 명령하여 RPM이 레드-존(Red-Zone)으로 가면 연료를 차단하여 엔진을 보호하거나 연비 향상에 기한다[2].

그러나 지금까지의 전자제어시스템들은 대부분 독립적으로 운용되어지므로, 각각의 전자제어시스템들에게 개별적인 제어로 인한 문제로 많은 비용과 설계시간이 투자되었고, 시스템의 고장에 대처하기위한 결함 허용성(Fault-Tolerance)을 보장하기가 어려웠다[1],[4],[5]. 최근에는 이러한 단점을 보완하고 신뢰성 확보를 위해서 여러 개의 소규모 전자제어 시스템들을 분산시켜 통신으로 정보를 공유할 수 있는 네트워크기반의 대규모 전자제어 시스템이 적용되는 추세이다[1],[2],[3]. 이러한 네트워크기반에서 운용되는 분산 제어시스템들은 이전의 독립적인 제어시스템들보다 더욱 적은 비용과 설계시간이 투자되게 되었고, 신뢰성 향상과 결함 허용성을 더욱 보장할 수 있게 되었다[1],[4],[5].

오늘날 분산된 전자제어 시스템간의 통신 프로토콜 규격은 안전성(safety)과 유용성(availability), 그리고 구성능력(composability)에 대한 강력한 요구조건을 가지고 있는 분산 실시간 시스템(distributed real-time system)을 목표로 하여 이벤트 트리거 (Event Trigger) 방식과 시간 트리거 (Time Trigger)방식으로 이루어진다[5],[6],[7][8],[12].

이벤트 트리거 방식은 전송되는 메시지가 시간의 흐름에 무관한 이벤트적인 메시지(event message)이며, 예측할 수 없는 사건으로부터 얻어진 시간적인 제어이다. 또한 메시지의 정해진 전송시간과 실제 메시지 전송시간 사이의 차이인 지터(Jitter)가 아주 크며, 드문드문 발생하는 이벤트적인 데

이터에 대해서 양호한 시스템이라 할 수 있겠다[7],[8]. 이에 반해 시간 트리거 통신방식은 전송되는 메시지가 시간의 흐름에 따른 상태를 담고 있는 상태 메시지(state message)이며, 예측 가능한 시간의 흐름으로부터 얻어진 시간적인 제어이다. 또한 지터(대략 10us 이하)가 거의 발생하지 않으며, 정규적이고 순환적인 데이터에 대해서 양호한 시스템이라 할 수 있겠다[7],[8].

이벤트-트리거 방식에 따른 통신 기술은 CAN(Controller Area Network)[9], LIN(Local Inter-Connect Network)[10], MOST(Media Oriented System Transport)[11]등이 있는데, CAN통신은 정보를 주고받는 전자 부품 및 기기를 위한 고속의 직렬통신버스 기술로 단순한 인터페이스와 범용성 때문에 대부분의 자동차 회사들에 선호되고 있다. 하지만 CAN 통신은 더 많은 대역폭을 요구하는 응용제품에는 부적합하다[9],[16]. 그리고 LIN통신은 저속의 단일 유선 시리얼 통신 방식으로, 자동차 ECU의 경우, 선루프, 전자 미러, 와이퍼 등의 그다지 속도를 요구하지 않는 간단한 아날로그 응용 제어에 적합한 통신방식으로 비용이 적게 드는 장점이 있다[10]. 또한 인포테인먼트(infotainment)를 위한 MOST 통신은 고리형의 네트워크 시스템으로 광섬유를 이용해서 최대속도 25Mbps에 이르는 복잡한 인터페이스 통신방식으로 일반적으로 시스템 제어용 정보보다는 텔레메틱스(Telematics)등의 정보처리를 위한 통신방식으로 적용되고 있다[11].

그러나 이벤트 트리거 통신방식의 통신 시스템으로는 결함허용기능을 구현하는 것에 한계가 있기 때문에 시간 트리거 통신방식이 많이 연구되고 있다. 현재까지 연구되어온 시간 트리거 통신방식으로는 FlexRay와 본 논문에서 제안하는 TTP(Time Triggered Protocol) 통신 기술이 제시되고 있다[6],[13].

TTP는 Hermann Kopetz에 의해 고안된 통신 프로토콜로서 미국 자동차 공학회(SAE)의 클래스 C 애플리케이션에 대한 요구사항을 충족시키는 하드 리얼-타임 통신시스템(Hard Real-time Communication System)이다[5],[6],[7][17]. 또한 TTA(Time-triggered Architecture)의 기반으로 설계되어졌기 때문에 지터가 거의 발생하지 않는다[8],[12]. 그리고 다양한 애플리케이션 서비스를 제공하는데 그중에 멤버십 서비스(membership service)는

각 통신 채널상의 노드들이 나머지 노드들의 상태를 항상 파악할 수 있게 하여 보다 효율적으로 고장(failure)에 대처하도록 하고 있다[6],[7]. 또한 기존의 네트워크 시스템 중에서 가장 안정적으로 결함허용의 기능을 지원할 수 있는 프로토콜이므로 높은 신뢰성과 실시간 제어가 요구되는 Safety-critical 시스템(항공시스템, 원자력시스템, 고속열차 시스템, 자동차 X-By-Wire 시스템) 등에 적용되고 있다[6],[7].

본 연구에서는 분산된 ECU들 사이의 결함허용 통신방식의 구현을 위해서 시간 트리거 통신방식인 TTP 통신 시스템을 제안하고, 통신 모듈의 제작 및 통신운영 방식에 대해서 기술하며, 제작된 TTP 통신모듈간의 하드웨어적인 결함과 소프트웨어적인 결함 상황에서 결함을 허용할 수 있는 통신 시스템임을 실험으로 입증한다.

제2장 TTP(Time-Triggered Protocol)

2.1 TTP의 개요

현재의 프로토콜 규격은 자동차와 전자 항공 분야에서 안정성(safety), 유용성(availability), 구성성(composability)에 대해 강력한 요구사항을 지닌 분산 실시간 시스템(distributed real-time system)에 초점을 두고 있다.

TTP(Time Triggered Protocol)는 결함허용(fault-tolerance)의 기능을 지닌 분산 실시간 시스템의 전기적 모듈들(e.g., ECUs)의 상호연결을 위한 실시간 통신 프로토콜이며, 오스트리아의 빈 공과대학에서 고안된 새로운 형태의 통신 프로토콜 방식이다. 또한 TTP는 TT(Time Triggered) 방식의 하드 리얼타임 시스템의 수행을 위해서 설계되었고, 자동차 공학회(SAE)의 Class C 애플리케이션을 충족시킨다.

TTP는 메시지 전송 시에 지터(jitter: 정해진 전송 시간과 실제 전송시간 상의 차이)를 거의 발생시키지 않고, 짧은 잠복기(low latency)를 가진다. 또한 하나의 노드(ECU)가 장애(failure)가 발생하더라도 나머지 다른 노드들은 정상적으로 작동을 하는 결함허용(fault-tolerance)의 기능을 가지며, 강인한 오류감지(error detection)능력과 회복(recovery)기능을 제공한다. 그리고 각각의 노드들이 다른 나머지 노드들의 통신 상태를 항상 알 수 있게 하는 멤버십 서비스(Membership Service)를 제공한다.

TTP는 클래스에 따라 TTP/A와 TTP/C로 분류한다. TTP/A 프로토콜은 결함허용(fault-tolerance)의 기능이 없는 필드 버스(field bus) 애플리케이션을 위해 고안된 다소 범위가 축소된 버전이며, 저가의 마이크로컨트롤러에서 볼 수 있는 표준 UART(Universal Asynchronous Receiver/Transmitter) 포트와 로컬 실시간 클록만을 필요로 한다. 또한 TTP/A 프로토콜 로직 회로는 마이크로컨트롤러의 소프트웨어에서 수행되어진다. 또 다른 하나는 TTP/C 프로토콜로서 결함허용(fault-tolerance)의 기능을 지닌 분산 실시간 시스템의 실행을 위해 필요한 모든 서비스를 제공하는 버전이다. TTP/C는 중복 통신 채널(Redundant Communication Channel)들과 서로

다른 복사 기법들로 구성된 FTU(Fault Tolerant Unit)를 지원한다(e.g., replicated fail-silent nodes 혹은 TMR(Triple Modular Redundancy) nodes). 특히 TTP/C는 별도의 프로토콜 기능 실행을 위한 하드웨어적인 메카니즘을 포함하여 설계된 통신 컨트롤러가 필요하다(e.g., AS8202와 같은 TTP/C 컨트롤러 칩).

Table. 1 Services of TTP/A and TTP/C

Service	TTP/A	TTP/C
Clock synchronization	Central Multi-master	Distributed, Fault-Tolerant
Mode Switches	yes	yes
Communication Error Detection	Parity	16/24 bit CRC
Membership Service	simple	full
External Clock Synchronization	yes	yes
Time-Redundant Transmission	yes	yes
Duplex Nodes	no	yes
Duplex Channels	no	yes
Redundancy Management	no	yes
Shadow Node	no	yes

2.2 TTP의 시스템 구조

TTP의 구조는 Fig. 1에서 보는바와 같이 두 개의 복사 채널을 통하여 연결된 '노드(node)'라고 불리는 전기적 모듈의 집합으로 이루어져있다. 결합된 전기적 모듈들과 TTP 네트워크를 묶어서 '클러스터(cluster)'라고 한다.

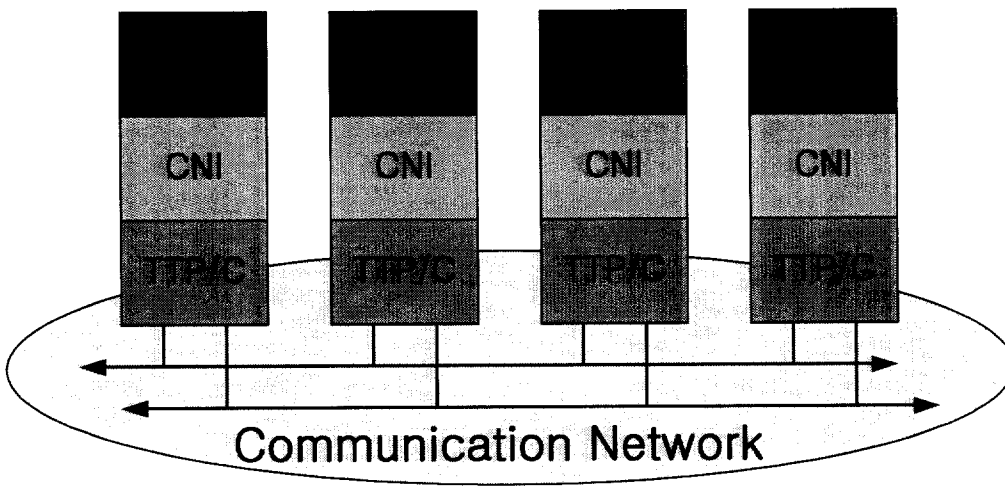

Fig. 1 Communication Network Interface of a TTP system

TTP에서 노드는 가장 작은 대체 가능한 요소(SRU: Smallest Replaceable Unit)로서 장애 발생시에 재구성이나 대체가 가능하다.

TTP 통신 프로토콜 시스템의 하드웨어적인 구조는 Fig. 2에 나타내었다. TTP 통신 프로토콜 시스템은 자동차의 센서와 액츄에이터에 대해 호스트 컴퓨터와 통신 컨트롤러 그리고 입/출력 인터페이스로 구성되며, 호스트 컴퓨터는 애플리케이션 소프트웨어를 실행한다.

CNI(Communication Network Interface)는 노드 내부의 TTP 컨트롤러와 호스트 컴퓨터 사이에 존재하고, TTP 컨트롤러와 호스트 컴퓨터에 대해 동시에 랜덤하게 접근을 허락하는 메모리영역으로 이루어진다. 메모리영역은 호스트 컴퓨터와 TTP 컨트롤러 사이의 데이터를 공유하는 인터페이스로써 DPRAM(Dual Ported Random Access Memory)이 사용된다. 또한

TTP 컨트롤러에서 호스트 컴퓨터로 가는 TTP 인터럽트 라인이 존재한다.

TTP 컨트롤러는 프로토콜 엔진과 TTP 제어 데이터를 위한 저장 공간(MEDL: Message Descriptor List), 그리고 버스 가디언(Bus Guardian)으로 구성된다. 버스 가디언은 컨트롤러의 타이밍 장애(timing failure)로부터 버스를 보호하기 위한 독립적인 하드웨어 요소이다.

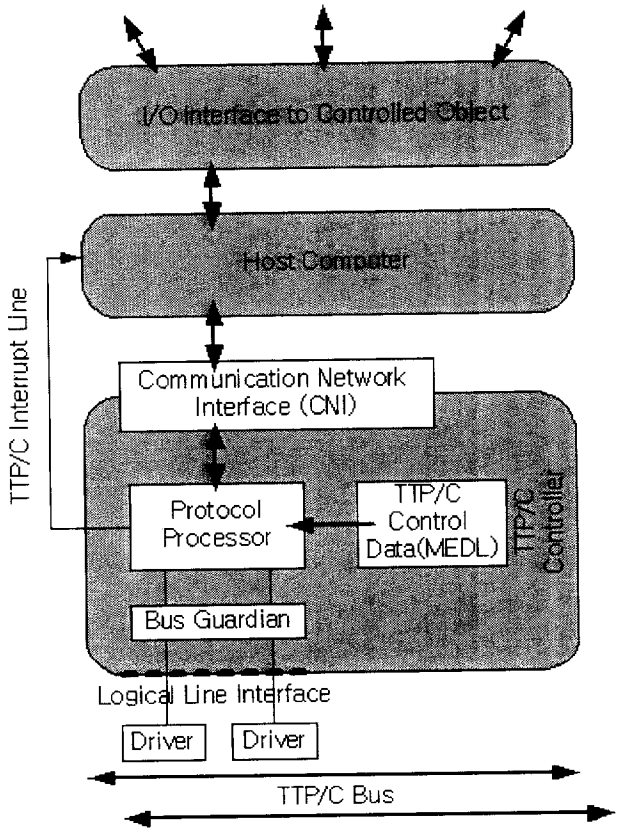

Fig. 2 Hardware structure of a TTP/C

2.3 TTP의 설계 원리

TTP는 정적으로 할당된 미리 정해진 타임 슬롯 동안에 각각의 모든 노드들이 공유된 통신 채널 상에서 송신하는 TDMA(Time Division Multiple Access) 방식을 사용한다. 이러한 주기적인 TDMA 시스템은 TTP 프로토콜을 최적화하기 위해서 사용되며, TTP의 7가지 설계 원리에 대해서 간략히 소개하겠다.

2.3.1 구성성(Composability)

TTP 통신 컨트롤러의 운용은 독립적이며, 결함허용(fault-tolerance) 기능의 글로벌 타임과 TTP 컨트롤러 내부에 존재하는 MEDL(Message Descriptor List)에 의해서 제어된다. TTP 컨트롤러와 호스트 컴퓨터사이에 존재하는 CNI는 값 영역(value domain)과 시간 영역(time domain)에 대해서 완전히 기술되어지므로 구조적으로 구성성(composability)을 지원한다(Fig. 3). 어떠한 제어신호도 CNI를 가로지르지 못하고 MEDL 또한 호스트 컴퓨터로 바로 접근을 할 수 없기 때문에, 호스트 컴퓨터들 중의 어느 한 곳에서 오류(소프트웨어적 혹은 하드웨어적 오류)가 발생한다 할지라도 통신 시스템의 운용을 방해할 수는 없다.

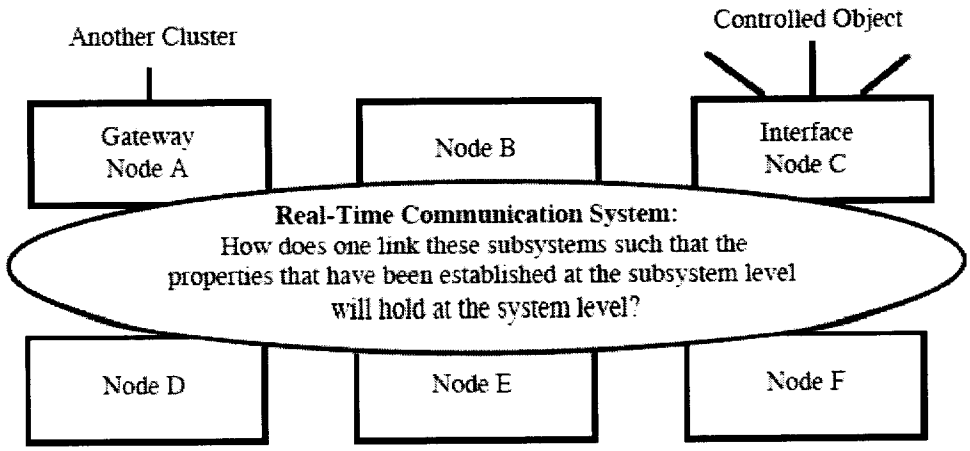

Fig. 3 The composability problem.

2.3.2 연역적 지식을 최대한 사용

TT구조에서, 시스템 동작에 관한 정보는 전체의 모든 노들들에 대해 설계 시점에서 알려진다. TTP는 이러한 연역적인 정보를 최대한 사용한다. 예를 들어, 수신단이 송신단에서 전송된 메시지를 빠뜨린다면 빠뜨린 메시지는 연역적으로 알려진 수신 시간이 경과된 후에 바로 검출될 수 있다.

2.3.3 명칭 부여(Naming)

메시지와 송신단의 명칭은 메시지의 일부분에 속할 수 없다. 왜냐하면 메시지와 송신단의 명칭은 MEDL로부터 복구되어지기 때문에, 이러한 데이터 요소의 명칭은 서로 다른 호스트 컴퓨터에서 다를 수 있다.

2.3.4 확인응답 방식(Acknowledgment Scheme)

TTP의 확인응답 방식은 통신 매체의 브로드캐스트(broadcast)에 대한 이점을 가진다. 연역적으로 알려진 데이터에 의해서 각각의 모든 올바른 멤버(혹은 노드)들은 올바른 수신단을 통해 전송된 각각의 모든 메시지를 상세히 알고 있다. 하나의 수신단이 송신단으로부터 메시지를 수신하였음을 알리자마자, 메시지가 올바르게 전송되어졌고 모든 올바른 수신단이 메시지를 수신하였다고 단정할 수 있다.

TTP 통신의 확인응답 방식은 Fig. 4와 같다. 여기서 TP는 데이터의 전송단계인 'Transmission Phase'를 나타내고, IFG는 피할 수 없는 프레임 사이의 간격이라 하여 'Inter-frame Gap'을 의미한다.

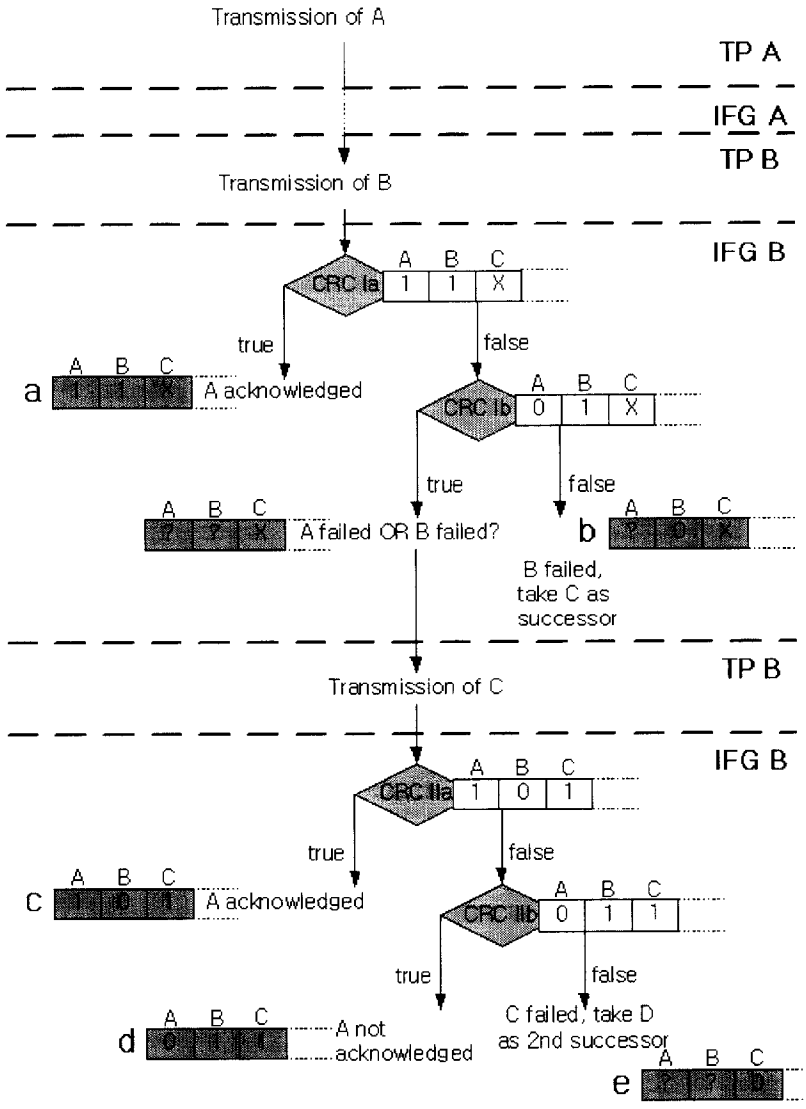

Fig. 4 Acknowledgment Scheme

2.3.5 시간 영역에서의 Fail Silence

TTP는 노드가 시간 영역(temporal domain)에서의 fail-silence의 개념을 지원하는데 기반을 두고, 이것은 시스템 단계에서 오류 제한을 실행하는데 유용하다. 시간 영역에서 노드의 fail-silence의 동작은 각 채널에 존재하는 독립적인 버스 가디언을 통해 구현된다. 멤버십 서비스(membership service)는 일관되게 짧은 잠복기(small latency)를 가진 노드의 장애를 검

출하기 위해서 제공한다.

2.3.6 값 영역에서의 Fail Silence

TTP 컨트롤러는 시간 영역에서 fail-silence를 제공한다. 값 영역에서 fail-silence를 설계하는 것은 순전히 호스트 컴퓨터에 의존한다. 호스트 소프트웨어(혹은 애플리케이션 소프트웨어)는 공간과 시간의 중복성(redundancy) 또는 공간이나 시간의 중복성을 통해서 보장되어야만 한다. 값 장애(value failure)는 TTP에서 제공하는 CRC 체크를 통해서 검출된다.

2.3.7 디자인 교환

TTP에서 노드에서의 처리 요구사항과 채널에서의 밴드 폭 요구사항 사이의 디자인 교환은 통신 컨트롤러에서 증가된 처리 부하(processing load)를 희생시키는 한이 있더라도 유효한 채널 밴드 폭의 최적 사용 쪽으로 기울어진다. 왜냐하면 VLSI 기술의 이점을 고려해서, 자동차 전자공학 응용분야에서 채널의 고유 밴드 폭을 제한하는 것이 통신 컨트롤러의 저장능력이나 처리능력에 대해 제한하는 것 보다 더욱 힘들기 때문이다.

2.4 TTP의 메시지 구성

TTP는 Fig. 5에서 보듯이 프레임, 애플리케이션 데이터, 메시지의 세 가지의 서로 다른 종류의 데이터들로 이루어진다.

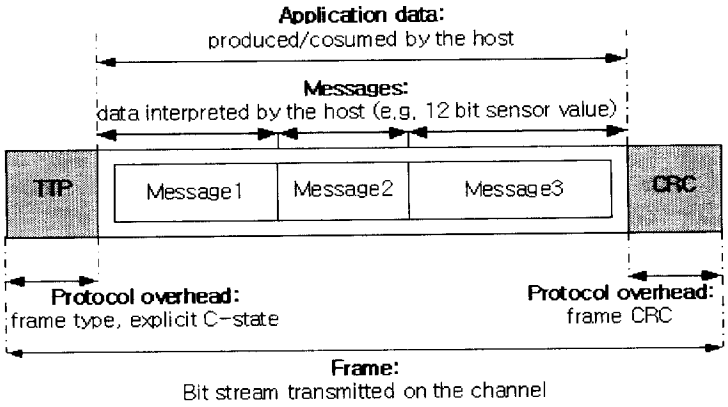

Fig. 5 Data frame of a TTP/C

2.4.1 Frame

프레임은 TTP 채널 상에서 정의된 길이와 코드의 비트 흐름을 전송한다. TTP 프레임은 프로토콜 오버헤드부분에 4비트로 이루어진 프레임의 형태를 결정하는 프레임타입비트, 모드변환요청비트 및 시스템의 상태를 담고 있는 C-state(Controller-state)비트와 데이터의 오류검출을 위한 2~3 바이트의 CRC(Cyclic Redundancy Check)가 존재하고, 최소 2 바이트에서 최대 16 바이트의 application data가 존재한다. 프레임은 프레임 안에 포함된 CRC에 의해서 보호된다.

2.4.2 Application Data

2~16 바이트로 구성된 애플리케이션 데이터는 단지 TTP 컨트롤러의 관점에서만 존재한다. 왜냐하면 프로토콜 운용에 필요하지 않은 모든 프레임 데이터가 호스트 컴퓨터의 범주에 속하기 때문이다. 수신의 경우에 TTP 컨트롤러는 CNI에다가 애플리케이션 데이터를 저장해 두었다가 송신할 때 CNI로부터 페치(fetch)한다.

2.4.3 Message

애플리케이션 데이터와 반대로 메시지는 호스트 컴퓨터의 관점에서만 존재한다. 왜냐하면 애플리케이션 데이터가 호스트 컴퓨터에 의해서 해석되어져서 메시지로 분리되기 때문이다. 메시지는 온도나 변수와 같은 상태를 나타내며 TTP 컨트롤러는 메시지에 관한한 그 어떤 사실도 알지 못한다.

2.5 TTP의 매체접근(Media Access)

TTP 통신 프로토콜 방식은 매체에 접근하기 위해서 Fig. 6에서 보인 것처럼 TDMA(Time Division Multiple Access) 방식을 사용한다.
물리 계층으로의 접근은 글로벌 시간 개념에서 발생된 TDMA 방식에 의해서 제어된다. 각각의 전기적 모듈(노드)은 각 중복 통신 채널(redundant communication channel) 상에서 적어도 하나의 전송을 수행한다. 그러한 노드 슬롯의 순서는 TDMA round의 형태를 이루며, TDMA round가 완료되고 난 후에는 동일한 시간의 접근 패턴을 가진 다음번의 TDMA round가 시작된다. TDMA round의 수는 클러스터 사이클의 주기를 결정한다. 클러스터 사이클이 끝나고 나면 전송 패턴은 다음 클러스터 사이클이 시작하는 지점에서 다시 시작한다.

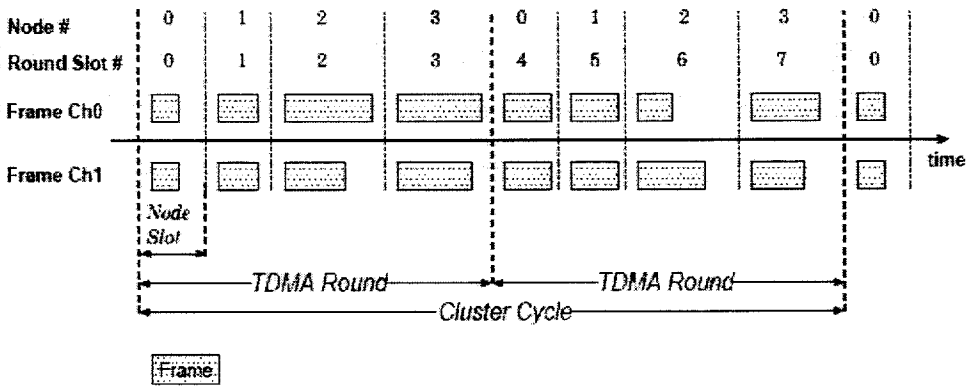

Fig. 6 Media access scheme

제3장 TTP의 운용

3.1 The Message Descriptor List(MEDL)

MEDL은 TTP 컨트롤러 내에 존재하는 정적 데이터 구조이다. MEDL은 통신채널로부터 메시지가 송/수신 시에 메시지를 제어하고, 메모리 영역인 CNI(Communication Network Interface)에 있는 데이터의 위치를 포함하고 있다(Fig. 7). MEDL의 길이는 클러스터 사이클의 길이에 의해서 결정된다.

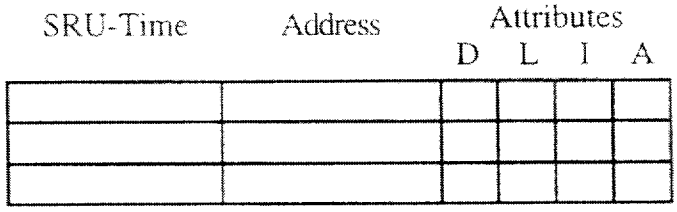

Fig. 7 Format of the MEDL

3.1.1 MEDL Entry

MEDL의 엔트리는 시간영역과 주소영역, 속성영역의 3가지 영역으로 구성된다(Fig. 7). 시간영역은 글로벌 타임의 지점(주소영역에서 명시된 메시지가 통신하는 시점)을 내포한다. 주소영역은 CNI 메모리 셀(cell)을 표시한다. 속성영역은 4개의 하부영역으로 구성된다.

(i) Direction subfield(D): 만약 메시지가 입력 메시지나 출력 메시지일 때, 명시한다.

(ii) Length subfield(L): 통신하는 메시지의 길이를 나타낸다.

(iii) Initialization subfield(I): initialization 메시지나 normal 메시지를 기술한다.

(iv) Additional parameter subfield(A): 모드 변화(mode change)와 노드 역할 변화(node role change)에 관해서 부가적으로 보호하기위한 정보를 내포한다.

임계-안정 시스템(Safety-critical system)에서 호스트 컴퓨터에 의해 요청된 모든 모드 변화는 MEDL에 의해서 차단될 수 있다.

3.2 프레임 형식

TTP의 일반적인 운용 중에, 노드는 SRU 슬롯 동안에 2 프레임을 전송한다. TTP/C의 프레임은 3개의 영역으로 이루어진다(Fig. 8). 이 영역은 4비트의 헤더(header)와 16 바이트까지 변형 가능한 길이 데이터 영역(variable-length data field) 그리고 2또는 3 바이트의 CRC 영역으로 이루어진다.

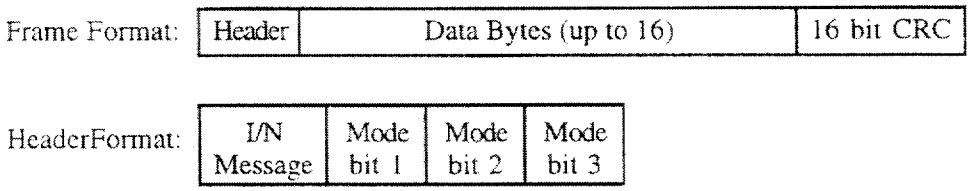

Fig. 8 Format of the TTP frame

헤더의 첫 번째 비트: 헤더의 첫 번째 비트는 메시가 I(initialization)-메시지인지 N(normal)-메시지인지를 알려준다. I-메시지는 시스템을 초기화할 때 사용한다. 또한 데이터 영역에서 송신단의 C-state(Controller-state)를 운반한다.

모드 비트: 3개의 모드 비트는 클러스터의 모든 노드에서 모드 변환을 요청하기위해 사용한다.

데이터 영역: 데이터 영역은 송신 노드에서 CNI로부터 최소 2에서 최대 16 바이트까지의 데이터로 구성된다.

CRC 영역: CRC 영역은 통신상의 오류를 검출하기위해서 CRC 체크 비트를 포함하여 데이터 프레임을 보호하는 역할을 한다.

3.3 CRC 계산

수신단에서의 CRC 체크는 수신단의 C-state(Controller-state)와 수신된 메시지 내용을 가지고 계산된다. 만약 수신단에서 CRC 체크의 결과 값이 negative(음)라면, 전송중인 내내 메시지는 오류를 일으켰거나 송신단과 수신단의 C-state 사이에 불일치가 존재했다는 것을 뜻한다.

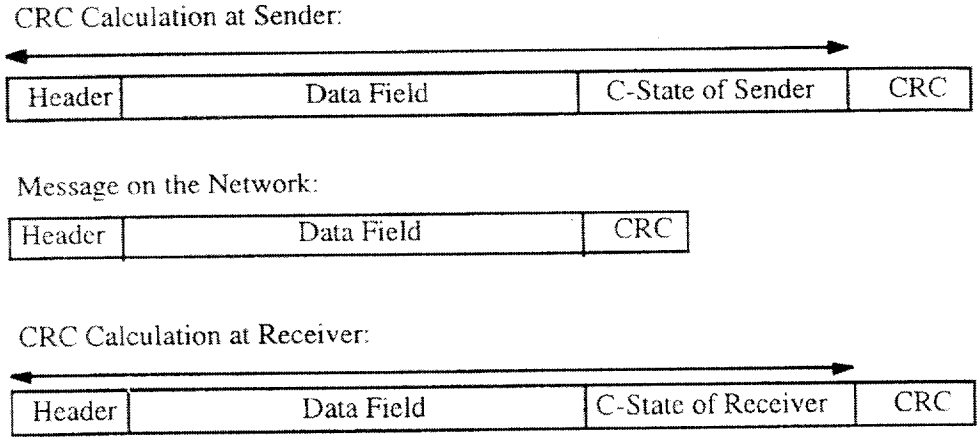

Fig. 9 Calculation of the CRC of normal messages

3.4 멤버십 서비스

TTP의 SRU 계층은 시기적절하게 노드 멤버십 서비스를 제공한다. 멤버십 영역에서 C-state의 비트 수(Fig. 8)는 클러스터의 최대 노드 수와 일치한다. 각 송신-노드 슬롯은 송신 노드에 대한 멤버십 지점이다. 만약 송신 노드의 중복 메시지(redundant message)중의 하나가 수신 노드로 올바르게 수신되었다면, 수신 노드는 이 멤버십 지점에서 송신 노드의 동작을 고려한다. 노드는 다음 TDMA 사이클의 멤버십 지점까지 동작을 고려한다. 만약 노드가 이 구간 안에서 고장이 발생했다면, 장애(failure)는 돌아오는 멤버십 지점에서 감지된다. 멤버십 정보의 지연은 적어도 1 TDMA 사이클이다. 그러므로 프로토콜은 장애 발생 후에 적어도 1 TDMA 사이클까지는 기다려야만 한다. 만약 예상되는 메시지가 아닌 다른 메시지가 CRC에 도달한다면, 수신단은 이 멤버십 지점에서 송신 노드를 '고장(fail)'이라고 간주하고 현재 SRU 슬롯의 끝 지점에서 이 노드의 멤버십 비트를 클리어(clear) 한다.

Fig. 10에서 IFG는 '피할 수 없는 프레임 사이의 간격(Inter-frame Gap)'이라고 하여 이전의 전송단계와 다음의 전송단계 프로토콜 태스크(task)를 실행하기위해서 컨트롤러에 요구되는 시 구간(time interval)을 말한다.

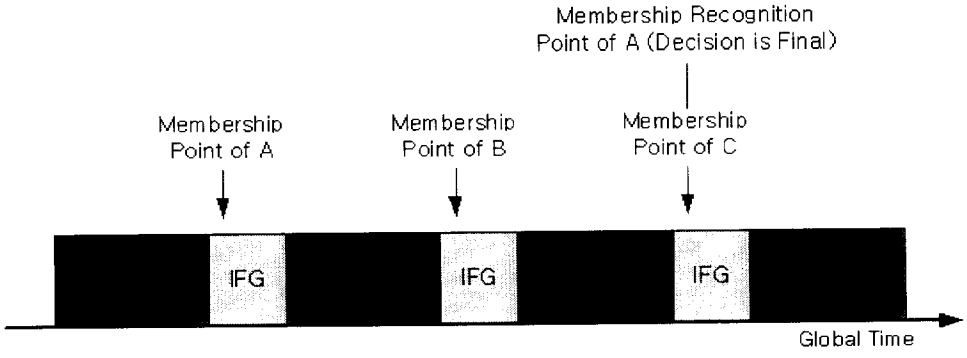

Fig. 10 Membership Decisions

3.5 내부 클록 동기화

내부 클록 동기화의 목적은 각 노드들의 실시간 로컬 클록의 변화하는 편차율(drift-rate)에도 불구하고, 모든 올바른 노드들의 글로벌 tick은 명시된 정밀도(Π: 데이터시트에 표기된 클록 발생기의 정밀도)의 범위 내에서 발생하는 것을 보장하는 것이다. 적절한 글로벌 타임 베이스의 효용이 실시간 분산시스템을 운용하는데 매우 중요하기 때문에, 클록 동기화는 결함허용(fault-tolerance)의 기능을 가져야만 한다. 전체 노드들의 클록 동기화는 다음의 3단계의 절차를 거친다.:

i) 매 클록마다 명확한 전체 클록의 시간 값을 읽는다.

ii) 매 클록마다 클록 동기화 알고리즘을 사용하여 정정 기간(correction term)을 계산한다.

iii) 매 클록마다 정정된 정정 기간을 각각의 분산시스템(노드)에 적용한다.

분산시스템의 모든 노드들은 오실레이터를 가지는데, 오실레이터는 그것의 물리적인 파라미터로 정해진 주파수를 가지며, 이것을 '(micro)tick'이라 한다. microtick의 부분집합을 '(macro)tick'이라하며, tick은 각 노드의 글로벌 타임 tick에 의해 해석되어진다. 이러한 글로벌 타임 tick은 각 노드의 글로벌 타임 계수기를 증가시킨다.

TTP는 글로벌 타임 베이스를 발생시키기 위해서 로컬 클록의 결함허용(fault-tolerance) 내부 동기를 제공한다. 모든 수신 노드들이 예상되는 메시지 도달 시간을 연역적으로(a priori) 알고 있기 때문에, 연역적으로 명시된 도달 시간과 관측된 도달 시간사이의 일탈은 수신단의 클록과 송신단의 클록사이의 클록 차이다.

이 절에서는 클록과 관련한 대략적인 설명과 함께 내부 클록 동기화에 대해 기술한다.

3.5.1 클록

과거에는 이벤트들 사이의 기간(duration) 측정이 주로 주관적인 판단에 근거했지만, 현대 과학의 출현으로 물리적인 클록을 사용함으로써 객관적인 판단에 의해서 시간의 경과를 측정하게 되었다. 이 절에서는 물리적인

클록과 TTP에서 클록의 기준이 되는 기준클록에 대해서 간략히 기술하도록 하겠다.

Physical Clock(물리적 클록): (물리적인)클록은 시간을 측정하기위한 수단이며, 계수기(counter)와 계수기의 증가에 따라 주기적으로 이벤트를 발생시키는 물리적 진동 장치(physical oscillation mechanism)로 구성된다. 이 주기적인 이벤트를 클록의 'microtick'이라하며, 두 연속적인 microtick 사이의 기간(duration)은 클록의 '세분성(g)'라 한다. 어떤 디지털클록의 세분성 g는 시간 측정에서 양자화 오류(digitalization error)를 발생시킬 수도 있다.

앞으로 나오게 될 수식에서 다음과 같은 표기법을 사용한다.: 클록은 자연수 '1, 2, …, n'으로 표기한다. 클록의 특징 표기에서 위첨자는 클록의 수를 표시하고, 아래첨자는 tick의 수를 표시한다. 예를 들면, k 클록의 i microtick은 $microtick_i^k$으로 표기한다.

Reference Clock(기준 클록): 분산 실시간 시스템에서 클록의 기준이 되는 기준클록에 대해 설명하기 전에, 모든 이벤트들을 관측할 수 있는 외부 관측자(external observer)가 있다고 가정하자. 이 관측자는 국제 표준시(international standard of time)에 완벽히 부합하는 f^z의 주파수를 가진 유일한 기준 클록 z를 가진다. 기준 클록의 계수기는 항상 국제 표준시처럼 항상 동일하며, $\frac{1}{f^z}$을 클록 z의 세분성 g^z라 한다. f^z가 $10^{15}(microticks/second)$ 만큼 아주 크다고 가정한다면, 세분성 g^z는 $10^{-15}(second)$이다. 그러므로 기준 클록의 세분성 g가 매우 작기 때문에 기준 클록의 양자화 오류는 앞으로 해석하게 될 부분에서 고려하지 않아도 된다.

관측자가 이벤트(e)의 발생을 감지할 때마다, 관측자는 기준 클록 z의 현재 상태를 즉시 기록하고, 이벤트 e에 대한 타임스탬프를 발생시킨다. 클록(이벤트)은 주어진 클록을 사용하여 발생된 타임스탬프를 표시한다. 클록 z가 시스템에서 단일 기준 클록이기 때문에, z(e)는 이벤트(e)의 '절대 타임스탬프(absolute timestamp)'라 한다.

두 이벤트들 사이의 기간(duration)은 두 이벤트들 사이의 구간(interval)

안에서 발생하는 기준 클록 z의 microtick을 카운트함으로써 측정한다. 주어진 클록 k의 세분성 g^k는 기준 클록 z의 microtick 수 n^k에 의해서 주어진다.

기준 클록의 두 연속적인 microtick 사이(g^z의 범위)에서 발생하는 이벤트의 시간적 순서는 절대 타임스탬프(absolute timestamp)로부터 재구성할 수 없다. 이것은 시간 측정에서의 기본적인 제한이다.

Clock Drift(클록 편차): microtick i와 microtick i+1사이의 물리적 클록 k의 편차(drift)는 클록 k와 기준 클록 z 사이에 주파수 비이다. 편차는 기준 클록 z와 함께 클록 k의 세분 구간을 측정함으로써 결정되고 다음과 같다.:

$$drift_i^k = \frac{z(microtick_{i+1}^k) - z(microtick_i^k)}{n^k}$$

양호한 클록은 1에 가까운 편차를 갖기 때문에, 표기상의 편의로 편차율 ρ_i^k의 개념은 다음과 같다.:

$$\rho_i^k = \left| \frac{z(microtick_{i+1}^k) - z(microtick_i^k)}{n^k} - 1 \right|$$

완벽한 클록은 0의 편차율을 갖지만, 실제의 클록은 환경의 영향에 따라 다양한 편차율을 가진다. 예를 들면, 주변 온도의 변화, 크리스탈 공진기(resonator)로 공급되는 전압 레벨의 변화나 크리스탈의 노화 등이 있다. 명시된 환경 파라메타에서, 공진기의 편차율은 최대 편차율 $\rho_{\max}^k$에 의해 한정된다. 전형적인 최대 편차율 $\rho_{\max}^k$는 10^{-2} ~ 10^{-7}(sec/sec)의 범위에서 존재하고, 가격이나 질(quality)에 따라 더 좋을 수도 있다. 모든 클록들이 0이 아닌 어떠한 편차율을 가지기 때문에, 설사 클록들이 시작 시에 완전히 동기화가 되었다고 할지라도 일정 시간이 지나면 제각각 다를 수밖에 없다.

Failure Modes of a Clock(클록의 고장모드): 물리적인 클록은 두 가지 형태의 고장을 나타낼 수 있다. 첫째, 계수기(counter)가 결함(fault)에 의해서 산산조각 나서 계수 값이 잘못된 상태에 이르게 된다. 둘째, 클록이 명시된 편차율(Fig. 11에서 색칠된 부분)보다 빠르거나 혹은 늦게 시작하기

때문에 클록의 편차율이 명시된 편차율에서 벗어난다.

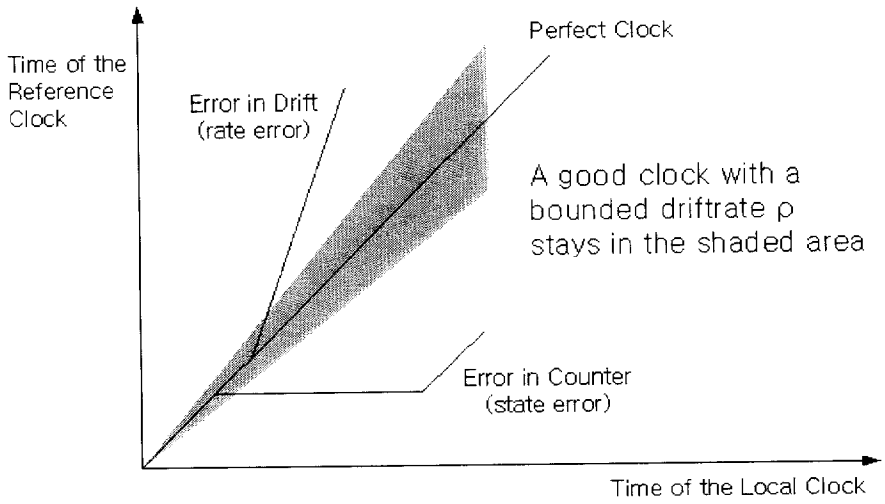

Fig. 11 Failure modes of a physical clock

3.5.2 정밀도와 정확성

Offset: 동일한 세분성 g를 가진 두 클록 j와 k 사이에서 microtick i는 다음과 같다.:

$$offset_i^{jk} = |z(microtick_i^j) - z(microtick_i^k)|$$

오프셋은 각각의 두 클록 j와 k의 microtick 사이의 시간 차이를 나타낸다.

Precision(정밀도): 클록의 전체 집합이 {1, 2, …, n}으로 주어지면, 전체 두 클록사이의 최대 오프셋을 정밀도 Π_i라 하고 다음과 같다.:

$$\Pi_i = \max_{\forall 1 \le j,k \le n} \{offset_i^{jk}\}$$

구간 전체에 걸친 Π_i의 최대치는 정밀도 Π라고 한다. 정밀도는 해당 구간의 주기 동안에 어떤 각각의 두 클록에 대한 microtick의 최대 오프셋을 나타낸다. 정밀도는 기준 클록의 microtick 수에 의해 표현되어진다.

물리적 클록의 편차율 때문에, 만약 주기적으로 재동기화(resynchronize)되지 않는다면 전체의 모든 클록들은 제각각 편차를 나타내게 된다. 제한

된 정밀도 Π를 유지하기 위해서 클록 전체의 상호 재동기화(mutual resynchronization)하는 처리과정을 '내부 동기화(internal synchronization)'라 부른다.

Accuracy(정확성): microtick i에서 기준 클록 z에 대한 클록 k의 오프셋은 $accuracy_i^k$라 한다. microtick i 전반에 걸친 최대 오프셋은 $accuracy^k$이다. 정확성은 외부 기준시간으로부터 주어진 클록의 최대 오프셋을 나타낸다.

기준 클록의 제한된 구간 안에서 클록을 유지하기위해서, 기준 클록과 함께 주기적으로 재동기화를 해야 한다. 이러한 재동기화 과정을 '외부 동기화(external synchronization)'라 부른다.

만약 전체의 모든 클록들이 정확성 A와 외부적으로 동기화된다면, 전체의 모든 클록들은 적어도 2A의 정밀도와 내부적으로 동기화되어야만 하고, 그렇지 못한 경우는 거짓이다. 만약 클록이 결코 외부시간과 재동기화하지 않는다면 내부적으로 동기화된 전체 클록들은 외부시간으로부터 드리프트 할 것이다.

3.5.3 동기화 조건

각 노드의 글로벌 타임 tick은 글로벌 타임 베이스를 형성하기위해서 노드의 전체 범위에서 주기적으로 동기화되어야 한다. 재동기화의 주기(period)를 '재동기화 구간(resynchronization interval)'이라 하며, 각 재동기화 구간의 끝에서 클록은 보정된다. 수렴 함수(convergence function: Φ)는 재동기화 직후의 시간 값 오프셋을 나타낸다. 그런 다음, 다음 재동기화 구간(R_{int})에서 재동기화 할 때까지 클록은 드리프트 된다(Fig. 12). 편차 오프셋(Γ)은 재동기화 구간(R_{int})의 길이에 의존하고, 명시된 클록의 최대 편차율(ρ)은 다음과 같다.:

$$\Gamma = 2\rho R_{int}$$

만약 수렴 함수(Φ)와 편차 오프셋(Γ) 그리고 정밀도(Π)가 다음의 동기화 조건을 만족한다면, 전체 클록은 동기화될 수 있다.:

$$\Phi + \Gamma \leq \Pi$$

Fig. 12에서 모든 클록들이 올바르게 동기화를 수행하기위해서는 음영부

분의 범위($\Phi + \Gamma \leq \Pi$)내에서 존재해야한다. 분산 실시간 시스템에서, 만약 모든 클록들이 동기화 조건을 만족시키지 못한다면 간헐적으로 발생하는 타이밍 오류인 비잔틴 오류(Byzantine Error)가 발생한다.

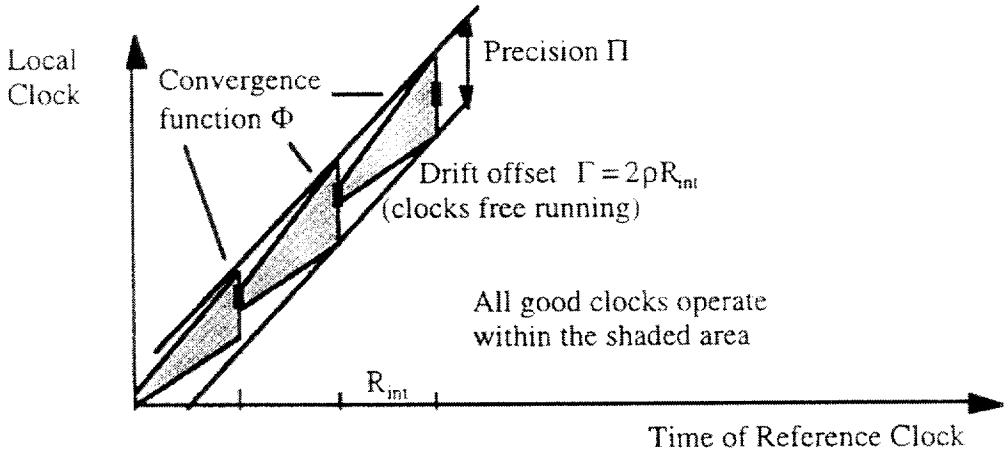

Fig. 12 Synchronization condition

Byzantine Error(비잔틴 오류): Fig. 13에서처럼 3개의 노드가 있다고 가정하자. 여기서, 클록 A와 B는 올바르게 동작하는 클록이고 클록 C는 나머지 올바른 클록들의 동기화를 방해하는 양면성을 지닌 악성 클록이다. 전체 3개의 노드에서 Fig. 13는 양면성을 지닌 악성 노드(malicious node)가 나머지 두 노드들의 동기화하는 것을 방해하는 것을 나타낸다. 여기서 동기화 조건을 5분의 범위라고 가정한다면, 클록 A와 클록 B는 5분의 시간차이가 나지만 동기화 조건을 만족하므로 이들 두 클록은 동기화하는데 아무런 지장을 주지 않는다. 하지만 클록 C와 같은 양면성(클록 A와 동기화를 수행할 때는 3:55분이었다가 클록 B와 동기화를 수행할 때는 4:10을 가리킴)을 지닌 악성 클록이 첨가된다면 악성클록 C 때문에 각 클록들이 동기화를 수행하는 과정에서 10분이라는 시간차가 발생하여 동기화 조건을 만족시키지 못함을 알 수 있다. 왜냐하면 클록 A와 B는 악성클록 C 때문에 올바른 시간 값을 가질 수 없으므로 동기화 조건을 만족시킬 수 없다.

이러한 양면성을 지닌 악성클록의 동작으로 인하여 발생하는 오류를 흔히 '악성 오류(malicious error) 또는 비잔틴 오류(Byzantine error)'라 한다.

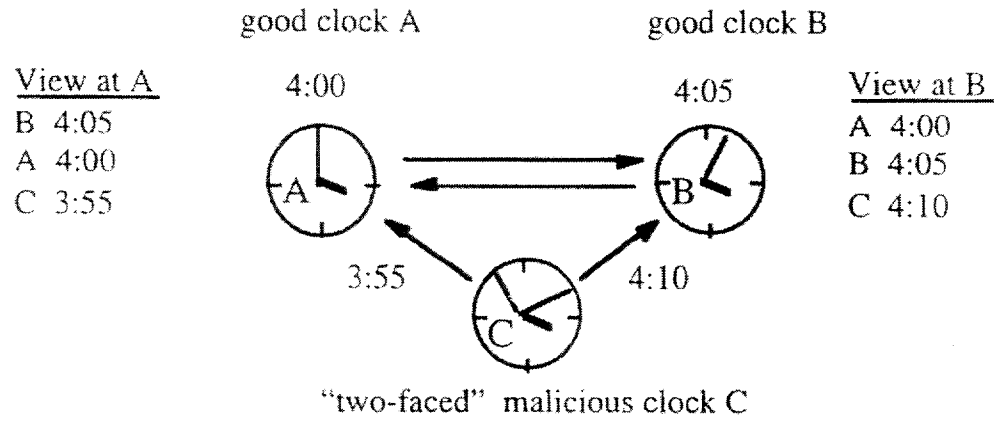

Fig. 13 Behavior of a malicious clock C

분산 시스템에서 비잔틴 오류가 1개 발생했다면 아래의 식에 따른다.:

$$E_{byz} = \Pi/(N-2k)$$

여기서, N은 노드의 수이고, k는 비잔틴 에러의 개수이다.

비잔틴 오류를 고려하여 분산 시스템의 최악의 상황에서, k 개의 비잔틴 오류가 발생했다면 아래의 식과 같다.:

$$E_{byz} = k\Pi/(N-2k)$$

동기 메시지에 대한 지터를 고려해보면, 재동기화 직후의 시간 값을 나타내는 수렴함수(Φ)는 다음과 같다.:

$$\Phi(N,k,\epsilon) = (k\Pi/(N-2k)) + \epsilon$$

여기서, ϵ은 지연 지터를 의미한다.

3.5.3절의 동기화 조건을 가지고 위의 식들을 조합하면 다음과 같은 정밀도(Π)를 구할 수 있다.:

$$\Pi(N,k,\epsilon,\Gamma) = (\epsilon+\Gamma)\frac{N-2k}{N-3k} = (\epsilon+\Gamma)\mu(N,k)$$

여기서, Γ는 편차(drift) 오프셋을 나타내고, $\mu(N,k)$는 비잔틴 오류 기간을 나타낸다. Table. 2는 비잔틴 오류 기간에 대해서 나타내었다.

Table. 2 Byzantine error term $\mu(N,k)$

Faults(k)	Number of nodes in the ensemble7							
	4	5	6	7	10	15	20	30
1	2	1.5	1.33	1.25	1.14	1.08	1.06	1.03
2				3	1.5	1.22	1.14	1.08
3					4	1.5	1.27	1.22

$\mu(N,k)$는 비잔틴 오류로 인하여 발생하는 클록의 불일치 때문에, 정밀도(Π)가 떨어지는 것, 즉 정밀도의 손실을 의미한다. Γ는 재동기화 구간의 길이와 사용자가 선택한 오실레이터의 특성에 의해 결정되어진다.

3.5.4 중앙 마스터 동기

단일 노드인 중앙 마스터(central master)는 동기 메시지에 있는 시간 계수 값을 모든 다른 노드(slave node)로 주기적으로 전송한다. 슬레이브 노드가 마스터로부터 새로운 시간 값을 수신하는 즉시, 슬레이브는 로컬-타임 카운터의 상태(메시지 도착 시간에서의 상태)를 기록한다. 동기화 메시지를 포함하고 있는 마스터 시간과 기록된 슬레이브의 메시지 도착시간사이의 차는 2 클록의 차이가 난다. 슬레이브는 이러한 마스터의 정보를 이용하여 슬레이브의 클록을 정정한다.

중앙 마스터 알고리즘의 수렴 함수(Φ)는 슬레이브 노드로의 최대 전송과 최소 전송간의 차이에 의해 결정된다. 즉, 마스터에서 클록을 읽는 이벤트와 모든 슬레이브에서 도착한 메시지 이벤트 사이의 지연 지터(latency Jitter: ϵ)이다.

동기화 조건을 적용하기위해서, 중앙 마스터 알고리즘의 정밀도는 다음과 같이 주어진다.:

$$\Pi_{central} = \epsilon + \Gamma$$

여기서 ϵ은 지연 지터이고 Γ는 편차 오프셋이다.

중앙 마스터 동기화는 주로 분산 시스템의 스타트-업(start-up) 단계에서 사용되어진다. 그것은 간단히 사용할 수 있지만, 결함허용(fault-tolerance)

의 기능을 지원하지는 않는다. 왜냐하면 마스터의 장애(failure)가 곧 나머지 슬레이브 노드들에게 영향을 미쳐서 재동기화를 할 수 없게 하기 때문이다. 또 다른 동기화 알고리즘으로는 멀티-마스터 방식(multi-master strategy)이 있다. 이 멀티-마스터 방식은 다음과 같은 조건을 만족해야한다.: 만약 활성 마스터(active master)가 장애(failure)를 일으키고 "shadow (새도)" 마스터에서 마스터 장애가 검출된다면, 새도 마스터중의 한 노드는 마스터의 역할을 대신하고 재동기화를 계속 진행한다.

3.5.5 분산 동기화 알고리즘

대개 결함허용(fault-tolerance)의 기능을 지닌 분산 클록의 재동기화는 3가지 단계를 거친다.:

i) 노드들 사이의 메시지 교환을 통하여 각각의 노드는 다른 노드들에서의 글로벌 타임 카운터 상태에 관련한 정보를 습득한다.
ii) 각각의 노드는 오류(error)를 검출하기위해서 수집된 정보를 분석하고, 로컬 글로벌 타임 카운터에 대한 정정 값을 계산하기위해서 수렴함수(Φ)를 수행한다. 만약 계산된 정정 값이 명시된 정밀도(Π)보다 크다면, 노드는 활성화되지 않는다.
iii) 노드의 로컬 타임 카운터가 계산된 정정 값으로 조정되어진다.

분산 실시간 시스템에서 동기화의 정밀도에 영향을 끼치는 가장 중요한 것은 시간-메시지의 지터(jitter)이다. 지터는 정해진 전송시간과 실제 전송시간상의 시간적인 'gap 또는 delay'를 의미한다. 정해진 두 노드간의 시간-메시지 전송에 대한 최소 지연시간은 연역적으로 정해놓은 지연-보상(delay-compensation)시간에 의해서 보상된다. 지연 지터는 시스템 레벨에 상당히 의존하는데, 지터가 시스템 레벨에 대한 의존도를 Table. 3에서 보여준다. Table. 3은 시스템의 서로 다른 단계에서 예상되는 지터에 대한 대략적인 값 범위를 나타낸다.

Table. 3 Approximate jitter of the synchronization message

synchronization message assembled and interpreted	approximate range of jitter
at the application software level	500 μs ~ 5 ms
in the kernel of the operating system	10 μs ~ 100 μs
in the hardware of the communication controller	less than 10 μs

위의 표에서도 알 수 있듯이, 지터는 다른 어떤 요소들보다도 더욱 시스템 레벨에 의존한다. 그러므로 지터를 최소화하기위해서는 TT 시스템과 같이 통신 컨트롤러의 하드웨어 단계에서 지터를 고려하여 설계를 해야 한다.

제4장 결함 허용성(Fault-Tolerance) 통신 실험

본 논문의 분산된 제어 노드간의 결함에 허용한 통신 시험을 위해 TTP/C 통신 모듈을 제작하였고, 4개의 통신모듈을 병렬 연결하여 노드간의 하드웨어적인 결함이나 소프트웨어적인 결함 상황에서도 정상적인 통신이 이루어지는 결함 허용성(fault-tolerance) 통신이 가능함을 실험으로 입증한다.

4.1 개발 보드

TTP/C 통신 프로토콜의 구현을 위해 제작한 통신 모듈의 호스트 프로세서는 MOTOROLA사의 32bit 리스크 타입 프로세서인 MPC555를 사용하였으며, 외부 메모리는 Flash/SRAM 메모리를 사용하였고, TTP/C 컨트롤러 칩(chip)은 AS8202를 사용하였다. 통신 버스라인의 중복성(redundancy)을 위해서 통신 드라이버는 두 개의 RS485를 써서 구성하였으며, Photo. 1과 같이 제작하였다.

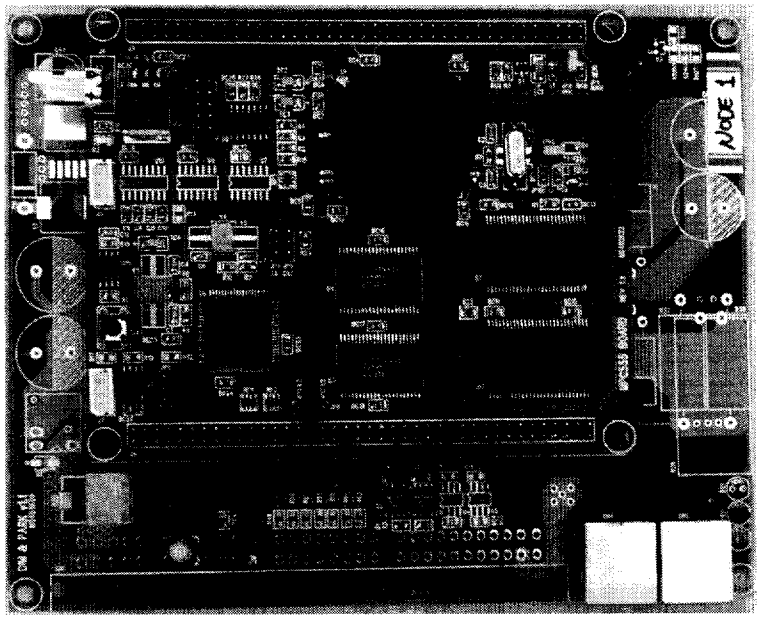

Photo. 1 TTP/C Communication Module

4.2 프로그램 개요

4.2.1 프로그램 소개

본 실험을 위해 사용한 프로그램 툴은 Math-works사의 Matlab/Simulink와 TTTech사의 상용화 툴을 사용하였으며, 컴파일러는 WindRiver사의 Diab Compiler를 사용하였고, 다운로드 장비로는 P&E사의 CABLEPPC를 사용하였다.

각 노드에서 작성되는 프로그램은 제어프로그램과 TTP/C 통신 프로토콜 프로그램으로 구성된다. 제어프로그램은 4개의 노드를 중심으로 순차적(1-2- 3- 4번 노드 순)으로 제어보드에서 스위치 신호를 받아 다음 순번의 노드에 자기 번호 신호를 보내고, 다음 순번의 노드에서는 받은 신호를 표시함과 동시에 다음번 노드에 보낼 신호를 대기하는 순으로 작성된다. 그리고 TTP/C 통신 네트워크 상에서 송수신되는 스케줄링 된 메시지가 들어있는 MEDL 파일을 작성하였다.

애플리케이션 파일은 Matlab/Simulink상에서 모델링 했으며, Real-Time Workshop을 이용하여 code generation하였다.

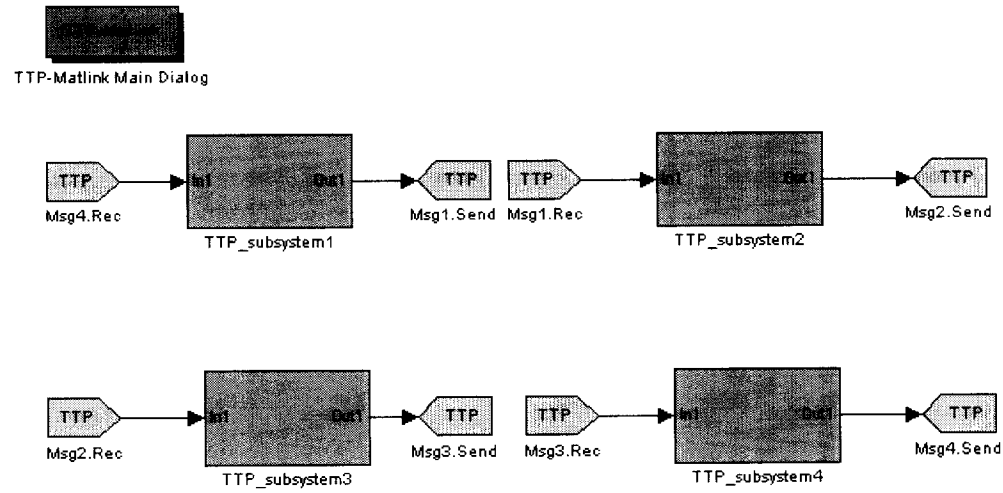

Fig. 14 TTP program block diagram

4.2.2 통신 시스템 설계

TTP 통신 시스템의 설계는 Fig. 14와 같이 우선 설계하고자 하는 TTP 통신 시스템을 Matlab/Simulink상에서 설계한 후, TTPplan에서 TTP 통신 프로토콜의 운용에 관련된 정보를 담고 있는 스케줄링 된 MEDL파일을 작성한다. 이 단계는 클러스터 레벨에서 이루어진다. MEDL파일을 작성하고 나면 TTPbuild를 통해서 클러스터 내에 존재하는 노드들을 디자인한다. TTPbuild가 끝나고 나면 Matlab/Simulink의 Real-time Workshop의 code-generator를 이용하여 code-generation한 후, Diab 컴파일러를 통하여 컴파일 한다. 컴파일까지 끝나고 나면 TTPload와 CABLEPPC를 이용하여 각 노드의 외부 메모리로 다운로드하면 일련의 절차는 끝이 난다. 이러한 일련의 절차는 Fig. 15에서 보여준다.

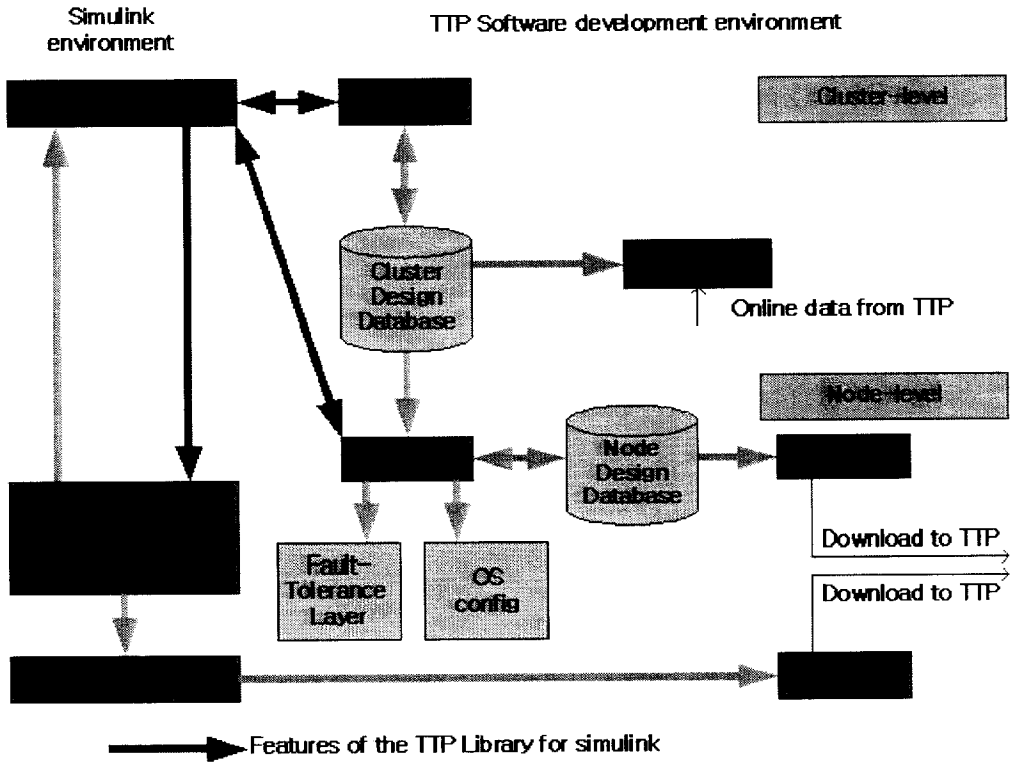

Fig. 15 Tool chain of a TTP

4.3 결함허용성(Fault-Tolerance) 통신 실험

결함 허용(fault-tolerance)이란, 어떠한 분산시스템에서 발생되는 결함(fault)을 허용하는 것을 말한다. 여기서 허용(tolerance)이란, 분산시스템에서 발생된 결함에도 불구하고 다른 나머지 시스템의 동작에는 지장을 주지 않는 것을 의미한다. 즉, 결함 허용 시스템은 발생된 결함으로 인한 시스템의 오작동을 방지하고, 계획된 동작을 수행할 수 있게 하는 시스템이다.

분산 시스템에서 결함(faults), 오류(errors), 장애(failures)는 시스템의 신뢰도를 떨어뜨리며, 최악의 상황에는 시스템이 회복불능의 상황에 도달할 수도 있다. Fig. 16은 결함, 오류, 장애의 인과관계를 나타내고 있다. 결함은 크게 하드웨어 결함과 소프트웨어 결함으로 나눌 수 있고, 고장을 일으키는 시스템의 원이 될 수도 있다. 하지만 결함이 발생했다고 하여 무조건 시스템의 오작동을 초래하는 것은 아니다. 결함이 시스템의 오작동에 영향을 미치게 된다면, 그 시스템(또는 서브시스템)은 오류가 발생하여 시스템의 상태는 사용자의 의도와는 다르게 변경될 것이다. 사용자가 의도하지 않은 방향으로 동작이 됨을 감지할 때 장애(또는 고장)이라고 한다.

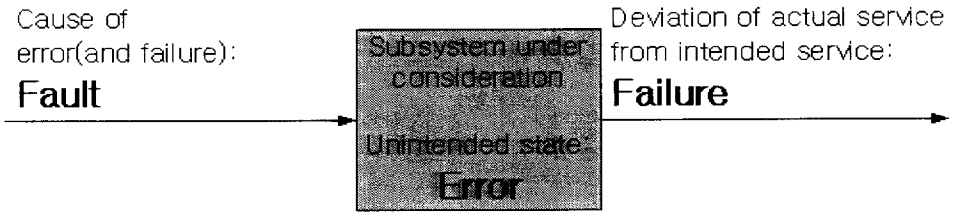

Fig. 16 Faults, Errors, and Failures

본 논문은 분산 시스템에서 오류와 장애를 유발시킬 수 있는 하드웨어/소프트웨어 결함에 TTP 통신 시스템을 적용하여 결함 허용성 통신을 실험한다.

4.3.1 통신 네트워크 인터페이스

결함 허용성 통신을 위한 하드웨어 결선은 Photo. 2와 같이 통신 채널 상에 중복을 두어 2개의 채널(채널 0과 1 또는 채널 A와 B)에 4개의 노드(ECU)를 병렬로 연결하여 구성하였다.

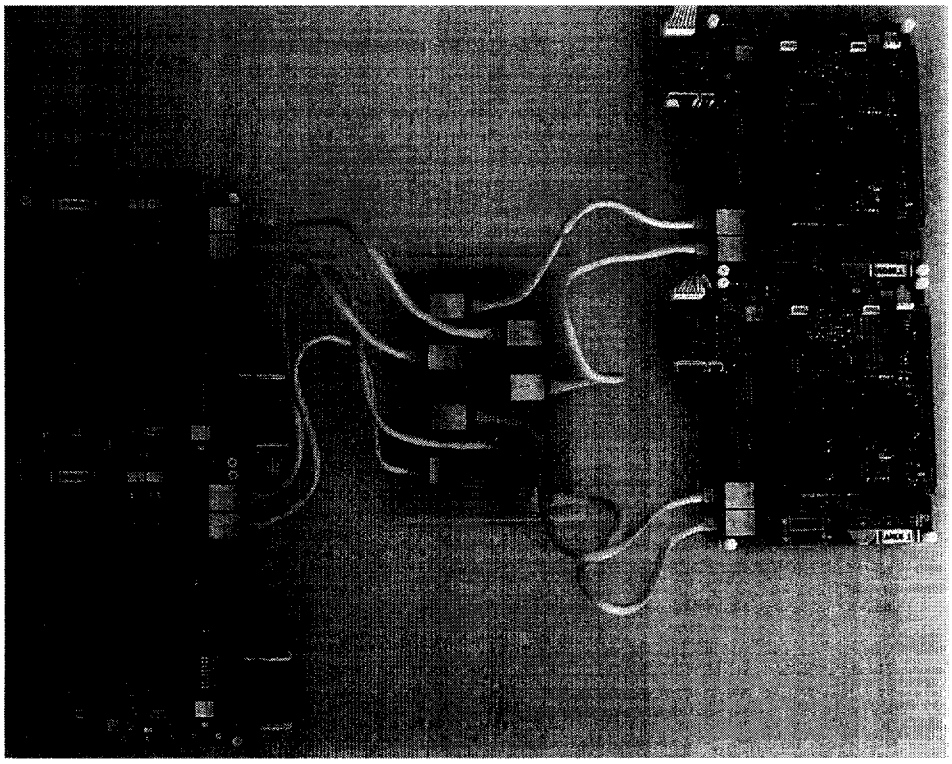

Photo. 2 Communication Network Interface

4.3.2 정상적인 통신 상태

정상적인 상황에서의 통신상태는 Photo. 3에서와 같이 노드 3과 노드 4 간의 송수신이 잘 됨을 보여준다. 여기서 노드 3의 7-세그먼트 4는 노드 4로 메시지를 보낸다는 송신의 의미이고, 노드 4의 7-세그먼트 3은 노드3에서 메시지가 도착했다는 수신의 의미이다. 그리고 각 노드의 우측하단에 존재하는 LED의 계속적인 깜빡임은 4개의 노드들이 잘 동기화가 되고 있음을 나타낸다.

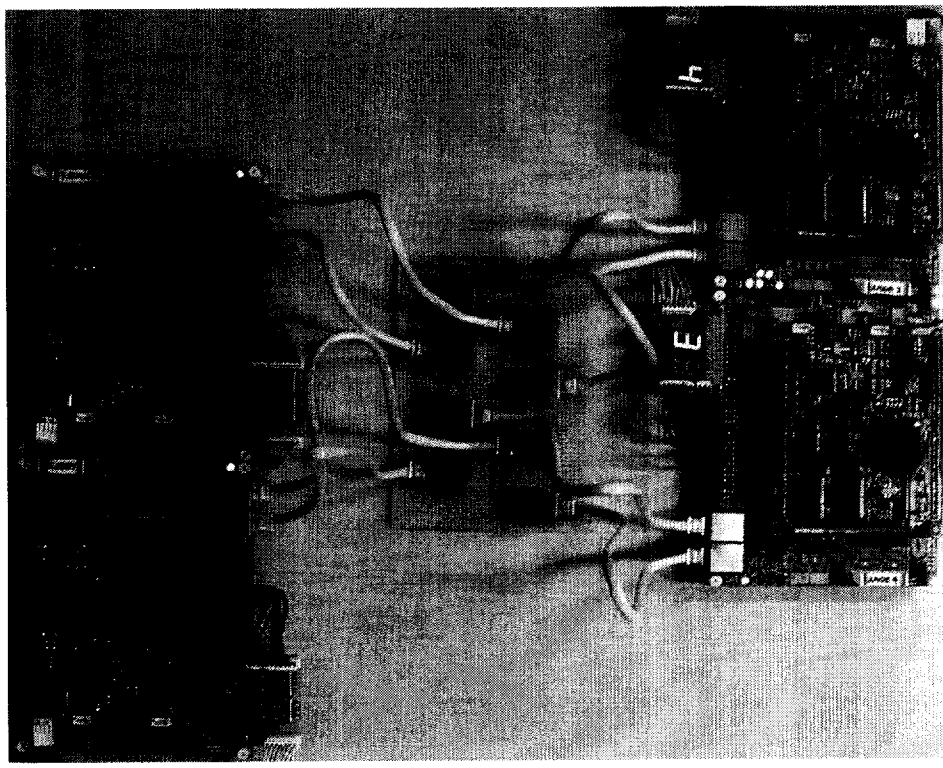

Photo. 3 Normal Communication

4.3.3 하드웨어적인 결함시의 통신 상태

Photo. 4에서 보듯이 하드웨어적인 결함상태에서는 두 개의 채널(channel 0, 1) 중에 하나의 채널을 끊음으로써 구현하였다. 하드웨어적인 결함상황에서도 노드간의 송수신에는 아무런 지장을 주지 않음을 알 수 있다. 또한 4개의 노드들의 동기화가 잘 이루어짐을 알 수 있다.

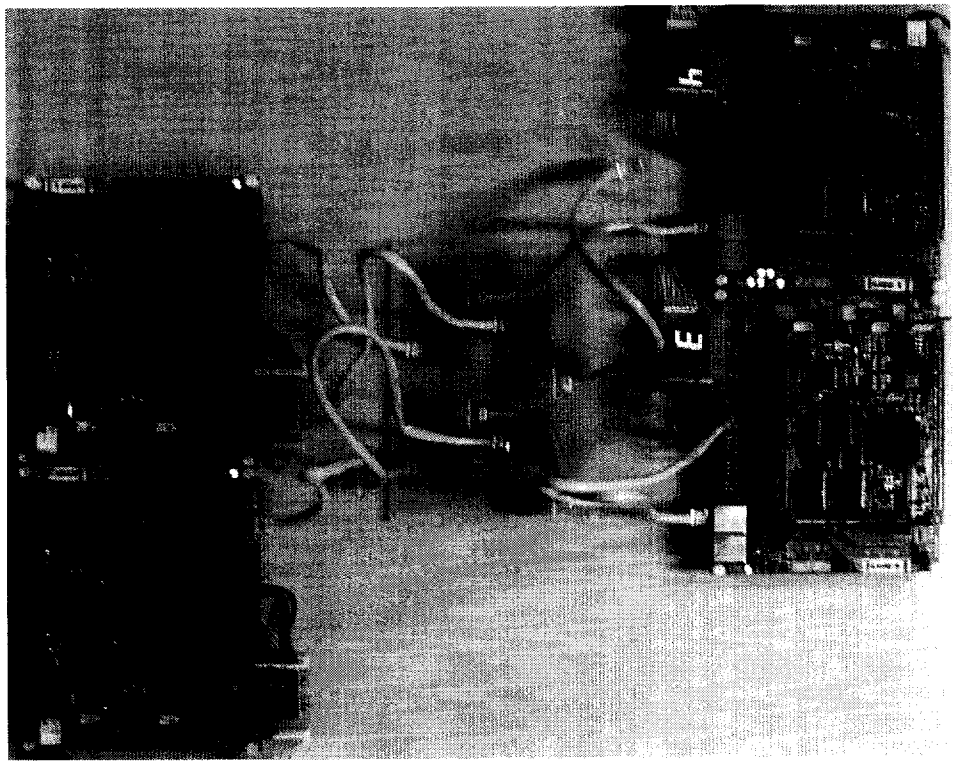

Photo. 4 Hardware Fault on the Communication Channel

4.3.4 소프트웨어적인 결함시의 통신 상태

소프트웨어적인 결함은 노드 1과 노드 2가 송수신 라인을 모두 지배하는 것으로서 구현하였다. Photo. 5에서 보듯이 두 노드(노드1, 2)가 송수신 라인을 모두 지배하는 상황에서도 그렇지 않은 나머지 노드들(노드3, 4)은 송수신 하는데 아무런 지장이 없음을 알 수 있다. 또한 나머지 2개의 노드들의 동기화가 잘 이루어짐을 알 수 있다.

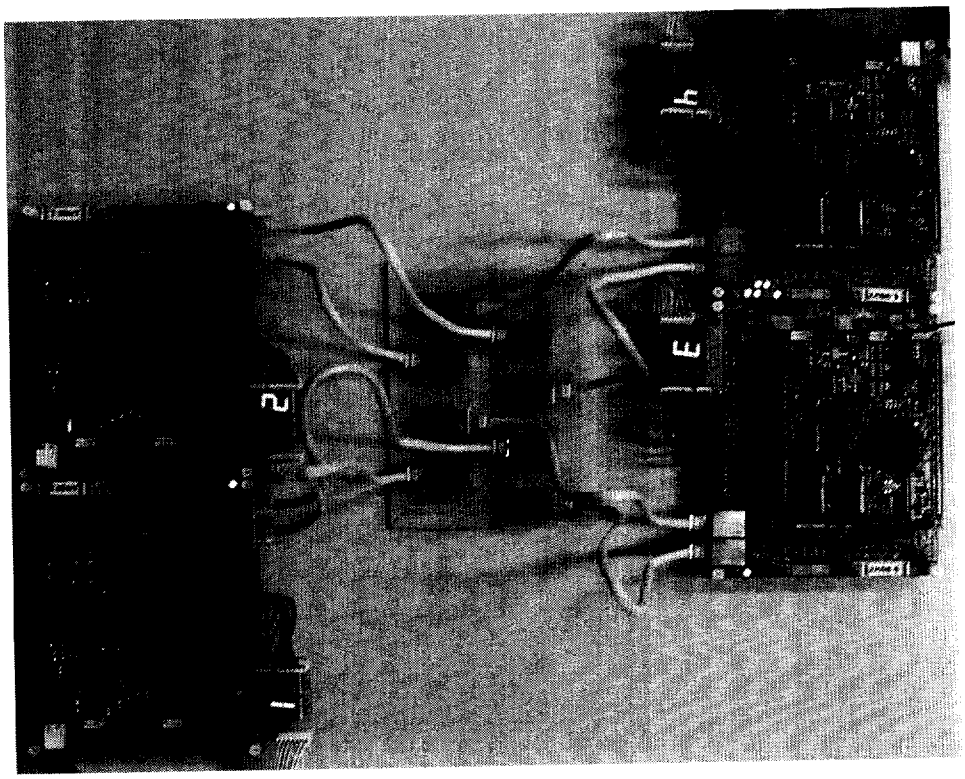

Photo. 5 Software Fault on the Communication Channel

제5장 결론 및 향후 계획

본 연구에서는 분산된 제어시스템(ECU)간에 소프트웨어적이거나 하드웨어적인 결함을 허용할 수 있는 통신을 위해 TTP 통신방식을 제안하였다.

과거에 사용하였던 분산된 제어기들은 독립적으로 운용되어지므로 각각의 분산된 제어시스템들에게 개별적으로 입력신호를 주어야하는 단점을 지니고 있었다. 따라서 각각의 분산된 제어시스템간의 보다 효율적인 자원의 공유와 확장성 및 결함 허용성을 보장하기 위해서 네트워크 기반의 전자제어 시스템이 필요하게 되었다. 그리고 기존의 이벤트 트리거(Event Trigger) 통신방식인 CAN(Controller Area Network), LIN(Local Inter-connect Network), MOST(Media Oriented System Transport) 등은 시간의 흐름에 무관한 통신 방식으로서 안전성이 극히 중요한 응용시스템의 요구사항인 결함 허용성(fault-tolerance)을 보장하는데 다소 무리가 있었다. 이러한 결점을 보완하기위해서 예측 가능한 시간의 흐름에 대한 이벤트 처리를 하는 시간 트리거(Time Trigger) 방식인 TTP 통신을 제안하였다.

본 논문에서는 TTP 통신의 개념과 통신방법을 설명하였고, 자체개발한 통신모듈을 이용하여 노드간의 정상적인 통신이 이루어지는 결함 허용성을 보여주었다.

본 논문의 연구결과로 향후 TTP 통신시스템을 이용하여 안전도가 극히 중요한(Safety-critical) 시스템인 자동차의 X-By-Wire용 제어기, 항공기의 분산제어기, 발전소의 분산제어기 또는 고속열차의 분산제어기 간의 통신에 결함을 허용하는 통신방법을 제안할 수 있다.

참 고 문 헌

[1] 이민광, 윤마루, 신민석, 선우명호, "차체제어시스템 개발을 위한 네트워크기반 HILS 환경 개발", KSAE, 2004

[2] 김창환, "IT 텔레매틱스 서비스 동향," 전자정보센터, 전자부품연구원, Oct. 2003

[3] 양성모 "자동차 네트웍 시스템 분석을 위한 시뮬레이션 알고리즘 개발에 관한 연구", 한양대 석사학위 논문, 1999

[4] Gunther Bauer, "Transparent Fault Tolerance in a Time-Triggered Architecture", Technical University of Vienna, April. 2001

[5] Hermann Kopetz, Andreas Damm, Christian Koza, Marco Mulazzani, Wolfgang Schwabl, Christoph Senft, Ralph Zainlinger, "Distributed Fault-Tolerant Real-Time Systems: The Mars Approach", IEEE, Feb. 1989

[6] TTTech, "Time-Triggered Protocol TTP/C High-Level Specification Document", July. 2002

[7] Hermann Kopetz, "REAL-TIME SYSTEMS - Design Principles for Distributed Embedded Applications", BOSTON. Kluwer Academic Publishers, 1977

[8] Reinhard Maier, "EVENT-TRIGGERED COMMUNICATION ON TOP OF TIME-TIGGERED ARCHITECTURE", IEEE, 2002

[9] Bosch, "CAN Specification v2.0", Bosch, 1991

[10] Wense. H, "Introduction to Local Interconnect Network", SAE World Congress, Detroit, March. 2000

[11] MOST Cooperation, "MOST Specification Rev 2.3", Aug. 2004

[12] Hermann Kopetz, "The Time-Triggered Architecture", IFP WG 10.4, Jan. 2004

[13] FlexRay-group, "FlexRay Communication System Protocol Specification Ver2.0", June. 2004

[14] Johan Nilsson, "Real-Time control Systems with Delays", PhD Thesis, Lund Institute of Technology, 1998

[15] Olof Bridal, Rolf Snedsbol, Lars-Ake Johansson, "On the Design of Communication Protocols for Safety-Critical Automotive Applications", IEEE, 1944

[16] Warrendale, "Control Area Network, an Invehicle Serial Communication Protocol", Society of Automotive Engineers, 1990

[17] Warrendale, "Class C Application Requirements", SAE Handbook, 1994

[18] Hakan Sivencrona, Chalmers, Johan Hedberg, SP Olle Bridal, Volvo, "Design Principles for dependable time-triggered control systems", December 13. 2000

[19] Howard Curtis and Robert France, "Time Triggered Protocol(TTP/C): A Safety-Critical System Protocol", EE382C Literature Survey, Oct 24. 1999

[20] Rene Hexel, "Validation of Fault Tolerance Mechanisms in a Time Triggered Communication Protocol using Fault Injection", Technical University of Vienna, Aug. 1999

[21] Roman Pallierer, "Prototype Implementation and Evaluation of TTP/C", Technical University of Vienna, Sept. 1996

[22] David Bradbury, "Simulation of a Time Triggered Protocol", University of Sydney, Australia

[23] H. Kopetz, "A Comparison of CAN and TTP", Technical University of Vienna

[24] H. Kopetz, "Composability in the Time-Triggered Architecture", Technical University of Vienna, Jan. 2000

[25] Motorola, "MPC555/556-USER'S MANUAL", Oct. 2000

감사의 글

미흡한 저의 논문이 완성되기까지 부족한 저에게 항상 진심어린 애정과 관심으로 지도하여주신 변기식 지도교수님께 진심으로 감사드립니다.

바쁘신 와중에도 저의 논문을 심사하시면서 부족한 저에게 지도와 격려를 아끼지 않았던 황용연 교수님, 김만고 교수님께 감사드립니다. 대학원에 들어와 석사과정을 마치기까지 여러 가지로 부족한 저에게 진심 어린 충고와 격려로 지켜봐 주신 이형기 교수님, 최연욱 교수님, 김남호 교수님, 안영주 교수님, 신춘식 선생님께도 진심으로 감사의 말씀을 드립니다.

아울러 함께하는 시간동안 항상 따뜻하게 저를 반겨주고 격려해주신 정상철 선배, 강경덕 선배, 김관형 선배, 배상범 선배, 얼마전에 결혼한 이상준 선배, 임상진 선배에게 감사드리고, 2년이라는 결코 짧지 않은 대학원 과정에 힘들 때면 서로의 말벗이 되어주었던 친구 박기원, 정광덕, 문홍득, 김동원, 서재현, 신재현에게 감사의 말을 우선 전하고, 신호처리 연구실에 있는 동안 동고동락하면서 힘이 되어준 공형식, 이종민, 박정민, 이형우, 유은상 후배, 사랑하는 김형수 후배, 로봇제어연구실의 김태곤, 문선호, 김병기 후배, uPC의 이철성 후배, 시스템제어연구실의 류나이 후배, 계측 및 신호처리연구실의 서현수, 류수원 후배에게도 감사의 말을 전합니다. 그리고 저의 졸업이 있기까지 많은 도움을 주신 정미숙 조교 선생님에게도 고마움을 전합니다.

오늘의 저를 있게 해준 사랑하는 부모님과 저에게 물질적, 정신적으로 큰 힘이 되어준 존경하는 형님과 형수님, 사랑하는 조카 경민이, 수민이, 그리고 항상 힘들 때 옆에서 용기와 힘을 북돋워주고 건강을 생각해서 항상 아침을 꼬박꼬박 챙겨준 사랑하는 아내와 더운 여름에 이 험난한 세상과 맞서 싸울 사랑하는 나의 아가에게 이 논문을 바칩니다.

2004년 12월

김 성 훈 올림