⊙ 研究論文

이중마이크로프로세서의 이용과 분산처리기법을 도입한 선박용 원격제어 시스템의 설계에 관한 연구

홍순철* · 정경열* · 류길수**

A Study on the Design of Ship's Distributed Remote Control System Using Dual - Microprocessor

S. C. Hong · K. Y. Chung · K. S. Rhyu

Key words : Remote control system(원격제어시스템), Integrated control system(통합제어 시스템), Dual - microprocessor(이중마이크로프로세서)

Abstract

In this paper design and implementation of ship's distributed remote control system using dual-microprocessor is presented for real time process. The proposed system is implemented with the single chip microprocessors in tightly coupled mode and results in speed up of s_p= 1.74. Under the assumption that the nodes are interconnected in multidrop, the overall system performance such as average throughout-delay characteristics and effective throughput are analyzed using M/G/1 queueing model, and results show that the proposed node can be used in medium sized distributed monitoring and control system.

1. 서 론

선박용 원격 제어 시스템의 주요 기능은 제어와 감시이다. 이를 과거에는 중앙 집중 방식으로 처리하였으나 최근에는 각 선박의 기능별로 분산하여 처리할 수 있는 분산형 제어방식을 많이 채택하고 있다.

분산형 제어방식은 모든 자원을 각 노드 프로세서 별로 분산하여 처리하기 때문에, 각 노드의 자율성이 어느 정도 인정되어 중앙 집중 제어 방식보다 시스템 성능의 안전성, 신뢰성의 향상, 자원 공유의 용이함, 그리고 확장성 등이 우수하며 또한 통합제어시스템(integrated control system)의 적용이 용이한 장점을 가지고 있다[1~3].

분산형 제어시스템의 구현을 위해서는 노드 프로세서간의 분산된 자원을 서로 공유하기 위한 통

* 정회원, 한국기계연구원
** 정회원, 한국해양대학교

신망이 필수적이며 분산 제어에서의 통신망은 신뢰성이 높고 구조가 간단해야 하며 확장성이 있어야 하기 때문에 보통 버스나 트리구조의 LAN (Local Area Network)으로 구성된다. 이 때 분산된 자원의 처리 및 긴급 메세지 송수신 등은 각 노드 프로세서가 제어와 통신을 처리하는 제한된 응답 시간 범위내에서 수행되므로 단일 프로세서를 사용하여 제어와 통신을 함께 처리하도록 노드 프로세서를 구성하면, 노드 프로세서는 메세지를 송수신하는 시간만큼 통신 부담(communication load)을 가지게 되어 전체적으로 제어사이클이 길어지는 단점이 있다. 즉, 제어 응답 시간이 늘어나 실시간 처리의 구현이 어려워지게 된다. 이러한 문제점을 해결하기 위해서는 이중 마이크로프로세서를 사용하여 제어와 통신을 적절히 분담시켜 동시에 수행할 수 있는 노드 프로세서의 설계가 요구된다. 일반적으로 이중 마이크로프로세서 시스템에서는 노드내의 두 프로세서간의 통신을 위해 공통 공유메모리구조가 많이 사용되고 있는데, 그 이유는 multiple - bus 공유메모리 구조가 multi-point - multibus 구조보다 데이타 전송 능력이 떨어지지만 개발과 구현이 간단하고 비용이 적게 들기 때문이다[4].

한편 분산형 제어 시스템의 성능은 각 노드의 성능 뿐만 아니라 사용된 통신망의 구조 및 시스템의 규모등이 종합적으로 검안되어 평가되어야 한다. 따라서 각 제어 노드의 설계를 바탕으로 분산 제어 시스템 전체의 성능 해석 방안이 뒷바침되면 노드의 설계해석을 좀더 융통성 있고 용이하게 할 수 있다.

이상의 관점으로부터 본 논문에서는 아직까지 정규화된 시스템 사양이 없는 선박용 원격제어 시스템을 대상으로 하여 분산형 원격제어시스템에서 제어와 통신을 각각의 프로세서에 분담하여 동시에 수행시킴으로써 실시간 처리 성능을 향상시키는 방법을 제안한다. 이를 위해 통신 기능이 내장된 8751 단일 칩 마이크로프로세서를 사용하여 이중 마이크로프로세서 노드시스템을 구현하였으며 실험을 통하여 속도제고율등 그 성능을 평가하였다. 또한 선박에 있어서 원격제어시스템은 열악한 환경에서도 신뢰성이 보장되어야 하므로 설계된 이중 마이크로프로세서 노드시스템을 사용하여 멀티드랍방식[5]의 분산형 원격제어시스템으로 구성하였으며, 이때 통신망에서의 전송 지연 시간을 M/G/1 待期모델로 해석하고, 각 프레임 크기에 따른 전송효율도 고찰함으로써 설계된 이중 마이크로프로세서 시스템의 실시간 분산 제어 노드로서의 타당성을 검토하였다.

2. 이중 마이크로프로세서시스템의 설계

2.1 다중프로세서 시스템의 구조

하나의 시스템이 통신과 제어와 같이 두 개 이상의 모듈을 동시적으로 수행하기 위해서는 다중프로세서의 구조가 필요하지만, 이러한 구조에서는 프로세서들이 동시적으로 버스의 접근을 막기위해 일정한 규칙을 따라 버스를 사용하도록 해야한다. 즉 요구 신호를 보내고 버스 사용권을 얻으면 버스를 사용하게 한다. 버스중재자는 여러 요구신호에 대해 하나의 프로세서만이 허가신호를 받도록 두 신호사이의 상호작용을 해주고, 허가신호들 사이에 적당한 보호시간을 두어 허가전환이 일어날 때 버스 경쟁이 일어나지 않도록 각 프로세서에 우선 순위를 정하여 하나의 프로세서만이 공통버스에 접근하게 해야한다. 우선 순위를 정하는 방법으로 데이지체인(daisy chain), 폴링, 독립 요구방식 등이 있다[3].

데이지 체인방식은 값이 싸고 선의 수가 적고 간단하지만, 우선 순위가 허가신호의 전달 순서에 의해 고정되고 느린 허가신호 전달로 인해 연결할 수 있는 프로세서의 수가 제한된다. 또 한 시스템이 고장나면 전 시스템이 동작을 할 수 없다.폴링 방식은 제어기가 각 모듈 주소를 순서적으로 발생시키고 버스 사용권을 요구한 프로세서가 자기주소를 받으면 busy선을 동작시키고 버스를 사용하며 폴링 순서를 바꿈으로써 우선 순위를 가변시킬 수 있다는 데 그 특징이 있다. 독립요구 방식은 각 요구 신호에 대해 우선순위 디코더를 사용하여 병렬적인 방법으로 우선순위를 정하므로 다른 방법보다 접근시간이 빠르나 버스제어 논리회로에 연결되는 선의 수가 많다.

따라서, 본 논문에서는 저가의 프로세서를 사용하면서 빠른 수행과 함께 우선 순위를 임의로 바꿀 수 있는 방식으로서 독립 요구방식을 택하고 노드 프로세서의 수를 확장할 수 있는 시스템으로 구성하기로 한다.

한편 프로세서의 수와 시스템 성능사이의 관계를 버스에 접근하는 평균시간과 전체 명령어를 수행하는데 소요되는 시간의 비율 즉 버스 사용률, BUF(Bus Utilization Factor)로서 판단할 수 있으며 이는 다음과 같이 정의 된다.

$$BUF=\frac{\text{평균버스접근시간}}{\text{명령어가 수행되는 전체 시간}} \qquad (1)$$

사용빈도가 높은 명령어와 그렇지 않는 것을 고려하여 BUF를 구해야 하나 응용되는 프로그램에 따라 명령어의 사용 횟수가 가변적이므로, 본 논문에서는 명령어별로 동일한 가중치를 주어 계산하기로 한다.

2.2. 이중마이크로프로세서 시스템의 구현

2.2.1. 하드웨어설계

이중 마이크로프로세서 시스템은 두 프로세서가 서로 독립적이면서 보완하는 관계를 가져야하고 또 프로세서의 BUF를 고려할 때 외부 데이타 메모리와 내부 프로그램 메모리 구조로 구성되어야 하며, 각각 I/O 장치와 전용의 메모리에 접근할 수 있는 로컬버스와 공유메모리에 접근할 수 있는 공통버스를 가지고 있어야 한다. 공통 공유메모리 구조는 하나의 공통경로를 사용하므로 주파수 대역과 속도의 한계가 있다. 공통버스 경쟁을 줄이는 방법으로 각 프로세서 모듈은 로컬 메모리에서 명령어나 참고 코드들을 가져오고 공유메모리는 두 모듈간의 통신으로 사용한다.

8751 프로세서의 메모리구조는 수행할 명령어나 참조 데이타를 저장하는 프로그램 메모리와 일반 데이타를 저장하는 데이타 메모리로 나누어져 있기 때문에 각 구조에 따른 BUF가 다르다. 프로그램 메모리와 데이타 메모리가 내부 메모리로 구성되는 경우는 시스템이 버스에 접근하지 않으므로 BUF와는 무관하다.

외부 데이타 메모리와 내부 프로그램 메모리인 경우는 CPU가 내부 프로그램 메모리로부터 실행할 때 PSEN 신호는 동작하지 않고, 프로그램 어드레스도 출력되지 않으므로 CPU는 버스에 접근하지 않는다. 외부 데이타 메모리에 접근하는 MOVX 명령어의 1사이클 S5에서 2사이클 S4까지 14클럭 동안 공통버스에 접근한다. 8751 명령어는 1사이클 명령어 45개, 4사이클 명령어 2개로 이루어져 있고 BUF를 구하기 위하여 명령어 종류별로 동일한 가중치를 주어 계산하면 전체 명령어 실행시간은 $(64+45\times2+2\times4)\times12=1944$(CLK)이고 평균 버스 접근 시간은 $4\times14=56$(CLK)이 되고 BUF는 56/1944=0.0288이다.

외부 프로그램 메모리와 내부 데이타 메모리일 때는 위와 반대인 경우이므로 전체 명령어 실행시간은 1944(CLK)이고 평균 버스 접근 시간은 1944 − 56 = 1888(CLK)이다. BUF는 1888/1944 = 0.971로 주어진다.

따라서 8751을 외부 RAM과 내부 ROM으로 구성하면 공통버스에 연결될 수 있는 프로세서의 갯수를 의미하는 BUF의 역수는 34.17(1944156)이 되므로 이중 마이크로프로세서 시스템은 확장 여유가 많다. 버스 제어 논리회로 설계는 기본적인 2개의 요구 신호외에 6개의 입력을 쉽게 확장할 수 있도록 하였다.

Fig. 1은 8751 프로세서로 구성한 공유메모리

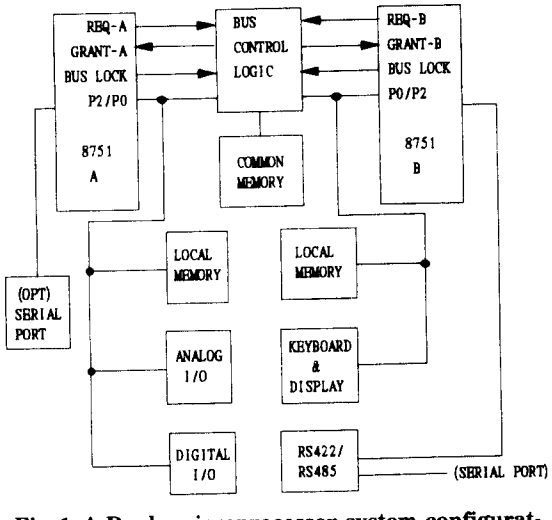


Fig. 1 A Dual microprocessor system configuration based on 8751

시스템을 보이고 있다. 공유 메모리의 어드레스는 두 프로세서에 똑같은 번지로 할당되고 포트 0은 어드레스와 데이타 이중포트로 사용되므로 데이타 입출력을 위한 양방향성 버퍼와 어드레스를 위한 버퍼를 설치했고 포트2는 어드레스 버스용으로 버퍼를 두었다. 각 프로세서들은 RD, WR 신호와 BCL(Bus Control Logic)의 출력신호와 조합하여 공유메모리에 접근할 수 있다. RD신호는 공유메모리에서 데이타를 읽을 때 메모리를 enable시킬 뿐만 아니라 양방향성 버퍼의 방향을 결정하는데 쓰인다. ALE신호는 BCL의 시스템 클럭으로 사용된다(Fig. 2).

BCL을 독립요구 방식으로 구현하였으므로 프로세서는 요구신호를 보낸뒤 허가신호가 올 때까지 wait 상태에 머물러야 하나 8751 프로세서는 wait 상태가 없으므로 사용자가 소프트웨어적으로 설정해야 한다.

제1단계 : 각 프로세서가 요구 포트에 레벨 변화로써 BCL에 버스 사용권을 요구한다.

제2단계 : busy 신호가 "0"이면 우선 순위에 따라 허가 신호를 발생시키고 busy 신호를 만든다.

제3단계 : 버스 사용권을 부여받은 상태로 1바이트를 읽거나 쓰는 과정에서 발생되는 RD/WR 신호는 요구신호를 리셋시킨다.

제4단계 : busy 신호를 T-F/F으로 클리어시킨다. busy 신호를 T-F/F을 사용하여 발생시키는데 이는 버스를 사용하기 시작하는 허가신호의 상승에지에서부터 버스사용을 완료하는 RD/WR신호의 상승에지까지만 버스 신호를 발생시킴으로써 버스 사용시간을 최소화한다.

제5단계 : BCL은 그동안 래치되어 있는 다른 요구신호들을 처리하거나 아니면 새로운 요구신호를 처리할 준비를 한다.

제6단계 : 래치 신호는 busy신호, ALE신호와

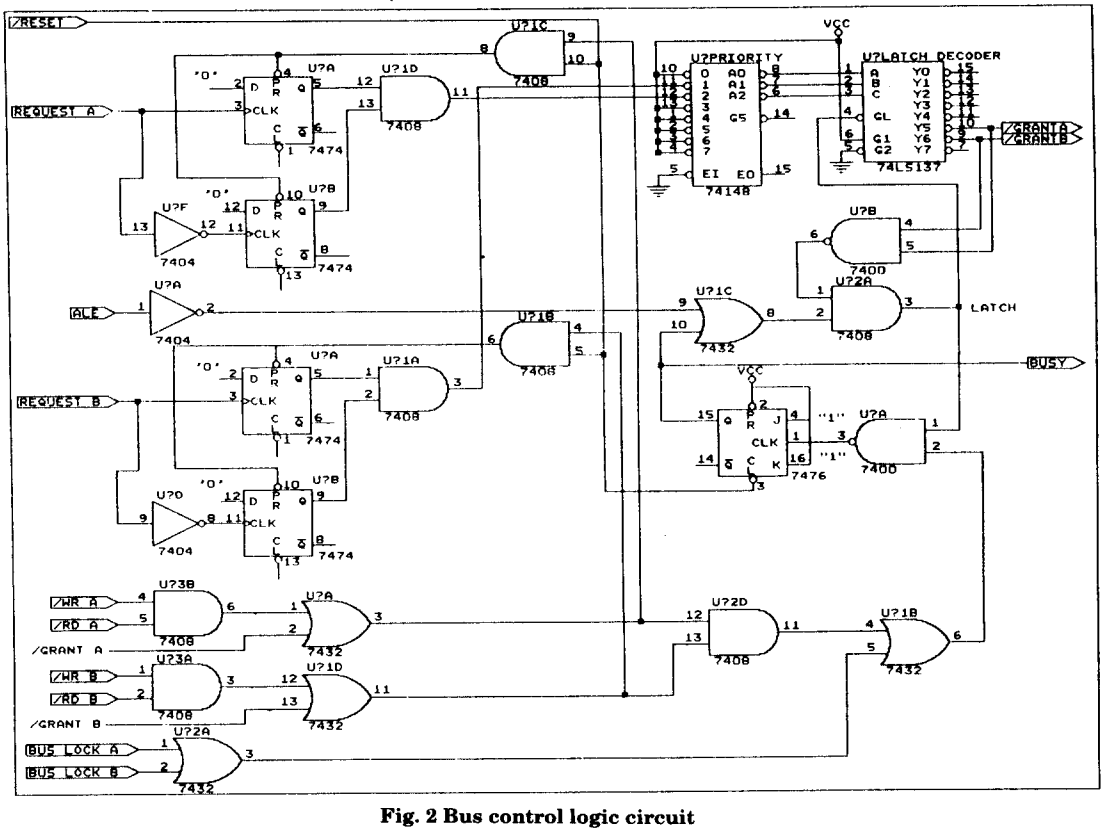


Fig. 2 Bus control logic circuit

함께 허가신호를 고정시켜 서비스를 받고 있는 요구신호를 서비스가 완료될 때까지 보호한다. 래치신호와 RD/WR 신호만의 결합으로 2개의 하강에지를 만들 수 있으나 다음 허가신호에서 하강에지를 만들기 위해 래치신호는 busy 신호가 리셋된 후 ALE에 의해 클리어되게 한다(Fig. 2 참조).

공유 메모리 구조로 된 시스템의 주된 병목현상은 여러 형태의 데이터를 공유하는데에서 비롯된다.

본 논문에서는 세마포어[4]를 이용하여 하나의 프로세서가 RMW(Read Modify Write) 동작을 수행하고 있는 동안 세그먼트락을 사용하여 다른 프로세서가 접근하지 못하도록 하고 있다. 세마포어는 두 개의 동작으로 나누어지며 각기 다른 메모리 위치에서 개별적으로 동작을 한다. P 동작은 다른 어떤 프로세서보다 우선하여 RMW 동작을 하도록 해야 하는데 BCL의 버스락(bus lock) 신호가 이를 가능하게 한다. P 동작은 프로세서가 버스 사용권을 얻고 버스락을 행한 후, S가 "1"이면 critical부분을 실행하고, S가 "0"이면 버스락을 해제하고 앞의 동작을 반복한다. P 동작이 끝나면 V 동작으로 S를 "0"으로 하여 세그먼트락을 해제한다. 두 모듈의 작용은 다음과 같은 순서로 이루어진다.

```
(1) P - operation(semaphore=1)
    TRY    : SEND REQ
    WAIT1  : IF(GRANT=0 ) WAIT1
             SET BUS_LOCK
             READ SEMAPHORE
             IF(SENAPHORE=0) OK
             CLR BUS_LOCK
             RESET GRANT
             JMP TRY
    OK     : SEMAPHORE=1
             CLR BUS_LOCK
             WRITE SEMAPHORE
(2) V - operation(semaphore=0)
             SEND REQ
    WAIT   : IF(GRANT=0) WAIT
             CLR SEMAPHORE
```

2.2.2. 망의 형태

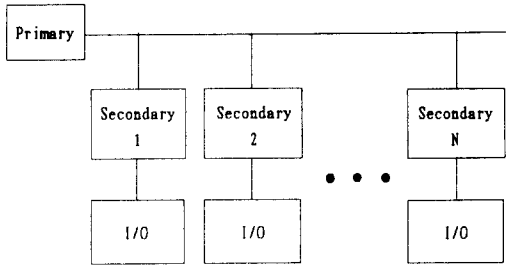


Fig. 3 Multidrop network structure

망의 선택기준은 자료의 형태와 통신의 효율과 속도를 결정하는 토폴로지와 전달 매체이다. 토폴로지는 보통 링, 스타, 버스/트리로 나누어진다. 이 중에서 버스/트리 토폴로지가 비교적 구성이 간단하고 신뢰성이 높다. 그러나 여러개의 노드들이 연결되어 있으므로 전달매체의 접근을 제어하는 문제점도 가지고 있다.

시스템 구성에 사용된 토폴로지는 Fig. 3에 나타낸 것과 같이 버스/트리의 한 형태인 멀티드랍형식으로 두개의 선으로 이루어진 half duplex이다. 프로토콜은 primary 스테이션(이하 primary)과 노드의 통신 프로세서에 해당하는 secondary 스테이션(이하 secondary)이 있는 폴링 방식으로 secondary는 primary가 poll을 할 때만 통신을 하므로 물리적 형태는 멀티드랍이지만 로직적으로 스타형태이다. 만일 기본적으로 physical layer와 data link layer를 제공하도록 구성하였고 physical layer는 RS485나 RS422등으로 구현할 수 있고 data link layer는 asynchronous multidrop이다.

Primary는 데이타 링크의 access와 링크 level의 error recovery, 그리고 정보의 흐름에 대해 조정을 해야 한다. error control는 time out을 사용한 두개의 ACK를 가진 enqury형식으로 구성하고 정보가 하나의 secondary에서 다른 하나의 secondary로 전달되도록 하기 위해 메세지 스위치 역할을 한다.

2.3. 소프트웨어 설계

secondary의 data link interface는 Fig. 4와 같이 세 부분으로 나누어진다.

Fig. 4에서 LDS(Logical Disconnect State)는 물리적으로 연결되어 있으나 정보 전달을 할 수 없

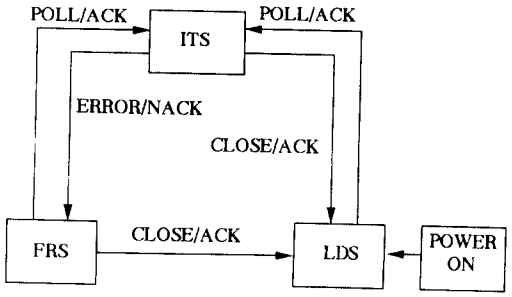


Fig 4. State diagram of secondary station

Table 1 Secondary station response primary station

Data Link States 0	POLL	CLOSE	READ	WRITE
Information Transfer States	ACK	ACK	NACK ACK	NACK ACK
Logical Disconnect States	ACK	LDS	LDS	LDS
Frame Reject State	ACK	ACK	FRS	FRS

는 상태이다. FRS(Frame Reject State)는 secondary가 primary에 대해 소프트웨어적인 동기를 잃어버리거나 어떤 종류의 error가 발생했을때 이 상태로 된다. ITS(Information Transfer State)는 primary가 poll을 하여 secondary와 정보를 주고 받는 상태이다.

Power가 전달될 때 secondary는 LMS에 있으며 primary가 POLL을 보내면 ITS로 들어가고 ACK를 primary에 보낸다. primary가 CLOSE를 전송하면 ITS에서 LDS로 간다. secondary가 ITS에서 primary가 보낸 command를 인식하지 못할 때, 전송 되어온 데이타에 에러가 있을때 FRS로 들어간다. Fig. 4와 Table 1은 secondary 스테이션 driver의 기능을 보여준다.

Primary는 데이타 링크의 접근과 링크 레벨의 에러 복원, 그리고 정보의 흐름에 대해 조정을 해야한다. 에러 제어는 time out을 사용한 두개의 ACK를 가진 enquiry 형식으로 구성하고 정보가 secondary에서 secondary로 전달되도록 하기 위해 메세지 스위치 역활을 한다. primary는 244개까지 연결되는 모든 스테이션의 주소를 알고, 각 스테이션에 관한 필요한 기록도 가지고 있어야 한다. 에러가 발생했을때 복원을 위해 time out 기능에서 primary가 기다리는 최소 wait 시간은 다음과 같은 성분의 합이다.

1. secondary까지의 전송 지연
2. clear to send time to DCE secondary
3. secondary processing 지연
4. primary까지 전송 지연
5. 최대 프레임의 전달 시간

3. 성능평가 및 고찰

3. 1. 이중마이크로프로세서 시스템의 성능

본 논문에서 제안한 이중 마이크로프로세서 시스템의 성능을 throughput, 비용 대 속도 비율, 신뢰성과 전송효율에 대해 고찰하였다.

이중 프로세서 성능 향상에 있어서 중요한 문제는 decomposition과 상호작용(interaction)을 어떻게 처리하느냐에 있다. 즉, 두 프로세서간에 업무를 균형있게 할당하여 두 프로세서의 업무 비중이 비슷하게 해야한다. 그 한 예로 $E=AB+CD$ 행렬 연산을 할 때 모든 데이타가 공유 메모리에 있다고 생각하고 decomposition은 A프로세서가 $A\times B$를 계산하는 동안 B프로세서는 $C\times D$를 계산한다. 행렬의 더하기 연산에서는 A 프로세서가 홀수행 부분을 B 프로세서가 짝수행 부분을 계산한다. 이 알고리즘은 문제해결에 두개의 프로세서들을 동시적으로 참여 시켰다. 프로세서들간의 상호작용은 공유 메모리를 통해 이루어진다.

Throughput은 문제 해결에 소요되는 시간과 반비례 관계를 가진다. 이중 마이크로프로세서 시스템과 단일 마이크로프로세서 시스템의 throughput을 비교하기 위해 실험적으로 위에서 언급한 행렬 연산 문제에 해결에 소요되는 시간을 측정하였다. 연산하기 직전에 8751의 내부 타이머를 동작시켜 연산이 완료되었을때의 타이머의 레지스터 값으로 실행시간을 측정하였다. Fig. 5은 행렬의 차수에 따른 실행시간을 보이고 있다.

행렬의 차수가 적을때는 단일 프로세서와 이중 프로세서는 비슷한 성능을 보이나 차수가 커짐에 따라 이중 마이크로프로세서 시스템이 더 나은 성능을 가진다. 속도 제고율은 각 차수에 따라 약간

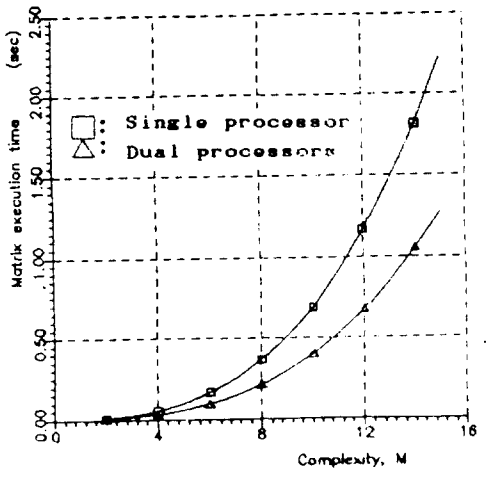


Fig. 5 Execution time versus matrix complexity for single microprocessor and dual microprocessor system.
Matrix Order Is(M,M)

차이가 있지만 평균적으로 Sp=1.74이고 한 프로세서의 병렬 연산 효율은 Ep=0.87 정도이다.

3.2. 망의 전송 성능 분석

제안된 시스템의 망의 성능을 전송효율[7], 평균 전송 지연시간[8]으로 알아보았다.

3.2.1 전송효율

DLC(Data Link Control) 프로토콜의 전송에 의한 효율을 고려하기 위해 효과적인 전송비율을 알아보면 Re는 다음과 같이 정의된다.

$$Re = \frac{\text{목적 스테이션에서 받는 데이타의 수}}{\text{데이타를 받기위해 걸리는 총 시간}} \tag{2}$$

정의에 따라 각 항을 정리하면[11]

$$Re = \frac{K - Kh}{\frac{K + Nca}{R} + 2Tp + Tc} \tag{3}$$

K : 전체 프레임 비트수
N_h : 오버헤드 비트 수
N_{Ca} : 명령어/응답 비트수
R : 채널 비트 비율
T_p : 전송지연시간
T_c : 총 clear to send 시간

$$Re = \frac{11\times(N+5) - 11\times5 - 3\times N}{\frac{11\times(N+5)+11\times4}{R} + 2Tp + Tc}$$

단, N : 바이트의 수(1, 2,…255, 256)

로 된다. 응답지연 시간과 전송의 에러가 없다고 가정하면

$$Re = \frac{R\times8\times N}{11\times(N+9)} \tag{4}$$

로 간단해진다. 하나의 비트에서 에러가 일어날 확률을 p라 하면 한 FRAME에서 에러가 날 확률은 $P=1-(1-p)^k$이고 j번째 FRAME 전송에서 성공할 확률은 $P^{j-1}(1-P)$이다. 평균적으로 전송해야 할 횟수는

$$\sum_{j=1}^{\infty} jP^{j-1}(1-P) = \frac{1}{1-P} \tag{5}$$

이고 이 횟수 만큼 전달시간이 길어지므로

$$Re = \frac{8\times(1-P)\times R\times N}{11\times(N+9)} (N = 1, 2, \cdots, 255, 256) \tag{6}$$

로 표현할 수 있다.

식(6)을 이용하여 전송비율과 바이트 수와의 관계를 그림으로 나타내 보면 Fig. 6과 같으며, 전송하고자 하는 데이타를 100 바이트 정도로 한 프레임을 구성하는 것이 효과적임을 알 수 있다.

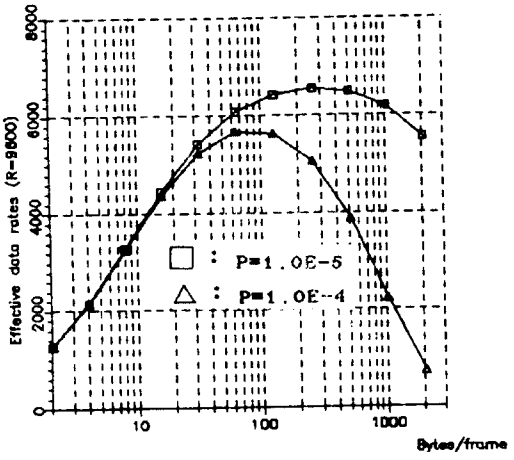


Fig. 6 Variation of effective transmission rate with frame length for error rates p, 10^{-4} and 10^{-5}, R=9600

3.2.2. 평균 전송 지연 시간

한 스테이션이 서비스를 받기 위해 기다려야 하는 기간이 망의 성능을 결정하는 중요한 요소이다. 우선 평균 폴링 사이클 시간 Tc를 알아 보기 위해 walk - time w를 설정한다[7].

Walk time w는 어떤 스테이션에서 다른 스테이션으로 poll을 전달하는 데 걸리는 시간과 중앙 스테이션과 동기를 맞추기 위해 소요되는 시간을 의미하며 poll을 전달하기 위한 모든 전달시간과 지연시간을 포함한다. 성능 분석에 앞서 몇가지 사항을 가정한다.

a. 각 스테이션에 도달되는 패킷은 통계적으로 평균 도달 비율 λ를 가지는 동일한 Poisson 프로세스이고, 패킷의 길이 X도 동일한 통계적 성질을 가진다.

b. 모든 스테이션사이의 walk time 및 채널 전파 지연 시간은 일정하고 같다.

[1] 노드 대 호스트 전송 지연 시간인 경우 : 스테이션의 수를 M, channel bit rate를 R, 평균 패킷의 수를 N_m이로 두면 평균 폴링 사이클 시간 $T_c=M[N_mX/R+\omega]$이고, $N_m=\lambda T_c$의 관계를 가지므로

$$Tc = \frac{Mw}{1-M\lambda X/R} = \frac{Mw}{1-S} \tag{7}$$

로 되고 S는 throughput을 의미하는 $M\lambda X/R$이다.

어떤 스테이션에 도착한 패킷이 서비스를 받기 위해 기다려야 할 시간 W는 두가지 성분으로 구별된다. W_1은 다른 스테이션이 서비스를 받을때 기다리는 시간이고 W_2는 특정 스테이션에서 먼저 전달되어온 데이타들이 서비스를 완료할 때까지의 지연시간이다(Fig. 7).

Service time 은 $\lambda TcX/R=\rho Tc$이고 $(1-\rho Tc)$은 지연 시간이 되므로 평균적인 지연시간은

$$W_1 = \frac{(1-\rho)T_c}{2} \tag{8}$$

$$= \frac{M_W(1-\rho)}{2(1-M\rho)} \tag{9}$$

특정한 스테이션에서 서비스를 받기위해 기다리는 시간 W_2는 Queueing 모델의 M/G/1 모델에서 구할 수 있다.

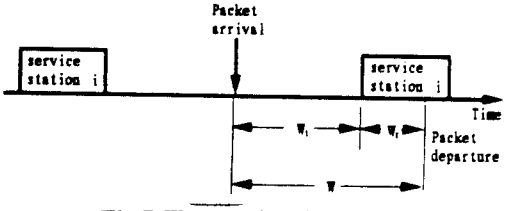


Fig 7. Waiting time for a facket

$$W_2 = \frac{SX^2}{2XR(1-S)} \tag{10}$$

한 스테이션이 polling 받기 위해 기다려야할 전체 평균 시간은 패킷을 전송하는데 걸리는 시간과 전달 지연 시간 τ, walk time 합이다.

$$T = \frac{X}{R} + \tau + W$$

$$= \frac{X}{R} + \tau + \frac{M\omega\left(1-\frac{S}{M}\right)}{2(1-S)} + \frac{SX}{2R(1-S)} \tag{11}$$

T를 한 FRAME으로 전송하는 시간 X/R로 나누어 정규화한 변수 $\hat{T}$를 구하면

$$T = \frac{\hat{T}}{\frac{X}{R}} = 1 + a + \frac{M\hat{w}\left(1-\frac{S}{M}\right)}{2(1-S)} + \frac{S}{2(1-S)}$$

$$\left(\hat{w} = \frac{wR}{X},\ a = \frac{\tau R}{X}\right) \tag{12}$$

이 된다.

[2] 노드 대 노드 전송지연시간인 경우 : 평균 polling 사이클 시간은 노드 대 호스트 전송과 거의 같은 방법으로 분석하며 앞의 polling 사이클 시간에 호스트에서 전송을 위한 시간 NmX/R과 지연되는 시간 d가 합해진다.

$$T_c = M\left[2N_m\frac{X}{R} + w + d\right] \tag{13}$$

$$N_m = \lambda T_c$$

$$T_c = \frac{M(w+d)}{1-2M\lambda\frac{X}{R}} = \frac{M(w+d)}{1-2S} \tag{14}$$

평균 지연시간은 다음과 같다.

$$W_1 = \frac{(1-\rho)T_c}{2} = \frac{M(w+d)(1-\rho)}{2(1-2M\rho)} \tag{15}$$

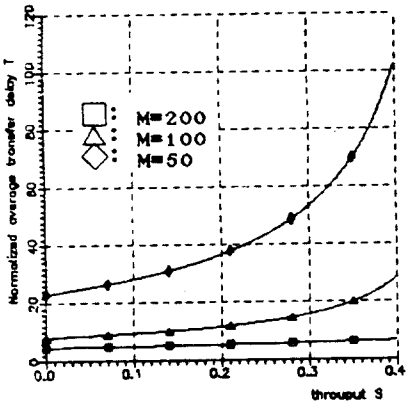
$$W_2 = \frac{SX^2}{2XR(1-S)} \tag{16}$$

전체 평균지연시간은

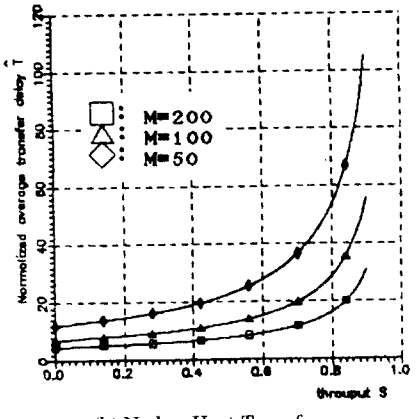
$$T = \frac{\overline{X}}{R} + 2\tau + W$$

$$= \frac{\overline{X}}{R} + \tau + \frac{M(w+d)\left(1-\frac{S}{M}\right)}{2(1-2S)} + \frac{SX}{2R(1-S)} \tag{17}$$

이고 정규화된 평균 지연시간은



(a) Node Node Transfer



(b) Node - Host Transfer

Fig 8. Normalized average transfer delay time $\hat{T}$

$$\hat{T} = \frac{T}{\frac{X}{R}} = 1 + a + \frac{M\hat{w}\left(1-\frac{S}{M}\right)}{2(1-2S)} + \frac{S}{2(1-2S)}$$

$$\left(\hat{w} = \frac{(w+d)R}{X},\ a = \frac{\tau R}{X}\right) \tag{18}$$

로 표현된다. Fig. 8은 $\hat{\omega}=0.1$일때 정규 평균 전송 지연 시간과 노드 갯수와의 관계를 보여주고 있다.

노드가 제공할 수 있는 서비스는 제어 사이클에 의해 결정되고 이는 전송 지연시간과 밀접한 관계를 가지며 서비스의 최대 응답 시간이 결정되면 망에 연결될 노드의 최대 갯수를 알 수 있다. 위의 경우를 종합하여 보면 본 논문에서 제안한 시스템은 frame size가 200 byte인 경우 약 255개의 노드 프로세서를 연결할 수 있다.

4. 결 론

본 논문에서는 분산형 원격제어시스템에서 실시간 처리 성능을 향상시키기 위하여 제어와 통신을 두 프로세서가 분담하여 동시 수행할 수 있는 이중마이크로프로세서 노드를 설계하고, 이중 마이크로프로세서 노드를 인텔사의 단일 칩 마이크로프로세서 8751들을 사용하여 구현하였다.

성능평가를 실험한 결과 $S_P=1.74$의 속도 제고율과 $E_P=0.87$의 병렬 연산 효율을 얻을 수 있었다. 이외에도 이중마이크로프로세서 시스템으로 구성한 경우 다음과 같은 장점이 있다.

(1) 하나의 프로세서로 운용되는 시스템에 약간의 하드웨어를 추가함으로써 구현되기 때문에 가격대 성능비가 높다.

(2) 메모리를 공유하고 있으므로 한쪽 프로세서에 고장이 생겨도 고유데이타의 손실을 막을 수 있으므로 신뢰성이 높다.

또한 멀티드랍 방식의 통신망을 전제로 전송 효율과 M/G/1 待期모델로 해석한 평균 전송 지연 시간을 고려해 볼 때 제안된 노드 프로세서는 중대형 크기의 선박용 원격제어시스템으로서 충분한 성능을 발휘할 수 있을 것으로 기대된다.

앞으로 이중 마이크로프로세서 시스템의 두 프로세서를 효율적으로 활용하기 위한 제어프로그램이 보완되어야 하겠다.

참고문헌

1) V. L. Narasimhan, J. K.Ramachandra and D. K. Avekar, "Design and evaluation of a dual-microcomputer shared memory system with shared I/O bus," ACM Conf. 1986

2) J. M. Cataifo and G. I.Cancelo, "LAN node for industrial environments," ACM. 1988.

3) Y. C. Liu and G. A. Gibson, Microcomputer Systems:The 8086/8088 Family, Architecture, Programming and Design, PRENTICE-HALL, Inc., 1984.

4) C. Weitzman and TRW Defence and Space System Group, Distributed Micro/Minicomputer Systems Structure, Implementation, and Application, PRENTICE-HALL, Inc., 1980.

5) W. M. Loucks, "Short-Packet Transfer Performance in Local Area Ring Network," IEEE Trans On. Computers vol. c-34, 1985.

6) D. Bursky, Microprocessor System Design and Applications, Hayden Book Company, 1980.

7) J. D. Spragins and J. L. Hammond, Telecommunications protocols and design, Addision Wesely Publishing Company, 1991.

8) J. L. Hammond and P. J. O'relly, Performance Analysis of Local Computer Network, Addition-Wesely Pudlishing Company, Inc., pp. 200~215, 1986.